Security of Modes of Operation and other provably secure cryptographic schemes

Ferdinand Sibleyras

Sorbonne Université

École Doctorale Informatique, Télécommunications et Électronique
ED130, Paris

Inria de Paris / Équipe-projet COSMIQ

Security of Modes of Operation and other provably secure cryptographic schemes

Thèse de doctorat d'informatique

présentée par

Ferdinand Sibleyras

et soutenue publiquement le 23 octobre 2020

devant un jury composé de :

Anne Canteaut	Inria	Directrice
Gaëtan Leurent	Inria	Directeur
Henri Gilbert	ANSSI Crypto Lab	Rapporteur
Bart Preneel	KU Leuven	Rapporteur
Raphaël Bost	DGA	Examinateur
Tetsu Iwata	Nagoya University	Examinateur
Antoine Joux	CISPA Helmholtz Center	Examinateur
Damien Vergnaud	Sorbonne Université	Examinateur

Financée à parts égales
par la Direction Générale
de l'Armement (DGA) et
Inria.

Équipe-projet
COSMIQ,
Inria de Paris,
2 rue Simone Iff,
75012 Paris.

Remerciements

Une thèse commence habituellement par des remerciements où l'humour vient maladroitement cacher l'émotion qui accompagne la fin. Malheureusement, je ne suis pas drôle. Je dédie donc ces remerciements :

À tous ceux qui ont rendu possible cette aventure. Cela commence avec Gaëtan Leurent pour un stage de master qui s'est si bien déroulé qu'il semblait évident de continuer à travailler ensemble en thèse. Et plus tôt encore avec Anne Canteaut, au MPRI, qui a diffusé cette offre de stage et est ensuite devenue ma directrice de thèse. Merci, donc, Anne et Gaëtan pour cet encadrement de qualité. J'ai particulièrement apprécié votre disponibilité et votre réactivité, que ce soit pour signer des documents ou bien pour travailler.

Merci aussi à ceux qui sont là pour la fin : à Henri Gilbert et Bart Preneel qui ont accepté de prendre de leur temps pour rapporter ce manuscrit, et à Raphaël Bost, Tetsu Iwata, Antoine Joux et Damien Vergnaud qui complètent le jury afin de valider le travail accompli. Je n'oublie pas non plus Damien Vergnaud et Brice Minaud qui ont assuré mon comité de suivi le long de la thèse.

À tous mes collègues. J'ai eu la chance de faire partie d'une excellente équipe de recherche, l'équipe SECRET devenue COSMIQ. Les occasions de se retrouver étaient nombreuses et toujours appréciées : les conférences, les workshops, les groupes de travail mais aussi la cantine de Bercy, les mots fléchés, le baby-foot... Je remercie donc tous les membres passés et présents de l'équipe que j'ai eu la chance de rencontrer en citant pêle-mêle et forcément avec des oublis : Anne, André (dit le Sage), Pascale, Gaëtan, Christelle, Anthony, María, Léo, Nicolas, Jean-Pierre, Christina, Xavier, Rémi, Kevin, Daniel, Thomas, Antonio, Antoine, Matthieu, Vivien, Rocco, Andrea, André (dit le Jeune), Clémence, Valentin, Mathilde,

Augustin, Clara, Nicolas (ou David), Pierre, Kaushik, Julia, Yann, Virginie, Sébastien, etc, etc.

Mais j'ai aussi eu l'occasion de rencontrer beaucoup d'autres chercheurs que je ne pourrais tous citer. Je mentionnerai tout de même mes coauteurs : Mridul, Donghoon, Nilanjan, Avijit, Bart et Somitra. Un remerciement particulier à Mridul Nandi, deux fois coauteurs, pour m'avoir également invité en Inde au workshop ASK qui s'est avéré fructueux.

À toute l'équipe NTT Secure Platform Laboratories. Je remercie chaleureusement Yu Sasaki de m'avoir invité à Tokyo. Un grand merci aussi à Kouji Wakasa et Damien Vergnaud pour avoir rendu cela possible. J'y ai passé un bon moment en compagnie de nombreuses personnes dont Yu, Mehdi, Akinori, Alexandre, Miguel, Thomas, etc.

Mon passage à NTT fut plus court que prévu mais je compte bien saisir l'opportunité d'y retourner pour un post-doctorat.

À toute ma famille. D'abord à mes parents Gérald et Pome qui m'ont toujours encouragé pendant mes longues études, à Sylvie aussi et à mes frères et sœurs Léon, Marius, Louise et Alfred. Une pensée aussi pour mes grands-parents et en particulier à Renate qui aurait adoré me voir docteur. Mais aussi je dois absolument remercier Tomomi qui fait maintenant partie de ma famille et qui accepte de me suivre entre la France et le Japon. Tu es aussi celle qui a assisté à l'écriture d'une grande partie de ce manuscrit pendant que nous étions confinés à Paris ce qui, malheureusement, n'a pas trompé ton ennui.

En particulier, pour ces trois dernières années, je remercie encore ma mère, d'abord, puis ma femme avec qui j'ai vécu et qui m'ont directement soutenu sans jamais vraiment comprendre ce que je pouvais bien faire de mes journées.

À tous mes amis. Je pense d'abord à Baptiste avec qui j'ai étudié du collège à l'université et qui aurait dû continuer à faire de la cryptographie [VN17]. Je souhaite aussi remercier Thomas Baignères qui m'a d'abord orienté vers l'EPFL puis vers le MPRI et enfin vers Inria ; ce sont peut-être, après mon mariage, les meilleurs choix que j'ai faits de ma vie.

Et puis il y a tous les autres qui m'ont permis, de temps en temps, de changer d'air, de m'amuser, de penser à autres choses. Cela peut être par des jeux, des bars ou simplement en gardant contact. Je pense bien

sûr aux *KRT*, au groupe *FRESHROOM*. Je pourrais mentionner Romain, Joris, Olivier, Benoît, Pierre, Fabien, Maxime, Ilhan, Natsuki, Quentin, Mathieu... mais je préfère m'arrêter rapidement plutôt que d'espérer naïvement n'oublier personne.

Enfin, je souhaite une bonne lecture aux plus téméraires qui s'aventureront au-delà de ces lignes.

Contents

Survol des Contributions

Que ce soit pour des transactions commerciales ou bien simplement pour échanger des messages, la sécurité de nos infrastructures numériques repose essentiellement sur de la cryptographie. D'une part, la cryptographie asymétrique, dite à clé publique, est capable de résoudre des problèmes complexes tels que le partage de clé via un canal public ou encore la signature numérique. D'autre part, la cryptographie symétrique, dite à clé privée, est de plus en plus sollicitée afin de chiffrer efficacement une masse de donnée chaque jour croissante.

Cette thèse traite de cryptographie symétrique et a comme sujet principal l'étude des modes d'opération ou, plus largement, des constructions dont la sécurité peut être formellement prouvée.

Chiffrement par bloc. De nos jours, une grande partie de la cryptographie symétrique repose sur les chiffrements par bloc tels que l'AES (Advanced Encryption Standard). Un chiffrement par bloc est défini comme une famille de permutations indexée par une clé. Un chiffrement par bloc E de taille n bits et admettant une clé de taille κ bits est donc une application $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. Typiquement, un chiffrement par bloc sûr est une permutation pseudo-aléatoire (PRP), cela signifie qu'une permutation prise aléatoirement parmi cette famille (en prenant une clé aléatoire) est indistinguable d'une permutation prise aléatoirement parmi toutes les permutations possibles.

Définition 1 (Distingueur PRP)**.** Soit $\mathcal{P}$ l'ensemble de toutes les permutations de n bits vers n bits et $\mathcal{A}^{f(\cdot)} \to 1$ l'événement "$\mathcal{A}$ renvoie 1 après interaction avec f". L'avantage de $\mathcal{A}$ dans le distingueur PRP d'un chiffrement par bloc E est :

$$\mathbf{Adv}_E^{\mathsf{prp}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{E_k(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{p(\cdot)} \to 1) \ ,$$

avec $k \xleftarrow{\$} \{0,1\}^\kappa$, $p \xleftarrow{\$} \mathcal{P}$ et les choix aléatoires de $\mathcal{A}$.

La sécurité PRP de E est l'avantage maximal parmi tous les adversaires $\mathcal{A}$ limités à t opérations et σ appels à l'oracle, notée $\mathbf{Adv}^{\mathsf{prp}}_E(t)$.

En fait, il est toujours possible d'énumérer toutes les clés pour vérifier si la permutation provient du chiffrement par bloc ou bien si elle a été tirée aléatoirement. Ceci est une attaque par recherche exhaustive sur la clé et s'applique à n'importe quelle PRP avec un coût de $t = \mathcal{O}(2^\kappa)$ opérations et quelques appels à l'oracle pour un avantage $\Omega(1)$. Un chiffrement par bloc est donc considéré sûr après σ appels si la recherche exhaustive est considérée trop coûteuse pour être utilisée en pratique et si aucune autre attaque n'est connue.

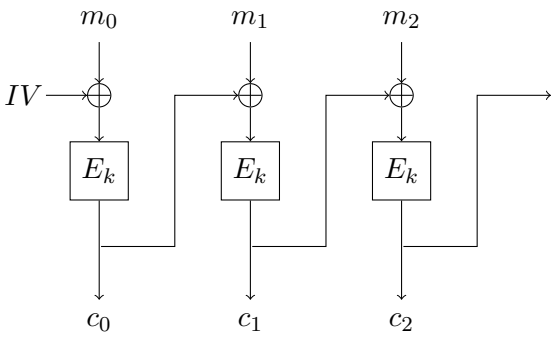


Schéma 1: Diagramme du mode de chiffrement Cipher Block Chaining (`CBC`) qui chiffre un message $m = m_0 \parallel m_1 \parallel \ldots$ à partir d'une clé k et d'une valeur initiale IV comme $c_0 = E_k(m_0 \oplus IV)$ et $c_i = E_k(m_i \oplus c_{i-1})$.

Les modes d'opération. Un s décrit la façon dont des primitives telles que les chiffrements par bloc, les chiffrements par bloc paramétrables ou encore les permutations peuvent être enchaînés dans un but cryptographique précis. Il s'agit là, par exemple, de chiffrer afin de garantir le secret, d'authentifier afin de garantir l'authenticité d'un message voire les deux en même temps. Les modes d'opération sont classés selon leurs buts ; respectivement les modes de chiffrement, les modes d'authentification et les modes de chiffrement authentifié. La représentation en diagramme comme celle du mode `CBC` en Schéma 1 est souvent utilisée pour décrire un mode, et le déchiffrement associé est le plus souvent évident.

Typiquement, la sécurité des modes d'opération est formellement décrite comme l'avantage maximum d'un jeu de type distingueur. Par exemple, nous utiliserons la notion d'indistinguabilité du chiffré d'une suite aléatoire avec messages choisis (IND$-CPA) pour évaluer la sécurité d'un mode de chiffrement, voir Définition 2.

Définition 2 (Distingueur IND$-CPA)**.** Soit un mode de chiffrement qui chiffre un message m avec une valeur initial IV et une clé k comme $c = \mathbf{Enc}_k^{IV}(m)$ et une fonction \$ qui à partir de IV et m renvoie une suite aléatoire de la même taille que le chiffré correspondant. L'avantage d'un adversaire $\mathcal{A}$ pour l'indistinguabilité du chiffré d'une suite aléatoire avec messages choisis est :

$$\mathbf{Adv}_{\cdot}^{\mathsf{IND\$\text{-}CPA}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{Enc}_k^{\cdot}(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{\$(\cdot,\cdot)} \to 1)\ ,$$

la sécurité IND$-CPA d'un tel mode est définit comme l'avantage maximum parmi tous les adversaires $\mathcal{A}$, notée $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}$.

La sécurité des modes d'opération peut ainsi être étudiée formellement en faisant l'hypothèse d'une primitive sûre. En effet, avec l'hypothèse que le chiffrement par bloc est une bonne PRP, il est possible de formellement étudier certains modes en remplaçant le chiffrement par bloc par une permutation parfaitement aléatoire.

Attaques Génériques. Puisqu'il est possible de prouver la sécurité des modes, il est aussi possible de monter des attaques génériques sur ceux-ci qui ne dépendent pas de la primitive utilisée. Une attaque peut directement décrire un adversaire avec un grand avantage pour un distingueur. Par exemple, il existe un adversaire IND-CPA contre le mode `CBC` avec une complexité de $\sigma = \mathcal{O}(2^{n/2})$ blocs de chiffré. Il suffit pour cela d'attendre une collision de deux blocs de chiffré :

$$\begin{aligned} c_i &= c_j \\ E_k(m_i \oplus c_{i-1}) &= E_k(m_j \oplus c_{j-1}) \\ m_i \oplus m_j &= c_{i-1} \oplus c_{j-1} \end{aligned}$$

puis de vérifier que le XOR des blocs de message est égal au XOR des précédents blocs de chiffré ce qui est toujours le cas en `CBC` et très peu probable pour une suite aléatoire. Or une collision sur n bits arrive en

moyenne après $\mathcal{O}(2^{n/2})$ valeurs selon le paradoxe des anniversaires ; c'est pourquoi $\mathcal{O}(2^{n/2})$ est aussi appelée la borne des anniversaires.

Au delà d'un simple distingueur, une attaque peut aussi se mettre dans des modèles plus réalistes et considérer la complexité en calcul dont les preuves ne parlent pas.

Organisation. Ce chapitre est un court abrégé en langue française des différentes contributions de cette thèse. Il sera suivi des introductions de l'état de l'art aux Chapitres 1, 2, 3 et 7 ainsi que des explications détaillées des contributions aux Chapitres 4, 5, 6, 8 et 9. Ces contributions correspondent respectivement à cinq articles publiés dans des conférences internationales avec comité de lecture et actes [LS18], [LNS18], [Cha+19b], [LS19] et [Sib20] (voir Page 25) et sont résumées ci-dessous dans cet ordre.

Le Problème de la Différence Manquante et ses Applications au Mode Compteur

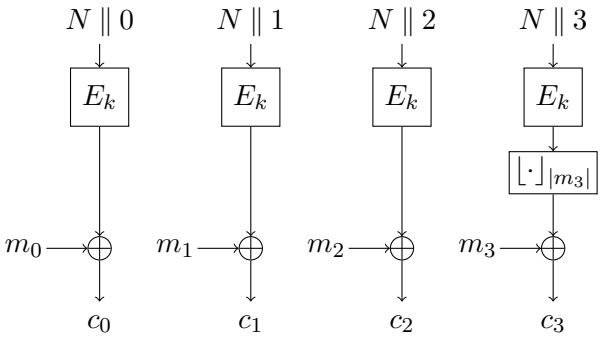


Schéma 2: Diagramme du mode compteur (`CTR`) qui produit les blocs de chiffrés c_i à partir des blocs de messages m_i, d'un nonce N, d'un chiffrement par bloc E et d'une clé k tel que $c_i = E_k(N \| i) \oplus m_i$. La valeur $E_k(N \| i)$ est appelée un bloc de suite chiffrante.

Notre première contribution s'intéresse au mode compteur (`CTR`, voir Schéma 2), un mode de chiffrement largement employé, notamment via le chiffrement authentifié `GCM`, et reconnu pour son efficacité et sa simplicité. Le mode `CTR` bénéficie d'une preuve de sécurité jusqu'à la borne des

anniversaires, soit jusqu'à $\mathcal{O}(2^{n/2})$ blocs de message avec n la taille du chiffrement par bloc utilisé. À partir de cette borne, une attaque de type distingueur est possible simplement en observant l'absence de collision parmi les blocs de suite chiffrante. En revanche, et au contraire de modes tels que `CBC`, ce distingueur ne semble presque rien révéler sur le message qui a été chiffré.

De nouveaux algorithmes. Ainsi, nous étudions les attaques qui extraient de l'information sur un message chiffré selon le mode `CTR`. Pour ce faire nous définissons un problème algorithmique, le problème de la différence manquante (Définition 3).

Définition 3 (Problème de la Différence Manquante)**.** Soit deux ensembles $\mathcal{A}$ et $\mathcal{B}$ et un ensemble d'aide $\mathcal{S}$ tous inclus dans $\{0,1\}^n$. Trouvez une valeur $S \in \mathcal{S}$ telle que :

$$\forall (a,b) \in \mathcal{A} \times \mathcal{B},\ S \neq a \oplus b\ .$$

Nous proposons deux algorithmes pour résoudre efficacement ce problème dans deux cas différents : lorsque $\mathcal{S} = \{0\}^z \times \{0,1\}^{n-z}$, le crible à préfixe connu requiert $\mathcal{O}\left((n-z)\cdot 2^{n-z} + \sqrt{n-z}\cdot 2^{n/2}\right)$ opérations; lorsque $\mathcal{S} = \{0,1\}^n$, le crible par convolution rapide requiert $\mathcal{O}(n\cdot 2^{2n/3})$ opérations grâce à la transformée rapide de Walsh-Hadamard.

Cryptanalyse. Ensuite, nous montrons comment la résolution du problème de la différence manquante permet d'extraire de l'information à partir d'un chiffré selon le mode `CTR` avec une complexité proche de la borne des anniversaires. Pour ce faire, remarquons d'abord que si l'on connaît le message qui a été chiffré alors nous pouvons en déduire le bloc de suite chiffrante qui a été utilisé en calculant $E_k(N \parallel i) = c_i \oplus m_i$ pour de nombreux i ; notons $\mathcal{A}$ l'ensemble de tous les blocs de suite chiffrante observés. D'autre part, nous imaginons avoir accès à de nombreux chiffrements d'un bloc de secret S, c'est à dire à $c_j = E_k(N \parallel j) \oplus m_j$ pour de nombreux $j \neq i$; notons $\mathcal{B}$ l'ensemble de tous les blocs de chiffré de S. Enfin, appelons $\mathcal{S}$ l'ensemble des valeurs possibles de S qui matérialise ainsi notre connaissance à priori sur le secret. Ainsi, résoudre le problème de la différence manquante avec les ensembles $\mathcal{A}$, $\mathcal{B}$ et l'ensemble d'aide $\mathcal{S}$ revient à retrouver le secret S pour peu que la solution soit unique.

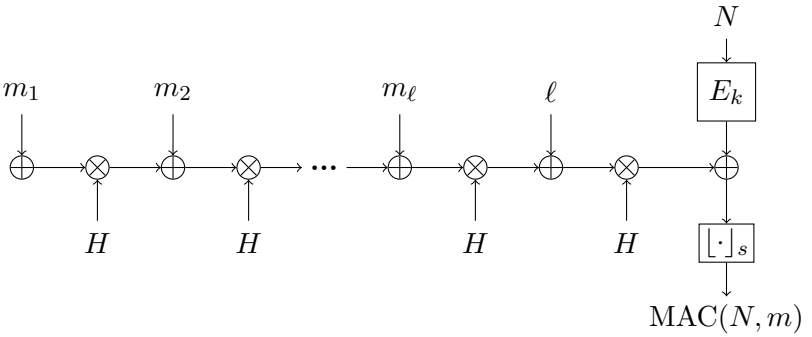


Schéma 3: Diagramme du MAC de Galois `GMAC` pour un message de ℓ blocs, une clé de hachage H et une clé de chiffrement par bloc k.

Plus concrètement, nous supposons avoir accès à de multiples chiffrés avec deux types de préfixes connus, H^1 et H^2, de longueurs respectives d'un bloc et demi et d'un bloc ($3n/2$ et n bits) et auxquels est systématiquement concaténé le même secret S de longueur arbitraire. Ceci est inspiré de scénarios d'attaques réalistes où les préfixes seraient des requêtes banales tandis que S serait un code d'authentification.

Comme H^1 est de longueur d'un bloc et demi, les premiers $n/2$ bits de S sont chiffrés dans le même bloc que les derniers $n/2$ bits du préfixe connu H^1. Nous pouvons donc utiliser l'algorithme du crible à préfixe connu avec $z = n/2$ pour récupérer un demi-bloc de secret S. Ensuite, comme H^2 est de taille n, alors le premier bloc de S sera chiffré par un seul bloc de suite chiffrante parmi les chiffrés de $H^2 \parallel S$. Sachant qu'un demi-bloc de S est maintenant connu, nous pouvons encore utiliser le crible à préfixe connu. Finalement, le secret S est intégralement récupéré en répétant un algorithme de complexité $\mathcal{O}\left(n/2 \cdot 2^{n/2}\right)$.

L'algorithme du crible par convolution rapide a une complexité plus élevée, mais ne demande pas de connaître à priori d'information sur S (le crible à préfixe connu requiert 2^n calculs lorsque $z = 0$). On peut aussi l'utiliser pour attaquer des modes d'authentification (MACs) qui suivent une construction de type Wegman-Carter-Shoup tels que `GMAC` (Schéma 3) et `Poly1305`. En effet, il est possible de récupérer la clé du hachage polynomial en résolvant le problème de la différence manquante. Par exemple, `GMAC` authentifie un message M de taille d'un bloc par $\mathrm{MAC}(N, M) = M \cdot H^2 \oplus H \oplus E_k(N)$ avec N un nonce et donc

$E_k(N)$ ne se répète jamais. L'idée est de fixer deux messages M_1 et M_2 d'une taille d'un bloc, de retrouver la différence $M_1 \cdot H^2 \oplus M_2 \cdot H^2 \neq \mathrm{MAC}(N_i, M_1) \oplus \mathrm{MAC}(N_j, M_2)$ pour tout $i \neq j$, puis de résoudre l'équation en H pour récupérer la clé de hachage. Ainsi, à l'aide du crible par convolution nous montrons une attaque de type forge universelle sur ce genre de MACs utilisant une fonction de hachage polynomiale avec un complexité de $\tilde{\mathcal{O}}(2^{2n/3})$ requêtes, mémoire et opérations. En particulier, cela constitue la première attaque dans ces conditions (nonces respectés et tag non tronqué) sur ces modes en moins de 2^n opérations.

Ceci est un travail commun avec Gaëtan Leurent [LS18].

Attaques Génériques sur les Constructions Double-block Hash-then-Sum MACs

En seconde contribution, nous nous intéressons à la sécurité de plusieurs constructions MACs de type *Double-block Hash-then-Sum* bénéficiant d'une preuve de sécurité au delà de la borne des anniversaires. Ces constructions utilisent un chiffrement par bloc et possèdent un état interne de la taille de deux blocs (c'est à dire de taille $2n$ bits). Concrètement, nous étudions `SUM-ECBC`, `PMAC+`, `3kf9`, `GCM-SIV2`, et des variantes (`LightMAC+`, `1kPMAC+`). Au moment de la publication de ces résultats, ces MACs avaient une preuve de sécurité jusqu'à $2^{2n/3}$ courtes requêtes mais aucune attaque n'avait été décrite en moins de 2^n requêtes. Notre attaque requiert $\mathcal{O}(2^{3n/4})$ courtes requêtes. Cela s'est avéré optimal : une nouvelle preuve de sécurité améliorée par Kim, Lee et Lee [KLL20] pour les modes `SUM-ECBC`, `PMAC+`, `LightMAC+` et `3kf9` montre qu'il est impossible d'attaquer ces modes en moins de $\mathcal{O}(2^{3n/4})$ courtes requêtes sans exploiter de particularité du chiffrement par bloc.

Nous montrons aussi une attaque à la borne des anniversaires contre `1kf9`. `1kf9` avait été retiré pour cause de problème dans sa preuve, mais cette attaque montre qu'une preuve au delà de cette borne est impossible.

Le Tableau 1 résume la complexité de nos attaques par rapport aux preuves.

Stratégie Générique. Les constructions étudiées sont des MACs de la forme $\mathrm{MAC}(m) = E(\Sigma(m)) \oplus E'(\Theta(m))$ avec deux parties de n bits $\Sigma(m)$

Tableau 1: Résumé de la sécurité des BBB MACs étudiés et de la complexité de nos attaques. q est le nombre de tags, ℓ_m la taille maximum d'un message signé, σ le nombre total d'appels au chiffrement par bloc. La borne inférieure et les complexités des attaques sont pour des messages courts ($\ell_m = \mathcal{O}(1)$). "U" signifie forge universelle et "E" signifie forge existentielle.

Mode	Bornes venant des preuves		Attaque (contribution)		
	Avantage	Tags	Tags	Temps	Type
`SUM-ECBC` [Yas10]	$\mathcal{O}\left(\frac{q^4{\ell_m}^3}{2^{3n}}\right)$ [KLL20]	$\Omega(2^{3n/4})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	U
			$\mathcal{O}(2^{6n/7})$	$\widetilde{\mathcal{O}}(2^{6n/7})$	U
`GCM-SIV2` [IM16]	$\mathcal{O}\left(\frac{q^3{\ell_m}^2}{2^{2n}}\right)$	$\Omega(2^{2n/3})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	U
			$\mathcal{O}(2^{6n/7})$	$\widetilde{\mathcal{O}}(2^{6n/7})$	U
`PMAC+` [Yas11]	$\mathcal{O}\left(\frac{q^4{\ell_m}^2}{2^{3n}} + \frac{q{\ell_m}^2}{2^n}\right)$ [KLL20]	$\Omega(2^{3n/4})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	E
`LightMAC+` [Nai17]	$\mathcal{O}\left(\frac{q^4}{2^{3n}}\right)$ [KLL20]	$\Omega(2^{3n/4})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	E
`1kPMAC+` [Dat+17]	$\mathcal{O}\left(\frac{\sigma}{2^n} + \frac{q\sigma^2}{2^{2n}}\right)$	$\Omega(2^{2n/3})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	E
`3kf9` [Zha+12]	$\mathcal{O}\left(\frac{q^4{\ell_m}^6}{2^{3n}}\right)$ [KLL20]	$\Omega(2^{3n/4})$	$\mathcal{O}(\sqrt[4]{n} \cdot 2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{5n/4})$	U
`1kf9` [Dat+15]	$\mathcal{O}\left(\frac{q{\ell_m}^2}{2^n} + \frac{q^3{\ell_m}^4}{2^{2n}} + \frac{q^4{\ell_m}^4}{2^{3n}} + \frac{q^4{\ell_m}^6}{2^{4n}}\right)$	$\Omega(2^{2n/3})$	$\mathcal{O}(2^{n/2})$	$\widetilde{\mathcal{O}}(2^{n/2})$	U

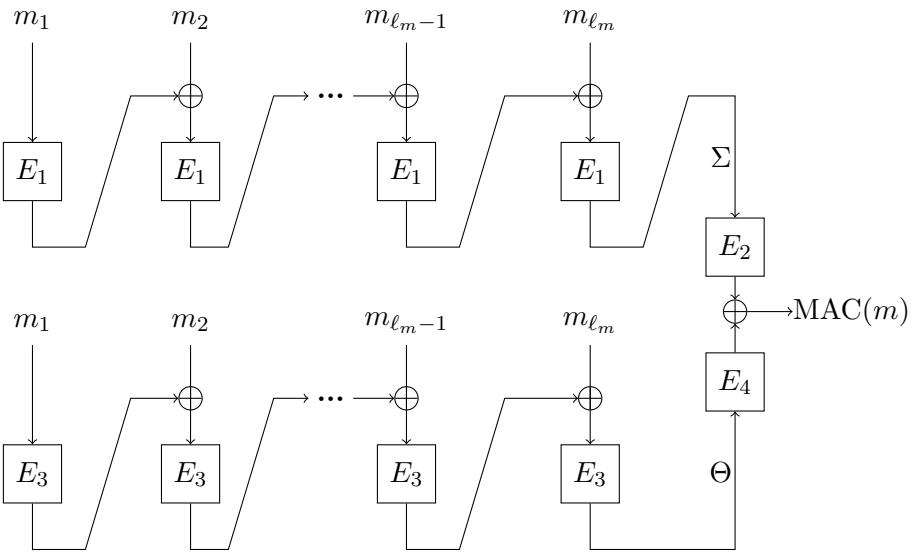


Schéma 4: Diagramme du mode `SUM-ECBC` avec ℓ_m blocs de message.

et $\Theta(m)$ (*Double-bloc Hash-then-Sum*). Voir le diagramme de `SUM-ECBC` (Schéma 4) pour un exemple. Notre cryptanalyse recherche des quadruplets de messages avec 4 collisions deux à deux sur une moitié de l'état interne. Cela signifie que nous cherchons un quadruplet de messages (X, Y, Z, T) tel qu'il satisfasse la relation $\mathcal{R}$:

$$\mathcal{R}(X,Y,Z,T) := \begin{cases} \Sigma(X) = \Sigma(Y) \\ \Theta(Y) = \Theta(Z) \\ \Sigma(Z) = \Sigma(T) \\ \Theta(T) = \Theta(X) \end{cases}$$

qui implique la relation :

$$\mathcal{R}(X,Y,Z,T) \implies \mathrm{MAC}(X) \oplus \mathrm{MAC}(Y) \oplus \mathrm{MAC}(Z) \oplus \mathrm{MAC}(T) = 0.$$

La forme des messages X, Y, Z, T est ensuite choisie de façon à ce que cette relation $\mathcal{R}$ soit non-seulement possible mais surtout forme une relation linéaire de rang trois, c'est à dire que :

$$[\Sigma(X) = \Sigma(Y) \text{ et } \Theta(Y) = \Theta(Z) \text{ et } \Sigma(Z) = \Sigma(T)] \implies \Theta(T) = \Theta(X).$$

Ainsi, en moyenne un quadruplet parmi 2^{3n} pris au hasard va satisfaire $\mathcal{R}$. Réunir 2^{3n} quadruplets requiert $2^{3n/4}$ différentes valeurs des messages X, Y, Z et T ce qui nous donne une complexité de $\mathcal{O}(2^{3n/4})$ en nombre de requêtes pour cette attaque.

En effet, à partir d'un quadruplet respectant $\mathcal{R}$ il est toujours facile de construire des forges en construisant d'autres messages X', Y', Z', T' dont on peut prévoir qu'ils respecteront $\mathcal{R}$ et, ainsi, prévoir $\mathrm{MAC}(X') = \mathrm{MAC}(Y') \oplus \mathrm{MAC}(Z') \oplus \mathrm{MAC}(T')$.

Complexité de Calcul. Concrètement, nous construisons un filtre en exploitant entre autre la relation sur les MACs pour chercher un quadruplet en résolvant un problème algorithmique de type 4-XOR (Définition 4).

Définition 4 (Problème du 4-XOR)**.** Soit 4 listes L_0, L_1, L_2 et L_3. Trouvez un quadruplet $(e_0, e_1, e_2, e_3) \in L_0 \times L_1 \times L_2 \times L_3$ tel que $e_0 \oplus e_1 \oplus e_2 \oplus e_3 = 0$.

Afin de garder une complexité en requête optimale avec des listes de taille $2^{3n/4}$, le meilleur algorithme connu requiert $\mathcal{O}(2^{3n/4})$ de mémoire et $\mathcal{O}(2^{3n/2})$ calculs. La complexité de calcul dépasse donc les 2^n même si d'un point de vue de la théorie de l'information ces MACs ne sont plus sûrs au-delà de $\mathcal{O}(2^{3n/4})$ tags de courts messages.

Néanmoins, nous parvenons à exploiter une propriété des quadruplets des modes `SUM-ECBC` et `GCM-SIV2` pour obtenir une variante de l'attaque avec une complexité de calcul et de requête $\tilde{\mathcal{O}}(2^{6n/7})$. Ceci est la première attaque en moins de 2^n calculs contre un MAC déterministe de type *Double-bloc Hash-then-Sum*.

Ceci est un travail commun avec Mridul Nandi et Gaëtan Leurent [LNS18].

Déchiffrements non-Vérifiés : Preuve de Sécurité Unifiée et Application à `ANYDAE`

Déchifrements non-vérifiés. Un chiffrement authentifié permet typiquement d'assurer la confidentialité et l'authenticité des messages échangés. Quand un chiffrement n'est pas valide, le déchiffrement ne renvoie rien (ou bien renvoie $\perp$). En pratique, il n'est pas toujours évident de cacher le déchiffrement jusqu'à ce que le tag soit vérifié, en particulier dans certains

modes qui nécessite de déchiffrer le message pour en vérifier l'authenticité. Si l'attaquant à accès au résultat du déchiffrement avant qu'il soit vérifié, la confidentialité et l'authenticité de futurs chiffrés pourraient être compromises. La notion de sécurité en présence de telles fuites, sécurité RUP, a été formalisée par Andreeva et al. [And+14]. Nous proposons un modèle unifié avec celle du chiffrement authentifié classique et adapté aux modes déterministes que nous appellons AERUP, et nous prouvons son équivalence avec l'ensemble des notions classiques de sécurité du chiffrement authentifié couplées à celles de RUP.

Application. Nous décrivons ensuite le chiffrement authentifié ANYDAE (Schéma 5) inspiré de SUNDAE de Banik et al. [Ban+18]. ANYDAE est un mode déterministe à bas coût utilisant un chiffrement par bloc et des fonctions de traitement.

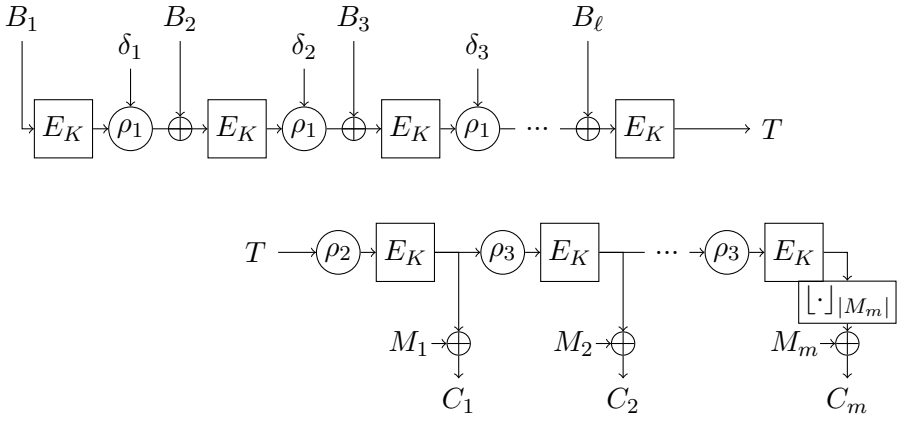


Schéma 5: Diagramme du mode de chiffrement authentifié ANYDAE où les données A et M sont d'abord utilisées dans une fonction de formattage pour calculer $(B[1 \dots \ell], \delta[1 \dots \ell - 1]) \leftarrow \mathsf{Fmt}(A, M)$.

Nous prouvons le Théorème 1, à savoir que ANYDAE est résistant aux fuites de déchiffrements non-vérifiés et est, de fait, AERUP. Une simple attaque prouve en revanche que SUNDAE n'offre aucune protection dans ce scénario.

Définition 5. Soit $\epsilon > 0$, $n \in \mathbb{N}$ et une function $\rho : \{0,1\}^n \times \mathcal{T} \to \{0,1\}^n$ pour un ensemble non-vide $\mathcal{T}$.

- $\rho(X,t)$ est dit ϵ-quasi uniforme si pour tout $t \in \mathcal{T}$ et tout $Y \in \{0,1\}^n$,

$$\mathbf{Pr}(X \xleftarrow{\$} \{0,1\}^n : \rho(X,t) = Y) \leq \epsilon \ .$$

- $\rho(X,t)$ est dit ϵ-quasi XOR-universelle (ϵ-AXU) si pour tout t et $t' \in \mathcal{T}$ distincts et tout $Y \in \{0,1\}^n$,

$$\mathbf{Pr}(X \xleftarrow{\$} \{0,1\}^n : \rho(X,t) \oplus \rho(X,t') = Y) \leq \epsilon \ .$$

Théorème 1 (Sécurité AERUP d'`ANYDAE`)**.** *Soit* `ANYDAE` *avec les functions de formattage et de traitement* Fmt, ρ_1, ρ_2, ρ_3 *et utilisant le chiffrement par bloc* $E\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$.
Nous notons $\mathcal{F}_1$ *l'ensemble des valeurs possible du premier bloc en sortie de* Fmt. *Si*

1. Fmt *est injective et qu'aucune image de* Fmt *n'est le préfixe d'une autre image de* Fmt*;*
2. ρ_1 *est* ϵ_1*-AXU et* γ_1*-quasi uniforme;*
3. ρ_2 *est* γ_2*-quasi uniforme;*
4. ρ_3 *est* γ_3*-quasi uniforme;*
5. $|\mathcal{F}_1 \cap \mathbf{Im}(\rho_2)| = 0$ *et* $|\mathcal{F}_1 \cap \mathbf{Im}(\rho_3)| = \Omega$,

alors

$$\mathbf{Adv}^{\mathit{AERUP}}_{\texttt{ANYDAE}}(\sigma, q_v, t) \leq \mathbf{Adv}^{\mathit{PRP}}_{E}(\sigma, t') + \binom{\sigma}{2}\Big(\frac{1}{2^n} + \max\{\epsilon_1, \gamma_1, \gamma_2, \gamma_3\}\Big) + \Omega\sigma \cdot \gamma_3 + \frac{q_v}{2^n} \, ,$$

pour tout adversaire limité à σ *utilisations du chiffrement par bloc,* q_v *appels à l'oracle de vérification et opérant respectivement en temps* t *et* $t' \simeq t$.

Enfin, nous proposons deux instances concrètes de `ANYDAE` : `MONDAE` et `TUESDAE` qui réunissent toutes les conditions nécessaires à la notion de sécurité robuste AERUP et rivalisent avec `SUNDAE` en termes d'efficacité et d'optimalité. En particulier, `MONDAE` montre que le simple ajout d'une function $\texttt{fix}_1$, qui fixe le bit de poids faible à 1, à `SUNDAE` permet d'obtenir

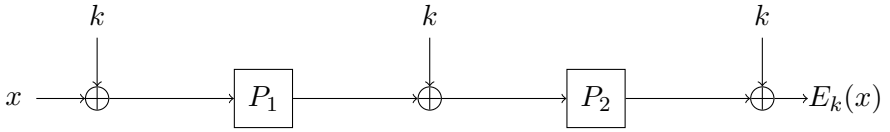


Schéma 6: Diagramme de deux tours de construction Even-Mansour (2EM) avec clé unique k et deux permutations indépendentes P_1 et P_2. Ainsi, $E_k(x) = k \oplus P_1(k \oplus P_2(k \oplus x))$.

un mode plus robuste, alors que `TUESDAY` vise à optimiser le nombre d'utilisation du chiffrement par bloc par rapport à la longueur des messages en entrée.

Ceci est un travail commun initié au huitième *Asian Workshop on Symmetric Key Cryptography* avec Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi et Somitra Sanadhya [Cha+19b].

Cryptanalyse Efficiente en Mémoire de Deux Tours d'Even-Mansour via le Problème du 3-XOR

Even-Mansour. La construction d'Even-Mansour itérée est une façon élégante de construire un chiffrement par bloc à partir de permutations publiques. C'est aussi une façon d'abstraire les constructions de type Substitution-Permutation utilisées entre autres par l'AES. Dans ce travail, nous nous concentrons sur deux tours d'Even-Mansour avec une seule clé (Schéma 6) qui est la construction la plus simple offrant des garanties de sécurité au delà de la borne des anniversaires. En effet, il existe une preuve de sécurité jusqu'à $2^{2n/3}$ évaluations des permutations et du chiffrement. En revanche, les meilleurs attaques connues ont une complexité en temps d'environ $2^n/n$ opérations.

Définition 6 (Problème du 3-XOR)**.** Soit trois fonctions f_0, f_1, f_2 à valeurs dans $\{0,1\}^w$ pour un $w \in \mathbb{N}$. Trouvez trois entrées (x_0, x_1, x_2) telles que $f_0(x_0) \oplus f_1(x_1) \oplus f_2(x_2) = 0$.

Cryptanalyse. Afin d'obtenir de nouvelle cryptanalyse de cette construction, nous faisons le lien avec le problème du 3-XOR (Définition 6)

Tableau 2: Comparaison des complexités asymptotiques des attaques contre 2EM. "Données" est le nombre de chiffrements, et "Requêtes" est le nombre de calculs des permutations publiques P_i.
$0 < \lambda < 1$; $\log n \leq \beta \lll n$; KP : Message connu; CP : Message choisi.

Ref		Données	Requêtes	Temps	Mémoire	Stratégie
[NWW14]	KP	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	Multi-collisions
[Din+13]	CP	$2^n\sqrt{\ln n/n}$	$2^n\sqrt{\ln n/n}$	$2^n\sqrt{\ln n/n}$	$2^n\sqrt{\ln n/n}$	Diff. m-c (clés indep.)
[Din+13]	KP	$2^{\lambda n}$	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	Multi-collisions
[Din+16]	CP	$2^n/\lambda n$	$2^n/\lambda n$	$2^n/\lambda n$	$2^{\lambda n}$	Algèbre lineaire
[IS17]	CP	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	MitM
	CP	$2^{\lambda n}$	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	MitM
	CP	$2^n\beta/n$	$2^n/2^\beta$	$2^n\beta/n$	$2^n/2^\beta$	MitM
S. 8.3.1	KP	n	$2^n/\sqrt{n}$	$2^n/\sqrt{n}$	$2^n/\sqrt{n}$	3-XOR [Jou09]
S. 8.3.2	KP	2^d	$2^{n-d/2}$	$2^n/n$	$2^{n-d/2}$	3-XOR [BDF18]
S. 8.3.2	KP	2^d	$2^{n-d/2}$	$2^n \ln^2 n/n^2$	$2^{n-d/2}$	3-XOR [BDP08]
S. 8.3.3	KP	λn	$2^n/\lambda n$	$2^n/\lambda n$	$2^{\lambda n}$	Peu de données

avec des éléments de taille $w = 2n$. Le 3-XOR est un problème algorithmique bien connu qui, pour être résolu, nécessite au minimum $2^{w/3}$ requêtes mais dont les meilleurs algorithmes demandent $2^{w/2}/w$ opérations. Cela correspond à l'état de l'art pour la cryptanalyse de 2 tours d'Even-Mansour.

Concrètement, il est facile de voir que résoudre le problème du 3-XOR avec les fonctions :

$$\begin{aligned} f_0(x) &:= x && \| \; x \oplus E(x) \\ f_1(y) &:= y \oplus P_1(y) && \| \; y \\ f_2(z) &:= z && \| \; P_2(z) \end{aligned}$$

est équivalent à récupérer la clé k car pour une solution x, y, z nous avons avec bonne probabilité $k = x \oplus y$. Grâce à ce lien, nous décrivons de nouvelles attaques sur 2 tours d'Even-Mansour. Nous parvenons ainsi à décrire la première cryptanalyse où le nombre de chiffrés et de mémoire requis sont significativement en dessous de 2^n. D'un point de vue pratique, les attaques demandant près de 2^n de mémoire et/ou blocs de chiffrés ont peu de chance de rivaliser avec une simple recherche exhaustive de la clé.

Grâce à cette réduction, n'importe quel algorithme pour le 3-XOR peut être utilisé pour attaquer 2EM. Cela signifie qu'une future amélioration des algorithmes pour résoudre le 3-XOR améliorera directement l'état de l'art de la cryptanalyse de 2EM. De plus, en utilisant une variante pour le 3-XOR d'un algorithme pour le 3-SUM de Baran, Demaine et Pătraşcu [BDP08], nous obtenons un algorithme avec une complexité asymptotique en calcul de $\mathcal{O}(2^n \ln^2 n/n^2)$, asymptotiquement meilleur que $\mathcal{O}(2^n/n)$, mais peu performant pour des valeurs pratiques.

En exploitant la forme particulière de notre instance de 3XOR, nous proposons un autre algorithme qui requiert très peu de données et est efficient en mémoire. Pour une constante $0 < \lambda < 1$, cet algorithme utilise seulement λn paires de message/chiffré et $2^{\lambda n}$ mémoire. Par exemple, en prenant $n = 64$ et $\lambda = 1/2$ la mémoire reste tout à fait raisonnable et nous gagnons un facteur 32 par rapport à la recherche exhaustive.

Une comparaison asymptotique de nos résultats et d'autres cryptanalyses est donnée au Tableau 1.

Ceci est un travail commun avec Gaëtan Leurent [LS19].

Attaques Génériques sur les Constructions FX Itérées Paramétrables

Notre dernière contribution s'intéresse à une construction générique pour construire des chiffrements par bloc paramétrables à partir de chiffrements par bloc classiques. Alors qu'un chiffrement par bloc classique de taille n avec une clé, sécrète, de taille κ est une application $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, un chiffrement par bloc paramétrable admet en plus un paramètre (tweak), public, de taille τ et est donc une application $\widetilde{E} : \{0,1\}^\kappa \times \{0,1\}^\tau \times \{0,1\}^n \to \{0,1\}^n$. La sécurité des chiffrements par bloc paramétrables a été formalisée par Liskov, Rivest et Wagner [LRW11].

Notre construction générique généralise une technique fréquement employé pour ce genre de construction : d'abord la taille de la clé est augmentée via une construction FX, puis la clé maître est mixée avec la valeur du paramètre pour produire les sous-clés nécessaires. La construction peut naturellement se répéter en série.

La construction générique. Décrivons formellement la construction FX paramétrable itérée. Soit $E_{1,2,\ldots,r}(u,\cdot)$ r chiffrements par bloc admettant une clé u de κ bits avec une entrée et sortie de n bits. Soit k la clé maître du chiffrement par bloc paramétrable de taille $\widetilde{\kappa}$ bits, t le paramètre de longueur arbitraire. Enfin, soit $\gamma_i(k,t)$ la sous-clé de taille κ de E_i pour $1 \leq i \leq r$ et $\lambda_i(k,t)$ la sous clé de taille n pour $0 \leq i \leq r$. Pour toute paire d'entrée message/paramètre (m,t), la sortie $\widetilde{E}_k(t,m) = c$ de r tours de construction FX paramétrable est définie comme :

$$\begin{aligned} s_0 &:= m \oplus \lambda_0(k,t) \\ s_i &:= E_i(\gamma_i(k,t), s_{i-1}) \oplus \lambda_i(k,t) \qquad , \text{ pour } 1 \leq i \leq r \\ c &:= s_r \ . \end{aligned}$$

Voir le Schéma 7 de la construction pour $r = 2$ tours.

Cryptanalyse. Notre résultat sur deux tours est une attaque théorique possible lorsque $\kappa \leq 2n$ en complexité, toutes requêtes confondues, de :

$$Q = \mathcal{O}\left(2^{\frac{2}{3}(n+\kappa)} \cdot \sqrt[3]{\widetilde{\kappa}/n}\right) \ .$$

Sous l'hypothèse raisonnable que $\widetilde{\kappa} = \mathcal{O}(n)$ alors $Q = \widetilde{\mathcal{O}}(2^{\frac{2}{3}(n+\kappa)})$.

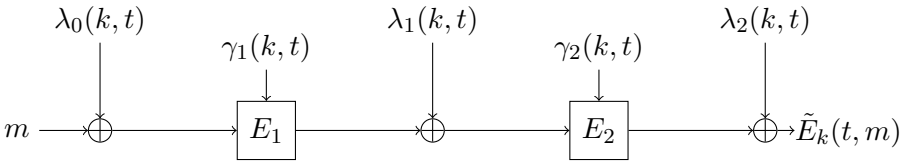


Schéma 7: Diagramme de deux tours de la construction FX paramétrable $\tilde{E}_k(t,m) = E_2(\gamma_2(k,t), E_1(\gamma_1(k,t), m \oplus \lambda_0(k,t)) \oplus \lambda_1(k,t)) \oplus \lambda_2(k,t)$.

Plusieurs constructions dans la littérature suivent ce paradigme sur deux tours pour construire des chiffrements paramétrables et notamment la construction récente `XHX2` par Lee et Lee [LL18] qui est un cas particulier de notre généralisation où $\lambda_1(k,t) = \lambda_0(k,t) \oplus \lambda_2(k,t)$. [LL18] prouve que la sécurité de `XHX2` est $\min\{2^{\frac{2}{3}(n+\kappa)}, 2^{n+\kappa/2}\}$. En particulier, la sécurité est de $2^{\frac{2}{3}(n+\kappa)}$ lorsque $\kappa \leq 2n$ ce qui correspond à la complexité de notre cryptanalyse. Ainsi, notre attaque prouve l'optimalité de leur preuve, et inversement.

Ensuite, nous généralisons cette attaque pour un nombre arbitraire de tours de cette construction et nous obtenons une attaque sur r tours lorsque $\kappa \leq rn$ en :

$$Q = \mathcal{O}\left(2^{\frac{r}{r+1}(n+\kappa)} \cdot \sqrt[r+1]{\widetilde{\kappa}/n}\right)$$

Encore une fois, si on suppose raisonnablement que $\widetilde{\kappa} = \mathcal{O}(n)$ alors $Q = \mathcal{O}(2^{\frac{r}{r+1}(n+\kappa)})$.

Le Tableau 3 montre plusieurs constructions en comparant la meilleure attaque connue avec notre attaque générique. Le principe de notre attaque générique ressemble à l'attaque sur la construction classique FX de Gaži [Gaž13] et prouve qu'une complexité totale de requête plus basse que $\mathcal{O}(2^{\frac{r-1}{r}n+\kappa})$ est atteignable lorsque $\kappa > r/n$ dans le cas d'une construction paramétrable. Néanmoins l'idée reste la même : nous collectons suffisamment de données pour pouvoir reconstruire les valeurs intermédiaires qu'aurait prit l'état interne avec chaque valeur possible de clé maître. Ainsi, nous pouvons vérifier si le résultat que l'on a reconstruit correspond au résultat obtenu en appelant directement l'oracle de chiffrement. Si ce n'est pas le cas, alors la valeur choisit ne correspond pas à celle de la clé maître.

Tableau 3: Comparaison des bornes asymptotiques connus et de notre attaque sur divers constructions.

Ref	Construction	r	Preuve	Attaque connue	Attaque générique
[LRW11]	`LRW2`	1	$2^{n/2}$	$2^{n/2}$	$2^{\frac{1}{2}(n+\kappa)}$
[Men15]	$\widetilde{F}[1]$	1	$2^{\frac{2}{3}n}$	2^n	2^n (as $\kappa = n$)
[Men16]	`XPX`	1	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$ (as $\kappa = 0$)
[Jha+17]	`XHX`	1	$2^{\frac{1}{2}(n+\kappa)}$	$2^{\frac{1}{2}(n+\kappa)}$	$2^{\frac{1}{2}(n+\kappa)}$
[Jha+17]	`GXHX`	1	$2^{\frac{1}{2}(n+\kappa)}$	$2^{\frac{1}{2}(n+\kappa)}$	$2^{\frac{1}{2}(n+\kappa)}$
[Men15]	$\widetilde{F}[2]$	1	2^n	2^n	N.A.
[LRW11]	`LRW1`	2	$2^{n/2}$	$2^{n/2}$	$2^{\frac{2}{3}(n+\kappa)}$
[LST12]	`CLRW2`	2	$2^{2n/3}$	$2^{3n/4}$	$2^{\frac{2}{3}(n+\kappa)}$
[LL18]	`XHX2`	2	$2^{\frac{2}{3}(n+\kappa)}$	$2^{n/2+\kappa}$	$2^{\frac{2}{3}(n+\kappa)}$

Telle qu'elle est décrite, cette attaque demande autant de calculs qu'une recherche exhaustive, c'est à dire $\mathcal{O}(2^{\widetilde{\kappa}})$. Le but de cette attaque n'est donc pas d'être pratique mais plutôt de répondre aux preuves. En particulier, en restant à ce niveau de généralité qui englobe de nombreuses constructions, cette attaque répond à la question suivante : en utilisant la stratégie des constructions FX itérées paramétrables pour construire un chiffrement par bloc paramétrable, quel niveau de sécurité peut-on espérer atteindre ? En effet, pour toute construction à r tours, notre attaque montre qu'aucune preuve ne pourra garantir une sécurité au-delà de $\mathcal{O}(2^{\frac{r}{r+1}(n+\kappa)})$ requêtes avec $\kappa \leq rn$.

Ceci est un travail personnel [Sib20].

Conclusion

Cette thèse aura ainsi exploré et contribué à de nombreux sujets concernant la sécurité des constructions cryptographiques symmétriques formellement prouvables. Nous avons décrit des attaques considérant la complexité total de temps et de mémoire (sur le mode `CTR`, `GMAC`, 2EM, `SUM-ECBC`, etc) ainsi que des attaques plus théoriques qui ne regardent que la complexité en requêtes et répondent ainsi aux preuves (sur les *Double-bloc Hash-then-*

Sum MACs et les constructions FX itérées paramétrables). Nous avons également étudié des notions de sécurité plus robustes, sécurité RUP, en dérivant la preuve de sécurité du chiffrement authentifié `ANYDAE`.

De nombreuses constructions cryptographiques, techniques de preuves et problèmes algorithmiques étudiés tout au long de cette thèse sont abordés plus en détail dans ce manuscrit.

Publications

[LS18] Gaëtan Leurent and Ferdinand Sibleyras. "The Missing Difference Problem, and Its Applications to Counter Mode Encryption". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, Apr. 2018, pp. 745–770 (cit. on pp. 8, 11, 28, 63, 101).

[LNS18] Gaëtan Leurent, Mridul Nandi, and Ferdinand Sibleyras. "Generic Attacks Against Beyond-Birthday-Bound MACs". In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Heidelberg, Aug. 2018, pp. 306–336 (cit. on pp. 8, 14, 28, 76, 131, 257).

[Cha+19a] Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi, Somitra Sanadhya, and Ferdinand Sibleyras. "Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE". In: *IACR Trans. Symm. Cryptol.* 2019.4 (2019), pp. 119–146. ISSN: 2519-173X (cit. on pp. 8, 17, 28, 161).

[LS19] Gaëtan Leurent and Ferdinand Sibleyras. "Low-Memory Attacks Against Two-Round Even-Mansour Using the 3-XOR Problem". In: *CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. LNCS. Springer, Heidelberg, Aug. 2019, pp. 210–235 (cit. on pp. 8, 19, 28, 98, 213).

[Sib20] Ferdinand Sibleyras. "Generic Attack on Iterated Tweakable FX Constructions". In: *CT-RSA 2020*. Ed. by Stanislaw Jarecki. Vol. 12006. LNCS. Springer, Heidelberg, Feb. 2020, pp. 1–14 (cit. on pp. 8, 22, 28, 239).

On Provably Secure Schemes

In symmetric cryptography, provably secure schemes are constructions that can be proven secure under some assumptions. They all rely on some lower-level cryptographic function that we call a primitive and typically come with a proof that reduces their security to the primitive's security under some conditions.

Though the proof and cryptanalysis techniques are somewhat similar, we classify provable secure schemes into two parts. Part I is about modes of operation which describe ways to use an actual primitive to achieve a concrete cryptographic goal such as encryption or authentication. Part II is about idealized designs which describe how to build new primitives upon existing primitives. Examples of what we call idealized designs are the Even-Mansour construction that builds a block cipher from a permutation, and the FX construction that builds a block cipher with a stronger resistance to generic attacks from another block cipher.

As the name suggests, provably secure schemes allow for formal proofs to be made. However, most of this manuscript focuses on the cryptanalysis of such schemes, but we nevertheless introduce proofs techniques and apply them in Chapter 6. Cryptanalysis can be seen as the dual of proofs: while proofs state sufficient conditions that make a scheme secure, cryptanalysis states necessary conditions for its security.

Organisation. The chapters are organized as follows. A general introduction to cryptography is given in Chapter 1 along with some useful results and a motivation for the study of modes of operation. Chapter 2 and Chapter 7 present the state of the art regarding modes of operations and idealized designs to introduce Part I and II, respectively. Chapter 3 pursues the presentation of the state of the art and details generic attacks on various modes of operation. Although it belongs to Part I, algorithmic

techniques introduced in Chapter 3 will remain relevant in Part II when talking about cryptanalysis of idealized designs.

The remaining chapters correspond to the contributions of a publication each. In Part I, Chapters 4, 5, and 6 describe the contributions of [LS18], [LNS18], and [Cha+19a], respectively. And, in Part II, Chapters 8 and 9 describe the contributions of [LS19] and [Sib20], respectively. See the list of publications on page 25.

Notations

$\{0,1\}^n$: The set of all n-bit words.

0^n : The n-bit all 0 word.

$\mathcal{S} \times \mathcal{T}$: The Cartesian product of sets $\mathcal{S}$ and $\mathcal{T}$.

$a \wedge b$: Bitwise AND.

$a \oplus b$: Bitwise exclusive-or, XOR.

$a \parallel b$: Concatenation of a and b.

$a \stackrel{?}{=} b$: Test for equality of a and b.

$|a|$: The bit length of a.

$a_{[i]}$: The bit i of a.

$a_{[i:j]}$: The bits i to $j-1$ of a.

$\lfloor a \rfloor_s$: The s most significant bits of a.

$\lceil a \rceil_s$: The s least significant bits of a.

$\mathbf{Pr}(x)$: Probability of event x happening.

$\mathbf{Im}(f)$: The image set of function f.

$\mathcal{O}$: The big O notation, or Bachmann-Landau notation, for asymptotic behavior.

$\widetilde{\mathcal{O}}$: The soft O notation ignoring log factors.

$\xleftarrow{\$} \mathcal{S}$: Uniformly randomly drawn from the set $\mathcal{S}$.

$\log(\cdot)/\ln(\cdot)$: the base 2 logarithm / the natural logarithm, respectively.

$E_k(m)$: Block cipher E with key k and message m.

$\widetilde{E}_k^t(m)$: Tweakable block cipher $\widetilde{E}$ with key k, tweak t, message m.

$\mathbf{Enc}_k^{IV}(m)$: Encryption of a message m under the key k and (optional) initial value IV.

$\mathbf{Dec}_k^{IV}(m)$: Decryption of a message m under the key k and (optional) initial value IV.

Chapter 1
Modern Cryptography

1.1 Cryptography from Antiquity to Today

Though cryptography is now a computer science field, the need for secure communication largely predates the Internet and the rise of information theory. Interestingly, the earliest evidences of cryptography may not have had the goal of hiding information but were mainly used to give a sense of mystery. Examples of such use are to be found in the Egyptian tomb of Khnumhotep II, 1900 BC, where some new hieroglyph-like symbols replace the usual ones, or again in the Old Testament where some words are written in Atbash which reverses the Hebrew alphabet (in the Latin alphabet 'a' would become 'z', 'b' would become 'y' and so on).

Domination of Cryptanalysis. Following early ideas, most encryption schemes used were some kind of substitution cipher where each letter is replaced by another letter or symbol. However, those ciphers were routinely broken by cryptanalysts using frequency analysis. The idea is that the most frequently used letter heavily depends on the language and hence corresponds with good probability to the most frequent symbol in the ciphertext.

The earliest known description of cryptanalysis by frequency analysis dates back to the 9th century by Al-Kindi [Al-ry]. It has later been rediscovered in Europe and, by the 15th century, it became very usual among ambassadors to encrypt important messages while employing cryptanalysts to break other's communications. Overall, cryptanalysts were able to break through most ciphers with a certain success despite innovative counter-measures such as adding useless symbols or employing poor orthography. From the 14th to early 20th centuries, cryptanalysts routinely overcame most ciphers by adapting the frequency analysis technique.

Many people would still rely on encryption to hide from curious eyes or, more dangerously, to get a false sense of security. This was the fall of Mary, Queen of Scots, who held confidence throughout her trial in 1586 that her letters couldn't have been deciphered. She believed no one could prove that she approved of the Babington Plot to assassinate the Queen of England Elizabeth I [Sin00]. Her trust relied on her multiple ciphers where additional symbols were used for both useless distractions (Nulles) and common words. Figure 1.1 shows one of Mary's cipher which employs many tricks to deter cryptanalysis by frequency analysis. Nevertheless, all of her letters were systematically brought to the cryptanalyst Thomas Phelippes which had no trouble understanding their contents. Moreover, in order to know who were involved in the Babington Plot, Phelippes imitated Mary's writing and asked in a post-scriptum for all the plotters' name in Mary's behalf by coding it properly. This is a great example of a forgery attack where the secrecy of the cipher wrongly authenticates the author. Following her trial, Mary was executed the next year in 1587.

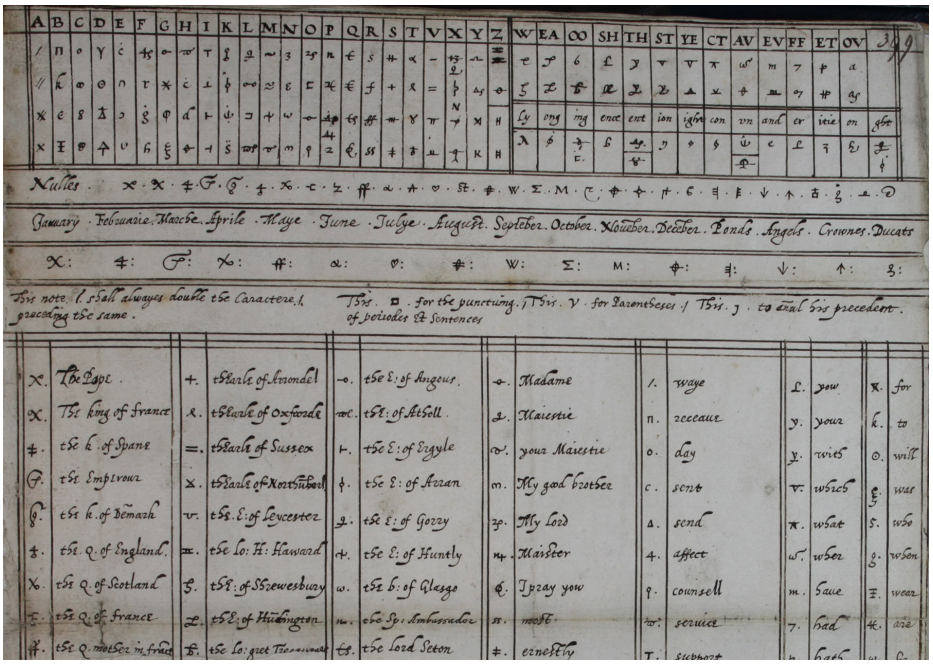


Figure 1.1: One of more than 100 ciphers used by Mary Queen of Scots and seized in her apartment in 1586. Collection of The National Archives (UK) [Que86].

Cryptography during the First World War. In the early 20th century the need for secure communications became more critical than ever. Every army during World War I made extensive use of a new technology: wireless telecommunications. This allowed for faster and more reliable communications than ever, especially at sea, but it also allowed the enemy to easily wiretap any signal sent and intercept the ciphertext.

Interestingly, no notable encryption scheme came out of World War I. Cryptanalysts kept breaking new ciphers as quickly as they came out. Messages of critical importance were routinely decrypted in matter of hours. The most notorious example is probably the invitation by the German Foreign Minister sent to Mexico asking to form an alliance and declare war on the USA. This message, though carefully encrypted, was mostly deciphered within a day by British Intelligence's cryptanalysts Montgomery and de Grey [Sin00]. In the end, this message is what finally convinced the USA to go to war.

1.2 Modern Symmetric Cryptography

Perfect Encryption. The early 20th century also saw the development of the science of information theory. As information started to be thought of at a bit level, a simple and elegant idea for encryption surfaced: the One-Time-Pad. The idea is to randomly flip, or not, each bit of the message. To do this is easy: produce a random key k of the same length of the message m and XOR each bit of the message with each bit of the key to produce the ciphertext c as $c = m \oplus k$. Indeed, at a bit level, if the i^{th} bit of key is 1 then the i^{th} bit of message is flipped in the ciphertext; otherwise it is not flipped. Given a random key, this is equivalent to independently flip each bit with probability 1/2.

In 1919, Vernam deposed a patent for an encryption scheme relying on the XOR operation, thus the One-Time-Pad is also sometimes called the Vernam's cipher.

There is, of course, a heavy drawback: the two parties need to share a key which is at least as long as the message sent, and the key cannot ever be reused again. However, the benefit is just as huge: in 1949 Shannon published a proof for the unconditional security of such a system [Sha49] given a perfectly random key.

Unconditional or perfect security has a precise meaning here that knowing the ciphertext does not change our a priori knowledge of the

message content or, equivalently, that the ciphertext is independent from the message. Formally speaking, let $M = m$ the event "the message is m" and $C = c$ the event "the ciphertext is c", then perfect security means that the following holds:

$$\forall \ell \in \mathbb{N}, \forall m, c \in \{0,1\}^{\ell} \ : \ \mathbf{Pr}(M = m | C = c) = \mathbf{Pr}(M = m) \ ,$$

for any adversary with no knowledge of the key.

Notice that this condition does not ensure that the scheme is sound. Indeed, sending a truly random ciphertext might be secure, but the receiver might not be able to recover the message from the ciphertext. However, this is very easy with the One-Time-Pad when knowing the key as we know what are the flipped bits. We simply need to compute $m = c \oplus k$.

Asymmetric Cryptography. Two major breakthroughs will give rise to modern cryptography as we know it. The first one solves the issue of key distribution. Indeed, sharing a secret key value is far from trivial. It was traditionally done by hand: cryptographic keys would be physically brought to the front line or loaded in a ship before departure. However, in 1976, Diffie and Hellman [DH76] published a key exchange algorithm that solves this issue and is now known as the Diffie-Hellman key exchange. Given an authenticated but public channel, two parties can now share pieces of information and securely derive a common secret key. This secret key can subsequently be used to enable private communication over the channel.

The second major breakthrough came shortly after, and public-key cryptography was invented. In 1977, Rivest, Shamir and Adleman described the RSA cryptosystem [RSA78], the first asymmetric encryption scheme using both a public key and a secret key. Using RSA, one can use its private key to make a signature over a document which can be verified by anyone using the corresponding public key. Conversely, anyone can use one's public key to encrypt a message which can only be deciphered using the corresponding secret key. The keys for encryption and decryption are not the same anymore, and one of them, the public key, can be safely broadcasted. This is called public-key cryptography, or asymmetric-key cryptography.

On the other hand, private-key cryptography, or symmetric-key cryptography, assumes that the communicating parties already share a common and secret key. In practice, symmetric cryptography is much faster than

its asymmetric counterpart. Most communications are actually encrypted via symmetric encryption using a key that was either shared via a Diffie-Hellman key exchange or encrypted with a public-key cryptosystem like RSA. Far from replacing symmetric cryptography, asymmetric cryptography allowed everyone to authenticate themselves and safely share secret keys over the Internet. Nowadays, a massive amount of data is encrypted daily using the combination of asymmetric and symmetric cryptographic schemes in a way that is almost oblivious to the users.

Practical Cryptography. Modern symmetric cryptography uses many types of building blocks or primitives, but the most prominent ones are surely stream ciphers and block ciphers. Stream ciphers can take a key of limited size and produce a key stream of arbitrary size which is then XORed with the message. It can be seen as a practical way of applying the One-Time-Pad but with a pseudo-random key generated by a smaller master key.

On the other hand, block ciphers can be used in multiple ways and are the most common primitive for modes of operation.

Concretely, we define a permutation as a bijective function $P : \{0,1\}^n \to \{0,1\}^n$ and a block cipher as a family of permutations indexed by a secret key. An n-bit block cipher E with a κ-bit key is therefore an application $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. When one knows the key, one knows how to transform any n-bit value into an n-bit codeword. Cryptographers often refer to those n-bit values as words and the transition table as a dictionary. In some sense, given a key, a block cipher provides a compact way of describing a dictionary that enciphers every n-bit words just like we used to transform letters back in early cryptography.

1.2.1 Security of Block Ciphers

Informally, a block cipher is considered secure if, when given a random key, its outputs look like the output of a random permutation. We talk about pseudo-random permutations (PRP) as the randomness in block ciphers really comes from the key while the rest is deterministic.

PRP Security. The security of a block cipher is formally defined using a security game. The game is a distinguisher: a hypothetical adversary $\mathcal{A}$ is given a black-box access to a function f, also called an oracle, and

must distinguish between the ideal and real worlds. In the real world, the function is the block cipher $f(\cdot) = E_k(\cdot)$ with a random key $k \xleftarrow{\$} \{0,1\}^\kappa$, that is a random permutation from the family defined by the block cipher E. In the ideal world, the function is a random permutation.

Definition 1.1 (PRP Security)**.** Let $\mathcal{P}$ be the set of all n-bit to n-bit permutations and let $\mathcal{A}^{f(\cdot)} \to 1$ the event "$\mathcal{A}$ outputs 1 when interacting with the function f". The PRP security game advantage of an adversary $\mathcal{A}$ for a block cipher E is defined as:

$$\mathbf{Adv}_E^{\mathsf{prp}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{E_k(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{p(\cdot)} \to 1) \ ,$$

with the randomness of $k \xleftarrow{\$} \{0,1\}^\kappa$, $p \xleftarrow{\$} \mathcal{P}$ and $\mathcal{A}$.

Then, the PRP security is $\mathbf{Adv}_E^{\mathsf{prp}}(t)$, that is the maximum advantage over all adversaries $\mathcal{A}$ running in time t.

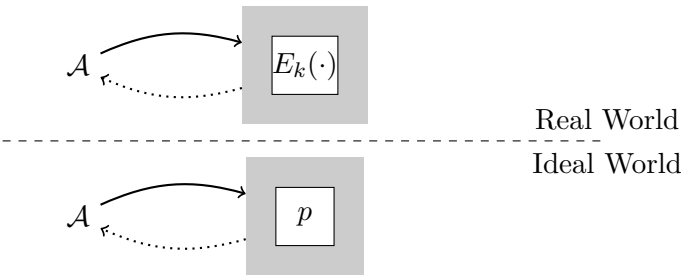


Figure 1.2: Distinguishing game for the prp security of a block cipher E where p is a random permutation and k a random key value.

A low advantage bound makes for a good PRP security as it means that any algorithm or adversary $\mathcal{A}$ will behave the same whether it uses a pseudo-random permutation from the block cipher E or a truly random permutation. In the case where the adversary has access to both f and its inverse f^{-1}, we talk about the strong pseudo-random permutation security or sPRP security.

Definition 1.2 (sPRP Security)**.** The sPRP security game advantage of an adversary for a block cipher E is defined as:

$$\mathbf{Adv}_E^{\mathsf{sprp}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{E_k(\cdot),E_k^{-1}(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{p(\cdot),p^{-1}(\cdot)} \to 1) \ .$$

with the randomness of $k \xleftarrow{\$} \{0,1\}^\kappa$, $p \xleftarrow{\$} \mathcal{P}$ and $\mathcal{A}$.

Then, the sPRP security is $\mathbf{Adv}_E^{\mathsf{sprp}}(t)$, that is the maximum advantage over all adversaries $\mathcal{A}$ running in time t.

Pseudo-Random Functions. A random permutation is a permutation taken at random among all n-bit to n-bit permutations. Similarly, we define a random function as a function taken at random among all functions and a PRF as a family of functions indexed by a key. Therefore, we also define the prf security game that measures how close is a keyed construction from a truly random function.

Definition 1.3 (PRF Security)**.** Let $\mathcal{F}$ be the set of all $\mathcal{X} \to \mathcal{Y}$ functions and let $\mathcal{A}^{f(\cdot)} \to 1$ the event "$\mathcal{A}$ outputs 1 when interacting with the function f". The PRF security game advantage of an adversary for a construction $G : \mathcal{K} \times \mathcal{X} \leftarrow \mathcal{Y}$ is defined as:

$$\mathbf{Adv}_G^{\mathsf{prf}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{G_k(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{f(\cdot)} \to 1) \ ,$$

with the randomness of a parameter $k \xleftarrow{\$} \mathcal{K}$, $f \xleftarrow{\$} \mathcal{F}$ and $\mathcal{A}$.

Then, the PRF security is $\mathbf{Adv}_G^{\mathsf{prf}}(t)$, that is the maximum advantage over all adversaries $\mathcal{A}$ running in time t.

The PRP/PRF Switching Lemma. Consider a random n-bit to n-bit function. The set of all n-bit to n-bit functions surely includes permutations, but it is easy to see that the set of all permutations is a relatively small subset. In other words, such a random function is extremely unlikely to be a permutation as n grows. The fundamental difference between a PRF and a PRP is that a permutation is bijective, that is every different input maps to a different output, while a PRF has collisions with overwhelming probability for large n. We can compare a PRP with a PRF by looking at the PRP advantage of a PRF or vice-versa:

Lemma 1.1 (PRP/PRF Switching Lemma)**.** *Let $n \geq 1$ an integer and $\mathcal{A}$ an adversary performing at most σ oracle queries. Then*

$$\mathbf{Pr}(\mathcal{A}^{f(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{p(\cdot)} \to 1) \leq \frac{\sigma(\sigma-1)}{2^{n+1}}$$

with f a function uniformly drawn from the set of all n-bit to n-bit functions and p a permutation uniformly drawn from the set of all n-bit to n-bit permutations.

This is the PRP/PRF Switching Lemma 1.1 that was first proved by Bellare and Rogaway [BR06]. It implies that a PRF and a PRP are indistinguishable up to $\sigma \ll 2^{n/2}$ computations. Indeed, after $2^{n/2}$ computations we expect to observe a repetition in the output of a pseudo-random function which cannot happen for a pseudo-random permutation.

It is critical to limit the power of the adversary to t time or operations even though it makes formal proof difficult. Indeed, an all-powerful $\mathcal{A}$ can always brute-force the key and see whether the permutation belong to the family defined by E or not. Enumerating all the keys requires $\mathcal{O}(2^\kappa)$ operations giving a substantial advantage.

In practice, sustained cryptanalysis efforts allow us to conjecture that there are no better attacks than brute-force to win the PRP or sPRP security game. The AES block cipher [AES], for one, has earned a great deal of trust, and it is now routinely conjectured that the advantage is $\mathbf{Adv}^{\mathsf{sprp}}_{\mathrm{AES}}(t) \leq 2^{t-128}$ with a 128-bit key size.

This conjecture means that any adversary limited to a realistic computation time has only a low advantage in the PRP and sPRP security game. But what is a realistic computation time? Let us make a comparison: the blockchain community is a good example of people going to great lengths to compute some cryptographic functions over and over again. In March 2020 they reached a rate of 120 quintillions hash per second [BCC], that makes for around 2^{67} computations every second. This is huge. Nevertheless, at this rate and according to the AES conjuncture, it would still take about 73 billions years to reach 2^{128} computations and break a single instance of AES or, alternatively, one billion years to reach a 1/73 PRP advantage.

1.2.2 The Need for Modes of Operation

A good block cipher along with a random secret key thus defines a codebook or a dictionary from any n-bit words to another n-bit words that looks random enough for any adversary. However, messages are of arbitrary size while a block cipher has a typical size of $n = 64$ or 128 bits. Modes of operation allow us to deal with those arbitrary long messages.

A mode of operations is a description of how to treat any messages using a secure primitive such as a block cipher. Typically, the message is first split into n-bit blocks and padded if necessary such that $m = m_1 \,\|\, m_2 \,\|\, ... \,\|\, m_\ell$. It is then processed according to the mode's specifications. We like to

represent a mode of operation with a diagram like the one for CBC on Figure 1.3 that produces a ciphertext c from a message m and, optionally, an initial value IV.

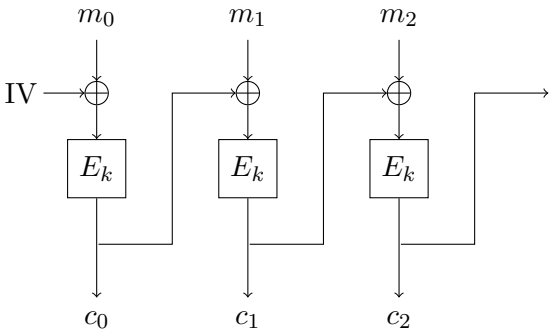


Figure 1.3: Diagram of the Cipher Block Chaining mode of operation (CBC) where $c_0 = E_k(m_0 \oplus IV)$ and $c_i = E_k(m_i \oplus c_{i-1})$.

One could be tempted to use a block cipher just like one used substitution ciphers in earlier times. Simply split and pad the message into n-bit blocks and independently pass them through the block cipher to get the ciphertext as $c_i = E_k(m_i)$. This is the Electronic Code Book mode (ECB), and we show in Section 2.1.2 how it actually fails to provide any satisfying level of security.

From Assumptions to Proofs. A mode of operation usually comes with a formal proof of security. A formal proof is possible with the assumption that the underlying primitive is secure. A secure mode using a block cipher typically assumes a bound on the PRP or sPRP advantage of Definition 1.1. Concretely, modes of operation can be proven secure after replacing the block cipher by a random permutation. If an adversary can break a given construction when instanced with a block cipher but not when instanced with a random permutation, this would be a distinguisher for the sprp security game and break the sPRP security assumption.

In practice, there are many proofs that start by replacing the block cipher by a random function instead of a random permutation. This includes the proof of ANYDAE shown in Chapter 6. The idea is to first replace

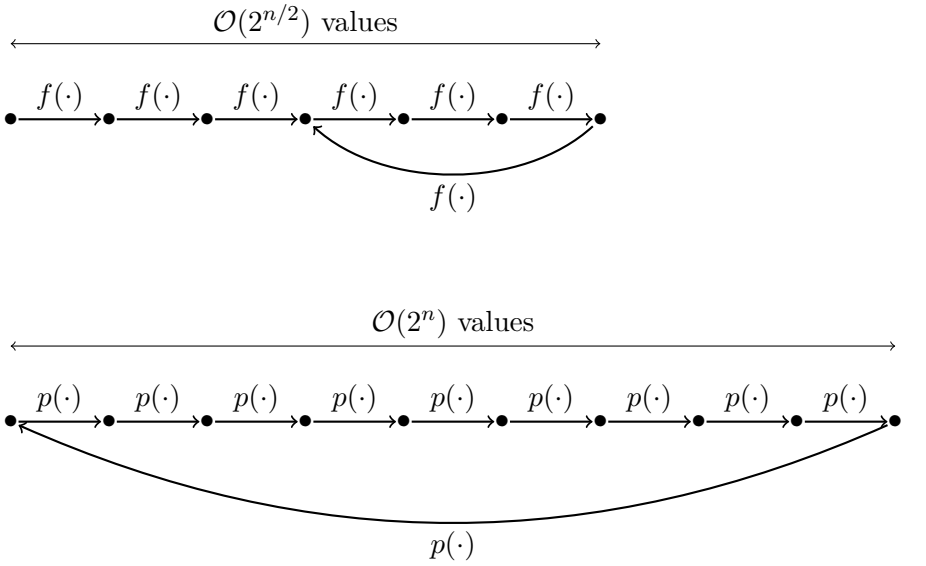


Figure 1.4: Iterating a random function $f(\cdot)$ versus iterating a random permutation $p(\cdot)$.

the block cipher by a random permutation (PRP security assumption) and then apply the PRP/PRF Switching Lemma 1.1.

1.2.3 Random Functions and Permutations

Here we wish to give some insights on the difference in behavior of random functions and random permutations which will be useful throughout this manuscript. Unlike a permutation, a random function can be defined with arbitrary input length though in this Section we limit ourselves to n-bit-input functions. The behavior we expect from random permutations is the same as the behavior we expect from secure PRPs and the same can be said for random functions and PRFs.

Iterating Behavior. While random permutations and random functions are indistinguishable up to $2^{n/2}$ computations, their respective expected behavior diverges afterwards. This is especially true when iterating over and over the function as $x \to f(x) \to f(f(x)) \to ....$

A random function is expected to reach a collision after about $2^{n/2}$ computations which makes the sequence starts looping over previously seen

values. As the image set $\{0,1\}^n$ is finite, any permutation will also have to cycle through previously seen values. However, since a permutation has no collision, it will necessarily loop back to the starting value. In fact, we know the order of magnitude of the length of such cycles. For a random function, it is $\mathcal{O}(2^{n/2})$ as it corresponds to the first collision. This comes from the birthday paradox, see Section 3.1.1 for more details. For a random permutation, it is $\mathcal{O}(2^n)$. Indeed, knowing that a random permutation was iterated σ times without closing a cycle, the probability of going back to the starting value in the next iteration is only $1/(2^n - \sigma)$, thus it most probably won't loop until σ is of order of magnitude $\mathcal{O}(2^n)$. We illustrate the difference in behavior in Figure 1.4.

Why Permutations? At first glance, the reason why we wish for a block cipher to define a family of permutations is so that it is invertible. However, we see in Chapter 2 that many modes actually allow for decryption without the use of the inverse. This is an appreciated feature as it lowers the implementation cost. Some of the most widely used modes, including the Counter mode shown in Section 2.1.4, actually benefits from a better security when used with a PRF instead of a PRP.

The reason may instead be that PRPs are easier to build. We present a few design strategies for building block ciphers in Chapter 7. Such iterative strategies can't be applied to build a pseudo-random function as composing PRFs is a delicate thing to do.

For instance, composing two random permutations p_1 and p_2 as $p(\cdot) = p_1(p_2(\cdot))$ is perfectly fine: it is akin to shuffling a deck twice instead of once. Hence, it provides for another random permutation that is absolutely equivalent to being randomly drawn from the set of all permutations. When composing the same random permutation as $p(\cdot) = p_1(p_1(\cdot))$, the conclusion is less obvious. If we consider the iterating behavior of such p against p_1, all the loops of even size will be halved so that p will contain an unexpected number of loops of odd size. Nevertheless, Minaud and Seurin [MS15] proved that such a p is indistinguishable from a truly random permutation up to $\Omega(2^n)$ computations, meaning that the difference in behavior (the sPRP advantage) is mostly negligible.

On the other hand, composing twice a random function like $f(\cdot) = f_1(f_1(\cdot))$ is only behaving like a true random function up to $\mathcal{O}(2^{n/2})$ computations. The intuition is that composing makes a collision more likely to happen and the iterating behavior (Figure 1.4) will be shorter in

expectation for every loop. Concretely, the first collision happens after $\sqrt{\frac{\pi}{4}2^{n/2}}$ computations instead of $\sqrt{\frac{\pi}{2}2^{n/2}}$ [CL14]. This leads to a prf security game distinguisher running in $\mathcal{O}(2^{n/2})$. This distinguisher even works when composing different random functions like $f(\cdot) = f_1(f_2(\cdot))$. Those results show how subtle it is to deal with random functions while hoping to preserve the expected behavior.

Part I
Modes of Operation

Chapter 2
Introduction to Modes of Operation

Modes of operation describe how we can use some primitive to achieve a concrete cryptographic goal. The primitive can be a permutation, a hash function, a block cipher, a tweakable block cipher, etc. Typically, this primitive is assumed to be secure which allows for the security to be proven information theoretically. The modes of operation are thus naturally classified according to their goals.

In this chapter we successively introduce modes that aim at encrypting messages in Section 2.1, authenticating messages in Section 2.2 and both encrypting and authenticating in Section 2.3. Finally, we conclude by dwelling deeper on various security notions of modes of operation with Section 2.4 that will motivate the contributions shown in later Chapters. Most of the presented modes will rely on the use of block ciphers as it is to date the most widely deployed type of primitive thanks to the standardization and popularity of the DES followed by the AES. However, there are also modes relying on public permutations and others relying on tweakable block ciphers. We give example of such in Section 2.3.4.

2.1 Modes for Encryption

Privacy or confidentiality is the cryptographic goal of an encryption. We want to hide all information about the message, also called plaintext, into a ciphertext.

They are now many modes for encryption but not all of them are standardized and much less in use. In this section we'll introduce some of the most well-known modes and comment on their merits with a focus on the security profile. Not only must an encryption hide the message content but it must also allow the recipient to recover the message knowing the key. Such a mode therefore takes a secret key k, a message m and,

sometimes, an initial value IV as input and outputs a ciphertext c that will be sent along with the IV. Thus, $c = \mathbf{Enc}_k^{IV}(m)$. A decryption algorithm, $m = \mathbf{Dec}_k^{IV}(c)$, is also necessary for completeness but is most of the time obvious given the specification of the encryption.

2.1.1 Security Game

Indistinguishability from Random Bits. To assess the security of a mode of operation doing encryption we imagine the following security game. As for any distinguishing game, the adversary has a black box access to a function, we call it an oracle, and must distinguish between two worlds, the real and the ideal worlds. In the real world, a key k is chosen at random and the adversary has access to the encryption of any message m under a possibly chosen IV and gets the proper corresponding ciphertext: $\mathbf{Enc}_k^{IV}(m)$. In the ideal world, whatever the adversary requests he will get a random bit string of corresponding size, that is the random function $\$(IV, m) \xleftarrow{\$} \{0,1\}^{|\mathbf{Enc}_0^{IV}(m)|}$.

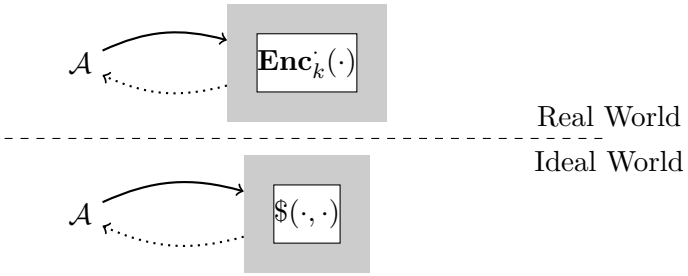


Figure 2.1: Distinguishing game for the IND$-CPA security of an encryption mode where k is a random key value and $\$(\cdot,\cdot)$ is a random function.

The attacker advantage for the indistinguishability from random bits against Chosen Plaintext Attack, IND$-CPA, of a mode of operation is defined as:

$$\mathbf{Adv}_{\cdot}^{\mathsf{IND\$\text{-}CPA}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{Enc}_k^{\cdot}(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{\$(\cdot,\cdot)} \to 1)\ , \tag{2.1}$$

the IND$-CPA security of a mode is understood as the highest advantage against all attacker $\mathcal{A}$, that is $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}$. Because (IV, m) query repetitions would lead to a trivial distinguisher, this game has some variants for restricting the adversary. If there is no IV, $\mathcal{A}$ cannot repeat queries;

the notion is deterministic IND$-CPA like in Section 2.1.2. If the IV is random, $\mathcal{A}$ is not allowed to choose it and the IV is instead randomly drawn before each encryption; we'll call this notion IND$-CPA-rIV as used in Section 2.1.3. At last, if the IV is a nonce, usually denoted N, $\mathcal{A}$ can freely choose the IV but can't repeat twice the same value even for two different messages; we'll call this notion IND$-CPA-N as used in Section 2.1.4.

An IND$-CPA secure mode means that any adversary cannot distinguish the ciphertext from a random bit string uncorrelated with the plaintext. The advantage really measures how close we are from perfect security of the ideal world where every ciphertext is equally likely and independent of the message.

Other Security Notions. There exists other security notions. For example there is the indistinguishability notion that asks the attacker to distinguish the encryption of the input message against the encryption of a random message; or the left-or-right notion where the attacker inputs two messages m_0 and m_1 and receives the encryption of only one of them before guessing which message was encrypted.

The main point is that the discussed IND$-CPA notion is one of the strongest security notion and implies the other ones. In particular, IND$-CPA implies the indistinguishability and the left-or-right security notions.

On Chosen Ciphertext Attack. In this section we won't consider security under Chosen Ciphertext Attack (CCA) where the adversary can freely choose the ciphertext as well as the plaintext. IND-CCA notions are common in public key cryptography but translate poorly in the symmetric setting. This is particularly true for encryption modes whose aim is not to provide security against malleability of the ciphertext. We discuss later ways protect the integrity of the message. In some sense, CCA security will be achieved by the AE notion shown in Section 2.3.1

Assumptions. To allow for a proof of a mode of operation, we need the assumption that the underlying primitive is secure. For instance modes of operation using a block cipher E will be proven by first assuming E is either PRP or sPRP secure and then replacing the block cipher by a truly random permutation. Consider the CBC mode of Section 2.1.3 and let p

be a truly random permutation, then:

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathtt{CBC}-E}(t) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(t) + \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathtt{CBC}-p} .$$

Therefore, we can give a bound of $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathtt{CBC}-E}(t)$ by independently bounding the PRP security of E and proving the security of the mode instanced with a random permutation. While the primitive has a computational security depending on the running time t, the security of modes using a perfect primitive are typically proven in the information theoretic setting and thus does not depend on t. Instead, it will depend on the size and number of queries made.

Leakage. It is harder for an adversary to recover any unknown bit of information on an encrypted message than to win the IND$-CPA security game. This IND$-CPA notion thus captures the leakage of any kind of bit of information.

The only leakage allowed here is the message length. A ciphertext necessarily leaks a maximum message length so avoiding this kind of leakage is beyond the goal of such modes of operation.

2.1.2 An Insecure Mode

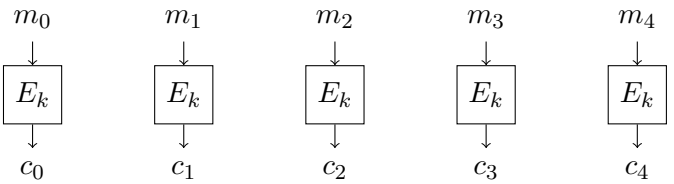


Figure 2.2: Electronic Code Book mode (ECB) where $c_i = E_k(m_i)$.

The Electronic Codebook mode of operation (ECB) is one of the earliest mode of operation as well as one of the simplest. We give its diagram in Figure 2.2. Though it has been standardized as early as 1980 by the FIPS81 document of the NIST for use along with the DES block cipher [FIPS81], it is also a good example of how to NOT use a block cipher.

Algorithm 2.1 IND$-CPA distinguisher for ECB mode for encryption.

1: **input:** f is either encryption or random.
2: **output:** 1 if $f(\cdot) = \mathbf{Enc}_k(\cdot)$, 0 otherwise.
3: **procedure** ECBDISTINGUISHER($f(\cdot)$)
4: Take any $m \in \{0,1\}^n$.
5: $c_0 \parallel c_1 \leftarrow f(m \parallel m)$
6: **return** $c_0 \stackrel{?}{=} c_1$ ▷ return 1 if true, 0 otherwise.

Distinguisher. We can easily show that ECB is not IND$-CPA secure by describing a distinguisher that only needs a single query: the attacker $\mathcal{A}$ requests a 2-block message with two identical blocks $m \parallel m$, gets the encryption $c_0 \parallel c_1$ and simply check whether $c_0 \stackrel{?}{=} c_1$ which will happen every time with ECB but is very unlikely in the random case. We explicitly describe such an adversary in Algorithm 2.1.

The advantage of such an adversary $\mathcal{A}$ that implements Algorithm 2.1 can be easily computed: $\mathbf{Pr}(\mathcal{A}^{\mathbf{Enc}_k(\cdot)} \to 1) = 1$ by construction of ECB mode and $\mathbf{Pr}(\mathcal{A}^{\$(\cdot)} \to 1) = 2^{-n}$ since there is a $\frac{1}{2^n}$ chance that equality of the cipher occurs randomly. Thus, there exists an $\mathcal{A}$ that has an advantage $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA}}_{\mathtt{ECB}}(\mathcal{A}) = 1 - 2^{-n}$ after a single query and a single comparison.

Independently of the cipher used, the ECB mode therefore offers no security guarantees after a single encryption.

In Practice. The ECB mode is well known to spectacularly fail at encrypting raw images. Images have a lot of redundancy thus the input will often repeat and recognizable patterns will be preserved through encryption.

ECB is still largely supported mainly for legacy reasons but, surprisingly, we sometimes also see new products implementing it. For example, a report from Marczak and Scott-Railton analyzing the web meeting service Zoom showed that the AES128 block cipher is used in ECB mode [MS20]. A later white paper by the company itself confirmed the use of ECB though it claims to use the AES256 block cipher [Zoo20].

2.1.3 Legacy Modes for Encryption

Among the earliest modes of operation we can cite the Cipher Block Chaining mode (CBC), but also the Output Feedback mode (OFB) and the

Cipher Feedback mode (CFB). They were standardized early with the ECB mode in the FIPS81 document [FIPS81] to be used with the DES block cipher. Thus, they are largely deployed and still supported by many systems.

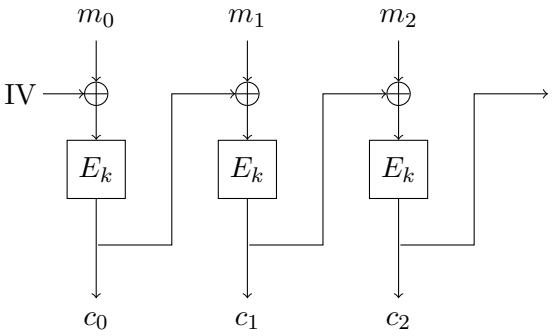


Figure 2.3: Cipher Block Chaining mode (CBC) where $c_0 = E_k(m_0 \oplus IV)$ and $c_i = E_k(m_i \oplus c_{i-1})$.

Cipher Block Chaining. The CBC mode (Figure 2.3) is probably the most popular encryption mode out of the 4 early standardized modes. The security of CBC was first proved in 1997 by Bellare et al. [Bel+97], nearly 17 years after its standardization. They used the left-or-right notion of security, and their result also holds for the IND$-CPA-rIV notion of security according to Rogaway [Rog11]. We won't go into the details of the proof but we give its statement:

$$\mathbf{Adv}^{\text{IND\$-CPA-rIV}}_{\text{CBC}-E} \leq \mathbf{Adv}^{\text{prp}}_{E} + \sigma^2/2^n, \tag{2.2}$$

with σ the number of calls to the block cipher which corresponds to σn bits of processed plaintext.

Output Feedback. The OFB mode (Figure 2.4) can be seen as a stream cipher built with a block cipher. It produces a key stream by applying iteratively the block cipher and then simply XORs the key stream with the plaintext to produce the ciphertext. OFB possesses some advantages over CBC: the decryption is the same as the encryption, it only uses the block cipher in the forward direction, and padding of the message to a

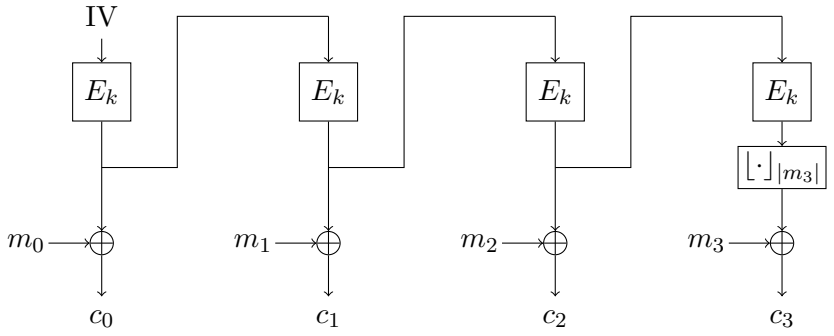


Figure 2.4: Output Feedback mode (OFB) where where $c_0 = E_k(IV) \oplus m_0$ and $c_i = E_k(m_{i-i} \oplus c_{i-1}) \oplus m_i$.

length multiple of n is not needed as one can truncate the unnecessary bits of key stream.

The proof of security of OFB is a direct corollary of the security of CBC. Indeed, the key stream can be seen as the CBC encryption of an all zero $0^{\sigma n}$ message. As long as CBC is IND$-CPA-rIV, the encryption of any message is undistinguishable from a random string. Simply XORing a random string to the message to produce the ciphertext is perfectly secure as it is a One-Time-Pad. Thus,

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}rIV}}_{\mathtt{OFB}-E} \leq \mathbf{Adv}^{\mathrm{prp}}_{E} + \sigma^2/2^n, \tag{2.3}$$

where σ is the number of block cipher calls.

Cipher Feedback. The CFB mode (Figure 2.5) can be seen as a self-synchronizing stream cipher. The message is split into blocks of r bits, and we only use the r most significant bits of the output of the block cipher while throwing out the rest. A proof of security was shown by Alkassar et al. [Alk+02] with the same bound as the CBC. Again their result can easily be translated into the IND$-CPA-rIV notion of security:

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}rIV}}_{\mathtt{CFB}-E} \leq \mathbf{Adv}^{\mathrm{prp}}_{E} + \sigma^2/2^n, \tag{2.4}$$

with σ the number of block cipher calls (processing σr bits of message).

An advantage of CFB over CBC is the use of the block cipher exclusively in the forward direction and the absence of padding. However, when $r < n$ the CFB mode requires more computations for the same message length.

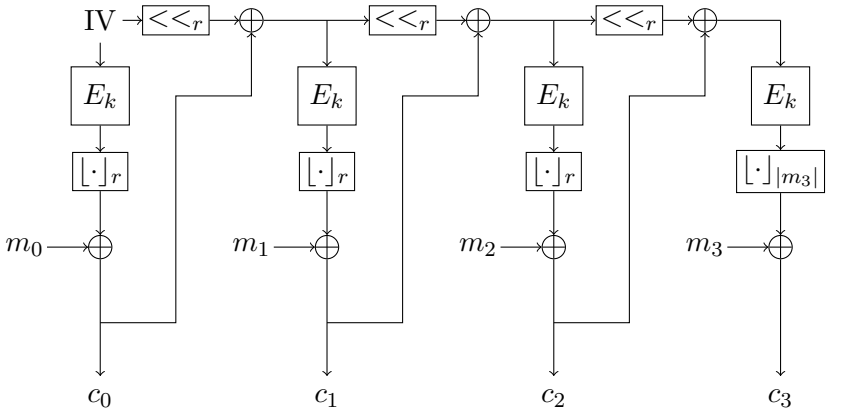


Figure 2.5: Cipher Feedback mode (CFB) where $S^0 = IV$, $c_0 = \lfloor E_k(S^0)\rfloor_r \oplus m_i$ and $S^i = S^{i-1}_{[0:(n-r)]} \,\|\, c_{i-1}$, $c_i = \lfloor E_k(S^i)\rfloor_r \oplus m_i$.

Random IV. The security of modes CBC, OFB and CFB can only be proven secure for random IVs. Concretely, the security game of Section 2.1.1 does not allow the adversary to choose the IV for encryption. Instead, the IV value is sampled randomly by the real world oracle and is returned to the attacker along with the ciphertext. There are indeed easy distinguishers if we let the attacker choose the next IV value to be used.

This can be considered as a weakening of the concept of the IND$-CPA security. In practice, this makes those modes more prone to implementation errors and implicitly requires additional works to properly produce random values.

Matching Distinguishers. All these modes have a matching distinguisher based on collision searching. In every of those modes an attacker can detect a collision in the outputs of a block cipher and thus deduce a collision in the inputs. For example, with the CBC mode, when one detects $c_i = c_j$ for some $i \neq j$, the equality in input can be verified as $c_{i-1} \oplus m_i \stackrel{?}{=} c_{j-1} \oplus m_j$. Since this equality won't hold with great probability in the ideal world, this makes for a distinguisher.

The probability for a random collision goes up with σ and is roughly $\sigma^2/2^n$. This is the birthday paradox, and the implied $2^{n/2}$ security bound is the birthday bound. Notice that the security upper bound also grows like $\sigma^2/2^n$. This is a matching distinguisher showing the proof is tight.

Since the attacker can choose the message, a random collision is unavoidable for CBC and CFB. For the OFB mode, the situation is slightly different: one avoids collision by encrypting its messages as a single very long message. It suffices to use a single IV and chain the permutation. This ensures that the only possible collision is when the internal state goes back to the initial IV. We've seen in Section 1.2.3 that the expected number of blocks before the internal state goes back to the starting value (initial IV) is in the order of $\mathcal{O}(2^n)$, much greater than $2^{n/2}$.

However, even in that case, there is a matching distinguisher as one can detect the absence of collision. In the ideal world the values $m_i \oplus c_i$ are completely random, thus collisions are bound to happen after $2^{n/2}$ block encryptions. The absence of such collision when σ approaches the birthday bound is sufficient to build a matching distinguisher. Therefore, the OFB mode used in this way shares the same distinguisher as the CTR mode shown in Section 2.1.4 and even shares the more advanced plaintext recovery attack of Chapter 4.

Conclusion. Those three legacy modes of encryption have the great advantage of being secure, especially compared with the ECB mode. Among those, the popularity of CBC was probably due the vague protection against tampering of the message. If one changes a bit of the ciphertext, it will make a whole block of plaintext looks like garbage after decryption. But it will still predictably change another block of plaintext which may be problematic.

Drawbacks of the CBC mode therefore include this fake sense of protection but, more importantly, the need for random IVs as well as the need for a good padding scheme of the message to a multiple of a block size or some ciphertext stealing. While the CBC mode seems easy, it is just as easy to get it wrong. On the other hand, the Counter mode of encryption (Section 2.1.4) solves those issues while keeping it simple.

2.1.4 The Counter Mode

At the time of this writing the Counter mode (CTR) is arguably the most deployed and used standard for encrypting data, in particular for TLS secured connections. The idea of the Counter mode is due to Diffie and Hellman [DH79] as early as the legacy modes of Section 2.1.3, but its standardization is more recen. The NIST started recommending the CTR in 2001 [Dwo01], and it is mostly used along with the AES block cipher.

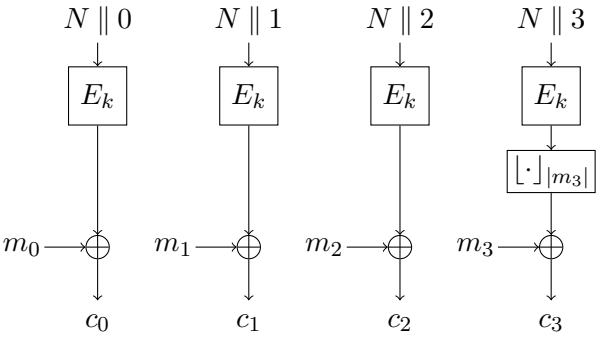


Figure 2.6: Diagram of the counter mode (`CTR`) where $c_i = E_k(N \| i) \oplus m_i$.

The `CTR` mode (Figure 2.6) can be seen as a stream cipher built upon a block cipher. It uses a nonce N as an IV to initialize a counter that goes as input of the block cipher. The message is encrypted with a simple XOR of the output with the plaintext to produce the ciphertext as $c_i = m_i \oplus E_k(N \parallel i)$.

Thus, the block cipher is only used in the forward direction, and decryption is the same as encryption. It also doesn't require any padding as it can simply cut the key stream to the exact message size. The `CTR` mode is also highly parallelizable.

Security Proof. Not only is the `CTR` mode provably secure, but its proof is actually quite simple. The proof is mainly about bounding the advantage an adversary has in distinguishing E from a random function, that is bounding the value $\mathbf{Adv}^{\text{prf}}_E(t)$. Then we simply need to realize that the `CTR` mode used with a random function is perfectly secure: as the counter ensures the input is never repeated, the output is always a "fresh" random value and perfectly masks the plaintext like a One-Time-Pad would do.

The proof unfolds:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}N}}_{\mathtt{CTR}-E} &\leq \mathbf{Adv}^{\mathrm{prf}}_{E} + \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}N}}_{\mathtt{CTR}-\mathrm{prf}} \\
&\leq \mathbf{Adv}^{\mathrm{prf}}_{E} && \text{As } \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}N}}_{\mathtt{CTR}-\mathrm{prf}} = 0. \\
&\leq \mathbf{Adv}^{\mathrm{prp}}_{E} + \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{prp}} && \text{As } E \text{ is a block cipher.} \\
&\leq \mathbf{Adv}^{\mathrm{prp}}_{E} + \sigma(\sigma-1)/2^{n+1} && \text{Apply Lemma 1.1.}
\end{aligned} \tag{2.5}$$

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}N}}_{\mathtt{CTR}-E} \leq \mathbf{Adv}^{\mathrm{prp}}_{E} + \frac{\sigma^2}{2^{n+1}} \,,$$

with σ the number of block cipher calls, that is the number of encrypted blocks of message.

The step 2.5 uses the PRP/PRF switching lemma (Lemma 1.1) to bound $\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{prp}}$.

Matching Distinguisher. A matching distinguisher directly comes from the fact that a block cipher is a pseudo-random permutation and not a pseudo-random function. As the input never repeats, every block of key stream will be different and no collision is ever possible in the $m_i \oplus c_i$ using the CTR mode. On the other hand, in the ideal world, a completely random cipher will end up giving an n-bit collision at the birthday bound, that is after about $2^{n/2}$ blocks.

Thus, after about $2^{n/2}$ blocks a distinguisher can discriminate with good probability between the ideal and real worlds by looking at the presence or absence of collision between blocks, respectively. In fact, the upper-bound of the security proof and the lower-bound of this distinguisher both grow at the same rate that is roughly $\sigma^2/2^n$. Therefore this is a matching distinguisher.

Nonce-based Encryption. Per definition, a nonce never repeats. If the inputs of the block cipher repeat a single time, the security of the CTR mode completely collapses as it will reuse the same key stream twice. In our description of the CTR these inputs are described as $N \,\|\, i$ which avoids any repetition. However, any sound way of incrementing the input would be secure. In practice, N is often a message counter.

For the IND$-CPA-N security, the adversary can choose the IV himself but it is strictly forbidden to use the same IV twice. This is an improvement

over random IVs based encryption that couldn't allow the adversary to freely choose this value.

Nonce-based encryption like the CTR mode is therefore more robust and less prone to implementation errors. However, one still has to be careful of not repeating the nonces which still makes room for devastating attacks.

Conclusion. The Counter mode is probably the best choice for encryption among the ones presented so far. It achieves a stronger notion of security with an easy proof while keeping the specification simple. Being nonce-based, it may be less prone to implementation errors though one still have to carefully avoid any repetitions.

The late adoption of the CTR mode may be due to its high malleability. Flipping a bit of the ciphertext will predictably flip the corresponding bit of the plaintext. While avoiding malleability is not the goal of an encryption scheme, it is true that CTR must be combined with an authentication mode of operation in order to avoid tampering of the ciphertext.

2.2 Modes for Authentication

The goal of authentication modes of operation is to avoid any undetected tampering of the message. Concretely, an authentication mode outputs an authentication code, also called a tag or a MAC, that is sent along with the message. Cryptographers routinely use the word MAC to describe both the authentication mode and its output, the tag. The message can again be of arbitrary size while the output of a MAC is of fixed size, say s bits, meaning that $\text{MAC}(m) \in \{0,1\}^s$. The message itself is not encrypted.

They are few situations where one needs authentication without encryption but it happens. For example, the Network Time Protocol (NTP) is a protocol to synchronize several distant clocks as precisely as possible which only requires authentication [Leu15]. While the timing is not a secret, it must be authentic as it may be critical to detect replay attacks or to assess the validity of certificates. Authentication-only usually makes much more sense than encryption without authentication. Moreover, a good understanding of authentication modes of operation allows us to build authenticated encryption modes in the next Section 2.3.

2.2.1 Security Game

Existential Unforgeability. To attack an authentication mode of operation, the adversary has to build a forgery. A forgery is defined as a valid message and tag pair that was produced without using the key. Any message will do as long as the tag is correct, that is an existential forgery. As we assume the adversary can choose the message and adapt from the answer, we define the notion of Existential Unforgeability under Adaptive Chosen-Message Attack, we'll call it EUF-ACMA or simply EUF security.

In the EUF security game, a key k is first randomly chosen and the adversary is given access to two oracles:, a MAC generation and a MAC verification. The MAC generation takes any message m and returns $\mathrm{MAC}_k(m)$. The MAC verification asks for a pair $(m, T) \in \{0,1\}^* \times \{0,1\}^s$ and outputs $V(m, T) = \top$ or $\bot$ whether the MAC is valid or not, respectively. So, we define the EUF security advantage for an authentication scheme MAC as:

$$\mathbf{Adv}^{\mathsf{EUF}}_{\mathrm{MAC}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathrm{MAC}_k(\cdot),V_k(\cdot,\cdot)} \text{ forges}) \ , \tag{2.6}$$

with the randomness of k and $\mathcal{A}$. Forging in this context means finding (m, T) such that $V_k(m, T) = \top$ for an m never queried before.

In the case of a nonce-based MAC, the adversary can choose a nonce when querying for a tag $T = \mathrm{MAC}_k(N, m)$ but cannot repeat it for two different queries. However, it can repeat the nonces for verification queries $V_k(N, m, T)$, we'll call this notion EUF-N:

$$\mathbf{Adv}^{\mathsf{EUF\text{-}N}}_{\mathrm{MAC}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathrm{MAC}_k(\cdot,\cdot),V_k(\cdot,\cdot,\cdot)} \text{ forges}) \ . \tag{2.7}$$

Provable Security. In order to be able to prove anything about such statement, one must again assume the underlying primitive to be secure. Block cipher-based authentication modes typically assume the PRP security of the block cipher.

Then, when dealing with provable security, we want to look at the number of queries to the MAC generation oracle, q_t, the number of queries to the MAC verification oracle, q_v and the query length, q_ℓ. For instance, an attacker can always randomly guess the MAC of a given message with an advantage of $q_v/2^s$. It is a kind of brute-force attack that is generic for any authentication mode. A typical proof statement will give an upper-bound of the advantage growing in function of q_ℓ, q_t and q_v.

PRFs are good MACs. The EUF security notion does a good job at capturing the security of an authentication mode but it may often be simpler to compare the authentication mode to a pseudo-random function.

Remember the PRF security Definition 1.3. We apply it with the construction being the MAC function that takes as input any message $m \in \{0,1\}^*$ and outputs an s-bit value that is:

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathrm{MAC}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathrm{MAC}_k(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{f(\cdot)} \to 1)\ , \tag{2.8}$$

where $k \xleftarrow{\$} \{0,1\}^\kappa$ and $f \xleftarrow{\$} \mathcal{F}$ where $\mathcal{F}$ is the set of all $\{0,1\}^* \to \{0,1\}^s$ functions.

While a MAC that is a good PRF is EUF secure, the converse is not true: a MAC could for example leak some information about the message and still be hard to forge, but it would immediately miss the PRF requirement. On the other hand, a non-generic attack on the EUF security directly translate into an attack on the PRF property of the studied authentication mode. More precisely, Bellare, Goldreich, and Mityagin [BGM04] showed that:

$$\mathbf{Adv}^{\mathsf{EUF}}_{\mathrm{MAC}}(t) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathrm{MAC}}(t + q_t + q_v) + \frac{q_v}{2^s}$$

with q_t and q_v the number of tagging and verification queries, respectively. We recognize $q_v/2^s$, the probability of success of randomly guessing the correct tag. Indeed, even a perfect PRF would be subject to this generic attack.

Proofs for deterministic MAC therefore often tend to use this PRF specification, as it will be the case in Chapter 6, while cryptanalysis often looks at the EUF security game as in Chapter 5. On the other hand, nonce-based MACs like the Wegman-Carter in Section 2.2.3 cannot be described as PRFs.

2.2.2 CBC-MAC

One way to build an authentication mode of operation is to adapt the CBC mode to produce a tag or a MAC. As the tag must depend on the whole message and must be of length $s \leq n$, the only value of interest is the last output of a CBC style encryption.

CBC-MAC variants shown here are all secure as PRFs with birthday-bound provable security.

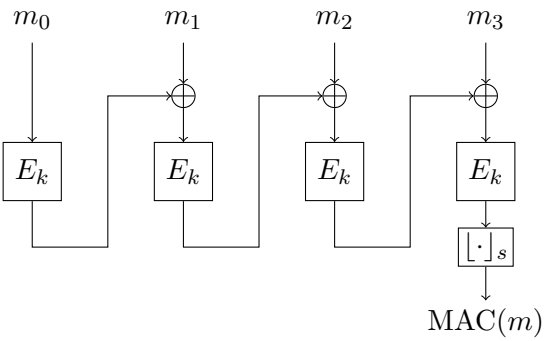


Figure 2.7: Raw CBC-MAC. Outputs the last block of a CBC encryption.

Raw CBC-MAC. A raw CBC-MAC authentication mode is shown in Figure 2.7: it simply truncates the last output of the block cipher to an s-bit tag.

Used as it is CBC-MAC is clearly insecure. There is an easy forgery attack assuming $s = n$: first query the tag $T_1 = \text{MAC}(m_1)$ for some message m_1, then query $T_2 = \text{MAC}(T_1 \oplus |m_2)$. By construction, T_2 is a valid tag for the message $m_1 \| m_2$.

Notice that the attacker can freely choose messages m_1 and m_2 before interacting with the oracles. Therefore, this attack is stronger than an existential forgery attack. We call it a universal forgery attack. Note that we can't deduce the security of CBC-MAC from the IND$-CPA security of CBC encryption as CBC-MAC is deterministic and doesn't use any IV.

However, the mode can actually be made secure by adding the requirement that the processed messages be all prefix-free.

Definition 2.1 (Prefix-free Formatting)**.** A formatting function Fmt is prefix-free if and only if no output is the prefix of another output, that is for any two messages m, m' with $\mathsf{Fmt}(m)$ and $\mathsf{Fmt}(m')$ of lengths ℓ and ℓ' respectively and $\ell < \ell'$ we have $\mathsf{Fmt}(m) \neq \lfloor \mathsf{Fmt}(m') \rfloor_\ell$.

Some padding schemes can act like prefix-free formatting functions (Definition 2.1) by encoding the length of the message in the first block. Another way to do it is to restrict the messages to a fixed length, but that defeats the point of dealing with arbitrary size messages.

Encrypt-last-block CBC-MAC. Alternatively, one can improve the security to deal with any messages by encrypting the output of a raw CBC-MAC

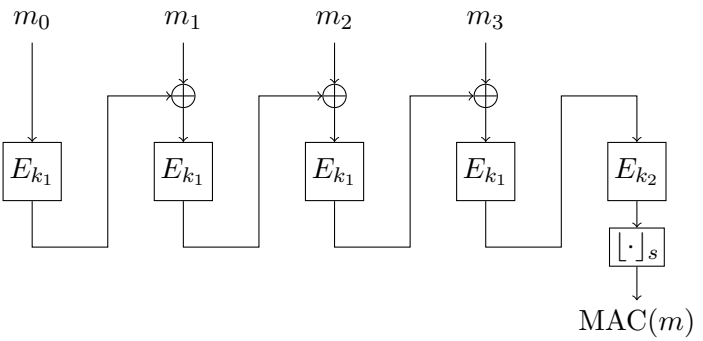


Figure 2.8: Encrypt-last-block Cipher Block Chaining Authentication mode (`ECBC`).

under a second key. This is the `ECBC` construction shown in Figure 2.8. At the price of an additional block cipher call, `ECBC` can be proven secure without requiring prefix-free formatting. A classic 10^* padding is enough: append the message with a 1 and add as many 0 as needed to get to a length multiple of n.

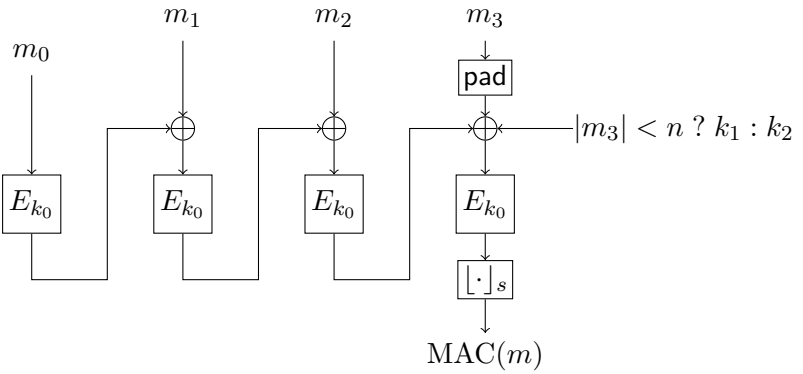


Figure 2.9: Cipher-based MAC Authentication mode (`CMAC`).

`CMAC`. One can do even better with `CMAC`, Figure 2.9. `CMAC` uses three different keys, k_0 for the block cipher, k_1 when padding is required and k_2 when it is not. Simply masking the last input with a key is enough to guarantee security and avoids the need for an additional block cipher computation. Avoiding this kind of additional computation is especially

good for short messages. Moreover, the use of two different keys allows CMAC to optimally deal with messages of length multiple of n. A 10^* padding would append a full block to such a message, but with CMAC it is only used when necessary.

Macthing Forgery Attacks. Matching attacks can be shown for all CBC-MAC modes. In fact, a generic forgery attack by Preneel and van Oorschot [Pv95] implies a matching birthday-bound cryptanalysis against all iterated deterministic MAC with an n-bit internal state. We detail this generic attack in Section 3.1.2.

2.2.3 The Wegman-Carter Construction

The idea of Wegman and Carter [WC81] is to use a keyed almost XOR-universal (AXU) hash function h along with a pseudo-random function F. The Wegman-Carter MAC is $\mathrm{MAC}(m) = h_{k'}(m) \oplus F_k(N)$ with the secret keys k and k' and a nonce N.

This construction benefits from strong security guarantees as it is as strong as the PRF and the universal hash function employed. Let a construction $\mathtt{WC} - h, F$ using a δ-almost XOR-universal hash function h (Definition 2.2) and a family of functions F then:

$$\mathbf{Adv}^{\mathsf{EUF\text{-}N}}_{\mathtt{WC}-h,F}(t) \leq \mathbf{Adv}^{\mathsf{prf}}_{F}(t + q_t + q_v) + q_v\delta + 2^{-n} \ ,$$

against nonce-respecting adversaries with q_v the number of verification queries.

Definition 2.2. Let $\delta > 0$ and a function $h : \mathcal{K} \times \mathcal{T} \to \mathcal{Y}$ for non-empty binary string sets $\mathcal{K}, \mathcal{T}, \mathcal{Y}$.
$h(k, t)$ is said to be δ-almost XOR-universal (AXU) if for any distinct t and $t' \in \mathcal{T}$ and any $y \in \mathcal{Y}$,

$$\mathbf{Pr}\big(k \xleftarrow{\$} \mathcal{K} : h(k, t) \oplus h(k, t') = y\big) \leq \delta \ .$$

Wegman-Carter-Shoup. In practice, we like to use block ciphers as it is often the most readily available primitive. Block ciphers are pseudo-random permutations, but the PRP/PRF switch (Lemma 1.1) tells us that

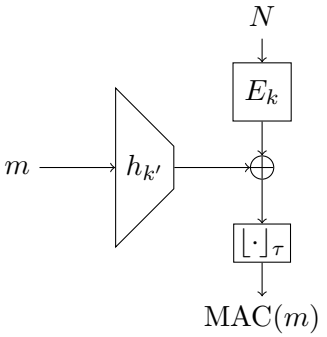


Figure 2.10: The Wegman-Carter-Shoup construction.

a pseudo-random permutation actually behaves like a pseudo-random function up to the birthday bound. The Wegman-Carte-Shoup construction exploits that fact and uses a block cipher E for the function F (Figure 2.10) to compute $\mathrm{MAC}(m) = h_{k'}(m) \oplus E_k(N)$.

The drawback is that the provable security falls down to the birthday bound. One can easily see that by using the PRP/PRF switch Lemma 1.1 and get the bound:

$$\mathbf{Adv}^{\mathsf{EUF\text{-}N}}_{\mathtt{WC}-h,E}(t) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(t + q_t + q_v) + \frac{(q_t + q_v)(q_t + q_v - 1)}{2^{n+1}} + q_v\delta + 2^{-n} \ ,$$

since $(q_t + q_v)$ is the number of block cipher calls.

However, there's been a few works trying to prove that the security still holds as we get near the $2^{n/2}$ bound. Shoup [Sho96] first showed that as long as $q_t < \sqrt{1/\delta}$, then:

$$\mathbf{Adv}^{\mathsf{EUF\text{-}N}}_{\mathtt{WC}-h,E}(t) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(t + q_t + q_v) + 2q_v\delta \ .$$

Later, Bernstein [Ber05a] again improved the bound showing that one could go slightly beyond $\sqrt{1/\delta}$ authenticated messages and still have some good security property. `GMAC` and `Poly1305` are good examples of deployed authentication modes following this strategy.

Galois MAC. A popular example is `GMAC` authentication mode that follows the Wegman-Carter-Shoup construction with the Galois Hash (`GHASH`) AXU secure hash function. Concretely, the `GHASH` function evaluates a Galois Field polynomial, whose coefficients are the message blocks and

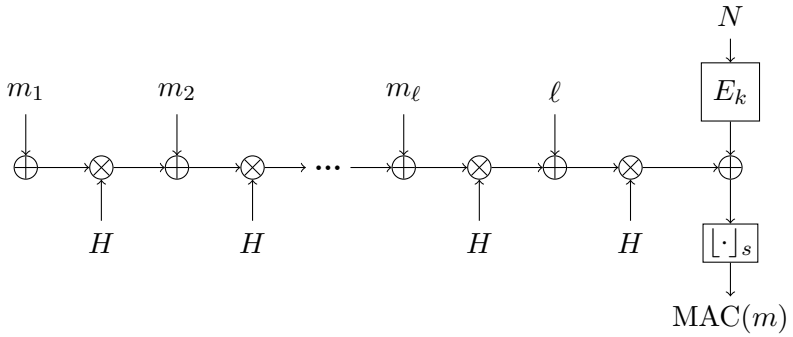


Figure 2.11: The Galois/counter Message Authentication Code `GMAC` construction for an ℓ-block message with an hash key H and a block cipher key k.

block length, at a secret value H that is the hash key. For example, a 3-block message is processed as $\texttt{GHASH}(m) = m_1H^4 \oplus m_2H^3 \oplus m_3H^2 \oplus 3H$.

An advantage of the `GMAC` construction is the possibility of computing the hash in parallel as well as sequentially. As a drawback, we note the need to implement Galois Field arithmetics to perform quick multiplications.

Matching Attack. In the nonce respecting model and untruncated tag ($s = n$), a forgery attack on `GMAC` in the information theoretic setting by Preneel and Luykx [LP18] matches the latest bound by Bernstein [Ber05a]. Independently, and at the same time, Leurent and I [LS18] also found a similar forgery attack but with a greater focus on time and memory complexity which gives the first attack in this setting in time complexity less than $\mathcal{O}(2^n)$.

The main idea of these forgeries is to solve to recover a difference $h_{k'}(m) \oplus h_{k'}(m')$ and deduce the hash key k' by solving the equation. Therefore, this approach works for all Wegman-Carter-Shoup constructions with h a polynomial hashing function. This includes `GMAC` and `Poly1305`. We give details on Chapter 4 for this attack and notably introduce the missing difference problem to recover the difference $h_{k'}(m) \oplus h_{k'}(m')$.

2.3 Modes for Authenticated Encryption

Whenever a secure communication is needed, both privacy and authenticity are required. Hence, authenticated encryption schemes are what we usually aim for: those modes satisfy both encryption and authentication security requirements.

The reason why we often study encryption and authentication separately is because they are generic constructions that can combine a secure encryption with a secure authentication to build a secure AE mode.

Typically, an AE encryption ($\mathbf{AEnc}(m)$) takes a message m as input and returns both a ciphertext c and a tag T. The AE decryption ($\mathbf{ADec}(c, T)$) takes the ciphertext and tag as inputs and outputs m if the tag is correct, $\perp$ otherwise.

Authenticated Encryption with Associated Data, or AEAD mode, simply adds as input some associated data that is a message we wish to authenticate but not to encrypt. It is, most of the time, trivial to deal with associated data in an AE scheme to build an AEAD scheme.

2.3.1 AE Security Game

AE Security. The security we aim for an authenticated encryption mode of operation is roughly the addition of the IND$-CPA and EUF notions in the sense that the mode must satisfy both games.

The IND$-CPA game in this context considers the MAC as being part of the ciphertext, while the EUF game can query for ciphertext and considers as a forgery any successful decryption that was not the result of a previous encryption query. We'll call those notions IND$-CPA-AE and EUF-AE respectively:

$$\mathbf{Adv}_{\cdot}^{\text{IND\$-CPA-AE}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{AEnc}_k(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{\$(\cdot)} \to 1) \tag{2.9}$$

$$\mathbf{Adv}_{\cdot}^{\text{EUF-AE}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{AEnc}_k(\cdot),\mathbf{ADec}_k(\cdot,\cdot)}\text{forges.}) \tag{2.10}$$

Thus, the AE security advantage can be defined as:

$$\mathbf{Adv}_{\cdot}^{\text{AE}} \leq \mathbf{Adv}_{\cdot}^{\text{IND\$-CPA-AE}} + \mathbf{Adv}_{\cdot}^{\text{EUF-AE}} \tag{2.11}$$

Unifying the Security Games. To prove the AE security of a scheme, we can give two proofs: a proof for each game. However, there is a way

to define a single game that allows us to prove the AE security in one go. Let $\mathbf{AEnc}_k(\cdot)$ and $\mathbf{ADec}_k(\cdot,\cdot)$ be the authenticated encryption and decryption oracles respectively. Let $\$(\cdot)$ be a random function that returns a random bit string of length $|\mathbf{AEnc}_k(\cdot)|$ and $\perp(\cdot,\cdot)$ be a function that always returns $\perp$. Then, the AE unified security game is:

$$\mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{AEnc}_k(\cdot),\mathbf{ADec}_k(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{\$(\cdot),\perp(\cdot,\cdot)} \to 1)\ , \tag{2.12}$$

with the randomness of k, $\mathcal{A}$ and $\$(\cdot)$. Of course, the attacker cannot pass a previously queried ciphertext and tag to the verification oracle. For nonce-based modes the attacker is also required to never repeat the nonce in any encryption query, but he can repeat it for decryption ones.

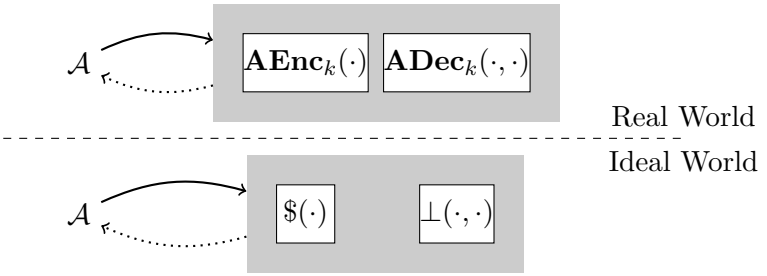


Figure 2.12: Distinguishing game for the AE security of an authenticated encryption mode where k is a random key value, $\$(\cdot)$ is a random function and $\perp(\cdot,\cdot)$ returns $\perp$ on all inputs.

Game Reduction. Let us formally show that this AE security game is indeed the combination of IND$-CPA-AE and EUF-AE notions.

First, we show that breaking IND$-CPA-AE breaks AE or, equivalently:

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{AE}}_{\cdot}$$

Indeed, for any IND$-CPA-AE adversary $\mathcal{A}_1$, there exists an AE adversary $\mathcal{A}_2$ with the same complexity with $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot}(\mathcal{A}_1) \leq \mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}_2)$. $\mathcal{A}_2$ simply runs $\mathcal{A}_1$ and answers its queries with the encryption oracle, that is $\mathbf{AEnc}_k(\cdot)$ or $\$(\cdot)$, and outputs the same conclusion.

Moreover, breaking EUF-AE breaks AE or, equivalently:

$$\mathbf{Adv}^{\mathsf{EUF\text{-}AE}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{AE}}_{\cdot}$$

Indeed, for any EUF-AE adversary $\mathcal{A}_1$ there exists an AE adversary $\mathcal{A}_2$ with the same complexity with $\mathbf{Adv}^{\mathsf{EUF\text{-}AE}}_{\cdot}(\mathcal{A}_1) \leq \mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}_2)$. $\mathcal{A}_2$ runs $\mathcal{A}_1$ answering with its oracles ($\mathbf{AEnc}_k(\cdot)$, $\mathbf{ADec}_k(\cdot,\cdot)$) or ($\$(\cdot)$, $\perp(\cdot,\cdot)$). If the decryption query doesn't output $\perp$, then it is a forgery and $\mathcal{A}_2$ outputs 1. Thus, in the real world, the probability of outputting 1 is the same as the EUF-AE advantage of $\mathcal{A}_1$. If the decryption always output $\perp$, then $\mathcal{A}_2$ outputs 0, and the probability of outputting 1 is 0 in the ideal world. Therefore, the advantage of $\mathcal{A}_1$ and $\mathcal{A}_2$ are the same in their respective game.

Finally, breaking AE either breaks IND$-CPA-AE or EUF-AE or, equivalently:

$$\mathbf{Adv}^{\mathsf{AE}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot} + \mathbf{Adv}^{\mathsf{EUF\text{-}AE}}_{\cdot}$$

For any AE adversary $\mathcal{A}_1$, we build an EUF-AE adversary $\mathcal{A}_2$ and an IND$-CPA-AE adversary $\mathcal{A}_3$. $\mathcal{A}_2$ forwards all $\mathcal{A}_1$ queries, while $\mathcal{A}_3$ forwards $\mathcal{A}_1$ encryption queries but answers $\perp$ at all its decryption queries then mimics decision. Let F be the forgery event, that is "$\mathbf{ADec}_k(\cdot,\cdot)$ does not answer $\perp$ on some query", then:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}_1) =& \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{ADec}_k(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}_1^{\$(\cdot),\perp(\cdot,\cdot)} \to 1)\\
\mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}_1) =& \mathbf{Pr}(F)\mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{ADec}_k(\cdot,\cdot)} \to 1|F)\\
&+ (1-\mathbf{Pr}(F))\mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{ADec}_k(\cdot,\cdot)} \to 1|\neg F)\\
&- \mathbf{Pr}(\mathcal{A}_1^{\$(\cdot),\perp(\cdot,\cdot)} \to 1)\\
\mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}_1) \leq& \mathbf{Adv}^{\mathsf{EUF\text{-}AE}}_{\cdot}(\mathcal{A}_2)\\
&+ \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\perp(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}_1^{\$(\cdot),\perp(\cdot,\cdot)} \to 1)\\
\mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}_1) \leq& \mathbf{Adv}^{\mathsf{EUF\text{-}AE}}_{\cdot}(\mathcal{A}_2) + \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot}(\mathcal{A}_3)
\end{aligned}$$

□

This result seamlessly applies whether we have random IVs, AE-rIV, or nonces, AE-N.

2.3.2 Generic Construction for Authenticated Encryption

Both encryption and authentication schemes are rarely used by themselves as we rather combine them to get secure Authenticated Encryption AE modes. There are multiple ways to combine them:

- Encrypt-then-MAC (EtM), that is $\mathbf{Enc}(m) \,\|\, \mathrm{MAC}(\mathbf{Enc}(m))$;

- Encrypt-and-MAC (E&M), that is $\mathbf{Enc}(m) \,\|\, \mathrm{MAC}(m)$;
- MAC-then-Encrypt (MtE), that is $\mathbf{Enc}(m \,\|\, \mathrm{MAC}(m))$.

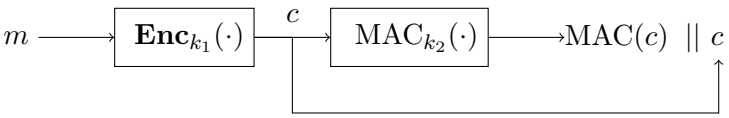


Figure 2.13: Encrypt-then-MAC generic Authenticated Encryption construction.

Encrypt-then-MAC. Bellare and Namprempre [BN00] showed that the only generically proven secure scheme is the Encrypt-then-Mac construction (Figure 2.13) with two independent keys for encryption and authentication each. The other constructions can be used, and are actually used, but they require a careful examination or even a new proof to guarantee their security.

Associated Data. Given a good AE scheme, adding in associated data to make for an AEAD mode is often trivial. In the case of EtM, one can simply compute $c = \mathbf{Enc}_{k_1}(m)$ then authenticate and send the associated data a as $a \,\|\, c \,\|\, \mathrm{MAC}_{k_2}(a \,\|\, c)$.

2.3.3 Concrete Examples

Widely deployed through the TLS protocol, the AEAD modes of operation `GCM` and `CCM` are securing a significant part of the web traffic.

Galois/Counter Mode. The `GCM` mode is a design by McGrew and Viega [MV04]. It is a combination of the Counter mode for encryption and Galois MAC for authentication. It basically follows the Encrypt-then-MAC construction with the important difference that the keys are not independent. The master key k is used to derive the hash key H as $H = E_k(0)$ and to derive the key stream both for encryption and Wegman-Carter-Shoup authentication.

McGrew and Viega [MV04] also proposed a proof of the AE security of `GCM`. However, flaws in their proof was later shown by Iwata, Ohashi

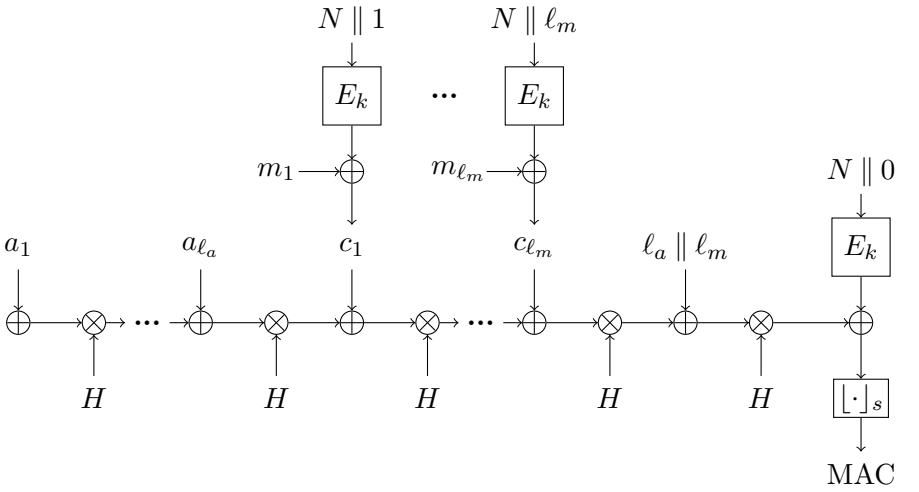


Figure 2.14: The Galois/Counter Mode GCM AEAD scheme for an ℓ_a-block authenticated data $a_1 \| \ldots \| a_{\ell_a}$ and an ℓ_m-block message $m_1 \| \ldots \| m_{\ell_m}$ with a block cipher key k and an hash key $H = E_k(0)$.

and Minematsu [IOM12] who repaired it and gave the following bounds when the nonce is restricted to 96 bits:

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}N}}_{\mathtt{GCM}-E}(t) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(t) + \frac{\frac{1}{2}(\sigma + q_e + 1)^2}{2^n}$$

$$\mathbf{Adv}^{\mathsf{EUF\text{-}N}}_{\mathtt{GCM}-E}(t) \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(t) + \frac{\frac{1}{2}(\sigma + q_e + q_v + 1)^2}{2^n} + \frac{q_v(q_\ell + 1)}{2^s}$$

for adversaries running in time t with σ the number of block cipher calls, q_e and q_d the number of encryption and verification queries respectively and q_ℓ the maximum block size of $a \| c$ over all queries.

In both bounds the first term is dominated by quadratic terms over 2^n, so this confirms the birthday bound security that is security up to $\mathcal{O}(2^{n/2})$ processed blocks.

The term $q_v(q_\ell + 1)/2^s$ shows a security degradation as we shorten the tag length s and allows for longer messages to be processed. This loss is stronger than the expected $q_v/2^s$ from generic tag guessing. Moreover, Ferguson [Fer05a] showed that this term is no artifact: there is a forgery attack using a single encryption of a message of length $2^{s/2+1}$ blocks

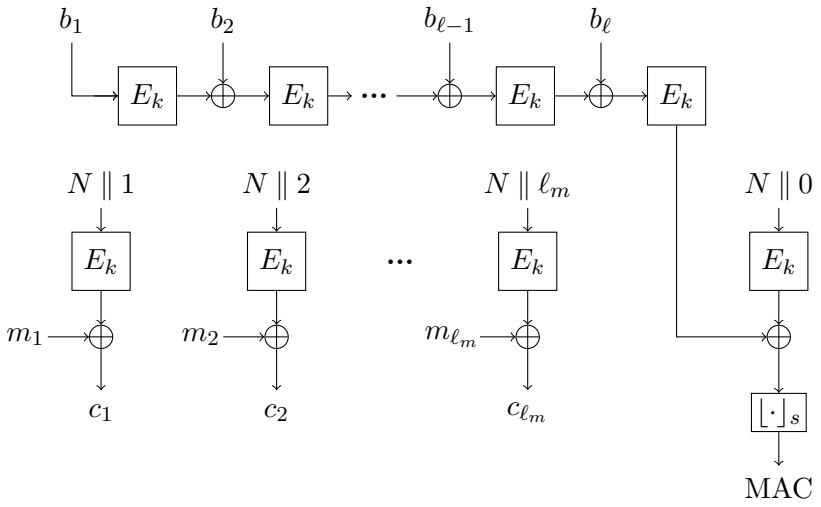


Figure 2.15: The CCM AEAD scheme ℓ_m-block message m, a block cipher key k and $b_0 \| b_1 \| ... \| b_\ell = \mathsf{Fmt}(N, a, m)$.

followed by $2^{s/2}$ expected short verification queries. This can be a problem for short tags, however, for untruncated tags $s = n$, the first birthday-bound term clearly dominates.

CCM. The CCM mode, Counter with CBC-MAC (Figure 2.15), is an AEAD mode by Whiting, Housle, and Ferguson that has also been standardized by the NIST [Dwo04]. It is thought as a combination of the CTR mode with a raw CBC-MAC in an MAC-then-Encrypt fashion. The particularity of the scheme is that it only uses a single key, the same key for encryption and authentication.

As shown in Figure 2.15, CCM seems simple but all its complexity might actually be hidden in the format function $\mathsf{Fmt}(N, a, m)$. The NIST document [Dwo04] imposes some restrictions on this function derived from the security proof of CCM. First, $\mathsf{Fmt}(N, a, m)$ must be prefix-free as per Definition 2.1. This is not surprising considering it's using raw CBC-MAC for authentication. Moreover, the first block of output b_1 must uniquely determine the nonce N and must be distinct from all other input blocks during the CTR encryption. This, again, is not hard to do, but it imposes some restrictions on the nonce and counter size as they must fully enter into the n-bit state while ensuring domain separation. For instance, the

counter function used in practice forces the first 5 bits of the input to be 0 and encode b_1 such that its first 5 bits can't be 0.

Given those properties for the format function, the security of CCM has been proven by Jonsson [Jon03] who showed:

$$\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}N}}_{\mathtt{CCM}-E} \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(t) + \frac{\sigma^2}{2^n} \,, \tag{2.13}$$

$$\mathbf{Adv}^{\mathsf{EUF\text{-}N}}_{\mathtt{CCM}-E} \leq \mathbf{Adv}^{\mathsf{prp}}_{E}(t) + \frac{\sigma^2}{2^n} + \frac{q_v}{2^s} \,, \tag{2.14}$$

with σ the number of block cipher calls, q_v the number of verification queries and t the running time of the attackers. Thus, we can directly deduce the AE security of CCM:

$$\mathbf{Adv}^{\mathsf{AE\text{-}N}}_{\mathtt{CCM}-E} \leq \mathbf{Adv}^{\mathsf{prp}}_{E} + \frac{2\sigma^2}{2^n} + \frac{q_v}{2^s} \,. \tag{2.15}$$

Notice that the value σ depends on the specification of the format function and is at least twice the block lengths of all encrypted plaintext. Hence, CCM benefits from strong security guarantees up to birthday bound and truncating the tag to s bits induce a loss of security as expected by generic tag guessing that succeeds with probability $q_v/2^s$.

With respect to GCM, CCM is probably easier to implement as it reuse the same block cipher without the need for Galois field arithmetics but it surely is less efficient. One of the reason is that using CBC-MAC for authentication is inherently sequential and must be done from scratch for every nonce.

2.3.4 Tweakable Block Cipher and Permutation based Modes

In this section we'll give AEAD modes that aren't directly built on a block cipher. We will show the OCB mode abstraction built on a tweakable block cipher and the sponge duplex built on a permutation.

Permutation and Tweakable Block Cipher. An n-bit permutation is simply a bijection $P : \{0,1\}^n \to \{0,1\}^n$. As it is deterministic and keyless, the security definition of a cryptographically secure permutation is a bit tricky to define. Informally speaking, a secure permutation is expected to behave just like a random permutation. However, there is no

security game that can formally define such a security notion. We call distinguisher any test that makes the permutation behaves differently from a random one although the compact and deterministic description of any practical permutations is, in itself, a distinguisher. In practice, the AES with the key set to 0 can be considered a good, albeit relatively small, permutation.

On the other hand, proofs of permutation based modes are still possible: they replace the public permutation by a random permutation only accessible through an additional oracle. We call this the ideal permutation setting. Concretely, the AE advantage of a scheme based on a public permutation P_0 is defined as:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{AEnc}_k(\cdot),\mathbf{ADec}_k(\cdot,\cdot),P(\cdot),P^{-1}(\cdot)} \to 1) \\ - \mathbf{Pr}(\mathcal{A}^{\$(\cdot),\perp(\cdot,\cdot),P(\cdot),P^{-1}(\cdot)} \to 1)\ ,\end{aligned}$$

where $\mathbf{AEnc}_k$ and $\mathbf{ADec}_k$ use P instead of P_0 with a random key k and a random permutation P. As P_0 does not appear in the game the studied construction is not, strictly speaking, the true construction. Nevertheless, a low AE advantage is still a good indicator that the construction is secure.

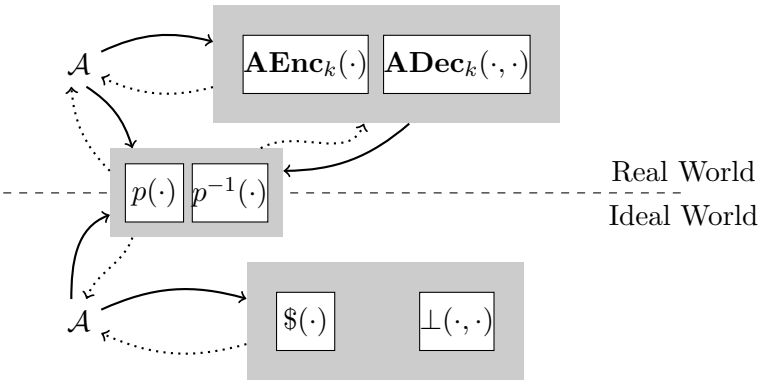


Figure 2.16: Distinguishing game for the AE security of an authenticated encryption mode based on a public permutation where p is a random permutation, k is a random key value, $\$(\cdot)$ a random function and $\perp(\cdot,\cdot)$ returns $\perp$ on all inputs.

We remind that a block cipher is defined to be a family of n-bit permutations indexed by a κ-bit key k that is an application $E : \{0,1\}^\kappa \times$

$\{0,1\}^n \to \{0,1\}^n$. Then, we define a tweakable block cipher as a family of n-bit permutations indexed by both a κ-bit key k, meant to be secret, and a τ-bit tweak t, meant to be public, that is an application $\widetilde{E} : \{0,1\}^\kappa \times \{0,1\}^\tau \times \{0,1\}^n \to \{0,1\}^n$. Ideally, a tweakable block cipher behave like independent PRPs for each different tweak.

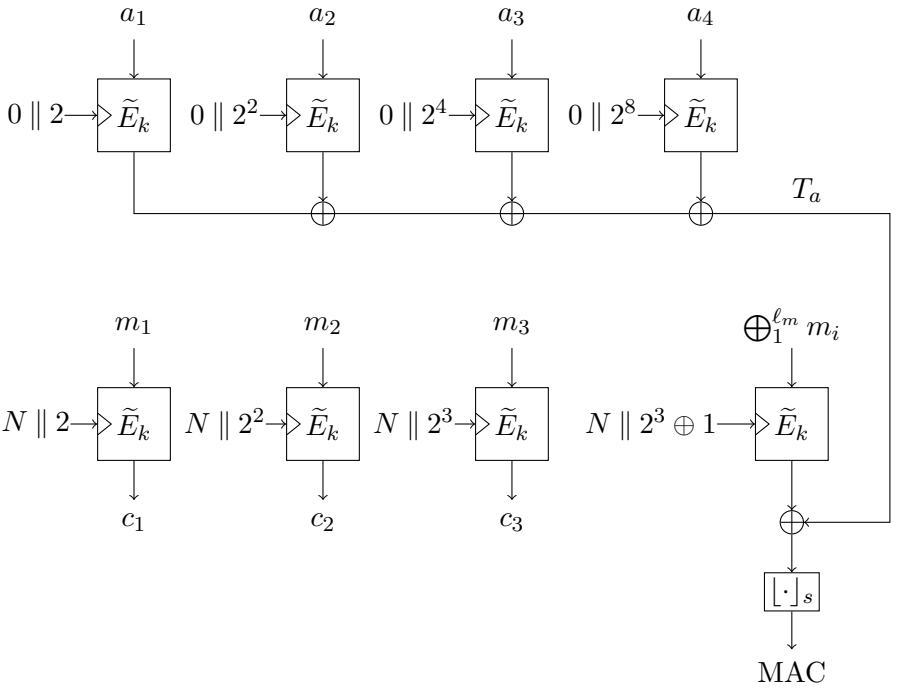


Figure 2.17: The abstracted `OCB` mode with authenticated data a of block length $\ell_a = 4$ and message m of block length $\ell_m = 3$. Formula $\bigoplus_1^{\ell_m} m_i$ represents the checksum of all blocks of message and sequential doubling are done inside a Galois Field of proper size. The ciphertext is computed as $c_i = \widetilde{E}_k^{N \| 2^{i-1}}(m_i)$.

The `OCB` mode. The first version of `OCB` was developed by Rogaway with Bellare, Black and Krovetz [Rog+01] as an AE scheme before going through multiple evolutions `OCB1`, `OCB2` up to the current `OCB3` AEAD mode of operation patented by Krovetz and Rogaway in RFC 7253 [KR14]. Plain `OCB` usually refers to the latest version which is currently `OCB3`.

As it is described by the RFC, the OCB AEAD mode is based on a classical block cipher, but its security actually relies on an abstraction suggested by Liskov, Rivest and Wagner [LRW02]. Indeed, the OCB mode can be thought to be built upon a tweakable block cipher, this is the abstraction we study in this section. Of course the actual OCB mode specifies how to implement a tweakable block cipher based on a secure block cipher. We'll talk about this kind of construction in Chapter 7.

The security of OCB relies on the nonce ensuring that each tweak is used only once except for computing T_a for authenticating associated data, Figure 2.17. One can see the sequential doubling $N \parallel 2^{i-1}$ as some sort of counter that guarantees non repetition. At this level of abstraction the OCB mode is fully secure that is as secure as the underlying tweakable block cipher or generic tag guessing. The security of the actual OCB mode falls down to birthday bound only because the underlying implementation of the tweakable block cipher is secure up to birthday bound. However, one should still be careful when using this kind of hybrid arguments. Indeed, Inoue and Minematsu [IM18] found an easy attack on the second version of OCB, OCB2, using a handful of queries by exploiting the relations between the tweak values used. With the help of Iwata and Poettering [Ino+19] this attack soon developed into a full break of the authenticity and confidentiality of the mode.

Overall the OCB mode is quite efficient. Galois field operations are light: doubling is done with a 1-bit shift followed by a conditional XOR. We say OCB is rate-1 (one block cipher call and one negligible doubling per processed block) while other AEAD schemes are rate-2 (GCM needs one block cipher call and one Galois Field multiplication while CCM needs two block cipher calls per processed block). Moreover, it is possible to precompute the value T_a as, in practice, associated data are often static to encode public information like IP addresses. It also deals nicely with padding by handling the last block differently (this is not shown in Figure 2.17) hence only padding when necessary. On the downside we note the need to implement the inverse of the tweakable block cipher for decryption which may not be ideal especially in hardware.

AEAD from Sponge. The sponge construction has gained popularity as a way to build hash functions with Keccak by Bertoni, Daemen, Peeters and Van Assche who won the SHA-3 NIST competition and thus became known as the SHA-3 standard [Dwo15]. The same authors also showed

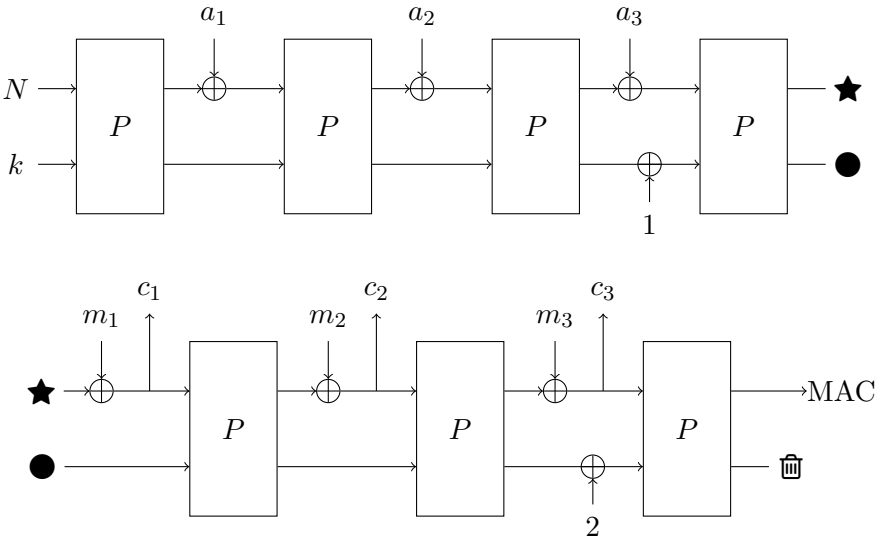


Figure 2.18: Diagram of a variant of `SpongeWrap` for AEAD mode processing blocks of size α bits with an n-bit permutation P and a capacity $\beta = n - \alpha$ bits. The state is initialized by the key k and a nonce N before absorbing three blocks of AD a and three blocks of message m. The output is $c_1 \parallel c_2 \parallel c_3 \parallel \text{MAC}$.

the duplex construction that outputs a random key stream for encryption while absorbing the message for authentication [Ber+12]. They called it `SpongeWrap` of which we show a variant in Figure 2.18 that makes for an AEAD mode of operation based on a public permutation.

The security of the `SpongeWrap` can be directly deduced from the security of the sponge construction for a hash. The n-bit state is separated into two parts: the α-bit rate and the β-bit capacity such that $n = \alpha + \beta$. The rate α influences the efficiency of the scheme, message will be absorbed and encrypted by block of α bits. On the other hand the capacity β influences the security of the scheme.

Indeed, the sponge construction for a hash function enjoys a provable collision-resistance security up to the birthday bound of the capacity. That is up to $2^{\beta/2}$ computations of the permutation P and so does the AEAD mode variant of `SpongeWrap` for the nonce based AE-N security notion. Notice that the MAC can be expanded to any size by iterating the permutation and concatenating many r-bit blocks of the rate part just

like one would expand the hash output of SHA-3.

2.4 On the Security of Modes of Operation

In the previous sections we've introduced some modes of operation with their security notions that provide provable guarantees that we can trust. Here we present some advanced notions and open questions that explore the various security profiles of modes of operation and will motivate the contributions of this thesis.

2.4.1 Quest for Concrete Security

Advantage lower and upper-bounds. Since proofs give an upper-bound on the adversary advantage, we can deduce safe parameters' bounds that guarantee security. Typical parameters include the length and number of messages that we can process before rekeying. However, proofs can get better and modes of operation really benefit from improved proofs as they can run with improved parameters. The only way we can know that there is no way one can get a better proof is when there is a matching cryptanalysis for the considered security game. Hence, the security profile of a mode shall not be complete without a proper distinguisher that gives a lower-bound matching the upper-bound of the proof.

The Sponge's Case. All modes we've seen so far have a matching distinguisher except one: the `SpongeWrap` using the duplex construction for AEAD. There are so far no distinguisher matching the $2^{\beta/2}$ bound, the generic attack on the hash function cannot work in this setting. An improved proof would definitely improve the mode's parameters. For example, having the rate and capacity size set to $n/2$ bits would improve the efficiency of the scheme, but it would set the provable security down to $2^{n/4}$ queries as it stands.

In fact, Chakraborti, Datta, Nandi and Yasuda presented another sponge-based mode `Beetle` [Cha+18] that mainly adds a small feedback function in the rate part, and they are able to improve the proof up to $2^{\min(\beta-\log(\alpha),n/2,\alpha)}$ calls. This would allow for a larger rate α without lowering the provable security guarantees. In the absence of a matching distinguisher, and at the time of this writing, the true security of the original `SpongeWrap` construction is still an open question.

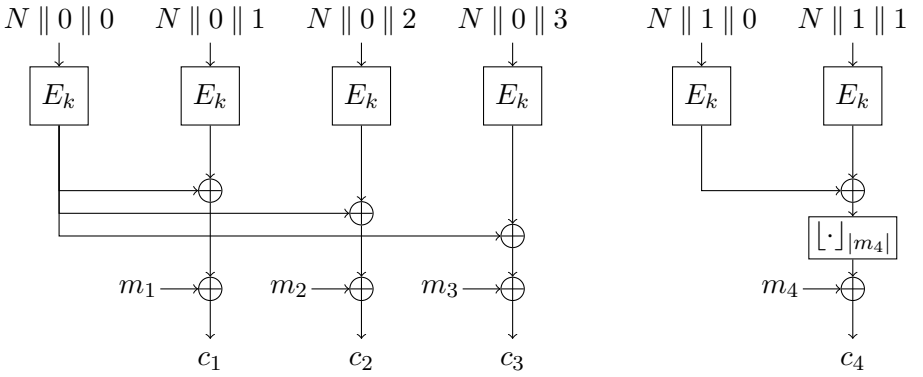


Figure 2.19: The CENC beyond birthday-bound secure mode for encryption for $\omega = 3$. Let $i = k\omega + j$ with $1 \leq j \leq \omega$ then we have $c_i = E_k(N||k \,\|\, 0) \oplus E_k(N \,\|\, k \,\|\, j) \oplus m_i$.

Beyond-Birthday-Bound modes. Going further, many new designs try to achieve beyond-birthday-bound security at a minimal cost. For instance the CENC mode for encryption (Figure 2.19) is a beyond-birthday-bound IND$-CPA-N secure encryption for the cost of $1 + 1/\omega$ cipher calls per block of plaintext with ω the size of the frame. It has first been proposed by Iwata [Iwa06] with a proof showing security up to $\mathcal{O}(2^{2n/3}/\omega)$ block cipher calls. Iwata, Mennink and Vizár [IMV16] later improved the bound up to $\mathcal{O}(2^{n/}/\omega)$ calls which matches an obvious distinguisher: repetitions inside windows of ω blocks cannot happen in CENC but happen with probability $\omega(\omega - 1)/2^n$ in the random case. Thus, after σ encrypted blocks we have σ/ω windows hence a collision probability of $\sigma(\omega - 1)/2^n$ which is $\Omega(1)$ when σ is $\mathcal{O}(2^n\omega)$.

We've also seen that the Wegman-Carter construction can be made beyond birthday-bound secure when used with a good pseudo random function. For instance using the CENC mode to build a seemingly random key stream would build an EUF-N secure beyond birthday bound MAC. There are also recent works trying to build a deterministic EUF secure beyond birthday bound MAC. Such modes that follow the double-block hash-then-sum strategy like SUM-ECBC [Yas10] or PMAC+ [Yas11] came with a proof of security up to $\mathcal{O}(2^{2n/3})$ short tags. In Chapter 5, as a contribution of this thesis and in collaboration with Leurent and Nandi [LNS18], we show a generic attack on those modes using $\mathcal{O}(2^{3n/4})$ tags effectively reducing the cryptanalysis/proof gap. Later Kim, Lee and Lee [KLL20]

actually improved the proof to $\mathcal{O}(2^{3n/4})$ tags, matching our distinguisher and finally closing the gap.

Proof and Cryptanalysis. Thus, proofs and cryptanalysis are needed to concretely assess the theoretical security of a scheme. A proof can benefit from better parameters thanks to a better proof and only cryptanalysis tells us where we have hope to improve the security bound.

Cryptanalysis can also disprove a result. Doing proof is hard as it needs to exhaustively treat everything that could go wrong and bound its probability. It is not rare for a cryptanalysis to refute a proof showing that its upper-bound on the attacker advantage is wrong. It was the case of our distinguisher for double-block hash-then-sum MACs that refuted a now retired proof for `LightMAC+` [Nai18].

2.4.2 Quest for Practical Security

Limits of Information Theoretic Thinking. Security game with corresponding proofs and distinguishers for modes of operation are usually defined in the information theoretic setting. This means that we assume an attacker with unlimited computing power, but limited number of queries, attacking a scheme using a perfect primitive. The reason is that there is no proof techniques that can formally bound a limited adversary. We have to assume a limited adversary to conjecture the security of primitives but, again, it's only conjectures though back up by experience and analysis of generic attacks.

In practice, though, the real world is not information theoretic and no attacker has unlimited computing power. Optimizing the time and memory complexity of attacks is of interest. Moreover, distinguishers can range from recovering a single bit of information, which might not be threatening depending on the context, to a devastating key recovery revealing all previous and future plaintext. Forgery attacks can also range from finding the tag for a single seemingly random plaintext to a full control of the plaintext we want to sign, the latter is called a universal forgery attack.

As we already said, the information theoretic world allows for great proofs to be made and cryptanalysis in this setting is just as important to concretely assess the security of a mode. With this we can come up with parameter bounds where we can fully trust the security of the studied scheme. However, we also need to ask ourselves what happens after we

reach the said bound and think about cryptanalysis is more realistic, practical models.

Practical vs Theoretical gaps. They are quite many schemes that still have a gap between the information theoretic complexity of their attack and the actual time and memory needed to perform them. The Wegman-Carte-Shoup with polynomial AXU hash functions, like `GMAC` or `Poly1305`, is a good example. Using the techniques developed for the missing difference in Chapter 4.1 we could describe a partial key recovery attack on those schemes running in $\mathcal{O}(2^{2n/3})$ time and memory which is the first cryptanalysis with a running time lower than 2^n but still leave a gap with the information theoretic attack of complexity $2^{n/2}$.

Other examples include the double-block hash-then-sum constructions of Chapter 5 where the proposed forgeries, even though some are universal forgeries, still require $\tilde{\mathcal{O}}(2^{3n/2})$ operations. That's the reason why we also give an attack that is not optimal in queries, it uses $\mathcal{O}(2^{6n/7})$ queries, but runs in $\tilde{\mathcal{O}}(2^{6n/7})$ time, less than 2^n, though this is not generic and only applicable to the `SUM-ECBC` and `GCM-SIV2` MAC constructions.

Notice that the exact same phenomenon arises for provable ideal designs. In Chapter 8 we'll look at the 2-round Even-Mansour block cipher construction. This construction has an information theoretic proof holding up to $2^{2n/3}$ queries, but no actual distinguisher runs in less than $2^n/n$ time. The analysis we propose greatly reduces the memory and online query complexity while gaining some insights on why a faster cryptanalysis might be really hard to achieve.

The Beastly Attack Setting. The Beastly attack setting is a framework inspired by the BEAST attack of Duong and Rizzo [DR11] allowing for practical chosen plaintext attacks. The Beastly attack model is an example of a practical chosen plaintext attack: it assumes a malicious webpage that runs some JavaScript code on the target's computer. The malicious code can then send requests to another domain like a social network. The target's computer will naturally encrypt and send those requests with all the personal authentication token required. These are called cross-origin requests and are an intended feature for Cross-origin resource sharing (CORS). Even though the code can choose the content of the requests made, it is impossible to manipulate the answer. Thus, the security should be ensured by using a strong encryption key that is never

manipulated outside the transport layer in the TLS protocol. The model furthermore assume that the adversary can observe queries going in and out of the target's computer. This is a typical assumption in cryptography and can be done in practice by listening to communications in a public Wi-Fi or wiretapping on a router.

This setting has notably been used by Bhargavan and Leurent [BL16] to mount a practical plaintext recovery attack over HTTPS secured web connection using `CBC` mode for encryption along with the 3DES block cipher. Their attack named Sweet32 can recover a web authentication token to successfully log onto an account without knowing its password. It essentially takes advantage of the relatively small state size of 3DES, that is $n = 64$ bits, for a birthday bound attack. In fact, they noticed that many implementations did not put a limit on the number of processed blocks of plaintext before rekeying. Sweet32 led to a change in the NIST standardization of the `CBC` mode so that early rekeying far away from the birthday bound is enforced unambiguously. This example shows that there are practical Chosen Plaintext Attacks with direct consequences.

In Chapter 4 we ask ourselves what could be a practical attack on the `CTR` mode. We've seen in Section 2.1.4 that there is a matching distinguisher on the `CTR` mode that looks for collisions in the key stream blocks. However, looking for collisions won't help to recover block of plaintext for a good reason: key stream collision won't happen with the `CTR` mode. Therefore, we define the missing difference problem upon the resolution of which we can recover some encrypted plaintext, and then we show how to solve it in query, time and memory complexity close to birthday bound that is $\tilde{\mathcal{O}}(2^{n/2})$. For that we place ourselves in the Chosen-Prefix Secret-Suffix attack model by Hoang et al. [Hoa+15] where the attacker can choose a message M and get the encryption $\mathbf{Enc}(M \parallel S)$ for some fixed secret S. Notice that the Chosen-Prefix Secret-Suffix attack model is a simplified model in the Beastly attack setting where a secret authentication token is often appended right after the chosen request content. Hence, as a contribution of this thesis, we show how an attack on the `CTR` mode would work in a practical setting though we also observed that the `CTR` mode, unlike `CBC`, is mostly used along with the AES block cipher whose relatively large internal state, $n = 128$ bits, makes a birthday bound attack too costly to perform.

2.4.3 Quest for Robust Security

A Need for Robustness. Previously in Section 2.1.4 we've argued that nonce based security left less room for implementation errors compared to random IV based security. This is what robust security is about: strengthening our security definition to prevent or at least mitigate misuses of the mode. For example, the GCM mode is secure under the relatively robust nonce based security notion, but it is also true that a single nonce reuse leads to a leak of the hash key thus compromising all future authentication. Informally speaking, a more robust mode of operation is *harder to get it wrong*. Looking at implementation errors is outside the scope of this work, but they are the source of many practical attacks on cryptography.

With regard to provable security, a more robust security notion assume a more powerful attacker and a more robust mode is secure under a more robust notion. In that sense, a deterministic notion is more robust than its nonce-based version: the user has no additional IV or counter to deal with, the only allowed leaks are the equality of the messages. Also, a nonce-based notion is more robust than its random IV based version: an attacker can choose any nonce value for an IV while he has not control on the IV in the random IV version.

Notice though that robustness in practice depends on many more factors including human factors. For instance a blog entry by Matthew Green titled "Why I hate CBC-MAC" [Gre13] provides some high-level insights on the lack of robustness of raw CBC-MAC mode of authentication. Indeed, although raw CBC-MAC should be a fine and provable deterministic secure MAC, the need for prefix-free formatting, the lack of proper standardization and its similarity to CBC encryption with only subtle, but critical, differences makes it seemingly easy to get wrong.

Using Synthetic IV. In order to gain more robustness, Rogaway and Shrimpton [RS06] proposed a construction to combine an IND$-CPA-rIV secure mode for encryption with a prf into a deterministic AE secure AEAD mode of operation. This is the synthetic IV construction or SIV. Figure 2.20 shows the SIV construction for encryption. It is basically a MAC-and-Encrypt construction where the MAC is required to be a secure prf and the tag is also interpreted as an IV for encryption. Thus, the IV is then used both for decryption and authentication in the SIV decryption, Figure 2.21. Let MOD be an IND$-CPA-rIV secure mode for encryption, F a

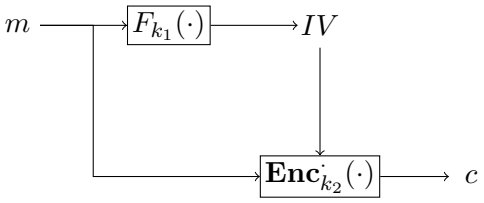


Figure 2.20: The SIV construction parametrized by two independent key k_1 and k_2 for a prf secure function F and an IND$-CPA-rIV secure encryption **Enc** respectively. It takes a message m as input and outputs $IV \parallel c = F_{k_1}(m) \parallel \mathbf{Enc}^{F_{k_1}(m)}_{k_2}(m)$.

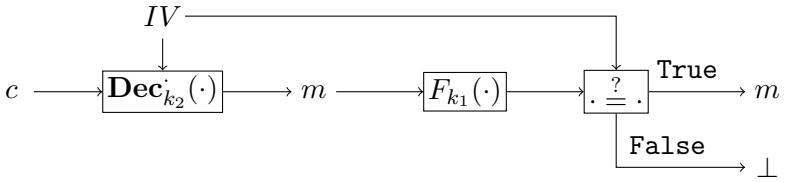


Figure 2.21: The SIV decryption parametrized by two independent key k_1 and k_2. It takes IV and c as inputs and outputs $m = \mathbf{Dec}^{IV}_{k_2}(c)$ if the ciphertext is valid, $\perp$ otherwise.

prf secure keyed function and $\mathtt{SIV}(\mathtt{MOD}, F)$ be the SIV construction using MOD and F. Then, Rogaway and Shrimpton proved that:

$$\mathbf{Adv}^{\mathsf{AE}}_{\mathtt{SIV}(\mathtt{MOD},F)} \leq \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}rIV}}_{\mathtt{MOD}} + \mathbf{Adv}^{\mathsf{prf}}_{F} + q/2^n \tag{2.16}$$

with q the total number of queries and for adversaries with about the same running time t. The original work also included associated data, but it is fairly easy to add to our diagrams in Figures 2.20 and 2.21: simply add the associated data as input to F to produce the IV and encrypt with this.

The main motivation for the SIV construction was to gain robustness against nonce misuse. Indeed, the counter mode is secure with a nonce but also with a random IV as a collision is not expected to happen before the birthday bound. From a deterministic AE secure scheme it is possible to enforce in the specification that it needs a nonce: simply to add it to, for example, the last block of associated data. This way you can ensure the messages will be all different thus randomizing the ciphertext. On the other hand, the leakage in case of nonce repetition is minimal as

any difference, whether in the plaintext or in the associated data, will be enough to prevent any leakage of information.

Following this work robust versions of commonly used AEAD modes were designed. We can cite `GCM-SIV` by Gueron and Lindell [GL15] followed by a different mode `AES-GCM-SIV` by Gueron, Langley and Lindell [GLL17] or again `CCM-SIV` by Kresmer and Zeh [KZ19].

Release of Unverified Plaintext. AEAD modes of operation are expected to decrypt and reveal the plaintext only after it has been properly authenticated. Some modes of operation can be severely compromised if the implementation releases the plaintext before completing the authentication. However, many schemes require that we decrypt beforehand to be able to authenticate. This includes paradigms such as MAC-then-Encrypt and MAC-and-Encrypt (and thus the SIV construction), the duplex and `OCB` constructions. In that context, an implementation has two choices: either keep the decrypted message in a secure memory before it's passed to the application layer or not storing anything but decrypt the message again after it's been verified. Both solutions come with constraints not welcomed in restricted environment where cryptography is costly. A third solution is to start using the message and roll back if the tag isn't verified at the end. This is more convenient but obviously insecure !

For this reason we ask ourselves what kind of security can we achieve under release of unverified plaintext, that is RUP security. To answer this question Andreeva et al.[And+14] proposed the notion of plaintext awareness (PA) security that deals with privacy in this setting and the notion of integrity under release of unverified plaintext (INT-RUP) for authenticity.

We formally describe the PA and INT-RUP security game in Section 6.1.1. Moreover, we also show the INT-RUP insecurity of `SUNDAE`, a deterministic AEAD mode by Banik et al.[Ban+18], by describing a forgery attack in this setting.

Unified notion. In Chapter 6 we'll propose a new notion, AERUP, that captures the RUP security we aim at for AEAD schemes. Basically an AERUP secure scheme is also AE, PA and INT-RUP secure. In Section 2.3.1 we've argued that unified notions like AE security allowed for proving security of a scheme with a single proof instead of one proof per security notion. Thus, we illustrate this by proving that a small change in `SUNDAE`

can make it AERUP secure. We prove the security of a generic scheme, named ANYDAE, in one single proof.

Chapter 3
Algorithms for Generic Attacks

This chapter is dedicated to generic attacks techniques and algorithms. Though this lays in the Modes of Operation part, similar techniques will be used in the second part about Idealized Designs.

3.1 Collisions

When exploring the cryptanalysis of modes of operation, looking for collisions is one of the most, if not the most, frequent approach. Distinguishers of `CBC`, `OFB`, `CFB`, `CTR` and all `CBC-MAC` variants rely on looking for such a collision.

Definition 3.1 generically summarizes the collision problem. Though in cryptanalysis we often use the collision problem with a unique function as in Definition 3.2, it is handled the same way as the generic one.

Definition 3.1 (Collision problem with Two Functions)**.** Given two n-bit functions f_0, f_1, find two inputs (x_0, x_1) such that $f_0(x_0) = f_1(x_1)$.

Definition 3.2 (Collision problem with a Single Function)**.** Given an n-bit function f, find two inputs $(x_0, x_1) : x_0 \neq x_1$ such that $f(x_0) = f(x_1)$.

Definition 3.3 (Collision problem with lists)**.** Given two lists of n-bit values L_0, L_1, find a couple $(e_0, e_1) \in L_0 \times L_1$ such that $e_0 = e_1$.

Moreover, we only discuss the random collision problem where the functions simply output a random value at a fresh input. This is the behavior we expect for cryptographically secure pseudo-random functions. The idea is that the attack can only get better if the underlying PRF has a bias.

3.1.1 Complexity

The Birthday Paradox. In a 30 people classroom, there is more than 70% chance of having two persons with the same birthday. More of a counterintuitive fact than a true paradox, the birthday problem answers this question: How many data do we need to collect before there is a collision? The birthday paradox, or birthday problem, states that we need about $\mathcal{O}(\sqrt{N})$ random values among N possibilities before two of them collide with high probability. More precisely, among N possible values the collision probability grows with the number of data, d, as $1 - \frac{N!}{(N-d)! \cdot N^d}$. This can be approximated to $1 - e^{-\frac{d \cdot (d-1)}{2 \cdot N}}$ for large N and d.

Thus, after collecting $d = 2^{n/2}$ n-bit values ($N = 2^n$), the probability of a collision is about 40%. This is why we say birthday bound secure schemes are secure up to $2^{n/2}$ block cipher calls, but one must actually be careful to stay sufficiently away from that bound. For instance taking $n = 64$ bits, after 2^{30} blocks of data the probability of collision is still around 3%, too high to properly guarantee security (3 out of 100 such encryptions are expected to be insecure). Hence, a birthday bound secure scheme needs to be rekeyed way before birthday bound to make sure collisions won't happen.

Nevertheless, when using a 128-bit block cipher, and thus $n = 128$ bits values, the birthday bound is far enough so that rekeying won't be needed. Indeed, while 2^{32} 64-bit blocks amount to 32 GiB of data, 2^{64} 128-bit blocks makes for 256 exbibytes that is unlikely to be processed under a single key. To give a comparison, the company Cisco estimated that the 2016 global IP traffic summed up to 83.3 exbibytes per month [Sys17].

The main point is that when we talk about birthday bound complexity, the constant hidden in the $\mathcal{O}(2^{n/2})$ notation is often actually smaller than 1 as illustrated by Luykx and Paterson [LP16] who derived concrete security limits from the proofs statements for various AE modes.

Sorting. There are multiple algorithms to find collisions but most of the time, as we gather values in a list, we require some sort of sorting as in Algorithm 3.1. It is well known that we can sort N values in $\mathcal{O}(N \log N)$ time and $\mathcal{O}(N)$ memory. However, in our case, we can take advantage of the randomness of our values to reduce the time complexity to $\mathcal{O}(N)$ using a radix sort or a hash table.

Algorithm 3.1 Collision in Lists

input: $L_0, L_1 \subset \{0,1\}^n$.
output: $(i,j) : L_0[i] = L_1[j]$.
procedure COLLISION(L_0, L_1)
    Sort(L_0) ▷ Sort with any order.
    Sort(L_1)
    $(i,j) \leftarrow (0,0)$
    while $i < |L_0|$ and $j < |L_1|$ **do**
        if $L_0[i] == L_1[j]$ **then**
            return (i,j) ▷ Collision found.
        else if $L_0[i] < L_1[j]$ **then** ▷ Assuming ascending values.
            $i \leftarrow i+1$
        else if $L_0[i] > L_1[j]$ **then**
            $j \leftarrow j+1$
    return $\emptyset$ ▷ No Collision.

Algorithm 3.2 Collision in Functions

input: $f_0, f_1 : \{0,1\}^n \to \{0,1\}^n$.
output: $(x_0, x_1) \in \mathcal{X}_0 \times \mathcal{X}_1 : f_0(x_0) = f_1(x_1)$.
procedure COLLISIONFUN($f_0(\cdot), f_1(\cdot), \mathcal{X}_0, \mathcal{X}_1$)
    $L_0 \leftarrow \{(x, f_0(x)) : x \in \mathcal{X}_0\}$
    $L_1 \leftarrow \{(x, f_1(x)) : x \in \mathcal{X}_1\}$
    ▷ Comparison on the second-hand value only.
    $((x_0, f(x_0), (x_1, f(x_1)) \leftarrow$ COLLISION(L_0, L_1)
    return (x_0, x_1)

Collision with lists. When the values are stored in sorted lists the complexity of looking for a collision is simply the complexity of going through the lists once. The complexity of Algorithm 3.1 is therefore the complexity of sorting, that is $\mathcal{O}(|L_0| + |L_1|)$.

In the case where we need to find multiple collision, say c collisions, then the complexity becomes $\mathcal{O}(|L_0| + |L_1| + c)$. Algorithm 3.1 is describes a birthday bound matching distinguisher that makes $d = \mathcal{O}(2^{n/2})$ queries and thus runs in time and memory $\mathcal{O}(2^{n/2})$.

Memoryless Collision. The collision problem with functions, Definitions 3.1 and 3.2, can be solved the same way by building one or two lists

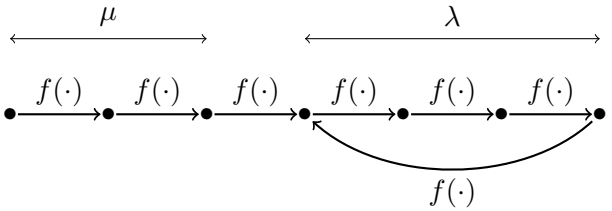


Figure 3.1: Behavior of an iterated random function f with path of length μ and cycle of length λ.

and looking for collision as in Algorithm 3.2. However, when using functions we are free to manage memory as we wish since we can recompute any value whenever we need it.

Therefore, we can look for a collision using only a negligible amount of memory with ideas from Pollard's rho and Floyd's cycle detection algorithms. The idea of such techniques (Algorithm 3.3) is to exploit the iterating behavior of random functions and detect when it enters a cycle, see Figure 3.1. So let μ be the distance travelled before entering a cycle of length λ, we need to find the moment where the function enters the cycle as this is where lies the collision, that is after μ iterations.

The Algorithm 3.3 uses two pointers, one goes a twice the speed of the other such that the distance separating them is always equal to the distance travelled. After μ steps both pointers are in the cycle and after at most λ additional steps the faster pointer necessarily reach the slower one (the faster pointer loops twice while the slow one loops only once). Moreover, when both pointers collide it means they are necessarily separated by a distance that is a multiple of the cycle length λ thus the total travelled distance (total number of steps) is a multiple of λ. After the pointers collide once, one of them is sent back to the starting point, afterwards they both pursue forward one step at a time. Therefore, after μ steps one of the pointer has travelled a distance of exactly μ and the other has travelled a distance of μ plus a multiple of λ, and they will collide at the collision point we were looking for.

The number of steps in Algorithm 3.3 is at most $2\mu+\lambda$ so its complexity depends on the expected values of μ and λ. An asymptotic analysis by Flajolet and Odlyzko [FO90] shows that in expectation μ and λ are $\sqrt{\frac{\pi}{8}}2^{n/2}$. This provides for an interesting trade-off as it still runs in time and query $\mathcal{O}(2^{n/2})$ but can use a negligible amount of memory. Those techniques are

Algorithm 3.3 Memoryless Collision with Pollard's Rho / Floyd's cycle detection.

```
1: input: f : {0,1}^n → {0,1}^n .
2: output: (x, y) : f(x) = f(y), x ≠ y .
3: procedure COLLISION(f)
4:     a ← 0                          ▷ Choose any arbitrary starting point.
5:     x ← f(a)
6:     y ← f(f(a))
7:     while x ≠ y do
8:         x ← f(x)
9:         y ← f(f(y))

10:    x ← a
11:    while f(x) ≠ f(y) do
12:        x ← f(x)
13:        y ← f(y)
14:    return (x, y)
```

extensively used in cryptanalysis. In public key cryptography they can also be used for generically computing discrete logarithm. In symmetric key cryptography one can use them to find collisions in hash functions. There are a few variants of this strategy and notably a parallelizable version by van Oorschot and Wiener [vW99]. When applicable it is probably the best generic attack for finding collisions.

However, this algorithm and its variants don't fit quite well in the context of modes of operation. Indeed, it assumes that we can forget and later query again previous values. Generally this is not something the security games allows, either because the IV is random or because we can't repeat nonces nor repeat messages, exception made for the prf security notion of some deterministic MAC.

Of course, it is possible if we record all queries, but we loose the memory efficient appeal of the technique. This is why Algorithm 3.1 is overall a better description of a birthday bound matching distinguisher for modes of operation.

On Complexity Trade-offs. For the same cryptanalysis, different techniques can provide a variety of trade-offs between the data, time and mem-

ory complexities. There is no objective way to compare those trade-offs. Whether is it a chosen plaintext attack (CPA) or a known plaintext attack (KPA), a low data complexity will make the attack easier to set up as one would need to capture less ciphertext. A low time and memory complexity will make the attack run faster and/or require less hardware.

Even comparing time/memory trade-offs is tricky. One could be tempted to simply add the time and memory complexity, but you could also multiply them arguing that a memory cell could be replaced by a computing cell performing multiple operations. We won't provide a definitive answer but simply state that $2 \cdot 2^{n/2}$ time with small memory is certainly better than $2^{n/2}$ time and memory which is certainly better than 2^n time with small memory. This question has some importance when comparing new attacks with the generic brute-force attack: it is usually easy to recover a κ-bit key with a handful of queries, 2^κ computations and negligible memory.

3.1.2 Cryptanalysis

Plaintext Recovery. We saw in Chapter 2 that many distinguishers were based on looking for a collision. In the case of `CBC`, `CFB` and `OFB` a collision can lead to a stronger attack, namely a plaintext recovery attack. These attacks usually assume that some secret information is encrypted among other known plaintext. The complexity of looking for such collisions is discussed in Section 3.1 along with Algorithm 3.1.

`CBC` mode encrypts data as $c_i = E_k(m_i \oplus c_{i-1})$ so we'll look for a collision in the c_i's:

$$\begin{aligned} c_i &= c_j && ,i \neq j \\ E_k(m_i \oplus c_{i-1}) &= E_k(m_j \oplus c_{j-1}) \\ m_i \oplus c_{i-1} &= m_j \oplus c_{j-1} \\ m_i \oplus m_j &= c_{i-1} \oplus c_{j-1} \end{aligned}$$

and we recover the XOR of two plaintext blocks. For instance when one of the block is known and the other is secret, we recover a full block of secret information.

`CFB` mode encrypts data as $S^i = S^{i-1}_{[0:(n-r)]} \| c_{i-1}$ and $c_i = \lfloor E_k(S^i) \rfloor_r \oplus m_i$. Thus, we look for a collision in the S^i's. Since we know the IV and the

ciphertext it is easy to reconstruct S_i and to detect collisions.

$$\begin{aligned} S^i &= S^j && , i \neq j \\ \lfloor E_k(S^i) \rfloor_r &= \lfloor E_k(S^j) \rfloor_r \\ m_i \oplus c_i &= m_j \oplus c_j \\ m_i \oplus m_j &= c_i \oplus c_j \end{aligned}$$

Again we recover the XOR of two plaintext blocks which is likely to leak secret information.

OFB mode encrypts data as $c_i = E_k(m_{i-i} \oplus c_{i-1}) \oplus m_i$ so we'll look for collisions in $m_i \oplus c_i$'s for known blocks m_i.

$$\begin{aligned} m_i \oplus c_i &= m_j \oplus c_j && , i \neq j \\ E_k(m_i \oplus c_i) &= E_k(m_j \oplus c_j) \\ m_{i+1} \oplus c_{i+1} &= m_{j+1} \oplus c_{j+1} \\ m_{i+1} \oplus m_{j+1} &= c_{i+1} \oplus c_{j+1} \end{aligned}$$

Thus we get the XOR of two plaintext blocks. Notice that in this case we can iterate the reasoning to deduce $m_{i+2} \oplus m_{j+2} = c_{i+2} \oplus c_{j+2}$, etc. This is to be expected from OFB that works like a stream cipher: once the internal state collides the key stream output will be identical going forward.

Collision Free modes. As for the counter mode, looking for a collision works for a distinguisher, but it does not seem to leak any information of the plaintext as, by construction, there cannot be a collision on the key stream. This explains a folklore belief that the CTR mode does not leak anything useful about the plaintext even as we get to the birthday bound. However, we actually show in Chapter 4 a plaintext recovery attack running in data, time and memory complexity close to the birthday bound just like the other attack seen above.

To do that, we define the missing difference problem that is more suited to perform plaintext recovery on the CTR mode and devise efficient algorithms to solve it. The OFB mode processing one very long message also has the same property that collision won't happen, thus it can be attacked the very same way as CTR using the missing difference problem.

Length Extension Forgeries. In Section 2.2.2 we saw multiple variants of CBC-MAC all provably secure up to birthday bound. Here we explain

a matching generic forgery attack on all those variants in $\mathcal{O}(2^{n/2})$ queries that looks for a collision in the internal n-bit state of the `CBC` style chain.

In fact, Preneel and van Oorschot [Pv95] showed that this generic attack is not limited to `CBC-MAC` variants. It is generic to all iterated and deterministic authentication modes with an n-bit internal state.

Definition 3.4 (Collision Extension Property)**.** Given a mode for authentication MAC with internal state Σ. Let $\Sigma(m)$ the internal state after having processed m. We define the collision extension property as:

$$\Sigma(m) = \Sigma(m') \iff \Sigma(m \parallel x) = \Sigma(m' \parallel x)$$

for any messages m, m' and x of arbitrary size.

The idea is that since Σ is the internal state we have $\Sigma(m) = \Sigma(m')$ implies $\mathrm{MAC}(m) = \mathrm{MAC}(m')$. Once we detect an internal state collision in Σ we can use the collision extension property (Definition 3.4) to forge a tag with any suffix message we want: simply ask one to get a forgery for the other.

The collision extension property applies for most of iterated MAC constructions. It assumes the messages are processed left to right (more of a notation convention than anything) and so a collision in the internal state will propagate as we feed in exact same blocks of message. However, the collision extension property is inapplicable in a notable case: prefix-free formatting. Indeed, in that case one cannot simply add more blocks without modifying the previous ones. Schemes like raw `CBC-MAC` rely on that property to guarantee security in the first place and usually does so by prepending a block encoding the length of the message. To get around that limitation we can use what I'd called the collision zero extension property (Definition 3.5) that is implied by the previous collision extension property.

Definition 3.5 (Collision Zero Extension Property)**.** Given a mode for authentication with internal state Σ. Let $\Sigma(m)$ the internal state after having processed m. We define the collision zero extension property as:

$$\Sigma(m \parallel 0^{|x|}) = \Sigma(m' \parallel 0^{|x|}) \iff \Sigma(m \parallel x) = \Sigma(m' \parallel x)$$

for any messages m, m' of arbitrary size and x of fixed arbitrary size.

Thus, for any deterministic iterated MAC, if there is an internal state collision after processing separately m and m', then it will have the collision zero extension property.

To detect that the internal state collide the obvious way is to look for a collision in the tag. However, if two tags collide, it does not necessarily mean that their internal states collided: the processing function can be random and the tag can be truncated. In that case we can again use the collision zero extension property to arbitrarily increase the chance of having an internal state collision. Simply look for collisions in $(\text{MAC}(m \| 0) \| \text{MAC}(m \| 1) \| \text{MAC}(m \| 2) \| ...)$ for different m.

Therefore, this attack is truly generic to all iterated, deterministic n-bit internal state MAC functions, whether it uses a block cipher or not, and in particular apply to all `CBC-MAC` variants we've seen. A natural conclusion is that to achieve beyond-birthday-bound security one must either build a randomized mode, see Wegman-Carter of Section 2.2.3, or increase the internal state size, see double-block hash-then-sum of Chapter 5.

3.2 Generalized Birthday

Wagner [Wag02] proposed a generalization of the birthday problem, see Definition 3.6. The collision or birthday problem can be seen as a 2-XOR problem. As we'll see, the efficient resolution of these problems can lead to new cryptanalysis for different modes of operation.

Definition 3.6 (k-XOR problem)**.** Given k functions $f_0, f_1, ..., f_{k-1}$, find k inputs $(x_0, x_1, ..., x_{k-1})$ such that $f_0(x_0) \oplus f_1(x_1) \oplus ... \oplus f_{k-1}(x_{k-1}) = 0$.

In the following section we consider the random k-XOR problem where all the functions output a random w-bit value on a fresh input.

Data Complexity. A lower-bound on the complexity of the random k-XOR problem is given by the expected number of queries that must be made before a solution exists. Roughly speaking after d queries the number of k-tuples will grow like d^k with each tuple having a 2^{-w} chance of being a solution. Thus, we need at least $d = \mathcal{O}(2^{w/k})$ queries to hope for a solution.

This corresponds to $d = \mathcal{O}(2^{w/2})$ for the collision problem.

Definition 3.7 (k-XOR problem with lists)**.** Given k lists $L_0, L_1, ..., L_{k-1}$, find a k-tuple $(e_0, e_1, ..., e_{k-1}) \in L_0 \times L_1 \times ... \times L_{k-1}$ such that $e_0 \oplus e_1 \oplus ... \oplus e_{k-1} = 0$.

Simple Algorithm. A simple way to solve the k-XOR is to query only one value of every function except 2 of them, and solve the collision problem over those 2 functions. This approach runs in $\mathcal{O}(2^{w/2})$ queries and time. We can solve using Algorithm 3.1 or Pollard's rho techniques.

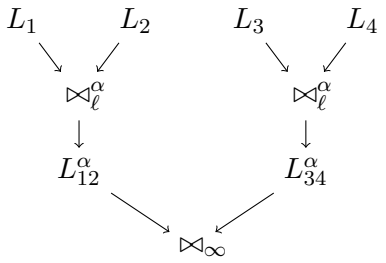
This approach may not be possible when given lists as in Definition 3.7. However, we can still decrease the number of lists by merging them until we get to the 2-XOR problem. To do this we build $L_{01} = \{(e_0 \oplus e_1, (e_0, e_1)) : (e_0, e_1) \in L_0 \times L_1\}$, we'll note $L_{01} = L_0 \bowtie L_1$. We record the couple (e_0, e_1) only to reconstruct the solution at the end. Of course $|L_{01}| = |L_0| \cdot |L_1|$.

3.2.1 Wagner's Algorithm

Generic Algorithm. In his work Wagner [Wag02] proposed a generic algorithm to solve the k-XOR problem. We saw that it is simple to reduce the number of functions or lists to get to the 2-XOR problem either by merging or considering a single query. What Wagner proposed is to reduce the number of lists to the first power of 2. That is reducing the k-XOR to the $2^{\lfloor \log(k) \rfloor}$-XOR. Then, he proposed an algorithm for k that is a power of 2 running in time $\mathcal{O}(k \cdot 2^{w/(1+\log k)})$.

Algorithm 3.4 Wagner's Algorithm for k-XOR

input: $L_i \subset \{0,1\}^w, |L_i| = 2^{w/(1+\log k)}$.
output: $(0, (e_1, e_2, ..., e_k) \in L_1 \times L_2 \times ... \times L_k) : e_1 \oplus e_2 \oplus ... \oplus e_k = 0$.
procedure kXORlist$(L_1, L_2, ..., L_k, n)$
    if $k \stackrel{?}{=} 2$ **then**
        return Collision(L_0, L_1) ▷ Collision on the first element.
    else
        $\ell \leftarrow w/(1 + \log k)$
        for $i = 1$ to $k/2$ **do**
            $L_{i,(i+k/2)} \leftarrow L_i \bowtie_\ell L_{i+k/2}$
        return kXORlist$(L_{1,k/2+1}, L_{2,k/2+2}, ..., L_{k/2,k}, w - \ell)$



for $\alpha \in \{0,1\}^\ell$ **do**
  $L_{12}^\alpha \leftarrow L_1 \bowtie_\ell^\alpha L_2$
  $L_{34}^\alpha \leftarrow L_3 \bowtie_\ell^\alpha L_4$
  $L \leftarrow L_{12}^\alpha \bowtie_\infty L_{34}^\alpha$
  if $L \neq \varnothing$ **then**
    $(0,(x,y),(z,t)) \leftarrow L[0]$
    return (x,y,z,t)
return $\perp$

Figure 3.2: Memory efficient algorithm for 4-XOR with lists of size $2^{w/4}$.

Wagner's Algorithm. Let us explain the idea of Algorithm 3.4. The trick is to merge lists in a way so that they won't grow in size. To do this we have to keep only a portion of the merged values so the idea of Wagner [Wag02] is to only keep values that starts with ℓ zeroes for lists of size 2^ℓ. Concretely we build $L_{01} = \{(e_0 \oplus e_1, (e_0, e_1)) : (e_0, e_1) \in L_0 \times L_1, \lfloor e_0 \rfloor_\ell \oplus \lfloor e_1 \rfloor_\ell = 0\}$, we note this as $L_{01} = L_0 \bowtie_\ell L_1$. Indeed, the expected size of L_{01} is $|L_0| \cdot |L_1|/2^\ell = 2^\ell$. Notice that $L_0 \bowtie_\ell L_1$ can be efficiently computed with Algorithm 3.1 looking for all collisions on ℓ bits.

We need to check that Algorithm 3.4 is consistent when it transforms a k-XOR instance into a $k/2$-XOR one. First, it is obvious to see that it produces $k/2$ lists with elements that are trivially mapped to $\{0,1\}^{w-\ell}$ since they all start by ℓ zeroes. Then, when replacing n by $w-\ell$, the lists of the $k/2$ instance are required to be of size $2^{(w-\ell)/(1+\log(k/2))} = 2^{(w-w/(1+\log k))/\log k} = 2^{w/(1+\log k)}$ which is consistent with the fact that the lists are expected to stay the same size. When $k = 2$, the lists are of size $2^{w/2}$ so there will be a collision with good probability.

Therefore, this algorithm has a time and memory complexity of $\mathcal{O}(k \cdot 2^{n/(1+\log k)})$ or, when we fix k to a constant, simply $\mathcal{O}(2^{w/(1+\log k)})$. This will reduce the complexity below the $2^{w/2}$ birthday bound as k grows large, but the complexity decreases slowly. In particular, for all k the complexity is $\mathcal{O}(k \cdot 2^{w/(1+\lfloor \log k \rfloor)})$ so that we only see a change of complexity every power of 2.

Memory Efficient Algorithm. Wagner's algorithm provides a good time complexity but it has two drawbacks: it is not efficient in terms of queries and it only finds a single solution. A low memory algorithm was first described by Chose, Joux and Mitton [CJM02] and shares some

Algorithm	Data	Time	Memory	Remarks
Wagner [Wag02]	$2^{w/3}$	$2^{w/3}$	$2^{w/3}$	Particular solution
Memory Efficient [CJM02]	$2^{w/4}$	$2^{w/2}$	$2^{w/4}$	Optimal data
Nikolic and Sasaki [NS15]	$2^{3w/8}$	$2^{3w/8}$	$2^{w/4}$	Unique function

Table 3.1: Various asymptotic complexity trade-offs for solving 4-XOR.

ideas with Wagner's algorithm. For $k = 4$ the algorithm essentially repeat Wagner's algorithm for every ℓ-bit value α, and merge on α (and not only on zero) as shown in Figure 3.2. Concretely, the lists are merged as $L_0 \bowtie_\ell^\alpha L_1 = \{(e_0 \oplus e_1, (e_0, e_1)) : (e_0, e_1) \in L_0 \times L_1, \lfloor e_0 \rfloor_\ell \oplus \lfloor e_1 \rfloor_\ell = \alpha\}$.

This algorithm can find all solutions as it keeps running. Therefore, if we're looking for a single solution, we need just the minimal number of queries to ensure such a solution exists in the random 4-XOR which implies $\ell = w/4$. The complexity becomes $\mathcal{O}(2^{w/4})$ queries, $\mathcal{O}(2^{w/4})$ memory and $\mathcal{O}(2^{w/2})$ time.

In the general case where lists are of size ℓ and there are 2^p expected solutions (due to some structure), this algorithm finds a solution in time $\mathcal{O}(2^{2\ell-p})$ with $\mathcal{O}(2^\ell)$ queries and memory.

Definition 3.8 (k-XOR problem with unique function)**.** Given a function f, find k distinct inputs $(x_0, x_1, ..., x_{k-1})$ such that $f(x_0) \oplus f(x_1) \oplus ... \oplus f(x_{k-1}) = 0$.

Exploiting Functions. Nikolic and Sasaki [NS15] showed that interesting trade-offs are achievable for a version of the 4-XOR with a unique function and distinct inputs as in Definition 3.8. They make a smart use of Hellman's tables to find a 4-XOR solution in $\mathcal{O}(2^t)$ time and queries for $\mathcal{O}(2^{w-2t})$ memory for $n/3 < t < n/2$.

A particular point in this trade-off is when memory is $\mathcal{O}(2^{w/4})$, as much as the memory efficient algorithm, then the time complexity is $\mathcal{O}(2^{3w/8})$ that is lower than the birthday bound but with also $\mathcal{O}(2^{3w/8})$ queries.

In Table 3.1 we show how the different approaches for 4-XOR compare.

3.2.2 A Hard Case: the 3-XOR Problem

The complexity of Wagner's algorithm for 3-XOR is the same as for collision, that is birthday bound. Wagner [Wag02] left a better 3-XOR algorithm as an open question.

Since then, the best algorithms have had a hard time going below the birthday bound. The best ones achieve $\mathcal{O}(2^{w/2}/\sqrt{w})$ time and memory. There are two main techniques to achieve this, one based on multi-collision and the other on linear algebra. We present each of these techniques and discuss the combination of them.

Definition 3.9 (3-XOR problem with lists)**.** Given three lists $L_0, L_1, L_2 \subset \{0,1\}^w$, find three elements $(e_0, e_1, e_2) \in L_0 \times L_1 \times L_2$ such that $e_0 \oplus e_1 \oplus e_2 = 0$.

Multi-collisions. As a first way of solving the 3-XOR problem faster than Wagner's algorithm Nikolic and Sasaki [NS15] proposed a multi-collision based approach. Their algorithm works in time and memory $\mathcal{O}(2^{w/2}/\sqrt{w/\ln(w)})$.

The algorithm is as follows. First, compute many outputs of f_0 and look for the most frequent $w/2$-bit prefix α appearing using any multi-collision search algorithm. Store all the values with this fixed prefix in a list L_0. Then, evaluate f_1 and f_2 with $2^{w/2}/\sqrt{|L_0|}$ different inputs each, and store the results in lists L_1 and L_2 respectively. Look in L_1 and L_2 for all pairs with a difference α in the first $w/2$ bits (this is a simple collision problem). In expectation, there will be $2^{w/2}/|L_0|$ such pairs, and there is a high probability that one of them sums to a value in L_0. According to their analysis, the optimal attack uses around $2^{w/2}/w$ evaluations of f_0, resulting in a multi-collision of size $\Theta(w/\ln(w))$. Therefore, this algorithm solves the 3-XOR problem with complexity $\mathcal{O}(2^{w/2}/\sqrt{w/\ln(w)})$.

Linear Algebra. A second approach, introduced by Joux [Jou09], exploits linear algebra and reaches a slightly better complexity of $\mathcal{O}(2^{w/2}/\sqrt{w})$. This attack uses just $w/2$ evaluations of f_0 stored in a list L_0, and $2^{w/2}/\sqrt{w/2}$ evaluations of f_1 and f_2 to build the lists L_1 and L_2. The goal is again to get values with a common prefix, but the trick is to use Gaussian reduction to find a non-singular matrix M such that the elements of $L_0 \cdot M$ start with $w/2$ zeroes.[1] Then, we focus on a modified 3-XOR instance:

$$L_0' = L_0 \cdot M \qquad L_1' = L_1 \cdot M \qquad L_2' = L_2 \cdot M.$$

[1]For instance, we write L_0 as a block matrix $\begin{bmatrix} A & B \end{bmatrix}$ with two $w/2 \times w/2$ sub-matrices. If B is non-singular, we can use $M = \begin{bmatrix} I & 0 \\ B^{-1}A & B^{-1} \end{bmatrix}$

The new instance has the same solutions ($L'_0[h] \oplus L'_1[i] \oplus L'_2[j] = 0 \Leftrightarrow L_0[h] \oplus L_1[i] \oplus L_2[j] = 0$), but the elements of L'_0 start with $w/2$ zeroes. Therefore, as in the previous attack, we can efficiently find a solution by looking for a partial collision on $w/2$ bits in L'_1 and L'_2.

Bouillaguet, Delaplace and Fouque [BDF18] later generalized this approach to deal with more balanced lists size: given three lists with $|L_0| \cdot |L_1| \cdot |L_2| = 2^w$, they solve the 3-XOR problem with complexity $\mathcal{O}(|L_0| \cdot (|L_1| + |L_2|)/w)$. In particular, with three lists of size $2^{w/3}$ this gives a time complexity of $\mathcal{O}(2^{2w/3}/w)$.

In addition, this algorithm can be combined with the clamping trick of Bernstein to reduce the memory: the attacker first filters the outputs of the function f_i to keep only values that start with $w/4$ zero bits, and solves a shorter 3-XOR instance on $3w/4$ bits. If we filter and store $2^{w/2}$ random outputs, the resulting lists still have $2^{w/4}$ elements which is sufficient to expect a solution. This gives an algorithm with time $\mathcal{O}(2^{w/2})$ and memory only $\mathcal{O}(2^{w/3})$. Arguably, this is more practical than algorithms using $\mathcal{O}(2^{w/2}/w)$ memory.

BDP Algorithm. Even before these two approaches, an algorithm proposed by Baran, Demaine and Pătraşcu [BDP08] for the 3-SUM problem (using modular additions instead of XORs) worked with the asymptotic complexity of $\mathcal{O}(2^{w/2} \cdot \ln^2(w)/w^2)$. This algorithm has been adapted to the 3-XOR problem by Bouillaguet et al. [BDF18] with the same complexity. This is the best known asymptotic complexity for the 3-XOR problem.

Generally, the BDP algorithm has an asymptotic speed-up of $\frac{w^2}{\ln^2(w)}$ compared to Wagner's algorithm which makes for an asymptotic time complexity of $\mathcal{O}(|L_0| \cdot (|L_1| + |L_2|) \cdot \ln^2(w)/w^2)$. However, as it relies on heavy precomputations, the algorithm is highly impracticable for all realistic values of w.

Combining Approaches. Since the multi-collision and the linear algebra approaches seem to follow a similar strategy it is natural to wonder how one can combine them into a better algorithm. Bouillaguet et al. [BDF18] write that *It seems very hard to combine these two improvements in a new algorithm.* We concurred with this analysis in [LS19].

However, the formulation may be a bit misleading: it is not hard to combine the two approaches but it may simply be useless. Indeed, both approach can be generalized by the following one:

1. Find a $w/2$-dimension affine subspace that contains $|L_0|$ outputs of f_0 and store them in L_0.

2. Consider now the modified 3-XOR instance with modified coordinates: the first coordinates define the found $w/2$-dimension linear subspace and the rest simply span to the whole w dimensions. Notice that all elements of L_0 now end by $w/2$ bits set to 0 as they belong to the linear subspace.

3. Take f_1 and f_2 to look for a partial collision on $w/2$ bits and verify if it yields a solution.

Step 1 is the crucial step here. Then, the complexity will be led by Step 3 that is $\mathcal{O}(2^{w/2}/\sqrt{|L_0|})$. The multi-collision approach solves Step 1 by fixing a priori a linear subspace, and looking for a constant that corrects the $w/2$-bit multi-collision to zero. On the other hand the linear algebra approach takes the first $w/2$ elements and directly deduce the corresponding $w/2$ dimension linear subspace (or $w/2+1$ elements in an affine subspace).

The question now is how big can be $|L_0|$. First, we assume we make $2^{w/2}$ queries as doing more queries would increase the total complexity. That is why the $w/2$-dimension affine subspace containing the most elements is unlikely to contain more than w elements. The intuition is that the number of element in a fixed $w/2$-dimension affine subspace is only 1 in expectation (each element is included with probability $2^{-w/2}$) with a variance of $1-1/2^{-w/2}$. The multi-collision technique looks at $2^{w/2}$ different subspaces and the maximum expected number of elements is $\Theta(w/\ln(w))$. On the other hand, using linear algebra allows us to easily find a subspace containing $w/2$ elements. There probably are some heuristic approaches looking at about $2^{w/2}$ different subspaces spanned by $w/2$ elements, but the maximum expected number of elements found is $w/2+\Theta(w/\ln(w))$, it only adds up. Notice that all interesting subspaces can be spanned by $w/2$ elements.

The point is that, even if we look at all spanned subspaces, the maximum expected L_0 is of order $\mathcal{O}(w)$. Therefore, the total complexity of the combined strategy is bound to be $\mathcal{O}(2^{w/2}/\sqrt{w})$.

Chapter 4
The Missing Difference Problem

Contributions brought forward in this chapter were published in Eurocrypt 2018 and are a joint work of Leurent and I [LS18].

Introduction

A Folklore Belief. While the information theoretic security of the counter mode (CTR) is well understood, one can wonder about its practical security. As we saw in Section 2.1, the CTR allows for a distinguisher at the birthday bound as the blocks of key stream never repeat. However, the lack of collisions does not seem to leak any useful information when actually using the CTR mode to encrypt secret data. It may therefore be believed that the CTR mode doesn't really leak any information even at birthday bound.

In fact, we can find in the literature the folklore belief that the leakage of the CTR mode is not as bad as the leakage of the CBC mode. For instance, we can quote Ferguson, Schneier and Kohno [FSK11, Section 4.8.2] in the context of a 128-bit block cipher who wrote:

> CTR leaks very little data. [...] It would be reasonable to limit the cipher mode to 2^{60} blocks, which allows you to encrypt 2^{64} bytes but restricts the leakage to a small fraction of a bit.
>
> When using CBC mode, you should be a bit more restrictive. [...] We suggest limiting CBC encryption to 2^{32} blocks or so.

Our contributions. As a first contribution of this thesis, we devise a plaintext recovery attack on the CTR mode beyond the simple distinguisher. We show a plaintext recovery attack in the Known-Suffix / Secret-Prefix model where a secret piece of information is repeatedly encrypted after

some known plaintext block. This situation is common with web cookies and is in line with other attacks against HTTPS [DR11; AlF+13; BL16]. Our cryptanalysis further assumes we can control the position of the secret data and this results in an efficient message recovery running in complexity $\widetilde{\mathcal{O}}(2^{n/2})$ in data, time and memory.

To do that we first follow the strategy of McGrew [McG12] and define the missing difference problem (Definition 4.1) upon the resolution of which plaintext recovery on `CTR` is possible. To efficiently solve this problem we give two new algorithms:

1. An algorithm with $\widetilde{\mathcal{O}}(2^{n/2})$ data and memory complexities and $\widetilde{\mathcal{O}}(2^{n/2} + 2^{\dim\langle\mathcal{S}\rangle})$, time complexity where we look for a secret value in $\mathcal{S}$ that is (a subset of) a linear subspace of $\{0,1\}^n$. In particular, when $\mathcal{S}$ is a linear subspace of dimension $n/2$, we reach a time and query complexity of $\widetilde{\mathcal{O}}(2^{n/2})$, while the searching algorithm of McGrew has a time and query complexity of $\widetilde{\mathcal{O}}(2^{3n/4})$.

2. An algorithm with data, time and memory complexities $\widetilde{\mathcal{O}}(2^{2n/3})$ for any arbitrary $\mathcal{S}$. In particular, for $\mathcal{S} = \{0,1\}^n$, the best previous algorithm had a time complexity of $\widetilde{\mathcal{O}}(2^n)$.

While the first algorithm we proposed is fit to attack the `CTR` mode, the second algorithm allows us to do a partial key recovery attack of some Wegman-Carter-Shoup MAC schemes using polynomial hash like `GMAC` and `Poly1305` running in time, query and memory $\widetilde{\mathcal{O}}(2^{2n/3})$. This is the first partial key recovery leading to a universal forgery on those schemes running in time less than $\mathcal{O}(2^n)$.

4.1 The Algorithmic Challenge

4.1.1 From `CTR` to Missing Difference

The Counter Mode. The `CTR` mode for encryption (Figure 4.1) was proposed as early as 1979 by Diffie and Hellman [DH79]. Nowadays it is largely used notably in TLS via the popular AEAD mode `GCM`. We introduced the `CTR` mode and its matching distinguisher in Section 2.1.4.

In this Chapter, and to simplify the notations, we denote the i^{th} block of key stream produced by a fresh nonce and the key k as a_i. Thus, we have the relation $c_i = a_i \oplus m_i$. Notice that it is easy to recover the value a_i when knowing the plaintext m_i and the ciphertext c_i.

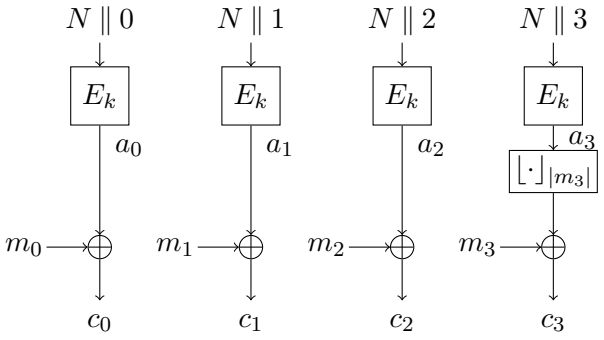


Figure 4.1: Diagram of the counter mode (`CTR`) where $a_i = E_k(N \| i)$ and $c_i = a_i \oplus m_i$.

Plaintext Recovery Strategy. So we know the value a_i for many i. Furthermore, we assume that a secret value S is repeatedly encrypted as $b_j = a_j \oplus S$ for many j. The distinguisher for `CTR` exploits the fact that the key stream never repeats which means that:

$$\begin{aligned} \forall i \neq j : a_i \neq a_j \\ \Rightarrow a_i \oplus a_j \oplus S \neq S \\ \Rightarrow a_i \oplus b_j \neq S. \end{aligned}$$

Therefore, the strategy is to collect enough a_i and b_j, store them in lists and use the inequality $a_i \oplus b_j \neq S$ to recover S.

The Missing Difference Problem. Formally let us denote as $\mathcal{A} \subseteq \{0,1\}^n$ the set of observed key stream blocks a_i, $\mathcal{B} \subseteq \{0,1\}^n$ the set of observed encryptions b_j and $\mathcal{S} \subseteq \{0,1\}^n$ the set of possible values of the secret which models the a priori known information about S. The missing difference problem is then defined as:

Definition 4.1 (Missing Difference Problem)**.** Given two sets $\mathcal{A}$ and $\mathcal{B}$, and a hint $\mathcal{S}$, find the value $S \in \mathcal{S}$ such that:

$$\forall (a,b) \in \mathcal{A} \times \mathcal{B},\ S \neq a \oplus b\ .$$

Alternatively the attacker can be given access to the sets $\mathcal{A}$ and $\mathcal{B}$ though some functions f and g as in Definition 4.2. This variant allows the attacker to actively optimize the size of the sets in the same way as we defined the problem of collision with function in Section 3.1.

Definition 4.2 (Missing Difference Problem with Functions)**.** Given two functions $f, g : X \to \{0,1\}^n$, and a hint $\mathcal{S}$, find the value $S \in \mathcal{S}$ such that:

$$\forall (x, y),\ S \neq f(x) \oplus g(y)\ .$$

It is clear that solving this problem is sufficient to recover some secret data S which makes for a plaintext recovery attack. The Known-Suffix / Secret-Prefix model is adapted to collect values and build the set $\mathcal{A}$ and $\mathcal{B}$ but before we show the details of the attack we need efficient way of solving the missing difference problem.

4.1.2 Previous Works

Data Complexity Lower Bound. To recover the secret S, it must be completely determined by the fact that $\forall (a, b) \in \mathcal{A} \times \mathcal{B},\ S \neq a \oplus b$. We can use the coupon collector's problem to estimate the number of such value we need. Concretely, the coupon collector's problem predicts that N out of N different coupons are found after $N \cdot H_N \simeq N \ln N$ uniformly random trials where H_N is the N^{th} harmonic number.

In our case there are $|\mathcal{S}| - 1$ values to eliminate, therefore we need to collect at least $\mathcal{O}(|\mathcal{S}| \ln |\mathcal{S}|)$ differences that fall into the set $\mathcal{S}$. Since we can reasonably assume that the differences $a \oplus b$ are uniformly distributed over $\{0,1\}^n \backslash S$ (we have to exclude S), they will fall in the set $\mathcal{S}$ with probability $(|\mathcal{S}| - 1)/(2^n - 1)$. Thus, the number of different $a \oplus b$ values $|\mathcal{A} \times \mathcal{B}|$ has to be in the order of $\Omega(2^n \ln |\mathcal{S}|)$.

Hence, one of the sets must be of size at least $\Omega(2^{n/2}\sqrt{\ln |\mathcal{S}|})$ which is a lower bound on the data and time complexity for this strategy. Notice that this is quite efficient as the IND$-CPA proof of the CTR mode implies that the data complexity must be at least $\Omega(2^{n/2})$ before a single bit of information can be leaked.

Simple Sieving. As explained by McGrew [McG12], an obvious way to solve the missing difference problem is to compute and record all values $a \oplus b$ in a sieve until only one value is left. This value will thus have to be the secret S. We call it the simple sieving algorithm, Algorithm 4.1. It is efficient in terms of data with a complexity $\mathcal{O}(2^{n/2}\sqrt{\ln |\mathcal{S}|})$ but takes time $\mathcal{O}(2^n \ln |\mathcal{S}|)$ and memory $|\mathcal{S}|$ to store the sieve.

Algorithm 4.1 Simple sieving algorithm

```
1: input: A, B, S ⊆ {0,1}^n .
2: output: {s ∈ S | ∀(a,b) ∈ A × B, a ⊕ b ≠ s} .
3: procedure SimpleSieve(A, B, S)
4:     for a in A do
5:         for b in B do
6:             Remove (a ⊕ b) from S;
7:     return S
```

Searching Algorithm. Moreover, McGrew [McG12] noticed that the simple sieving is actually wasting a lot of computation by computing values that fall outside the search space $\mathcal{S}$, especially when $\mathcal{S}$ is small.

To solve this issue he proposed an algorithm that tests and eliminates values of $\mathcal{S}$ by directly looping over its values. This is the searching algorithm, Algorithm 4.2. The searching algorithm builds the set $\mathcal{B}$ for quick membership testing (like a hash table) and then loops over $\mathcal{S}$ and $\mathcal{A}$ values. In fact, for all $s \in \mathcal{S}$ it tests whether $\exists a \in \mathcal{A}$ such that $s \oplus a \in \mathcal{B}$. If it exists, then s is removed from the search space $\mathcal{S}$, if it doesn't exist then s is a solution. Therefore, its time complexity is $\mathcal{O}(|\mathcal{B}| + |\mathcal{A}| \cdot |\mathcal{S}|)$.

In the case where the attacker freely chooses the size of the sets, like in Definition 4.2, then the time complexity is optimized when $|\mathcal{B}| = |\mathcal{A}| \cdot |\mathcal{S}|$. In addition, we know that $|\mathcal{B}| \cdot |\mathcal{A}| = \Omega(2^n \ln |\mathcal{S}|)$ for the algorithm to be successful. Therefore, the total optimized time complexity is $\mathcal{O}(2^{n/2}\sqrt{|\mathcal{S}| \ln(|\mathcal{S}|)})$ which is also the data complexity.

This algorithm is designed to be efficient when $\mathcal{S}$ is small. Indeed, it is almost optimal for small $|\mathcal{S}|$ since its time and data complexities tend to the theoretical lower bound $\widetilde{\mathcal{O}}(2^{n/2})$.

4.2 The Known-Prefix Sieving

The Simple Sieving Algorithm 4.1 has a huge time complexity and the Searching Algorithm 4.2 is only efficient for very small $\mathcal{S}$.

We present a first algorithm, the Known-Prefix Sieving, that remains optimal for relatively large searching space $\mathcal{S}$ given that it is included in an $n/2$ dimension affine space. The algorithm can easily be iterated over multiple affine subspaces to search for larger $\mathcal{S}$ increasing only the time but remaining optimal in data complexity.

Algorithm 4.2 Searching algorithm

```
1: input: A, B, S ⊆ {0,1}^n .
2: output: {s ∈ S | ∀(a,b) ∈ A × B, a ⊕ b ≠ s} .
3: procedure SEARCHING(A, B, S)
4:     Store B so that operation ∈ is efficient.
5:     for s in S do
6:         for a in A do
7:             if (s ⊕ a) ∈ B then
8:                 Remove s from S;
9:     return S
```

4.2.1 The Algorithm

The Affine Searching Space. To apply the Known Prefix Sieving Algorithm the search space $\mathcal{S}$ needs to be in an affine subspace of dimension $n/2$ or less. Let's assume $\mathcal{S}$ is an $n-z$ dimension affine subspace for some $n/2 \leq z < n$. Let ϕ be a bijective affine function mapping $\mathcal{S}$ unto $\{0\}^z \times \{0,1\}^{n-z}$ then for all n-bit values a, b:

$$\begin{aligned} S \neq a \oplus b &\Leftrightarrow \phi(S) \neq \phi(a \oplus b), && \text{as } \phi \text{ is a bijection.} \\ &\Leftrightarrow \phi(S) \neq \phi(a) \oplus \phi(b) \oplus \phi(0), && \text{as } \phi \text{ is affine} \end{aligned}$$

The missing difference problem is then rewritten with the transformed sets $\mathcal{A}', \mathcal{B}', \mathcal{S}'$ as follows:

$$\begin{aligned} \mathcal{S}' &:= \{0\}^z \times \{0,1\}^{n-z} \\ \mathcal{A}' &:= \{\phi(a) \mid a \in \mathcal{A}\} \\ \mathcal{B}' &:= \{\phi(b) \oplus \phi(0) \mid b \in \mathcal{B}\} \end{aligned}$$

which corresponds to the case where the secret S is known to start with z zeroes.

The Known Prefix Sieving described in Algorithm 4.3 assumes that the secret S starts with z zeroes but it is indeed applicable to all affine search spaces after the above transformation.

Description. The known prefix sieving (Algorithm 4.3) solves the issue of useless computation (that is $a \oplus b$ falling outside $\mathcal{S}$) by first looking a partial collision on z bits. Thus, the algorithm looks for all z-bit partial

Algorithm 4.3 Known prefix sieving algorithm

input: $\mathcal{A}, \mathcal{B} \subset \{0,1\}^n$, $z < n$, $\mathcal{S} \subseteq \{0\}^z \times \{0,1\}^{n-z}$.
output: $\{s \in \mathcal{S} \mid \forall (a,b) \in \mathcal{A} \times \mathcal{B},\ a \oplus b \neq s\}$.
procedure PrefixSieve($\mathcal{A}, \mathcal{B}$, $\mathcal{S}$)
    $h_B \leftarrow$ Empty hash table.
    for b **in** $\mathcal{B}$ **do**
        $h_B[b_{[0\dots(z-1)]}] \xleftarrow{\cup} \{b_{[z\dots(n-1)]}\}$

    for a **in** $\mathcal{A}$ **do**
        $v_a \leftarrow a_{[z\dots(n-1)]}$
        for v_b **in** $h_B[a_{[0\dots(z-1)]}]$ **do**
            Remove $\overline{0} \,\|\, (v_a \oplus v_b)$ from $\mathcal{S}$;
    return $\mathcal{S}$

collision between $\mathcal{A}$ and $\mathcal{B}$ and then removes the resulting XOR from the searching space. To do that it uses of a hash table indexed by the z first bits for quick access to all values that partially collide with a given element.

Notice that this gives another way of looking for collisions. Sorting the two lists and simultaneously going through them, like we saw with Algorithm 3.1, to find all partial collisions would also work with the same complexity.

4.2.2 Complexity Analysis

Data, Time and Memory. If there is a solution, then Algorithm 4.3 will find it.

The time complexity is the size of the lists, that is the data complexity, plus the number of collisions found. For a search space of size 2^{n-z}, we need about $\ln(2^{n-z}) \cdot 2^{n-z} = \ln 2 \cdot (n-z) \cdot 2^{n-z}$ collisions following the coupon collector analysis. Hence, the time complexity is $\mathcal{O}(|\mathcal{A}| + |\mathcal{B}| + \ln 2 \cdot (n-z) \cdot 2^{n-z})$.

Finally, the memory stores the lists of n-bit values and keeps track of the sieve so its complexity is $\mathcal{O}(n \cdot (|\mathcal{A}| + |\mathcal{B}|) + 2^{n-z})$.

When the sets $\mathcal{A}$ and $\mathcal{B}$ are balanced, the data complexity will be

optimal and the overall complexities for $\mathcal{S} = \{0\}^z \times \{0,1\}^{n-z}$ are:

$$\begin{aligned}
&\mathcal{O}\left(\sqrt{n-z}\cdot 2^{n/2}\right) && \text{queries}\\
&\mathcal{O}\left(2^{n-z} + n\sqrt{n-z}\cdot 2^{n/2}\right) && \text{bits of memory (sieve \& lists)}\\
&\mathcal{O}\left((n-z)\cdot 2^{n-z} + \sqrt{n-z}\cdot 2^{n/2}\right) && \text{operations (collisions \& lists)}
\end{aligned}$$

as $n/2 \leq z$ all those complexities are $\widetilde{\mathcal{O}}(2^{n/2})$.

Iterating for larger search space. Notice that the complexity analysis given above also holds for larger $\mathcal{S}$, in particular for $z < n/2$. However, this would lead to unnecessary use of memory. When $\mathcal{S}$ is larger and/or doesn't fit into an $n/2$ dimensional subspace the idea is to split $\mathcal{S}$ into multiple subspaces of smaller dimensions and look trough them with the known prefix sieving Algorithm 4.3.

By doing so, the data complexity will remain optimum and the time complexity will be multiplied by the number of iterations. Concretely, for a large search space $|S| = 2^s$ splittable into $2^{s-n/2}$ linear subspaces of dimension $n/2$, the data, time and memory complexities are $\mathcal{O}(\sqrt{s}2^{n/2})$, $\mathcal{O}(n/2 \cdot 2^s)$ and $\mathcal{O}(n\sqrt{s}2^{n/2})$ respectively.

Comparisons. The known prefix algorithm thus works very well with $\mathcal{S}$ in an $n/2$ dimensions subspace. With $z = n/2$ the complexities are $\widetilde{\mathcal{O}}(2^{n/2})$ while the searching Algorithm 4.2 requires $\widetilde{\mathcal{O}}(2^{3n/4})$ computations. For small $\mathcal{S}$, as small as $|S| = 2$, then the two algorithms perform similarly.

For large $\mathcal{S}$ though we mainly observe a gain in memory. In particular, when $\mathcal{S} = \{0,1\}^n$ (when we have no prior information on the secret), the known prefix sieving requires $\widetilde{\mathcal{O}}(2^n)$ time but $\widetilde{\mathcal{O}}(2^{n/2})$ data and memory while the simple sieving Algorithm 4.1 requires $\mathcal{O}(2^n)$ memory for the sieve.

Lower Bound on Success Probability. In addition to this analysis in expectation we now derive a lower bound to the probability of success of the sieving when $z = n/2$, depending on the query complexity. So let the number of couples (a, b) be $|\mathcal{A}| \cdot |\mathcal{B}| =: \alpha 2^n$ for some α. In expectation, we should get $\alpha 2^n / 2^{n/2} = \alpha 2^{n/2}$ partial collisions. More precisely, the Chernoff bound gives us a lower bound for the probability of finding at

least $(1-\delta)\alpha 2^{n/2}$ collisions:

$$\mathbf{Pr} \geq 1 - \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}} \right)^{\alpha 2^{n/2}}$$

for any $\delta > 0$.

Every partial collision is assimilated to a draw in the coupon collector problem. A formula in [RP95] for the tail of coupon collector problem probability distribution allows us to estimate the chance of success after obtaining $\beta \cdot 2^{n/2}$ partial collisions:

$$\mathbf{Pr} \geq 1 - 2^{-\beta/\ln(2)+n/2}$$

which is positive whenever $\beta \geq n/2 \cdot \ln(2)$.

Therefore, we bound the probability of success when collecting $|\mathcal{A}| \cdot |\mathcal{B}| = \alpha 2^n$ pairs as the probability of obtaining at least $(1-\delta)\alpha 2^{n/2}$ partial collisions multiplied by the probability of success after sieving $(1-\delta)\alpha 2^{n/2}$ values:

$$\mathbf{Pr} \geq \left(1 - \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}} \right)^{\alpha 2^{n/2}}\right) \cdot \left(1 - 2^{-(1-\delta)\cdot\alpha/\ln(2)+n/2}\right)$$

In particular, with two lists of size $\sqrt{n/2} \cdot 2^{n/2}$ (*i.e.* $\alpha = n/2$), we get $\mathbf{Pr} \geq 0.99$ as long as $n \geq 32$ (using $\delta = 2^{-8}$). This theoretical lower bound is compared with actual simulations for $n = 64$ bits in Figure 4.2

4.2.3 Simulations

The Setting. Simulations have been run with a block size $n = 64$ bits, and a secret S of size $n/2 = 32$ bits. Encryption is done using the block cipher Tiny Encryption Algorithm (TEA [WN95]) in `CTR` mode. We create many lists: multiple key stream outputs lists $a_i \in \mathcal{A}$, and single encryptions list $b_j = a_j \oplus (\overline{0} \parallel S) \in \mathcal{B}$. After computing and sorting a list $\mathcal{B}$ with 2^{32} elements, we iteratively produce, sort and sieve several lists $\mathcal{A}$ containing 2^{32} elements each, until the secret S is the only one remaining in the sieve.

Results. One simulation runs in around 20 minutes over 36 cores, as every step is trivially parallelizable: encryption, sorting and sieving. We

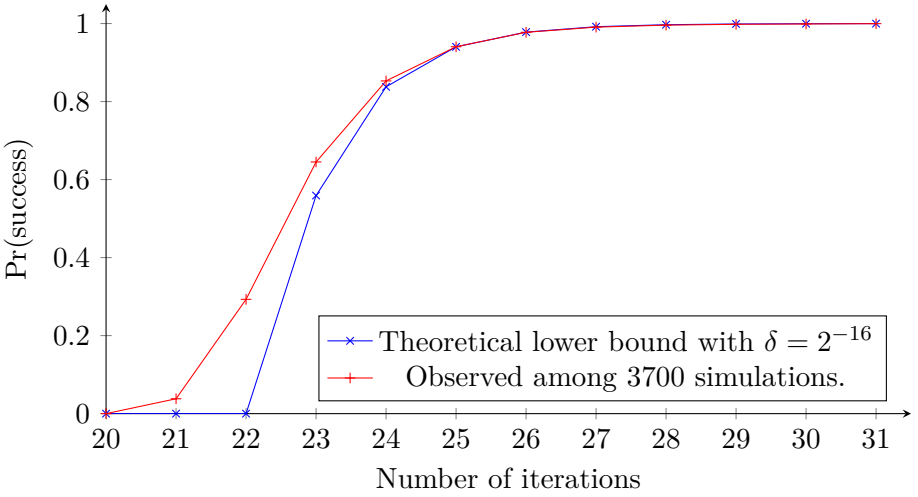


Figure 4.2: Probability of success of the known prefix sieving knowing 2^{32} encryptions of a 32-bit secret against the number of chunks of 2^{32} key stream blocks of size $n = 64$ bits used.

ran 3700 simulations and tracked how many lists of $2^{n/2} = 2^{32}$ key stream outputs were needed before a single value was left in the sieve. The coupon collector problem predicts that one needs on average $n/2 \cdot \ln(2) \cdot 2^{n/2}$ partial collisions which will be obtained after $n/2 \cdot \ln(2) \simeq 22.18 < 23$ rounds in expectation. And indeed, the simulations showed a 64.5% probability of success after 23 iterations.

Figure 4.2 shows the convergence between the theoretical lower bound of Section 4.2.2 and the success probability of the simulations. We also noticed that the discrepancy in the number of rounds required is largely due to the last few candidates remaining in the sieve. If we decided that the attack is successful whenever we are left with less than 1000 potential candidates for the secret, then the algorithm would successfully finish after 16 rounds. In fact, after 16 rounds, the number of candidates left varies from 419 to 560 in all the simulations we have run.

4.3 The Fast-Convolution Sieving

In the case of arbitrary $\mathcal{S}$ and no prior information on S, all algorithms seen so far fail to solve the missing difference problem in time less than

$\mathcal{O}(2^n)$.

In this section we show how to solve this particular case of the missing difference problem in time $\widetilde{\mathcal{O}}(2^{2n/3})$ with the fast-convolution sieving at the cost of a greater data complexity.

4.3.1 The Algorithm

Increasing the Data Complexity. At a high level, the idea of the algorithm is to increase the discrepancy between S and the other values. We do that by increasing the required number of draws $a \oplus b$ so that every value appears multiple times except for the secret S.

In fact, we only consider truncated values. Concretely, we look at the values $\lfloor a \oplus b \rfloor_m$ and see which one appears the less. Hopefully the truncated values $\lfloor S \rfloor_m$ will appear fewer times than the other values since S is unreachable. This allows us to recover the truncated value of S, and we can recover the rest via the known prefix sieving.

Description. Concretely, for $m < n$ bits we define a list of counters for a multi-set $\mathcal{X}$ as:

$$\forall i \in \{0,1\}^m : C_{\mathcal{X}}[i] = \left|\{x \in \mathcal{X} : \lfloor x \rfloor_m = i\}\right|.$$

The fast-convolution sieving algorithm (Algorithm 4.6) starts by building lists of counters $C_{\mathcal{A}}$ an $C_{\mathcal{B}}$ for the corresponding sets $\mathcal{A}$ and $\mathcal{B}$ with t the number of truncated bits (that is $m = n - t$). Then, the goal is to compute $C_{\mathcal{A}\oplus\mathcal{B}}$ efficiently from $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$, where $\mathcal{A} \oplus \mathcal{B}$ is the multi-set $\{a \oplus b \ : \ (a,b) \in \mathcal{A} \times \mathcal{B}\}$.

We observe that:

$$\begin{aligned}
C_{\mathcal{A}\oplus\mathcal{B}}[i] &= |\{(a,b) \in \mathcal{A} \times \mathcal{B} : \lfloor a \oplus b \rfloor_m = i\}| \\
&= \sum_{a \in \mathcal{A}} |\{b \in \mathcal{B} : \lfloor a \oplus b \rfloor_m = i\}| \\
&= \sum_{a \in \mathcal{A}} |\{b \in \mathcal{B} : \lfloor b \rfloor_m = i \oplus \lfloor a \rfloor_m\}| \\
&= \sum_{a \in \mathcal{A}} C_{\mathcal{B}}[i \oplus \lfloor a \rfloor_m] \\
&= \sum_{j \in \{0,1\}^{n-t}} C_{\mathcal{A}}[j] C_{\mathcal{B}}[i \oplus j]
\end{aligned}$$

This is a discrete convolution. In fact, this kind of convolution is efficiently computed with the Fast Walsh-Hadamard Transform (Algorithm 4.4) and this convolution is computed using in Algorithm 4.5 which works the same way we compute circular convolution using the Fast Fourier Transform.

The best guest for $\lfloor S \rfloor_m$ is the position of the smallest counter.

Optimisations. In order to increase the success rate of the algorithm, one can test several candidates for $\lfloor S \rfloor_m$ (using the lowest remaining counters), and use the known-prefix sieving to detect whether the candidate is correct.

Another option is to run multiple independent runs of the algorithm with different choices of the $n/3$ truncated bits. This would avoid some bad cases we have observed in simulations, where the right counter grows abnormally high and gets hidden in all the other counters.

4.3.2 Complexity Analysis

Data, Time and Memory. A balanced trade-off between the complexity and the success probability is achieved when $m = 2n/3$ (that is $t = n/3$). With this parameter the fast-convolution sieving requires:

$\mathcal{O}(\sqrt{n} \cdot 2^{2n/3})$	queries
$\mathcal{O}(n \cdot 2^{2n/3})$	bits of memory (counters)
$\mathcal{O}(n \cdot 2^{2n/3})$	computations (fast Walsh-Hadamard)

See the detailed proof below. Thus, each complexity is of order $\widetilde{\mathcal{O}}(2^{2n/3})$. Moreover, our simulations (Section 4.3.3) show that the constant hidden in the $\mathcal{O}$ notation is small: using lists of size $\sqrt{n} \cdot 2^{2n/3}$, the lowest counter corresponds to $\lfloor S \rfloor_m$ in at least 70% of our experiments.

Proof of success probability. Consider, without loss of generality and for blocks of size n, that we possess $|\mathcal{A}| = |\mathcal{B}| = \alpha \cdot 2^{2n/3}$ blocks of key stream and the same number of blocks of encrypted secret S with α a function of n. So in this setting we have $|\mathcal{A}| \cdot |\mathcal{B}| = \alpha^2 \cdot 2^{4n/3}$ different $(a \oplus b)$ values possible between the two lists, that we consider as independent and uniformly distributed over $\{0,1\}^n \backslash S$. Furthermore, we take $t = n/3$, that is, the fast-convolution algorithm considers the $2n/3$ most significant bits. Using the fast convolution (Algorithm 4.5) we compute $C_{\mathcal{A} \oplus \mathcal{B}}$ and we hope that the counter for $\lfloor S \rfloor_{2n/3}$, the good counter, will be lower than

Algorithm 4.4 Fast Walsh-Hadamard Transform

input: $|C_{\mathcal{A}}| = 2^m$.
output: The Walsh-Hadamard transform of $C_{\mathcal{A}}$.
procedure FWHT($C_{\mathcal{A}}$)
    for $d = m$ **downto** 0 **do**
        for $i = 0$ **to** 2^{m-d} **do**
            for $j = 0$ **to** 2^{d-1} **do**
                $C_{\mathcal{A}}[i \cdot 2^d + j] \leftarrow C_{\mathcal{A}}[i \cdot 2^d + j] + C_{\mathcal{A}}[i \cdot 2^d + j + 2^{d-1}]$
                $C_{\mathcal{A}}[i \cdot 2^d + j + 2^{d-1}] \leftarrow C_{\mathcal{A}}[i \cdot 2^d + j] - 2 \cdot C_{\mathcal{A}}[i \cdot 2^d + j + 2^{d-1}]$
    return $C_{\mathcal{A}}$

Algorithm 4.5 Fast convolution

input: $|C_{\mathcal{A}}| = |C_{\mathcal{B}}| = 2^{n-t}$.
output: $C_{\mathcal{A} \oplus \mathcal{B}}$.
procedure FastConvolution($C_{\mathcal{A}}$, $C_{\mathcal{B}}$)
    FWHT($C_{\mathcal{A}}$); FWHT($C_{\mathcal{B}}$);
    for $c = 0$ **to** 2^{n-t} **do**
        $C_{\mathcal{A} \oplus \mathcal{B}}[c] \leftarrow C_{\mathcal{A}}[c] \cdot C_{\mathcal{B}}[c]$
    FWHT($C_{\mathcal{A} \oplus \mathcal{B}}$);
    return $C_{\mathcal{A} \oplus \mathcal{B}}$

Algorithm 4.6 Sieving with fast convolution

input: : $\mathcal{A}, \mathcal{B} \subset \{0,1\}^n$, $t \leq n$.
output: : S s.t. $\forall (a, b) \in \mathcal{A} \times \mathcal{B},\ a \oplus b \neq S$.
procedure FastConvSieve($\mathcal{A}$,$\mathcal{B}$, t)
    $C_{\mathcal{A}}, C_{\mathcal{B}}, C_{\mathcal{A} \oplus \mathcal{B}} \leftarrow$ arrays of 2^{n-t} integers initialized to 0;
    for a **in** $\mathcal{A}$ **do**
        Increment $C_{\mathcal{A}}[\lfloor a \rfloor_{n-t}]$
    for b **in** $\mathcal{B}$ **do**
        Increment $C_{\mathcal{B}}[\lfloor b \rfloor_{n-t}]$

    $C_{\mathcal{A} \oplus \mathcal{B}} \leftarrow$ FastConvolution($C_{\mathcal{A}}, C_{\mathcal{B}}$)
    $u \leftarrow \text{argmin}_i\, C_{\mathcal{A} \oplus \mathcal{B}}[i]$ ▷ Success if $S_{0..(n-t-1)} = u$.
    return $u \oplus$ PrefixSieve($\mathcal{A}, \mathcal{B} \oplus \{u \,\|\, 0^t\}, \{0\}^{n-t} \times \{0,1\}^t$)

all the other counters, the bad counters. We want to find the value α such that the algorithm succeeds with probability $\Omega(1)$.

Let X_i^c represents the fact that the i^{th} value truncates to c, so that X_i^c follows a Bernoulli distribution and any counter can be written as $X^c = \sum_{i=1}^{\alpha^2 2^{4n/3}} X_i^c$. Now we have to discriminate between the distributions of the good and bad counters:

$$\begin{aligned}
\text{(Good) } c = \lfloor S \rfloor_{2n/3}: \quad & \mathbf{Pr}(X_i^{\lfloor S \rfloor_{2n/3}} = 1) = (2^{n/3} - 1)/(2^n - 1) \\
& \implies \mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}] = (2^{5n/3}\alpha^2 - 2^{4n/3}\alpha^2)/(2^n - 1)
\end{aligned}$$

$$\begin{aligned}
\text{(Bad) } c \neq \lfloor S \rfloor_{2n/3}: \quad & \mathbf{Pr}(X_i^c = 1) = (2^{n/3})/(2^n - 1) \\
& \implies \mathbf{E}[X^c] = (2^{5n/3}\alpha^2)/(2^n - 1)
\end{aligned}$$

Now we are interested in the probability that a bad counter gets a value above $\mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}]$ as a measure of how distinct the distributions are. Using Chernov Bound we get for all $c \neq \lfloor S \rfloor_{2n/3}$ that:

$$\begin{aligned}
\mathbf{Pr}(X^c \geq \mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}]) &= 1 - \mathbf{Pr}(X^c < \mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}]) \\
&= 1 - \mathbf{Pr}(X^c < (1 - 2^{-n/3})2^{5n/3}\alpha^2/(2^n - 1)) \\
&= 1 - \mathbf{Pr}(X^c < (1 - 2^{-n/3})\mathbf{E}[X^c]) \\
&\geq 1 - e^{-((2^{-n/3})^2 \cdot 2^{5n/3}\alpha^2)/(2(2^n - 1))} \\
&\geq 1 - e^{-\alpha^2/2}
\end{aligned}$$

And to compute the probability that no bad counter gets below $\mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}]$ we assume their independence, which is wrong, but we will come back later to discuss this assumption.

$$\begin{aligned}
\mathbf{Pr}(\forall c \neq \lfloor S \rfloor_{2n/3} : X^c \geq \mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}]) &= \prod_{c \neq \lfloor S \rfloor_{2n/3}} \left(1 - \mathbf{Pr}(X^c < \mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}])\right) \\
&\geq \left(1 - e^{-\alpha^2/2}\right)^{2^{2n/3}} .
\end{aligned}$$

To conclude, we need to find an $\alpha = \alpha(n)$ such that this probability remains greater than some positive value as n grows. This is clearly achieved with $\alpha = \mathcal{O}(\sqrt{n})$ as for example taking $\alpha = \frac{2\sqrt{n}}{\sqrt{3 \cdot \log_2(e)}} \simeq 0.96\sqrt{n}$ we get:

$$\begin{aligned}\mathbf{Pr}(\forall c \neq \lfloor S \rfloor_{2n/3} : X^c \geq \mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}]) &\geq (1 - e^{-\alpha^2/2})^{2^{2n/3}} \\ &\geq (1 - 2^{-2n/3})^{2^{2n/3}} \\ &\geq 0.25, \qquad \forall n \geq 3/2\end{aligned}$$

Therefore, we bound the probability of success by the events "$X^{\lfloor S \rfloor_{2n/3}} < \mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}]$", probability $\simeq 1/2$, and "$\forall c \neq \lfloor S \rfloor_{2n/3} : X^c \geq \mathbf{E}[X^{\lfloor S \rfloor_{2n/3}}]$", probability at least $1/4$. And we indeed have a probability of at least $1/8$ of having a successful algorithm. Hence, when $|\mathcal{A} \times \mathcal{B}| = \mathcal{O}(n \cdot 2^{4n/3})$ the algorithm has probability $\Omega(1)$ of succeeding.

Notice that when the list are balanced we have $|\mathcal{A}| = |\mathcal{B}| = \mathcal{O}(\sqrt{n} \cdot 2^{2n/3})$ but the proof only requires that $|\mathcal{A}| \cdot |\mathcal{B}| = n \cdot 2^{4n/3}$. This allows for a range of trade-offs for the lists size.

Regarding the independence of the counters, this is obviously wrong as they are bound by the relation $\sum_c X^c = \alpha^2 2^{4n/3}$. However, this relation becomes looser and looser as n grows so the approximation obtained should still be correct asymptotically. Moreover, the covariances implied are negative meaning that knowing one draw is big makes the other draws smaller in expectation to compensate. Small negative covariances will make the distribution look more evenly distributed in the sense that we can't observe too many extreme events in a particular direction which is good for the success rate of the algorithm. So the assumption of independence may be a conservative one for this complexity analysis and the expected behavior closely matches the one observed in our simulations.

4.3.3 Simulations

Estimating Success Probability. We ran simulations for block sizes $n = 12$, 24, 32 and 48 bits, so that we can estimate the success probability for this algorithm. We first create two lists of same size, one of raw key stream outputs and the other of `CTR` encryptions of an n-bit secret S. Then, we run Algorithm 4.5 counting over $m = 2n/3$ bits (unless specified otherwise) to get a list of counters for each possible XOR output on those m bits. The expected behavior of the attack would be to look for a solution whose m first bits correspond to the position of the lowest counter and test this hypothesis with the known prefix sieving Algorithm 4.3. If it returns a unique value, this is S, and we are done. If it returns an empty set, we test with the position of the second lowest counter, etc. As we

know the number of key candidates that would be required to recover S, we compute over many simulations an estimation of the probability of success after a given number of candidates.

Settings. For block sizes of 12 and 24 bits, we simulated a random permutation simply by shuffling a range into a list. For bigger sizes of 32 and 48 we used the lightweight cipher Simon designed by the NSA [Bea+15] with a random key as that is one of the rare block ciphers that can act on 48-bit blocks. We could quickly gather 10 000 runs for each setting except for the heavier 48-bit blocks simulation where we gathered 756 runs. Indeed, for $n = 48$ bits, one simulation took us 40 minutes over 10 cores (each step is highly parallelizable), and 64 gibibytes of RAM for the counter lists.

Observations. In general, we observe in Figure 4.6 that the algorithm has a good chance of success with the first few candidates when using the suggested parameters. Moreover, the sensibility with respect to the data complexity (Figure 4.4) and to the number of bits counted over (Figure 4.5) is fairly high. These results back up our complexity analysis and are a good indication that no big constant is ignored by the $\mathcal{O}()$ notation.

On the speed at which the probability increases we realized that, despite the log scale on the x axis, the curves take a straight (Figure 4.3) or concave shape (Figure 4.5 4.6). That means that the probability of success with the next key candidate decreases very quickly with the number of key candidates already tested and proved wrong. For example, for $n = 48$ bits (Figure 4.6) over 756 trials, the right key candidate was in the 2048 lowest counters in 98.1% of the time, but the worst case found was at rank 1 313 576. These "very bad" cases push the mean rank of the right key candidate to 2287 and its sample variance to 2 336 937 008. This shows that the analysis made in Section 4.3.2 wouldn't be improved significantly if we defined the algorithm's success by the right counter being among the lowest instead of strictly the lowest one.

4.4 Application

With those new approaches to solving the missing difference problem we now describe two applications: a cryptanalysis of the `CTR` mode using the known prefix sieving in Section 4.4.1 and a cryptanalysis of some

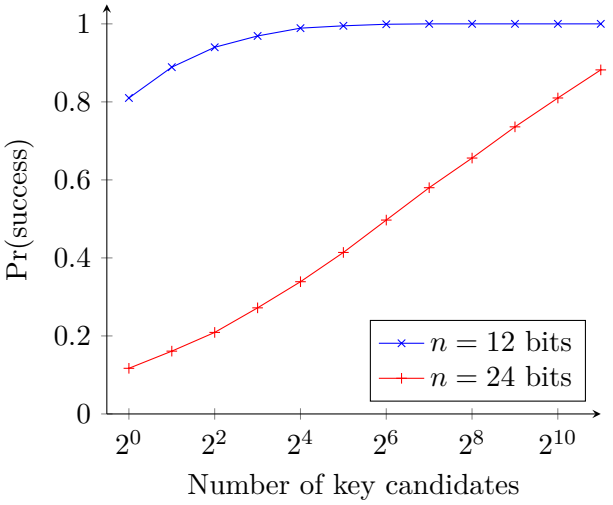


Figure 4.3: Results for lists size of $3 \cdot 2^{2n/3}$

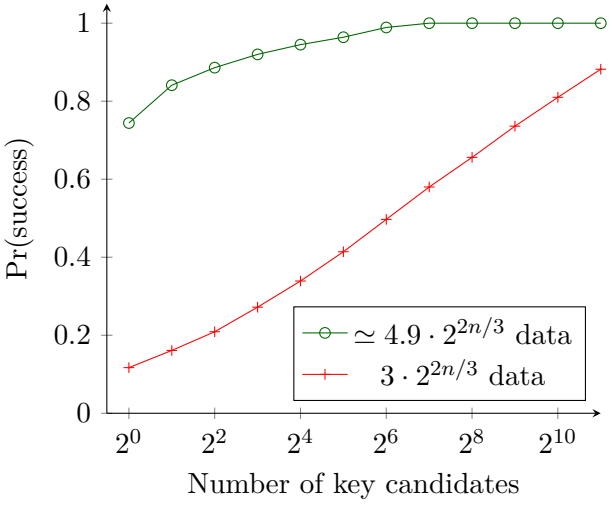


Figure 4.4: Results for $n = 24$ bits

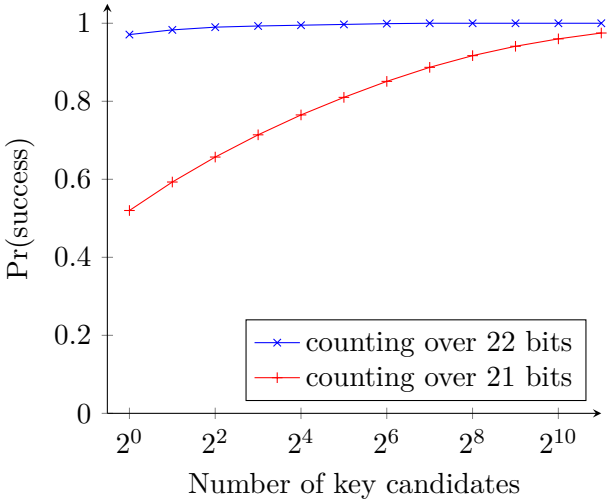


Figure 4.5: Results for $n = 32$ bits; $\sqrt{n}2^{2n/3} \simeq 5.66 \cdot 2^{2n/3}$ data

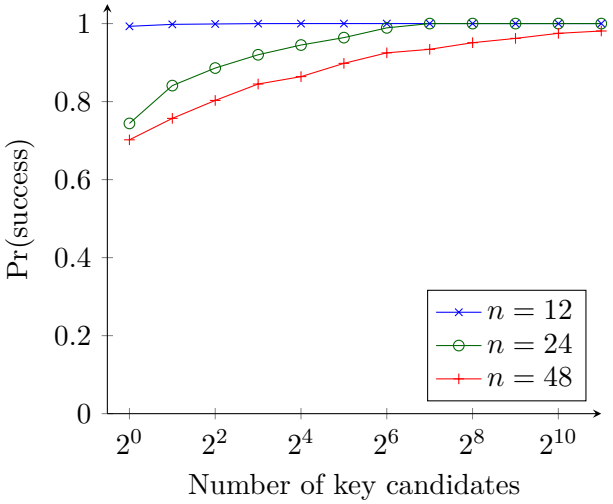


Figure 4.6: Results for $\sqrt{n}2^{2n/3}$ data; counting over $2n/3$ bits

Wegman-Carter-Shoup constructions using the fast convolution sieving in Section 4.4.2.

4.4.1 Plaintext Recovery of the Counter Mode

Direct Application. There are many settings where unknown plaintext will naturally lie in some known affine subspace, and the known prefix sieving algorithm can be used directly. For instance a credit card number (or any number) could be encoded in 16 bytes of ASCII before getting encrypted. Because in ASCII the encoding of any digit starts by `0x3` (`0x30` to `0x39`), we know half of the bits of the plaintext, and we can use the known-prefix sieving with $z = n/2$. Other examples are information encoded by `uuencode` that uses ASCII values `0x20` to `0x5F` (corresponding to two known bits) or HTML authentication cookies that are typically encoded to some subset of ASCII numbers and letters[1].

Chosen-Prefix Secret-Suffix. We saw in Section 2.4.2 the Beastly attack model which is a practical way to set up a plaintext recovery attack. Concretely we show an attack on the `CTR` mode in the Chosen-Prefix Secret-Suffix model [Hoa+15] where we can choose the length of a known head H and get the encryption of $\mathbf{Enc}(H \parallel S)$.

For the attack to work we need to control the way S is split among blocks. However, and unlike the direct application, we do not assume anything on S. It is fit to recover some authentication token with no apparent structure.

This is truly a block splitting attack: the attacker starts with a header length so that a small chunk of the secret message is encrypted together with known information, and he recovers this secret chunk. Then, he changes the length of the header to recover a second chunk of the message, using the fact that the first chunk is now known. Eventually, the full secret is recovered iteratively.

In our case, the easiest choice is to recover chunks of $n/2$ bits of secret one by one, using the known-prefix sieving Algorithm 4.3 with $z = n/2$. We illustrate this attack in Figure 4.7, assuming a two-block secret $S = S_1 \parallel S_2 \parallel S_3 \parallel S_4$, and a protocol that lets the adversary query an encryption of the secret with an arbitrary chosen prefix:

[1] For example, `wikipedia.org` encodes cookies with lower case letters and digits, this corresponds to two known bits.

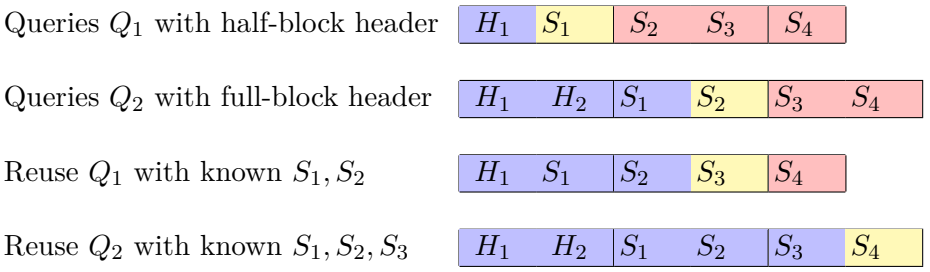


Figure 4.7: Example of an attack on two blocks secret $S = S_1 \| S_2 \| S_3 \| S_4$. Each step performs the known prefix sieving algorithm. Known information in blue, unknown information in red, attacked information in yellow.

1. The attacker makes two kinds of queries
   - Q_1 with a known half-block header H_1 ($\mathbf{Enc}([H_1 \| S_1] \| [S_2 \| S_3] \| [S_4])$);
   - Q_2 with a known full-block header $H_1 \| H_2$ ($\mathbf{Enc}([H_1 \| H_2] \| [S_1 \| S_2] \| [S_3 \| S_4])$).
2. He first recovers S_1 using the known-prefix sieving with the first block of each type of query. More precisely, he uses $\mathcal{A} = \{\mathbf{Enc}(H_1 \| H_2)\}$ and $\mathcal{B} = \{\mathbf{Enc}(H_1 \| S_1)\}$, so that the missing difference is $0 \| (S_1 \oplus H_2)$.
3. When S_1 is known, he can again use known prefix sieving to recover S_2, with the first and second blocks of Q_2 queries: $\mathcal{A} = \{\mathbf{Enc}(H_1 \| H_2)\}$ and $\mathcal{B} = \{\mathbf{Enc}(S_1 \| S_2)\}$, so that the missing difference is $(S_1 \oplus H_1) \| (S_2 \oplus H_2)$. To improve the success rate of this step, he can also consider the first block of Q_1 queries as known key stream.
4. When S_2 is known, another round of known prefix sieving reveals S_3, for instance with $\mathcal{A} = \{\mathbf{Enc}(H_1 \| H_2)\}$ and $\mathcal{B} = \{\mathbf{Enc}(S_2 \| S_3)\}$, the missing difference is $(S_2 \oplus H_1) \| (S_3 \oplus H_2)$.
5. Finally, S_4 is recovered with a last round of known prefix sieving using $\mathcal{A} = \{\mathbf{Enc}(H_1 \| H_2)\}$ and $\mathcal{B} = \{\mathbf{Enc}(S_3 \| S_4)\}$, with missing difference is $(S_3 \oplus H_1) \| (S_4 \oplus H_2)$.

This gives an algorithm with query complexity of $\mathcal{O}(\sqrt{n}2^{n/2})$ to recover repeated encryption of a secret over multiple blocks in the Chosen-Prefix Secret-Suffix model. In Section 4.2.3, we analyzed the constants in the $\mathcal{O}$ notation and run experiments with $n = 64$ using locally encrypted data. In particular, we have a success probability higher than 80% using two lists of 5×2^{32} queries with $n = 64$.

Generally, we show that for $n \geq 32$ the success probability of this attack is at least 99% with lists of size $\sqrt{n/2} \cdot 2^{n/2}$. With a one block secret, an optimal attack uses two lists of $\sqrt{n/2} \cdot 2^{n/2}$ two-block queries: queries $[H_1 \,\|\, S_1] \,\|\, [S_2]$ with a half-block header, and queries $[H_1 \,\|\, H_2] \,\|\, [S_1 \,\|\, S_2]$ with a full-block header. This translates to a data complexity of $4\sqrt{n/2} \cdot 2^{n/2}$ blocks. For comparison, an attack against the `CBC` mode requires on average $2 \cdot 2^{n/2}$ blocks of data in the ideal case.

Smaller Splitting. Alternatively, an attacker could recover the secret bit by bit. This leads to a less realistic attack in practice, and we can show that the complexity is similar. For this variant, McGrew's searching algorithm could also be used instead of our known-prefix sieving algorithm (because in this scenario, we have $|\mathcal{S}| = 2$). Overall, we find that the best strategy is still to split the secret and recover $n/2$ bits at a time: smaller chunks make the attack less practicable with no gain of complexity, while bigger ones would increase the time complexity.

Let us show the complexity of such an attack to recover a block of secret, taking into account the n steps necessary for this attack. For simplicity, we consider a setting where one query returns a block of key stream and the encryption of $0 \,\|\, s_i$ with an unknown bit s_i. We are interested in the query complexity for recovering n bits of secret one bit at a time; that is we need to know the first bit to ask for the second one, etc. Clearly this can be done in $\mathcal{O}(n \cdot 2^{n/2})$ queries by repeating n times the attack on one bit. However, we need fewer and fewer queries to uncover the next bit as we go forward and accumulate blocks of key stream.

Let:

$U_i \leftarrow$The expected number of encryption of $0 \,\|\, s_i$ to recover s_i.

$K_i \leftarrow$The expected number of raw key stream outputs to recover s_i.

From the definition of a query, the above description and because each time we find a bit of secret we can deduce a range of key stream blocks

for the next step we have the relations:

$$K_1 = U_1 \tag{4.1}$$
$$K_{i+1} = K_i + U_i + U_{i+1} \quad \text{for } i \geq 1 \tag{4.2}$$
$$K_i \cdot U_i = 2^n \quad \text{(in expectation)} \tag{4.3}$$

We consider the following proposition:

$$P_i : U_i = 2^{n/2}(\sqrt{i} - \sqrt{i-1}),$$

and, using (4.2), when P_k true for all $k \leq i$ we have:

$$K_i = 2\sum_{k=1}^{i-1} U_k + U_i = 2^{n/2}(\sqrt{i} + \sqrt{i-1}).$$

Moreover, (4.1) and (4.3) imply $K_1 = U_1 = 2^{n/2}$ so P_1 is true. Now suppose P_k true for all $k \leq i$, let's prove it holds for P_{i+1}:

$$\begin{aligned}
& K_{i+1} \cdot U_{i+1} = 2^n && \text{by (4.3)} \\
\implies & U_{i+1}^2 + (K_i + U_i) \cdot U_{i+1} - 2^n = 0 && \text{by (4.2)} \\
\implies & U_{i+1}^2 + 2^{n/2} \cdot 2\sqrt{i} \cdot U_{i+1} - 2^n = 0 && \text{by } P_i \\
\implies & U_{i+1} = 2^{n/2}(\sqrt{i+1} - \sqrt{i}) && \text{as } U_{i+1} \geq 0 \\
\implies & P_{i+1} \text{ is true.}
\end{aligned}$$

Now that we have a closed form for U_i we can deduce the expected number of queries needed to recover n bits of secret by summing over as $\sum_{i=1}^{n} U_i = 2^{n/2}\sqrt{n}$.

Therefore, the query complexity is really $\mathcal{O}(\sqrt{n} \cdot 2^{n/2})$ ignoring a constant depending on the length of a query. Notice that this complexity is the same as when sieving S as a whole showing that we don't increase the query complexity by more than a constant with this strategy.

About Rekeying. As for many modes of operation, the common wisdom to counter this kind of attacks asks for rekeying before the birthday bound, that is before $2^{n/2}$ blocks. However, rekeying too close to the birthday bound may not be enough. For example let's consider an implementation of a `CTR` based mode of operation that rekeys every $2^{n/2}$ blocks. Using the same model as previously for a one-block secret, an optimal

attack uses queries $[H_1 \parallel S_1] \parallel [S_2]$ with a half-block header, and queries $[H_1 \parallel H_2] \parallel [S_1 \parallel S_2]$ with a full-block header, where rekeying occurs after $2^{n/2-2}$ queries of each type. To recover S_1, we use the known prefix sieving algorithm as previously, but we can only use relations between ciphertext blocks encrypted with the same key. In each session of $2^{n/2}$ blocks, we consider 2^{n-4} pairs of ciphertext blocks; on average there are $2^{n/2-4}$ pairs with the correct prefix used for sieving. Since we need $n/2 \cdot 2^{n/2}$ draws to reduce the sieve to a single element with high probability, we use $8n$ sessions, that is $8n \cdot 2^{n/2}$ blocks of data in total. The same data can be reused to recover S_2 when S_1 is known. This should be compared with the previous data complexity of $4\sqrt{n/2} \cdot 2^{n/2}$ in the absence of rekeying.

However, rekeying every $2^{n/2-16}$ blocks makes the data complexity goes up to $2^{35}n$ sessions or $n \cdot 2^{19+n/2}$ blocks to recover the secret block. Notice that the security gain of rekeying is comparable with `CBC` where rekeying every $2^{n/2-16}$ blocks forces an increase of the data complexity from $2 \cdot 2^{n/2}$ to $2^{18} \cdot 2^{n/2}$ blocks.

The `CTR` Mode in Communication Protocols. The `CTR` mode is widely used in internet protocols, in particular as part of the `GCM` authenticated encryption mode [MV04], with the `AES` block cipher. For instance, Mozilla telemetry data show that more than 97% of HTTPS connections from Firefox nightly 92 use AES-GCM[2]. While attacks against modes with a 128-bit block cipher are not practical yet, it is important to limit the amount of data processed with a given key, in order to keep the probability of a successful attack negligible, following the guidelines of Luykx and Paterson [LP16].

Surprisingly, there are also real protocols that use 64-bit block ciphers with the `CTR` mode (or variants of the `CTR` mode), as shown below. Attacks against those protocols would be (close to) practical, assuming a scenario where an attacker can generate the encryption of numerous messages with some fixed secret.

SSH. Cipher-suites based on the `CTR` mode were added to SSHv2 in 2006 [BKN06]. In particular, `3DES-CTR` is one of the recommended ciphers, but actual usage of `3DES-CTR` seems to be rather low [Alb+16]. In practice, `3DES-CTR` is optionally supported by the Dropbear server, but it is not implemented in OpenSSH. According to a scan of the full IPv4 space by

[2] `https://mzl.la/32PT4x9`, accessed September 22, 2020

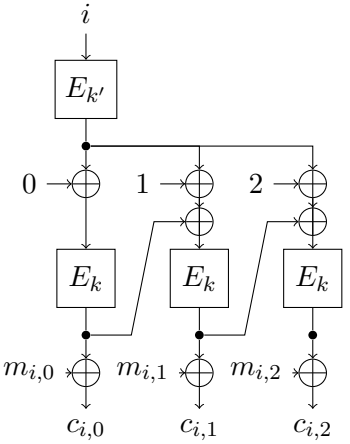


Figure 4.8: f8 mode (i is a message counter)

Censys.io[3], around 7.15% of SSH servers support `3DES-CTR`, but actual usage is hard to estimate because it depends on client configuration.

The SSH specification requires to rekey after 1 GB of data, but an attack is still possible, although the complexity increases.

3G telephony. The main encryption algorithm in UMTS telephony is based on the 64-bit block cipher Kasumi. The mode of operation, denoted as f8, is represented in Figure 4.8. While this mode in not the `CTR` mode and was designed to avoid its weaknesses, our attack can be applied to the first block of ciphertext. Indeed, the first block of message i is encrypted as $c_{i,0} = m_{i,0} \oplus E_k(E_{k'}(i))$, where the value $E_k(E_{k'}(i))$ is unique for all the messages encrypted with a given key.

There is a maximum of 2^{32} messages encrypted with a given key in 3G, but this only has a small effect on the complexity of attacks.

Because of the low usage of `3DES-CTR` in SSH, and the difficulty of mounting an attack against 3G telephony in practice, we did not attempt to demonstrate the attack in practice. However, the setting and complexity of our attacks are comparable to the Sweet32 attack on the `CBC` mode with 64-bit ciphers [BL16].

[3] https://web.archive.org/web/20200620131835/https://censys.io/ipv4/report?field=22.ssh.v2.support.client_to_server.ciphers&max_buckets=, scan performed over 18 400 820 servers on June 20, 2020

4.4.2 Partial Key Recovery of GMAC and Poly1305

Because the fast convolution algorithm requires no prior information on S, it can be adapted to other modes of operation based on CTR and particularly to Wegman-Carter-Shoup type of modes for authentication, see Section 2.2.3. Wegman-Carter-Shoup MACs use a keyed permutation E and a keyed universal hash function h, with k_1 and k_2 two private keys. The input is a message m and a nonce N, and the MAC is defined as:

$$\mathrm{MAC}(N, m) = h_{k_1}(m) + E_{k_2}(N) \ .$$

Again, the construction requires that all block cipher inputs are different. To apply our attack, we use two fixed messages m and m', and we capture many values $\mathrm{MAC}(N, m)$ in a list $\mathcal{A}$ and values $\mathrm{MAC}(N', m')$ in a list $\mathcal{B}$, all using unique nonces. Then, we solve the missing difference problem to recover $h_{k_1}(M) - h_{k_1}(M')$ as we know that $\forall N \neq N' : E_{k_2}(N) - E_{k_2}(N') \neq 0$. When the hash is a polynomial based AXU function, it is often sufficient to know this difference and the two messages m and m' to recover the key k_1. We give two examples with GMAC and Poly1305 algorithms. In addition, we show a simple forgery attack on Wegman-Carter-Shoup with a deterministic hash function applicable on the authentication part of the CWC mode of operation.

Galois/Counter Mode. GCM is an AEAD mode with associated data, combining the CTR mode for encryption and a Wegman-Carter MAC based on polynomial evaluation in a Galois field for authentication. When used with an empty message, GCM only performs authentication and it reduces to GMAC. In our attack, we use an empty message with one block of authenticated data a, so that the tag is computed as:

$$\mathrm{MAC}(N, a) = a \cdot H^2 \oplus H \oplus E_k(N),$$

with $H = E_k(0)$ the hash key and $(\cdot)$ the multiplication in a Galois Field defined by a public polynomial. So, for two different blocks of authenticated data a and a' we collect $\mathcal{O}(\sqrt{n} \cdot 2^{2n/3})$ MACs and perform the fast convolution algorithm to recover $a \cdot H^2 \oplus H \oplus a' \cdot H^2 \oplus H = (a \oplus a') \cdot H^2$. We know $a \oplus a'$ and the field is known, so we invert that value and recover H^2 then compute the square root and recover the hash key H.

In a concurrent work (published at the same venue) Luykx and Preneel [LP18] described a similar attack on GMAC in the information theoretic setting to match the best known proof [Ber05a]. Their approach

directly sieves out the wrong key candidates. That is for any known couple of 1-block messages m, m' they solve the inequality $\text{MAC}(N, m) \oplus \text{MAC}(N', m') \neq m \cdot H^2 \oplus m' \cdot H^2$ to eliminate a candidate of H.

The advantage of this version is that we don't need to manipulate the values to force the repetition of the authenticated messages. In particular, `GCM` authenticates the ciphertext which is known but isn't expected to be repeated.

On the other hand this cryptanalysis still requires $\mathcal{O}(2^n)$ time and memory in the computational model making it hardly practical and it is unclear how to apply our ideas to efficiently sieve in this context. Moreover, it can't be applied to modes such as `CWC` unlike our approach based on the missing difference.

Comparison with other cryptanalysis of `GMAC`. There are other known attacks against `GCM` and `GMAC`, but none of them seems to allow universal forgery with just $2^{2n/3}$ blocks of data and $2^{2n/3}$ computations. In particular, Handschuh and Preneel [HP08] gave a weak-key attack, that can also be used to recover the hash key without weak key assumptions, using roughly $2^{n/2}$ messages of $2^{n/2}$ blocks. Later work extended these weak key properties [Saa12; PC15], but an attack still requires about 2^n blocks in total when no assumptions are made about the key. We also note that these attacks require access to a verification oracle, while our attack only uses a MAC oracle.

Other attacks use specific options of the `GCM` specifications to reach a lower complexity, but cannot be applied with standard-length IV, and tag: Ferguson [Fer05b] showed an attack when the tag is truncated, and Joux [Jou06] gave an attack based on non-default IV lengths.

Poly1305. `Poly1305` [Ber05b] is a MAC scheme following the Wegman-Carter-Shoup construction, using polynomial evaluation modulo the prime number $2^{130} - 5$. It uses a keyed 128-bit permutation (usually `AES`), and the hash function key, r, has 106 secret bits (22 bits of the key are set to 0, including in particular the 4 most significant ones). The message blocks are first padded to 129-bit values c_i. Then, the MAC of a q-block message m with nonce N is defined as:

$$T(m, N) = (((c_1 r^q + c_2 r^{q-1} + ... + c_q r) \mod 2^{130} - 5) + E_k(N)) \mod 2^{128}.$$

With the same strategy as above, using two different messages m and m' we recover the missing difference

$$(((c_1 - c'_1)r^q + (c_2 - c'_2)r^{q-1} + ... + (c_q - c'_q)r) \mod 2^{130} - 5) \mod 2^{128}.$$

Moreover, we chose m and m' such that $c_i - c'_i = 0$ and $c_q - c'_q = 1$; since by design, $r < 2^{124}$ the value recovered is simply the hash key r.

Notice that `Poly1305` doesn't use the XOR operation but a modular addition, and we have to adapt our algorithms to this case. Luckily, the fast convolution algorithm can easily be tweaked. First, we keep the $2n/3$ least significant bits to avoid issues the carry, something the XOR operation doesn't have. Moreover, when the lists of counters are up, we need to compute their cyclic convolution, which is done with a fast convolution algorithm based on the fast Fourier transform (instead of fast Walsh-Hadamard). Finally, we verify the value suggested by the lowest counter by running the known prefix algorithm looking for collisions on the least significant bits and sieving the modular subtraction of the most significant bits. This adaptation has similar complexities and proofs than the one described earlier. Moreover, in the case of `Poly1305`, one can further adapt the algorithms to take into account the fact that 22 bits of the key r are fixed at 0 effectively reducing the dimensions of $\mathcal{S}$.

Simple Forgery on `CWC`. The `CWC` mode was designed by Kohno, Viega and Whiting [KVW04], shown in Figure 4.9. It follows the Wegman-Carter-Shoup construction with a twist: it encrypts the output of the hash function before masking it with a fresh nonce. This makes exploiting the property of the hash function to recover the key impossible. However, by solving the missing difference problem we can recover the value $\Delta = E_k(\texttt{CWC-HASH}_H(a)) \oplus E_k(\texttt{CWC-HASH}_H(a'))$ for fixed authenticated data a and a'. This is enough to make a forgery; for a fresh nonce N get the tag $\mathrm{MAC}(N, a)$ then deduce the tag for a' that is $\mathrm{MAC}(N, a') = \mathrm{MAC}(N, a) \oplus \Delta$.

This attack works as soon as the hash function used is deterministic. Indeed, the forgery exploits the fact that the difference in the output of the hash remains unchanged across multiple nonces.

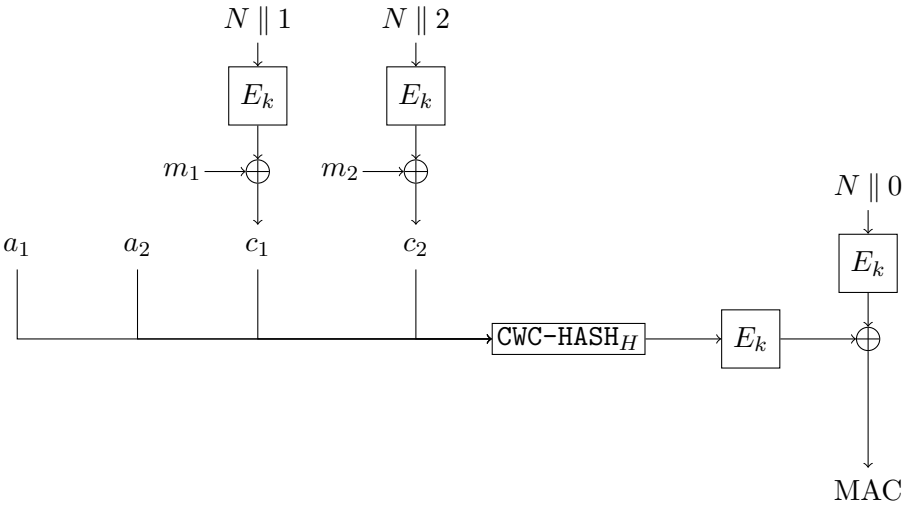


Figure 4.9: The CWC mode of operation for an 2-block message $m_1 \| m_2$ and authenticated data $a_1 \| a_2$ with a block cipher key k and an hash key $H = E_k(11 \| 0^*)$.

4.5 Conclusion

Attacks Summary. We have given efficient algorithms for the missing difference problem in two practically relevant cases: with an arbitrary missing difference, and when the missing difference is known to be in some low-dimension vector space. These algorithms lead to a message-recovery attack against the CTR mode with complexity $\widetilde{\mathcal{O}}(2^{n/2})$, and a universal forgery attack against some Carter-Wegman MACs with complexity $\widetilde{\mathcal{O}}(2^{2n/3})$.

In particular, we show that plaintext recovery attacks against the CTR mode and the CBC mode can be mounted with roughly the same requirements and the same complexity. This goes against the folklore assumption that the security loss of the CTR mode with large amounts of data is slower than of the CBC mode.

Mitigation. Therefore, the CTR mode with 64-bit block ciphers should be considered unsafe (unless strict data limits are in place). As a countermeasure, we recommend to use larger block sizes, and to rekey well before $2^{n/2}$ blocks of data. Concrete guidelines for 128-bit block ciphers have

been given by Luykx and Paterson [LP16]. Alternatively, if the use of small block is required, we suggest using a mode with provable security beyond the birthday bound, such as `CENC` [Iwa06; IMV16].

Chapter 5
Beyond-Birthday-Bound Secure MAC

Contributions brought forward in this chapter were published in Crypto 2018 and are a joint work of Leurent, Nandi and I [LNS18].

Introduction

In the quest for Beyond-Birthday-Bound Security many proposals for deterministic authentication schemes follow the Double-block Hash-then-Sum strategy explained in Section 5.1. They originally enjoyed a security proof up to $\mathcal{O}(2^{2n/3})$ and there were no known theoretical attack faster than the generic 2^n brute-force.

Our contributions. In this chapter we present a generic cryptanalysis for the Double-block Hash-then-Sum construction, Section 5.1.2. Then, we concretely show an attack on the authentication modes `SUM-ECBC`, `GCM-SIV2`, `PMAC+`, `LightMAC+`, `3kf9` and `1kPMAC+` in data complexity $\mathcal{O}(2^{3n/4})$.

Although they were all originally shown secure up to $\mathcal{O}(2^{2n/3})$ short queries, Kim, Lee and Lee [KLL20] later proved that our cryptanalysis are actually optimal for modes `SUM-ECBC`, `PMAC+`, `LightMAC+` and `3kf9` by providing a tight bound matching our attack in data complexity $\mathcal{O}(2^{3n/4})$ for short queries.

Most of the attacks shown are information theoretic as they have a time complexity greater than 2^n, but we show a variant for `SUM-ECBC` and `GCM-SIV2` that runs in data and time complexity $\mathcal{O}(2^{6n/7})$.

We also show a birthday bound attack on `1kf9` [Dat+15] that was already withdrawn due to a flaw in its proof. This attack shows that the scheme actually doesn't offer beyond birthday bound security. Our generic attack also invalidates an improved proof for `LightMAC+` by Naito [Nai18].

Mode	Provable security bounds		Attacks (this work)		
	Advantage	Queries	Queries	Time	Type
`SUM-ECBC` [Yas10]	$\mathcal{O}\left(\frac{q^4{\ell_m}^3}{2^{3n}}\right)$ [KLL20]	$\Omega(2^{3n/4})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	U
			$\mathcal{O}(2^{6n/7})$	$\widetilde{\mathcal{O}}(2^{6n/7})$	U
`GCM-SIV2` [IM16]	$\mathcal{O}\left(\frac{q^3{\ell_m}^2}{2^{2n}}\right)$	$\Omega(2^{2n/3})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	U
			$\mathcal{O}(2^{6n/7})$	$\widetilde{\mathcal{O}}(2^{6n/7})$	U
`PMAC+` [Yas11]	$\mathcal{O}\left(\frac{q^4{\ell_m}^2}{2^{3n}} + \frac{q{\ell_m}^2}{2^n}\right)$ [KLL20]	$\Omega(2^{3n/4})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	E
`LightMAC+` [Nai17]	$\mathcal{O}\left(\frac{q^4}{2^{3n}}\right)$ [KLL20]	$\Omega(2^{3n/4})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	E
`1kPMAC+` [Dat+17]	$\mathcal{O}\left(\frac{\sigma}{2^n} + \frac{q\sigma^2}{2^{2n}}\right)$	$\Omega(2^{2n/3})$	$\mathcal{O}(2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{3n/2})$	E
`3kf9` [Zha+12]	$\mathcal{O}\left(\frac{q^4{\ell_m}^6}{2^{3n}}\right)$ [KLL20]	$\Omega(2^{3n/4})$	$\mathcal{O}(\sqrt[4]{n} \cdot 2^{3n/4})$	$\widetilde{\mathcal{O}}(2^{5n/4})$	U
`1kf9` [Dat+15]	$\mathcal{O}\left(\frac{q{\ell_m}^2}{2^n} + \frac{q^3{\ell_m}^4}{2^{2n}} + \frac{q^4{\ell_m}^4}{2^{3n}} + \frac{q^4{\ell_m}^6}{2^{4n}}\right)$	$\Omega(2^{2n/3})$	$\mathcal{O}(2^{n/2})$	$\widetilde{\mathcal{O}}(2^{n/2})$	U

Table 5.1: Summary of the security for studied modes and our main results. q is the number of queries, ℓ_m is the maximum size of a query, σ is the total number of processed blocks. The expected lower bound and attack complexity is in number of constant length queries ($\ell_m = \mathcal{O}(1)$). We use "U" for universal forgeries, and "E" for existential forgeries.

5.1 Double-block Hash-then-Sum MACs

The Challenge. We can distinguish three types of modes for authentication: deterministic, nonce-based and probabilistic. Nonce-based MACs, like Wegman-Carter MACs [WC81], require to maintain a state to provide for a unique value, a nonce, and probabilistic MACs, like `RMAC` [JJV02] and `EHtM` [Min10], require some entropy to provide for a random value, a salt. On the other hand, deterministic MACs, like `CBC-MAC` [FIPS113; BKR00] and its variants, are the easiest to use in practice which explains their popularity.

However, a deterministic MAC scheme with beyond-birthday-bound security is far from trivial to build. In fact, we saw in Section 3.1.2 a generic attack by Preneel and van Oorschot [Pv95] that works on all deterministic iterated MACs using internal state collisions. Therefore, a deterministic MAC with an n-bit state cannot reach beyond birthday-bound security.

All the proofs mentioned in this chapter deal with the prf security notion (Definition 1.3) meaning that all Double-block Hash-then-Sum MACs we introduce are secure as PRFs. Although our attacks are described as forgery attacks, they can also be seen as distinguishers in the prf security game.

5.1.1 Generic Design

Increasing the State Size. To bypass the limitation of iterated deterministic MACs, one has to increase the internal state size of the mode. Double-block Hash-then-Sum MACs double it from n to $2n$ so that the generic attack of [Pv95] runs in $\mathcal{O}(2^n)$ complexity which is no better than a random guessing of the tag.

More precisely the internal state is divided into two n-bit parts that we denote Σ and Θ, and the final MAC is computed as:

$$\mathrm{MAC}(M) = E(\Sigma(M)) \oplus E'(\Theta(M)),$$

where E and E' denote a block cipher with potentially different keys. The functions Σ and Θ are typically two n-bit almost-xor universal (AXU, Definition 2.2) hash functions computed on the message, hence the name Double-block Hash-then-Sum MAC.

Yasuda [Yas10] was the first to propose and prove a construction following this design strategy with `SUM-ECBC`. The `SUM-ECBC` MAC came

with a proof guaranteeing security up to $\mathcal{O}(2^{2n/3})$ short queries which indeed is beyond the birthday bound.

A Prolific Research Topic. Following that there have been several works on the topic of building deterministic MACs secure beyond the birthday bound. Yasuda himself proposed `PMAC+` [Yas11] which is a BBB MAC achieving rate 1 meaning that it uses a single block cipher call per block of message while `SUM-ECBC` is only rate 1/2 (two block cipher calls per message block).

Later works proposed designs optimizing various aspects. We will study `3kf9` [Zha+12] that removes the needs for Galois Field doubling, `1kPMAC+` [Dat+17] and `1kf9` [Dat+15] that only use a single secret key, `LightMAC+` [Nai17] and the authentication part of `GCM-SIV2` [IM16]. All of them came with a proof of security up to $\mathcal{O}(2^{2n/3})$ short queries.

Working on Better Proofs. There has been work to unify and improve the initial proofs. Datta, Dutta, Nandi and Paul [Dat+18] unified the design strategy under the Double-block Hash-then-Sum paradigm and proposed an improved proof mostly improving the security bound for long queries. In particular, the security bound for short queries was still at $\mathcal{O}(2^{2n/3})$ until Kim, Lee and Lee [KLL20] later proved that the Double-block Hash-then-Sum paradigm is secure up to $\mathcal{O}(2^{3n/4})$ short queries. However, the bound of [KLL20] does not always dominate the one of [Dat+18] notably for long queries.

Nevertheless, the proofs are quite technical and require attention. For instance `1kf9` was withdrawn due to an issue with its proof, and we provide in Section 5.3.1 a birthday bound attack showing that the proof cannot be repaired. The original proof of `3kf9` was shown to be flawed and repaired by [Dat+18]. Finally, an improved proof for `LightMAC+` by Naito [Nai18] showed an advantage upper-bound that should make any attack using $\mathcal{O}(2^{3n/4})$ short queries and a single verification query impossible. Therefore, the existence of our attack contradicts the proof and, when contacted, the author recognized that there was a flaw in its proof.

5.1.2 Generic Attack

We present a generic strategy that leads to a cryptanalysis on all known Double-block Hash-then-Sum authentication modes in data complexity $\mathcal{O}(2^{3n/4})$ given an iterated and deterministic construction of the hash functions.

4-way Relation. Remember that a Double-block Hash-then-Sum MAC is composed of two n-bit AXU hash $\Sigma(\cdot)$ and $\Theta(\cdot)$ and is computed as:

$$\mathrm{MAC}(M) = E\big(\Sigma(M)\big) \oplus E'\big(\Theta(M)\big) \ .$$

Our generic strategy to mount forgery attacks consists in looking for a quadruple of messages (X, Y, Z, T) such that pairs of values collide for one half of the state. More precisely, we look for quadruples satisfying a relation $\mathcal{R}(X, Y, Z, T)$ defined as:

$$\mathcal{R}(X, Y, Z, T) := \begin{cases} \Sigma(X) = \Sigma(Y) \\ \Theta(Y) = \Theta(Z) \\ \Sigma(Z) = \Sigma(T) \\ \Theta(T) = \Theta(X) \end{cases}$$

which directly implies that:

$$\mathcal{R}(X, Y, Z, T) \implies \mathrm{MAC}(X) \oplus \mathrm{MAC}(Y) \oplus \mathrm{MAC}(Z) \oplus \mathrm{MAC}(T) = 0. \tag{5.1}$$

Constructing Messages. We build two-block messages using two different injection functions ϕ and ψ. The generic strategy consists in adding two distinct prefixes and letting the suffix free as for example:

$$\begin{aligned} \phi(i) &= 0 \,\|\, i & \psi(i) &= 1 \,\|\, i \\ X = \phi(x) &= 0 \,\|\, x & Y = \psi(y) &= 1 \,\|\, y \\ Z = \phi(z) &= 0 \,\|\, z & T = \psi(t) &= 1 \,\|\, t \ . \end{aligned}$$

It is important to interleave the injection functions so that all pairs of messages that need to collide on a state part, that are $(X, Y), (Y, Z), (Z, T)$ and (T, X) always involve both ϕ and ψ. Hence, a collision can occur in Σ or Θ when the second block cancels the difference induced by the first block.

Reducing the Linear System. What makes this attack possible is the fact that the messages X, Y, Z, T are constructed in such a way that the relation $\mathcal{R}$ is implied by a linear system of rank three. Concretely we design the attack such that:

$$\big[\Sigma(X) = \Sigma(Y) \text{ and } \Theta(Y) = \Theta(Z) \text{ and } \Sigma(Z) = \Sigma(T)\big] \Rightarrow \Theta(T) = \Theta(X).$$

Therefore, taken randomly, at least one quadruple out of 2^{3n} is expected to satisfy $\mathcal{R}$ Notice that 2^{3n} quadruples can be built with $4 \cdot 2^{3n/4}$ queries thus we can find a quadruple (X, Y, Z, T) satisfying $\mathcal{R}$ with an $\mathcal{O}(2^{3n/4})$ data complexity.

This part of the attack is the least generic in the sense that one has to look at the construction to verify that indeed good quadruples appear more often than expected. However, every MAC considered has this property.

Building a Forgery. In the very same way that the generic attack of Preneel and van Oorschot [Pv95] exploited the Collision Extension Property (Definition 3.4) we can define the 4-way Collision Extension Property as:

Definition 5.1 (4-way Collision Extension Property)**.** We consider a MAC following the Double-block Hash-then-Sum paradigm. Let $\Sigma(m)$ and $\Theta(m)$ be the two n-bit internal state after having processed m. We define the 4-way collision extension property as:

$$\mathcal{R}(X, Y, Z, T) \implies \mathcal{R}(X \parallel s, Y \parallel s, Z \parallel s, T \parallel s)$$

for any messages X, Y, Z, T and any suffix s where:

$$\mathcal{R}(X, Y, Z, T) := \begin{cases} \Sigma(X) = \Sigma(Y) \\ \Theta(Y) = \Theta(Z) \\ \Sigma(Z) = \Sigma(T) \\ \Theta(T) = \Theta(X) \end{cases}$$

Thus, with a proper quadruple (X, Y, Z, T) satisfying $\mathcal{R}$ we deduce a related quadruple $(X \parallel s, Y \parallel s, Z \parallel s, T \parallel s)$ also satisfying $\mathcal{R}$. Hence, we predict that $\mathrm{MAC}(X \parallel s) = \mathrm{MAC}(Y \parallel s) \oplus \mathrm{MAC}(Z \parallel s) \oplus \mathrm{MAC}(T \parallel s)$ so that asking for the tag of three of them allows us to forge a valid tag for the fourth one.

This 4-way Collision Extension Property really applies when the two part Σ and Θ are independently built as deterministic iterated functions just like in the case of MACs. More precisely, when appending a block s, the value $\Sigma(m \parallel s)$ (resp. $\Theta(m \parallel s)$) must only depends on $\Sigma(m)$ and s (resp. $\Theta(m)$ and s). For instance this is the case for MACs `SUM-ECBC`, `GCM-SIV2` and `PMAC+` but not for `3kf9`. We thus adapt or improve our attack for every considered modes in their respective sections.

Detecting Quadruple. As we look for a good quadruple, the relation $\mathrm{MAC}(X) \oplus \mathrm{MAC}(Y) \oplus \mathrm{MAC}(Z) \oplus \mathrm{MAC}(T) = 0$ in itself is too weak because we expect one quadruple out of 2^n to satisfy it randomly. However, we can usually amplify the filtering using related quadruples that satisfy $\mathcal{R}$ simultaneously, for instance exploiting the 4-way collision extension, Definition 5.1. Amplifying to a $3n$ bit filter will be sufficient as, then, only a single quadruple is expected to satisfy it randomly.

Therefore, in most of our attacks, detecting a quadruple is done by solving an instance of the 4-XOR problem with word size $w = 3n$. In the optic of an information theoretic attack that matches the proof, the data complexity is the most important, if not the only important, parameter to optimize. We saw multiple algorithms for the 4-XOR problem in Section 3.2.1, but only the memory efficient algorithm [CJM02] is suitable as it is also data efficient. Indeed, Wagner's algorithm [Wag02] as well as Nikolic and Sasaki's algorithm [NS15] would need $2^{w/3}$ queries or more which in our case makes for a 2^n data complexity, no better than random tag guessing.

This is why our attack has a data, time and memory complexities of $\mathcal{O}(2^{3n/4}), \mathcal{O}(2^{3n/2})$ and $\mathcal{O}(2^{3n/4})$ respectively. However, variants of our attack exploiting the actual structure of the MAC to get a time complexity below 2^n are possible at least for `SUM-ECBC` and `GCM-SIV2` where we describe an attack using $\mathcal{O}(2^{6n/7})$ data, time and memory.

5.2 Application to concrete MACs

In this section we go through authentication modes `SUM-ECBC`, `GCM-SIV2`, `PMAC+`, `LightMAC+`, `3kf9` and `1kPMAC+`. We give details on how to apply our generic strategy of Section 5.1.2 to build a forgery attack.

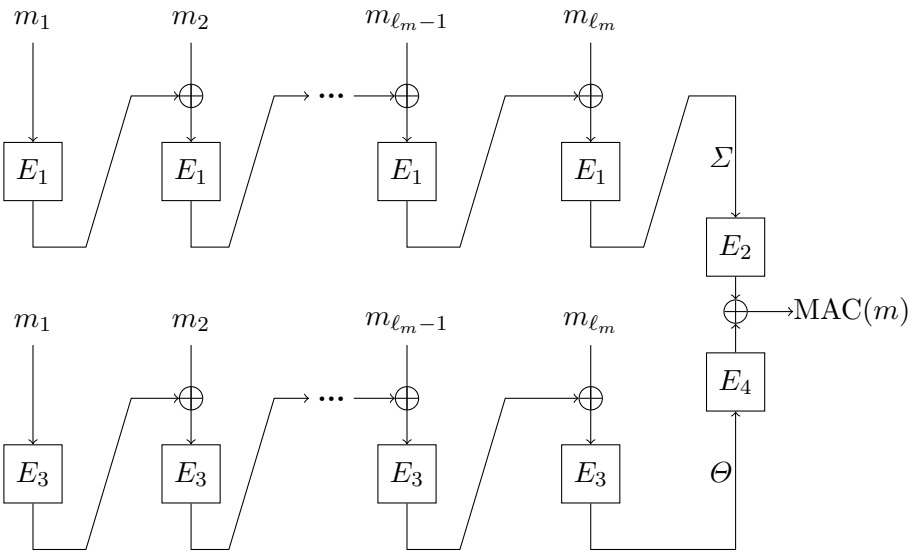


Figure 5.1: Diagram for `SUM-ECBC` with a ℓ_m-block message.

5.2.1 Attacking `SUM-ECBC`

`SUM-ECBC` was designed by Yasuda in 2010 [Yas10], inspired by MAC constructions summing two CBC-MACs in the ISO 9797-1 standard. The scheme uses a block cipher keyed with four independent keys, denoted as E_1, E_2, E_3, E_4. The message m is first padded with 10^* padding, and divided into n-bit blocks. In the following we ignore the padding and consider the padded message as the input: this makes our description easier, and any padded message whose last block is non-zero can be "un-padded" to generate a valid input message. The construction is defined as follows (see also Figure 5.1):

$$\begin{aligned} \Sigma(m) &= \sigma_{\ell_m} & \sigma_0 &= 0 \quad \sigma_i = E_1(\sigma_{i-1} \oplus m_i) \\ \Theta(m) &= \theta_{\ell_m} & \theta_0 &= 0 \quad \theta_i = E_3(\theta_{i-1} \oplus m_i) \\ \mathrm{MAC}(m) &= E_2(\Sigma(m)) \oplus E_4(\Theta(m)) \end{aligned}$$

Mounting the Attack. Following the framework of Section 5.1.2, we consider quadruple of messages built with two message injection functions:

$$\phi(i) = 0 \,\|\, i \qquad\qquad \psi(i) = 1 \,\|\, i$$

In particular, we have

$$\mathrm{MAC}(\phi(i)) = E_2\Big(\underbrace{E_1(i \oplus E_1(0))}_{\Sigma_0(i)}\Big) \oplus E_4\Big(\underbrace{E_3(i \oplus E_3(0))}_{\Theta_0(i)}\Big)$$
$$\mathrm{MAC}(\psi(i)) = E_2\Big(\underbrace{E_1(i \oplus E_1(1))}_{\Sigma_1(i)}\Big) \oplus E_4\Big(\underbrace{E_3(i \oplus E_3(1))}_{\Theta_1(i)}\Big)$$

Next, we build quadruples of messages X, Y, Z, T with

$$X = \phi(x) \qquad Y = \psi(y) \qquad Z = \phi(z) \qquad T = \psi(t),$$

and we look for a quadruple with partial state collisions for the underlying pairs, that is a quadruple following the relation:

$$\mathcal{R}(x,y,z,t) := \begin{cases} \Sigma_0(x) = \Sigma_1(y) \\ \Sigma_0(z) = \Sigma_1(t) \\ \Theta_0(z) = \Theta_1(y) \\ \Theta_0(x) = \Theta_1(t). \end{cases}$$

We have

$$\mathcal{R}(x,y,z,t) \Leftrightarrow \begin{cases} x \oplus E_1(0) = y \oplus E_1(1) \\ z \oplus E_3(0) = y \oplus E_3(1) \\ z \oplus E_1(0) = t \oplus E_1(1) \\ x \oplus E_3(0) = t \oplus E_3(1) \end{cases} \Leftrightarrow \begin{cases} x \oplus y \oplus z \oplus t = 0 \\ x \oplus y = E_1(0) \oplus E_1(1) \\ x \oplus t = E_3(0) \oplus E_3(1) \end{cases}$$

As promised in Section 5.1.2, $\mathcal{R}$ defines a $3n$-bit relation. We can easily observe when $x \oplus y \oplus z \oplus t = 0$, and we can also detect the relation on the sum of the MACs following Equation (5.1):

$$\mathcal{R}(x,y,z,t) \Rightarrow \mathrm{MAC}(\phi(x)) \oplus \mathrm{MAC}(\psi(y)) \oplus \mathrm{MAC}(\phi(z)) \oplus \mathrm{MAC}(\psi(t)) = 0$$

Moreover, we observe that $\mathcal{R}(x,y,z,t)$ is satisfied if and only if $\mathcal{R}(x \oplus c, y \oplus c, z \oplus c, t \oplus c)$ is satisfied for any constant c. We use this relation to build several related quadruples that satisfy $\mathcal{R}$ simultaneously:

$$\mathcal{R}(x,y,z,t) \iff \forall c,\ \mathcal{R}(x \oplus c, y \oplus c, z \oplus c, t \oplus c) \tag{5.2}$$

This leads to an attack with $\mathcal{O}(2^{3n/4})$ queries: we consider four sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{T}$ of $2^{3n/4}$ values, and we look for a quadruple $(x,y,z,t) \in \mathcal{X} \times$

$\mathcal{Y} \times \mathcal{Z} \times \mathcal{T}$ with:

$$\begin{cases} x \oplus y \oplus z \oplus t = 0 \\ \mathrm{MAC}(\phi(x)) \oplus \mathrm{MAC}(\psi(y)) \oplus \mathrm{MAC}(\phi(z)) \oplus \mathrm{MAC}(\psi(t)) = 0 \\ \mathrm{MAC}(\phi(x \oplus 1)) \oplus \mathrm{MAC}(\psi(y \oplus 1)) \oplus \mathrm{MAC}(\phi(z \oplus 1)) \oplus \mathrm{MAC}(\psi(t \oplus 1)) = 0. \end{cases} \tag{5.3}$$

Because we need a fair distribution of values $x \oplus y$ and $x \oplus t$ to find the good quadruple, we build the sets as:

$$\mathcal{X} = \{x \in \{0,1\}^n \ : \ x_{[0:n/4]} = 0\} \qquad \mathcal{Y} = \{x \in \{0,1\}^n \ : \ x_{[n/4:n/2]} = 0\}$$
$$\mathcal{Z} = \{x \in \{0,1\}^n \ : \ x_{[n/2:3n/4]} = 0\} \quad \mathcal{T} = \{x \in \{0,1\}^n \ : \ x_{[3n/4:n]} = 0\}$$

With this construction, there is exactly one quadruple $(x, y, z, t) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \times \mathcal{T}$ that satisfies $\mathcal{R}$, given by:

$$x = v_1|w_2|u_3|0 \quad y = w_1|v_2|0|u_4 \quad z = u_1|0|v_3|w_4 \quad t = 0|u_2|w_3|v_4,$$

where:

$$E_1(0) \oplus E_1(1) =: u_1|u_2|u_3|u_4$$
$$E_3(0) \oplus E_3(1) =: v_1|v_2|v_3|v_4$$
$$E_1(0) \oplus E_1(1) \oplus E_3(0) \oplus E_3(1) =: w_1|w_2|w_3|w_4.$$

We expect on average one random quadruple satisfying (5.3) (with 2^{3n} potential quadruples, and a $3n$-bit filtering), in addition to the quadruple satisfying $\mathcal{R}$. The correct quadruple can easily be checked with a few extra queries.

Concretely, we solve the generalized birthday algorithms with the memory efficient algorithm of Section 3.2.1 in order to optimize the query complexity of the attack. We consider four lists:

$$L_1 = \{x \parallel \mathrm{MAC}(\phi(x)) \parallel \mathrm{MAC}(\phi(x \oplus 1)) \ : \ x \in \mathcal{X}\}$$
$$L_2 = \{y \parallel \mathrm{MAC}(\psi(y)) \parallel \mathrm{MAC}(\psi(y \oplus 1)) \ : \ y \in \mathcal{Y}\}$$
$$L_3 = \{z \parallel \mathrm{MAC}(\phi(z)) \parallel \mathrm{MAC}(\phi(z \oplus 1)) \ : \ z \in \mathcal{Z}\}$$
$$L_4 = \{t \parallel \mathrm{MAC}(\psi(t)) \parallel \mathrm{MAC}(\psi(t \oplus 1)) \ : \ t \in \mathcal{T}\}$$

Notice that we can build those lists with $5 \cdot 2^{3n/4}$ queries as, by construction, for any element i of $\mathcal{Y}, \mathcal{Z}, \mathcal{T}$ the element $(i \oplus 1)$ also belongs to $\mathcal{Y}, \mathcal{Z}, \mathcal{T}$, respectively.

We solve the 4-XOR problem to find: $(x, y, z, t) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \times \mathcal{T}$ such that $L_1[x] \oplus L_2[y] \oplus L_3[z] \oplus L_4[t] = 0$ with $\widetilde{\mathcal{O}}(2^{3n/2})$ operations, using a

memory of size $\mathcal{O}(2^{3n/4})$. After finding a collision, we verify that it is not a false positive by testing the relation for another value c. As there are on average $\mathcal{O}(1)$ random quadruples, the attack is indeed using a total of $5 \cdot 2^{3n/4} + \mathcal{O}(1) = \mathcal{O}(2^{3n/4})$ queries.

Universal Forgeries. This attack can be extended to a universal forgery. Indeed, the fixed prefix 0 and 1 can be replaced by v and $v \oplus 1$ for any block v, and when we identify a right quadruple (x, y, z, t) we deduce the values $\Delta_1 = E_1(v) \oplus E_1(v \oplus 1)$ and $\Delta_3 = E_3(v) \oplus E_3(v \oplus 1)$. For the end of the message we can use the 4-way collision extension property (Definition 5.1) which implies that: if (x, y, z, t) is a right quadruple, then $\mathrm{MAC}(v \| x \| s) \oplus \mathrm{MAC}(v \oplus 1 \| y \| s) \oplus \mathrm{MAC}(v \| z \| s) \oplus \mathrm{MAC}(v \oplus 1 \| t \| s) = 0$ for any suffix s.

Therefore, if we want to forge a MAC for any message m of size $\ell_m \geq 2$ blocks we parse it as $m = v \| w \| s$ (where s has zero, one, or several blocks) and perform the attack to recover Δ_1 and Δ_3. Then, we can forge using the previous relation, and Equation (5.2):

$$\begin{aligned}\mathrm{MAC}(v \| w \| s) = &\mathrm{MAC}(v \oplus 1 \| w \oplus \Delta_1 \| s) \oplus \mathrm{MAC}(v \| w \oplus \Delta_3 \| s) \\ &\oplus \mathrm{MAC}(v \oplus 1 \| w \oplus \Delta_1 \oplus \Delta_3 \| s)\end{aligned}$$

Optimizing the time complexity. Equation (5.2) can also be used to reduce the time complexity below 2^n, at the cost of more oracle queries. Indeed, if we consider a subset $\mathcal{C}$ of $\{0,1\}^n$, we have:

$$\begin{aligned}\mathcal{R}(x, y, z, t) &\Leftrightarrow \forall c \in \mathcal{C}, \mathcal{R}(x \oplus c, y \oplus c, z \oplus c, t \oplus c) \\ &\Rightarrow \forall c \in \mathcal{C},\ \mathrm{MAC}(\phi(x \oplus c)) \oplus \mathrm{MAC}(\psi(y \oplus c)) \\ &\qquad \oplus \mathrm{MAC}(\phi(z \oplus c)) \oplus \mathrm{MAC}(\psi(t \oplus c)) = 0 \\ &\Rightarrow \bigoplus_{c \in \mathcal{C}} \mathrm{MAC}(\phi(x \oplus c)) \oplus \bigoplus_{c \in \mathcal{C}} \mathrm{MAC}(\psi(y \oplus c)) \\ &\qquad \oplus \bigoplus_{c \in \mathcal{C}} \mathrm{MAC}(\phi(z \oplus c)) \oplus \bigoplus_{c \in \mathcal{C}} \mathrm{MAC}(\psi(t \oplus c)) = 0\end{aligned} \tag{5.4}$$

If we select $\mathcal{C}$ as a linear subspace, the last expression does not depend on the full values (x, y, z, t) but only on their projection onto the space orthogonal to $\mathcal{C}$. Concretely, we use $\mathcal{C} = \{x : x_{[3n/7:n]} = 0\} = \{x : x < 2^{3n/7}\}$, so that the value $\bigoplus_{c \in \mathcal{C}} \mathrm{MAC}(\phi(x \oplus c))$ is independent of bits 0 to $3n/7 - 1$ of x.

Therefore, we consider the rewritten MAC function

$$\mathrm{MAC}'(v \,\|\, w) = \bigoplus_{c \in \mathcal{C}} \mathrm{MAC}(v \,\|\, w \oplus c),$$

the following message injections, with a $4n/7$-bit input

$$\phi'(i) = 0 \,\|\, i|0 \qquad\qquad \psi'(i) = 1 \,\|\, i|0,$$

and a reduced relation over $4n/7$-bit values:

$$\mathcal{R}'(x,y,z,t) := \begin{cases} x \oplus y = (E_1(0) \oplus E_1(1))_{[3n/7:n]} \\ y \oplus z = (E_3(0) \oplus E_3(1))_{[3n/7:n]} \\ z \oplus t = (E_1(0) \oplus E_1(1))_{[3n/7:n]} \\ t \oplus x = (E_3(0) \oplus E_3(1))_{[3n/7:n]} \end{cases}$$

$$\Leftrightarrow \begin{cases} x \oplus y \oplus z \oplus t = 0 \\ x \oplus y = (E_1(0) \oplus E_1(1))_{[3n/7:n]} \\ x \oplus t = (E_3(0) \oplus E_3(1))_{[3n/7:n]} \end{cases}$$

Thanks to Equation 5.4, we still have:

$$\mathcal{R}'(x,y,z,t) \implies \mathrm{MAC}'(\phi'(x)) \oplus \mathrm{MAC}'(\psi'(y)) \oplus \mathrm{MAC}'(\phi'(z)) \oplus \mathrm{MAC}'(\psi'(t)) = 0$$

Since the relation $\mathcal{R}'$ is now only a $12n/7$-bit condition, we can use shorter lists than before, with just $2^{3n/7}$ elements. We can also increase the filtering using the same trick as previously, considering the following lists:

$$L'_1 = \{x \,\|\, \mathrm{MAC}'(\phi'(x)) \,\|\, \mathrm{MAC}'(\phi'(x \oplus 1)) : x \in \{0,1\}^{4n/7},\ x_{[0:n/7]} = 0\}$$
$$L'_2 = \{y \,\|\, \mathrm{MAC}'(\psi'(y)) \,\|\, \mathrm{MAC}'(\psi'(y \oplus 1)) : y \in \{0,1\}^{4n/7},\ y_{[n/7:2n/7]} = 0\}$$
$$L'_3 = \{z \,\|\, \mathrm{MAC}'(\phi'(z)) \,\|\, \mathrm{MAC}'(\phi'(z \oplus 1)) : z \in \{0,1\}^{4n/7},\ z_{[2n/7:3n/7]} = 0\}$$
$$L'_4 = \{t \,\|\, \mathrm{MAC}'(\psi'(t)) \,\|\, \mathrm{MAC}'(\psi'(t \oplus 1)) : t \in \{0,1\}^{4n/7},\ t_{[3n/7:4n/7]} = 0\}$$

Finally, using the memory efficient algorithm of Section 3.2.1 with $\ell = 3n/7$ and $p = 0$, we can locate a right quadruple using $\widetilde{\mathcal{O}}(2^{6n/7})$ queries, $\widetilde{\mathcal{O}}(2^{6n/7})$ operations, and $\mathcal{O}(2^{3n/7})$ memory. This recovers only $4n/7$ bits of $E_1(0) \oplus E_1(1)$ and $E_3(0) \oplus E_3(1)$, but we can easily recover the remaining bits, either by brute force, or by repeating the attack with a different set $\mathcal{C}$.

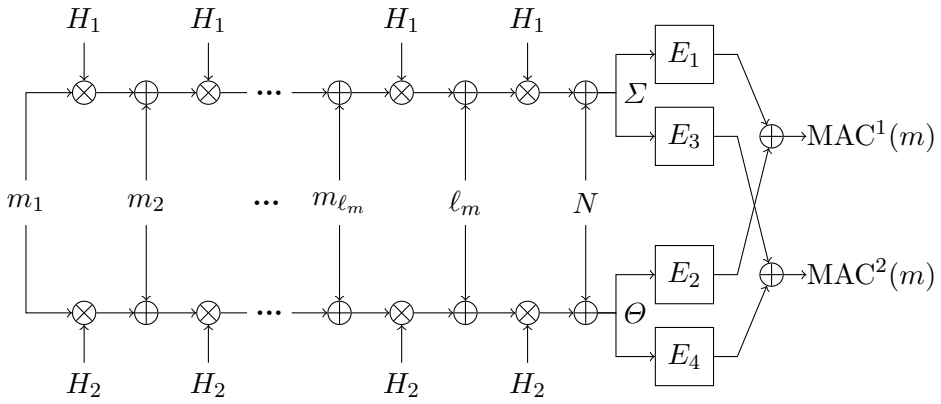


Figure 5.2: Diagram for authentication in `GCM-SIV2` using `GHASH` with a ℓ_m-block message, a nonce N, hash keys H_1 and H_2.

5.2.2 Attacking `GCM-SIV2`

`GCM-SIV2` is an authenticated encryption mode designed by Iwata and Minematsu [IM16] as a double-block hash-then-sum version of `GCM-SIV` (in the following, we consider `GCM-SIV2` with the Galois hash, `GHASH`, as the underlying universal hash function). For simplicity, we focus on the authentication part of `GCM-SIV2`, using inputs with a non-empty associated data, and an empty message. In this case, `GCM-SIV2` becomes a nonce-based MAC with a $2n$-bit output. The message m (considered as associated data for the mode) is zero-padded, divided into n-bit blocks, and the length is appended in an extra block. Then, the construction is defined as follows, with multiplication done in a finite field (see also Figure 5.2):

$$\begin{aligned}
\Sigma(N, m) &= N \oplus \ell_m \cdot H_1 \oplus \textstyle\bigoplus_{i=1}^{\ell_m} m_i \cdot H_1^{\ell_m+2-i} \\
\Theta(N, m) &= N \oplus \ell_m \cdot H_2 \oplus \textstyle\bigoplus_{i=1}^{\ell_m} m_i \cdot H_2^{\ell_m+2-i} \\
\mathrm{MAC}(N, m) &= E_1(\Sigma(m)) \oplus E_2(\Theta(m)) \,\|\, E_3(\Sigma(m)) \oplus E_4(\Theta(m))
\end{aligned}$$

Mounting the Attack. The structure of the authentication part of `GCM-SIV2` is essentially the same as the structure of `SUM-ECBC`, where the block cipher calls E_1 and E_3 are replaced by multiplication by H_1 and H_2. The finalization function has a $2n$-bit output $\mathrm{MAC}^1, \mathrm{MAC}^2$, but quadruples following $\mathcal{R}$ will collide on both outputs. Thus, we can essentially repeat the `SUM-ECBC` attack, but there is an important difference: `GCM-SIV2` is a

nonce-based MAC, rather than a deterministic one. Therefore, all queries must include a nonce N, and we should not query two different messages with the same nonce. We adapt the previous attack using message injection functions that output both a nonce and a message, so that we use two fixed messages, 0 and 1, with variable nonces:

$$\phi(i) = (i, 0) \qquad \psi(i) = (i, 1)$$

$$\begin{aligned}
\mathrm{MAC}(\phi(i)) &= \mathrm{MAC}(i, 0) \\
&= E_1(\underbrace{i \oplus H_1}_{\Sigma_0(i)}) \oplus E_2(\underbrace{i \oplus H_2}_{\Theta_0(i)}) \;\Big\|\; E_3(\Sigma_0(i)) \oplus E_4(\Theta_0(i)) \\
\mathrm{MAC}(\psi(i)) &= \mathrm{MAC}(i, 1) \\
&= E_1(\underbrace{i \oplus H_1 \oplus H_1^2}_{\Sigma_1(i)}) \oplus E_2(\underbrace{i \oplus H_2 \oplus H_2^2}_{\Theta_1(i)}) \;\Big\|\; E_3(\Sigma_1(i)) \oplus E_4(\Theta_1(i)).
\end{aligned}$$

We consider quadruples of nonce/messages X, Y, Z, T with

$$X = \phi(x) \qquad Y = \psi(y) \qquad Z = \phi(z) \qquad T = \psi(t),$$

and we have the same kind of relations as in the previous attack:

$$\mathcal{R}(x, y, z, t) := \begin{cases} \Sigma_0(x) = \Sigma_1(y) \\ \Sigma_0(z) = \Sigma_1(t) \\ \Theta_0(z) = \Theta_1(y) \\ \Theta_0(x) = \Theta_1(t). \end{cases} \Leftrightarrow \begin{cases} x \oplus y \oplus z \oplus t = 0 \\ x \oplus y = H_1^2 \\ x \oplus t = H_2^2 \end{cases}$$

$$\implies \mathrm{MAC}(\phi(x)) \oplus \mathrm{MAC}(\psi(y)) \oplus \mathrm{MAC}(\phi(z)) \oplus \mathrm{MAC}(\psi(t)) = 0\,.$$

Since the MAC output is $2n$-bit long, we can directly build an attack with $\mathcal{O}(2^{3n/4})$ queries: we consider four distinct sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{T}$ of $2^{3n/4}$ values, and we look for a quadruple $(x, y, z, t) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \times \mathcal{T}$, such that

$$\begin{cases} x \oplus y \oplus z \oplus t = 0 \\ \mathrm{MAC}(\phi(x)) \oplus \mathrm{MAC}(\psi(y)) \oplus \mathrm{MAC}(\phi(z)) \oplus \mathrm{MAC}(\psi(t)) = 0 \end{cases} \tag{5.5}$$

we expect to find one good quadruple that respects $\mathcal{R}$ along with $\mathcal{O}(1)$ quadruples that randomly satisfy the observable filter (5.5). This leads to an attack with $\mathcal{O}(2^{3n/4})$ queries and time $\widetilde{O}(2^{3n/2})$. Since we recover H_1 and H_2 (from $H_1^2 = x \oplus y$ and $H_2^2 = x \oplus t$), we can do universal forgeries. In addition, we can also easily adapt the attack with $\mathcal{O}(2^{6n/7})$ queries and time $\widetilde{O}(2^{6n/7})$.

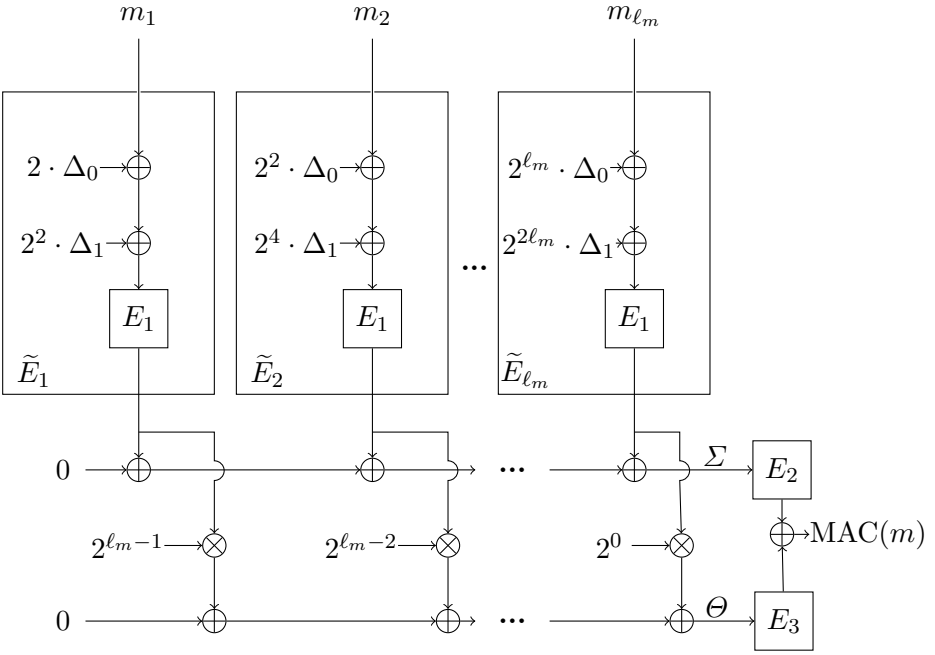


Figure 5.3: Diagram for PMAC+ with a ℓ_m-block message where $\Delta_0 = E_1(0)$ and $\Delta_1 = E_1(1)$.

5.2.3 Attacking PMAC+

PMAC+ was designed by Yasuda in 2011 [Yas11] based on PMAC [BR02] but with a larger internal state. The scheme internally uses a tweakable block cipher construction inspired by the XE construction [Rog04], that we denote as $\widetilde{E}_i$. The attack works independently of the construction, meaning that it works even if $\widetilde{E}_i$ is a true tweakable block cipher. The message m is first padded with 10^* padding, and divided into n-bit blocks, but for simplicity we ignore the padding in our description. The construction is shown in Figure 5.3[1]:

$$\Sigma(m) = \bigoplus_{i=1}^{\ell_m} \widetilde{E}_i(m_i) \qquad \widetilde{E}_i(x) = E_1(x \oplus 2^i \odot \Delta_0 \oplus 2^{2i} \odot \Delta_1)$$
$$\Theta(m) = \bigoplus_{i=1}^{\ell_m} 2^{\ell_m - i} \odot \widetilde{E}_i(m_i) \qquad \Delta_0 = E_1(0) \quad \Delta_1 = E_1(1)$$
$$\mathrm{MAC}(M) = E_2(\Sigma(m)) \oplus E_3(\Theta(m))$$

[1]The algorithm and the figure given in [Yas11] differ in the coefficients used to compute Θ. We use the algorithmic description because it matches later PMAC+ variants, but the attack can easily be adapted to the other case.

Mounting the Attack. As in the previous attack, we use message injection functions with two different prefixes, but we include an extra block u to define related quadruples:

$$\phi_u(i) = u \,\|\, 0 \,\|\, i \qquad\qquad \psi_u(i) = u \,\|\, 1 \,\|\, i$$

$$\begin{aligned}
\mathrm{MAC}(\phi_u(i)) &= \mathrm{MAC}(u \,\|\, 0 \,\|\, i) \\
&= E_2\Big(\underbrace{\widetilde{E}_1(u) \oplus \widetilde{E}_2(0) \oplus \widetilde{E}_3(i)}_{\Sigma_{u,0}(i)}\Big) \oplus E_3\Big(\underbrace{4\widetilde{E}_1(u) \oplus 2\widetilde{E}_2(0) \oplus \widetilde{E}_3(i)}_{\Theta_{u,0}(i)}\Big) \\
\mathrm{MAC}(\psi_u(i)) &= \mathrm{MAC}(u \,\|\, 1 \,\|\, i) \\
&= E_2\Big(\underbrace{\widetilde{E}_1(u) \oplus \widetilde{E}_2(1) \oplus \widetilde{E}_3(i)}_{\Sigma_{u,1}(i)}\Big) \oplus E_3\Big(\underbrace{4\widetilde{E}_1(u) \oplus 2\widetilde{E}_2(1) \oplus \widetilde{E}_3(i)}_{\Theta_{u,1}(i)}\Big).
\end{aligned}$$

Next, we build quadruples of messages X, Y, Z, T with

$$X = \phi_u(x) \qquad Y = \psi_u(y) \qquad Z = \phi_u(z) \qquad T = \psi_u(t),$$

and we look for a quadruple with partial state collisions for the underlying pairs, that is a quadruple following the relation:

$$\mathcal{R}(x,y,z,t) := \begin{cases} \Sigma_{u,0}(x) = \Sigma_{u,1}(y) \\ \Sigma_{u,0}(z) = \Sigma_{u,1}(t) \\ \Theta_{u,0}(z) = \Theta_{u,1}(y) \\ \Theta_{u,0}(x) = \Theta_{u,1}(t). \end{cases}$$

We have

$$\begin{aligned}
\mathcal{R}(x,y,z,t) \Leftrightarrow & \begin{cases} \widetilde{E}_3(x) \oplus \widetilde{E}_2(0) = \widetilde{E}_3(y) \oplus \widetilde{E}_2(1) \\ \widetilde{E}_3(z) \oplus \widetilde{E}_2(0) = \widetilde{E}_3(t) \oplus \widetilde{E}_2(1) \\ \widetilde{E}_3(y) \oplus 2\widetilde{E}_2(1) = \widetilde{E}_3(z) \oplus 2\widetilde{E}_2(0) \\ \widetilde{E}_3(t) \oplus 2\widetilde{E}_2(1) = \widetilde{E}_3(x) \oplus 2\widetilde{E}_2(0) \end{cases} \\
\Leftrightarrow & \begin{cases} \widetilde{E}_3(x) \oplus \widetilde{E}_3(y) \oplus \widetilde{E}_3(z) \oplus \widetilde{E}_3(t) = 0 \\ \widetilde{E}_3(x) \oplus \widetilde{E}_3(y) = \widetilde{E}_2(0) \oplus \widetilde{E}_2(1) \\ \widetilde{E}_3(t) \oplus \widetilde{E}_3(x) = 2\widetilde{E}_2(0) \oplus 2\widetilde{E}_2(1) \end{cases}
\end{aligned}$$

Again, $\mathcal{R}$ defines a $3n-$bit relation, and we can detect it through the sum of the MACs following Equation (5.1):

$$\mathcal{R}(x,y,z,t) \Rightarrow \mathrm{MAC}(\phi_u(x)) \oplus \mathrm{MAC}(\psi_u(y)) \oplus \mathrm{MAC}(\phi_u(z)) \oplus \mathrm{MAC}(\psi_u(t)) = 0$$

In addition, the relation $\mathcal{R}$ is independent of the value u, so that we can easily build several quadruples that satisfy $\mathcal{R}$ simultaneously. This leads to an attack with $\mathcal{O}(2^{3n/4})$ queries: we consider four sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{T}$ of $2^{3n/4}$ random values, and we look for a quadruple $(x, y, z, t) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \times \mathcal{T}$, such that

$$\forall u \in \{0, 1, 2\},$$
$$\mathrm{MAC}(\phi_u(x)) \oplus \mathrm{MAC}(\psi_u(y)) \oplus \mathrm{MAC}(\phi_u(z)) \oplus \mathrm{MAC}(\psi_u(t)) = 0 \ .$$

We expect on average one random quadruple (with 2^{3n} potential quadruples, and a $3n$-bit filtering), and one quadruple satisfying $\mathcal{R}$ (also a $3n$-bit condition). The correct quadruple can easily be checked with a few extra queries.

In practice, we use the memory efficient algorithm of Section 3.2.1 in order to optimize the complexity of the attack. We consider four lists:

$$\begin{aligned}
L_1 &= \{\mathrm{MAC}(\phi_0(x)) \,\|\, \mathrm{MAC}(\phi_1(x)) \,\|\, \mathrm{MAC}(\phi_2(x)) \ : \ x \in \mathcal{X}\} \\
L_2 &= \{\mathrm{MAC}(\psi_0(y)) \,\|\, \mathrm{MAC}(\psi_1(y)) \,\|\, \mathrm{MAC}(\psi_2(y)) \ : \ y \in \mathcal{Y}\} \\
L_3 &= \{\mathrm{MAC}(\phi_0(z)) \,\|\, \mathrm{MAC}(\phi_1(z)) \,\|\, \mathrm{MAC}(\phi_2(z)) \ : \ z \in \mathcal{Z}\} \\
L_4 &= \{\mathrm{MAC}(\psi_0(t)) \,\|\, \mathrm{MAC}(\psi_1(t)) \,\|\, \mathrm{MAC}(\psi_2(t)) \ : \ t \in \mathcal{T}\}
\end{aligned}$$

and we look for a quadruple $(x, y, z, t) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \times \mathcal{T}$ such that $L_1[x] \oplus L_2[y] \oplus L_3[z] \oplus L_4[t] = 0$. This can be done with $\tilde{\mathcal{O}}(2^{3n/2})$ operations, using a memory of size $\mathcal{O}(2^{3n/4})$. Finally, once a quadruple (x, y, z, t) satisfying $\mathcal{R}(x, y, z, t)$ has been detected, it can be used to generate forgeries. Indeed, we can predict the MAC of a new message by making three new queries using Equation (5.1):

$$\forall u, \ \mathrm{MAC}(\phi_u(x)) = \mathrm{MAC}(\psi_u(y)) \oplus \mathrm{MAC}(\psi_u(z)) \oplus \mathrm{MAC}(\phi_u(t))$$

Time-Query Trade-offs. As opposed to the `SUM-ECBC` attack, we don't have an analogue to Equation (5.2) that can be used to reduce the time complexity. However, the time complexity of the algorithm can be slightly reduced when using more than $\mathcal{O}(2^{3n/4})$ queries. If we consider sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{T}$ of size 2^ℓ with $3n/4 < \ell < n$, the resulting 4-XOR is slightly easier, because there are 2^{4t-3n} expected solutions. Using the memory efficient algorithm, this can be solved in time $\tilde{\mathcal{O}}(2^{3n-2\ell})$, using a memory of size $\mathcal{O}(2^\ell)$.

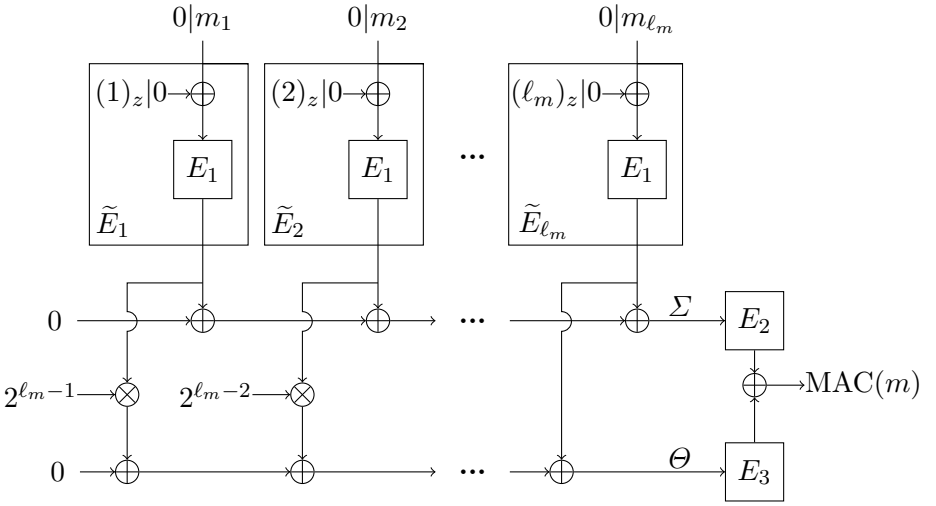


Figure 5.4: Diagram for `LightMAC+` with $(n-z)$-bit blocks of a ℓ_m-block message where $(v)_z$ is the value v written over z bits.

5.2.4 Attacking `LightMAC+`

`LightMAC+` was designed by Naito [Nai17] using ideas from `PMAC+` and `LightMAC`. If we consider `LightMAC+` as based on a tweakable block cipher $\widetilde{E}$, it follows the same structure as `PMAC+` (see Figure 5.4), but $\widetilde{E}$ takes a message block smaller than n bits:

$$\Sigma(m) = \bigoplus_{i=1}^{\ell_m} \widetilde{E}_i(m_i) \qquad \widetilde{E}_i(x) = E_1(i|x)$$
$$\Theta(m) = \bigoplus_{i=1}^{\ell_m} 2^{\ell_m - i} \odot \widetilde{E}_i(m_i)$$
$$\mathrm{MAC}(m) = E_2(\Sigma(m)) \oplus E_3(\Theta(m))$$

Mounting the Attack. Since our attack on `PMAC+` is independent of the way the tweakable block cipher is built, the same attack will be applicable to `LightMAC+`. The only difference from our point of view is that the message blocks are shorter than the block-size. As long as one message block is big enough to fit $2^{3n/4}$ different values, our attack will succeed.

This attack violates an improved security proof by Naito [Nai18], with a security bound of $\mathcal{O}(q_t^2 q_v / 2^{2n})$ (with q_t MAC queries and q_v verification queries). Indeed, our attack reaches a constant success probability with $q_t = \mathcal{O}(2^{3n/4})$ and $q_v = 1$. We have shared our attack with the author of the proof, and he agreed that his proof was flawed.

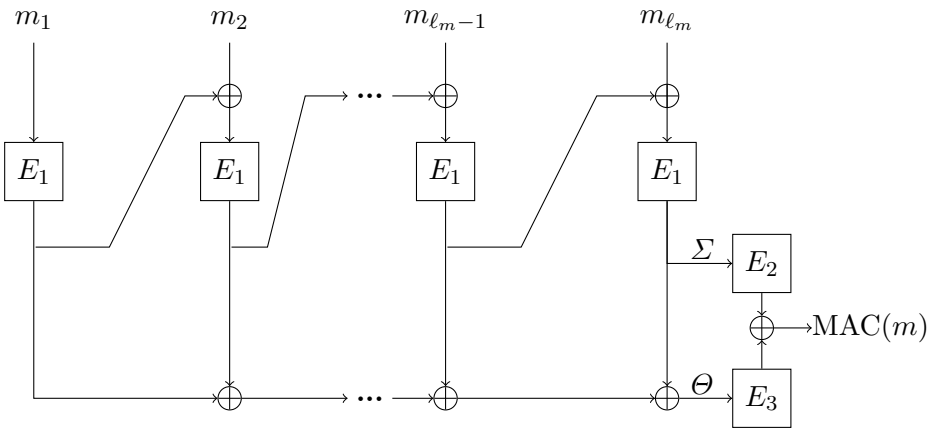


Figure 5.5: Diagram for `3kf9` with a ℓ_m-block message.

5.2.5 Attacking `3kf9`

Our third attack is applicable to `3kf9` [Zha+12] and similar constructions. We have a universal forgery attack with $\mathcal{O}(2^{3n/4})$ queries and $\widetilde{\mathcal{O}}(2^{5n/4})$ operations using memory $\mathcal{O}(2^n)$, with a possible time-memory trade-off.

`3kf9` [Zha+12], designed by Xhang, Wu, Sui and Wang, is a three-key variant of the `f9` mode used in 3G telephony. While the original `f9` does not have security beyond the birthday bound [KM03], `3kf9` is secure up to $2^{3n/4}$ queries [KLL20]. We describe `3kf9` in Figure 5.5:

$$\begin{aligned} \Sigma(m) &= \sigma_{\ell_m} & \sigma_0 &= 0 \quad \sigma_i = E_1(\sigma_{i-1} \oplus m_i) \\ \Theta(m) &= \textstyle\bigoplus_{i=1}^{\ell_m} \sigma_i \\ \mathrm{MAC}(m) &= E_2(\Sigma(m)) \oplus E_3(\Theta(m)) \end{aligned}$$

Mounting the Attack. Our attack follows the same structure as the previous attacks. We start with messages of the form:

$$\phi(i) = 0 \,\|\, i \qquad\qquad \psi(i) = 1 \,\|\, i,$$

and the corresponding MACs:

$$\mathrm{MAC}(\phi(i)) = E_2\Big(\underbrace{E_1(x \oplus E_1(0))}_{\Sigma_0(x)}\Big) \oplus E_3\Big(\underbrace{E_1(x \oplus E_1(0)) \oplus E_1(0)}_{\Theta_0(x)}\Big)$$
$$\mathrm{MAC}(\psi(i)) = E_2\Big(\underbrace{E_1(x \oplus E_1(1))}_{\Sigma_1(x)}\Big) \oplus E_3\Big(\underbrace{E_1(x \oplus E_1(1)) \oplus E_1(1)}_{\Theta_1(x)}\Big).$$

We use quadruples of messages X, Y, Z, T with

$$X = \phi(x) \qquad Y = \psi(y) \qquad Z = \phi(z) \qquad T = \psi(t),$$

and we look for a quadruple with partial state collisions for the underlying pairs, that is a quadruple following the relation:

$$\begin{aligned}
\mathcal{R}(x,y,z,t) := &\begin{cases} \Sigma_0(x) = \Sigma_1(y) \\ \Sigma_0(z) = \Sigma_1(t) \\ \Theta_0(z) = \Theta_1(y) \\ \Theta_0(x) = \Theta_1(t). \end{cases} \\
\Leftrightarrow &\begin{cases} x \oplus E_1(0) = y \oplus E_1(1) \\ z \oplus E_1(0) = t \oplus E_1(1) \\ E_1(z \oplus E_1(0)) \oplus E_1(0) = E_1(y \oplus E_1(1)) \oplus E_1(1) \\ E_1(x \oplus E_1(0)) \oplus E_1(0) = E_1(t \oplus E_1(1)) \oplus E_1(1) \end{cases} \\
\Leftrightarrow &\begin{cases} x \oplus y \oplus z \oplus t = 0 \\ x \oplus y = E_1(0) \oplus E_1(1) \\ E_1(x \oplus E_1(0)) \oplus E_1(t \oplus E_1(1)) = E_1(0) \oplus E_1(1) \end{cases} \\
\implies &\mathrm{MAC}(\phi(x)) \oplus \mathrm{MAC}(\psi(y)) \oplus \mathrm{MAC}(\phi(z)) \oplus \mathrm{MAC}(\psi(t)) = 0.
\end{aligned}$$

As in the previous attacks, $\mathcal{R}$ defines a $3n$-bit relation. Moreover, we can easily observe when $x \oplus y \oplus z \oplus t = 0$, and the relation $x \oplus y = E_1(0) \oplus E_1(1)$ can be verified across several quadruples. We don't have related quadruples satisfying $\mathcal{R}$ simultaneously as in the previous attacks, but we can use those properties to detect right quadruples. This leads to an attack with $\widetilde{\mathcal{O}}(2^{3n/4})$ queries: we consider four sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{T}$ of $\sqrt[4]{n} \times 2^{3n/4}$ random values, and we look for quadruples $(x, y, z, t) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \times \mathcal{T}$, such that:

$$\begin{cases} x \oplus y \oplus z \oplus t = 0 \\ \mathrm{MAC}(\phi(x)) \oplus \mathrm{MAC}(\psi(y)) \oplus \mathrm{MAC}(\phi(z)) \oplus \mathrm{MAC}(\psi(t)) = 0. \end{cases} \tag{5.6}$$

Since this is a $2n$-bit condition, we expect on average $n \cdot 2^n$ quadruples (x, y, z, t) satisfying (5.6). In order to filter out the right ones, we look at the value $x \oplus y$ for all these quadruples. While the wrong quadruples should have a random $x \oplus y$, the right ones have $x \oplus y = E_1(0) \oplus E_1(1)$. Therefore, with high probability, the most frequent value for $x \oplus y$ is equal to $E_1(0) \oplus E_1(1)$, and quadruples satisfying this extra relation are right quadruples with probability $1/2$. More precisely, we expect on average n wrong quadruples for each value of $x \oplus y$, and n right quadruples with $x \oplus y = E_1(0) \oplus E_1(1)$.

Success Probability. We want to prove the claim that one will need $\mathcal{O}(n \cdot 2^n)$ quadruples in order to find a right quadruple for `3kf9`. We say the attack finishes when we recover the target value $T = E(0) \oplus E(1)$.

Assuming each quadruple found respects $\mathcal{R}$ with probability $1/2^n$, we fill a list of counters with every suspected values of T; a random quadruples gives two random values and a right one gives one value equal to T and one random value. Therefore, we sum up the distribution of an observable value v as:

$$v \begin{cases} \xleftarrow{\$} \{0,1\}^n & \text{with probability } 1 - 1/2^{n+1} \\ \longleftarrow T & \text{with probability } 1/2^{n+1} \end{cases}$$

Let N be the number of observed values, and X_i^c represents the indicator that the i^{th} value equals c (following a Bernoulli distribution), so that the counter corresponding to c is $X^c = \sum_{i=1}^{N} X_i^c$. Now we have to discriminate between the distributions of X^c with $c \neq T$, and the distribution of X^T:

$$\begin{aligned} \Pr(X_i^T = 1) = \Pr(x = T) &= (1 - 1/2^{n+1})/2^n + 1/2^{n+1} \\ &= (3/2 - 1/2^{n+1})/2^n \\ &\implies \mathbf{E}[X^T] = N(3/2 - 1/2^{n+1})/2^n \\ \Pr(X_i^c = 1) = \Pr(x = c) &= (1 - 1/2^{n+1})/2^n \\ &\implies \mathbf{E}[X^c] = N(1 - 1/2^{n+1})/2^n \\ &\implies \mathbf{E}[X^T] \geq 3/2 \cdot \mathbf{E}[X^c] \end{aligned}$$

We use the Chernoff bound to get a lower bound on the probability

that a given counter is higher than the average value of X^T:

$$\begin{aligned}\Pr(X^c \geq \mathbf{E}[X^T]) &\leq Pr(X^c \geq 3/2 \cdot \mathbf{E}[X^c]) \\ &\leq e^{-N(1-1/2^{n+1})/2^{n+1}}\end{aligned}$$

and assuming the counters are independent:

$$\begin{aligned}Pr(X^c < \mathbf{E}[X^T]) &\geq 1 - e^{-N(1-1/2^{n+1})/2^{n+1}} \\ Pr(\forall c \neq T : X^c < \mathbf{E}[X^T]) &\geq (1 - e^{-N(1-1/2^{n+1})/2^{n+1}})^{2^n}\end{aligned}$$

This expression will asymptotically converge to a strictly positive constant when $e^{-N(1-1/2^{n+1})/2^{n+1}} = \Theta 2^{-n}$. Therefore, we use

$$N \simeq n \ln(2) \cdot \frac{2^{n+1}}{(1 - 1/2^{n+1})} = \mathcal{O}(n \cdot 2^n).$$

Since we observe 2 values per quadruples, this makes $\mathcal{O}(n \cdot 2^n)$ quadruples. Moreover, the event '$X^T \geq \mathbf{E}[X^T]$' has a probability close to 0.5, therefore after $\mathcal{O}(n \cdot 2^n)$ quadruples, we indeed have a $\Omega(1)$ probability that X^T is greater than all the other counters, which allows to recover the value T. Performing the attack with probability $\Omega(1)$ thus requires $\mathcal{O}(n \cdot 2^n)$ quadruples.

To get to this result some assumptions have been made, like the independence of the counters, but they all tend to be either conservative or asymptotically true.

Optimizing the time complexity. While the memory efficient algorithm of Section 3.2.1 would take time $\widetilde{\mathcal{O}}(2^{3n/2})$ with $\widetilde{\mathcal{O}}(2^{3n/4})$ queries, we can reduce the time complexity using sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{T}$ with some structure. More precisely, we use:

$$\begin{aligned}\mathcal{X} = \mathcal{Z} &= \left\{x \in \{0,1\}^n \ : \ x_{[0:n/4]} = 0\right\} \\ \mathcal{Y} = \mathcal{T} &= \left\{x \in \{0,1\}^n \ : \ x_{[n/4:n/2]} = 0\right\}\end{aligned}$$

so that quadruples can be written as

$$\begin{aligned}x =: x_3|x_2|x_1|0 \in \mathcal{X} \qquad & y =: y_3|y_2|0|y_0 \in \mathcal{Y} \\ z =: z_3|z_2|z_1|0 \in \mathcal{Z} \qquad & t =: t_3|t_2|0|t_0 \in \mathcal{T}.\end{aligned}$$

In particular, right quadruples satisfy $x \oplus y \oplus z \oplus t = 0$, therefore $x_1 = z_1$, $y_0 = t_0$, and $x_3|x_2 \oplus z_3|z_2 = y_3|y_2 \oplus t_3|t_2$. We use these properties to adapt the 4-XOR algorithm and locate the quadruples efficiently. First, we guess the $n/2$-bit value $\alpha_3|\alpha_2 := x_3|x_2 \oplus z_3|z_2 = y_3|y_2 \oplus t_3 t_3$. Then, for each $x = x_3|x_2|x_1|0$, there is a single candidate $z = (x_3 \oplus \alpha_3)|(x_2 \oplus \alpha_2)|x_1|0$ that could be part of a right quadruple. Similarly, every $y = y_3|y_2|0|y_0$ can be paired with a single $t = (y_3 \oplus \alpha_3)|(y_2 \oplus \alpha_2)|0|y_0$. Therefore, we consider the two following lists:

$$
\begin{aligned}
&L_1 := \\
&\{\mathrm{MAC}(\phi(x_3|x_2|x_1|0)) \oplus \mathrm{MAC}((x_3 \oplus \alpha_3)|(x_2 \oplus \alpha_2)|x_1|0) : x_3|x_2|x_1|0 \in \mathcal{X}\} \\
&L_2 := \\
&\{\mathrm{MAC}(\phi(y_3|y_2|0|y_0)) \oplus \mathrm{MAC}((y_3 \oplus \alpha_3)|(y_2 \oplus \alpha_2)|0|y_0) : y_3|y_2|0|y_0 \in \mathcal{Y}\}
\end{aligned}
$$

After sorting the lists, we look for matches, and the corresponding quadruples x, y, z, t are exactly the quadruples satisfying

$$
\begin{cases}
x \oplus y \oplus z \oplus t = 0 \\
(x \oplus z)_{[n/2:n]} = \alpha_3|\alpha_2 \\
\mathrm{MAC}(\phi(x)) \oplus \mathrm{MAC}(\psi(y)) \oplus \mathrm{MAC}(\phi(z)) \oplus \mathrm{MAC}(\psi(t)) = 0.
\end{cases} \tag{5.7}
$$

More precisely, a match $L_1[x] = L_2[y]$ suggests $z = x \oplus \alpha_3|\alpha_2|0|0$ and $t = y \oplus \alpha_3|\alpha_2|0|0$, but there are four corresponding quadruples: (x, y, z, t), (z, y, x, t), (x, t, z, y), (z, t, x, y), and two candidate values for $E_1(0) \oplus E_1(1)$: $x \oplus y$ and $x \oplus y \oplus \alpha_3|\alpha_2|0|0$.

We need $\widetilde{\mathcal{O}}(2^{3n/4})$ operations to generate those quadruples. We repeat this $2^{n/2}$ times to exhaust all $n/2$-bit values $\alpha_3|\alpha_2$ and generate all quadruples satisfying (5.6). Finally, we use an array to count the number of occurrences of each possible value of $x \oplus y$. Each counter receives on average two values, but the counter corresponding to $E_1(0) \oplus E_1(1)$ will receive three values on average. After repeating all the operations $\mathcal{O}(n)$ times, with some arbitrary constants in place of the zero bits, the highest counter corresponds to $E_1(0) \oplus E_1(1)$ with high probability, as we'll detail later. This gives an attack with $\widetilde{\mathcal{O}}(2^{3n/4})$ queries, $\widetilde{\mathcal{O}}(2^{5n/4})$ operations, and $\mathcal{O}(2^n)$ memory[2].

[2] We can actually reduce the polynomial factors by fixing only $(n - \log_2(n))/4$ bits to zero, in order to have sets of size $\sqrt[4]{n} \cdot 2^{3n/4}$.

Time-Memory Trade-offs. We can reduce the memory usage if we store only a subset of the counters, and repeat the whole algorithm until the whole set has been covered. Concretely, we store only the counters with a fixed value for bits $[0 : n/8]$ and $[n/4 : 3n/8]$ of $x \oplus y$. Because of the way the lists L_1 and L_2 are constructed, we have actually fixed $n/8$ bits of y_0 and x_1, and we can reduce the lists to size $2^{5n/8}$. Therefore, we evaluate $2^{3n/4}$ counters in time $\widetilde{\mathcal{O}}(2^{n/2} \cdot 2^{5n/8})$, using only $\mathcal{O}(2^{3n/4})$ memory. We repeat iteratively over the full counter set, so we need time $\widetilde{\mathcal{O}}(2^{n/4} \cdot 2^{n/2} \cdot 2^{5n/8}) = \widetilde{\mathcal{O}}(2^{11n/8})$. Generally, we have a time-memory trade-off with time $\widetilde{\mathcal{O}}(2^{5n/4+t/2})$ and memory $\mathcal{O}(2^{n-t})$ for $0 < t < n/4$.

Forgeries. Once we found a quadruple (x, y, z, t) that respects the relation $\mathcal{R}(x, y, z, t)$, we know that after processing the messages $\phi(x) = 0 \parallel x$ and $\psi(t) = 1 \parallel t$ there is no difference in the Θ part of the states $(\Theta_0(x) = \Theta_1(t))$. Moreover, we have $\Theta_0(x) = \Sigma_0(x) \oplus E_1(0)$ and $\Theta_1(t) = \Sigma_1(x) \oplus E_1(1)$; this implies that there is a difference $E_1(0) \oplus E_1(1) = x \oplus y$ in the Σ part of the state. Therefore, we can build a full state collision with message $0 \parallel x \parallel 0$ and $1 \parallel t \parallel x \oplus y$. In particular, the following relation can be used to create forgeries with an arbitrary message m (of any length):

$$\mathrm{MAC}(0 \parallel x \parallel 0 \parallel m) = \mathrm{MAC}(1 \parallel t \parallel x \oplus y \parallel m).$$

Universal Forgeries. We can even forge the tag of an arbitrary message of length at least $(2n + 2)$ blocks with complexity only $n + 1$ times the complexity of the simple forgery attack. The technique is more advanced and inspired by the multi-collision attack described by Joux [Jou04] (see Section 7.2.1). For ease of notation we'll show how to forge the signature for a message starting with $2n + 2$ blocks of zero, but this can be trivially adapted for any message.

First, we find a quadruple (x_1, y_1, z_1, t_1) as before. Then, we consider messages $0 \parallel 0$ and $1 \parallel x_1 \oplus y_1$. Since $x_1 \oplus y_1 = E_1(0) \oplus E_1(1)$, we have $\Sigma(0 \parallel 0) = \Sigma(1 \parallel x_1 \oplus y_1)$, as the Σ part of the state collides. Moreover, we know the difference in the Θ part: $\Theta(0 \parallel 0) \oplus \Theta(1 \parallel x_1 \oplus y_1) = x_1 \oplus y_1$.

Generally, at step i we use message injection functions

$$\phi_i(x) = \underbrace{0 \parallel 0 \parallel \ldots \parallel 0}_{\times 2(i-1)} \parallel 0 \parallel x \qquad \psi_i(x) = \underbrace{0 \parallel 0 \parallel \ldots \parallel 0}_{\times 2(i-1)} \parallel 1 \parallel x,$$

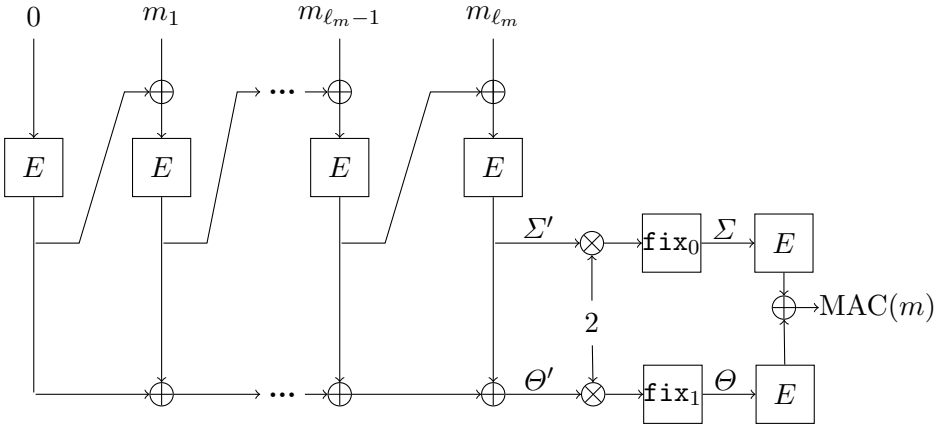


Figure 5.6: Diagram for `1kf9` with a ℓ_m-block message.

to look for a quadruple of messages

$$X_i = \phi_i(x_i) \qquad Y_i = \psi_i(y_i) \qquad Z_i = \phi_i(z_i) \qquad T_i = \psi_i(t_i).$$

When a right quadruple (x_i, y_i, z_i, t_i) has been identified, we can deduce that the MACs for $0 \,\|\, 0 \,\|\, \ldots \,\|\, 0 \,\|\, 0 \,\|\, 0$ and $0 \,\|\, 0 \,\|\, \ldots \,\|\, 0 \,\|\, 1 \,\|\, x_i \oplus y_i$ will match on the Σ branch and differ by $x_i \oplus y_i$ in their Θ branch.

After several iterations, we have actually built a multi-collision: all the messages $h_1 \,\|\, h_2 \,\|\, \ldots \,\|\, h_n \,\|\, h_{n+1}$ with $h_i \in \{(1 \,\|\, x_i \oplus y_i), (0 \,\|\, 0)\}$ collide on the Σ branch. In addition, we also know the difference in the Θ branch for those messages: it is equal to $\bigoplus_{\{i\,:\,h_i \neq 0\|0\}}(x_i \oplus y_i)$.

After at most $n+1$ steps, we can find a nonempty subset $\mathcal{I} \subseteq [1 : n+1]$ such that $\bigoplus_{i\in\mathcal{I}}(x_i \oplus y_i) = 0$ by simple linear algebra[3]. This gives a collision on the full state, using messages $m_0 = 0 \,\|\, 0 \,\|\, \ldots \,\|\, 0$ (with $2(n+1)$ blocks) and $h = h_1 \,\|\, h_2 \,\|\, \ldots \,\|\, h_n \,\|\, h_{n+1}$ with $h_i = 1 \,\|\, x_i \oplus y_i$ if $i \in \mathcal{I}$, $h_i = 0 \,\|\, 0$ otherwise. Since the full state collides, we have for any message m (of any length):

$$\mathrm{MAC}(h \,\|\, m) = \mathrm{MAC}(m_0 \,\|\, m).$$

5.2.6 Attacking and Breaking `1kf9`

`1kf9` is a tentative to build a single keyed version of `3kf9`. It uses `fix`$_b$ function that fixes the least significant bit to b and leave the rest untouched.

[3] We construct the kernel of the linear function $\lambda_i \mapsto \bigoplus_i \lambda_i(x_i \oplus y_i)$

This ensures domain separation, that is for all message m we have $\Sigma(m) \neq \Theta(m)$. We describe `1kf9` in Figure 5.6:

$$\begin{aligned}\Sigma(m) &= \mathtt{fix}_0(2 \cdot \sigma_{\ell_m}) \qquad \sigma_0 = E(0) \quad \sigma_i = E(\sigma_{i-1} \oplus m_i)\\ \Theta(m) &= \mathtt{fix}_1(2 \cdot \textstyle\bigoplus_{i=0}^{\ell_m} \sigma_i)\\ \mathrm{MAC}(m) &= E(\Sigma(m)) \oplus E(\Theta(m))\end{aligned}$$

However, `1kf9` was withdrawn to do a mistake in the proof leaving its true security as an open question. The attack on `3kf9` we describe in Section 5.2.5 is applicable but one can do better here. In fact, we answer the open question of `3kf9` security by showing a birthday-bound attack. This mode thus fails to guarantee security beyond the birthday bound despite its double-block hash-then-sum approach.

Attack on `1kf9`. In order to mount a birthday bound attack on `1kf9`, we use pairs of messages instead of quadruples. More precisely, instead of looking for a quadruple with pairwise collisions in Σ and Θ, we look for a pair of message X, Y colliding on Σ', and with a difference in Θ' that will be absorbed by the $\mathtt{fix}_1$ function. Therefore, we define the relation $\mathcal{R}$ as:

$$\begin{aligned}\mathcal{R}(X,Y) := &\begin{cases}\Sigma'(X) = \Sigma'(Y)\\ 2\Theta'(X) = 2\Theta'(Y) \oplus 1\end{cases}\\ &\Rightarrow \mathrm{MAC}(X) = \mathrm{MAC}(Y).\end{aligned}$$

We build the messages with different postfixes, parametrized by u:

$$X = \phi_u(x) = x \,\|\, u \qquad\qquad Y = \psi_u(y) = y \,\|\, u \oplus d,$$

where d is the inverse of 2 in the finite field. With this construction, we have

$$\begin{aligned}\Sigma'(\phi_u(x)) &= E\big(u \oplus E(x \oplus E(0))\big)\\ \Theta'(\phi_u(x)) &= E\big(u \oplus E(x \oplus E(0))\big) \oplus E\big(x \oplus E(0)\big) \oplus E(0)\\ \Sigma'(\psi_u(y)) &= E\big(u \oplus d \oplus E(y \oplus E(0))\big)\\ \Theta'(\psi_u(y)) &= E\big(u \oplus d \oplus E(y \oplus E(0))\big) \oplus E\big(y \oplus E(0)\big) \oplus E(0)\end{aligned}$$

In particular, we observe

$$\begin{aligned}E(x \oplus E(0)) \oplus E(y \oplus E(0)) = d \Leftrightarrow\ & \Sigma'(\phi_u(x)) = \Sigma'(\psi_u(y))\\ \Rightarrow\ & \Theta'(\phi_u(x)) \oplus \Theta'(\psi_u(y)) = d\\ \Rightarrow\ & \mathrm{MAC}(\phi_u(x)) = \mathrm{MAC}(\psi_u(y)). \quad (5.8)\end{aligned}$$

From this observation, we construct a birthday attack against `1kf9`. We build two lists:

$$L_0 = \left\{\mathrm{MAC}(\phi_0(x)) \ : \ x < 2^{n/2}\right\} \quad L_1 = \left\{\mathrm{MAC}(\psi_0(y)) \ : \ y < 2^{n/2}\right\},$$

and we look for a match between the lists. We expect on average one pair to match randomly, and one pair to match because of (5.8). Moreover, when we have a collision candidate $L_0[x], L_1[y]$, we can verify whether it is a right pair by comparing $\mathrm{MAC}(x \,\|\, 1)$ and $\mathrm{MAC}(y \,\|\, d \oplus 1)$.

Therefore, we find a pair satisfying $\mathcal{R}(X, Y)$ with complexity $2^{n/2}$, and this leads to simple forgeries using (5.8). This contradicts the security proof of `1kf9` given in [Dat+15]. Note that this attack is still valid if we use different multiplications for the two branches in the finalization function.

5.2.7 Attacking `1kPMAC+`

The `1kPMAC+` mode also uses the $\mathtt{fix}_b$ function that fixes the least significant bit to b and leave the rest untouched. `1kPMAC+` is essentially a single-key variant of `PMAC+`, as shown in Figure 5.7.

$$\begin{aligned} \Sigma'(m) &= \textstyle\bigoplus_{i=1}^{\ell_m} \widetilde{E}_i(m_i) & \Sigma(m) &= \mathtt{fix}_0(\Sigma'(m)) \\ \Theta'(m) &= \textstyle\bigoplus_{i=1}^{\ell_m} 2^{\ell_m+1-i} \odot \widetilde{E}_i(m_i) & \Theta(m) &= \mathtt{fix}_1(\Theta'(m)) \\ \mathrm{MAC}(m) &= E(\Sigma(m)) \oplus E(\Theta(m)) \end{aligned}$$

Mounting the Attack. Since the `fix` functions used in the finalization have collisions, we can build a variant of the attacks from Section 5.2.3 using differences in Σ' and/or Θ' that are absorbed by the `fix` functions. More precisely, we use the following relation $\mathcal{R}$ on quadruple of messages:

$$\mathcal{R}(X, Y, Z, T) := \begin{cases} \Sigma'(X) = \Sigma(Y)' \oplus 1 \\ \Theta'(Y) = \Theta(Z)' \oplus 1 \\ \Sigma'(Z) = \Sigma(T)' \oplus 1 \\ \Theta'(T) = \Theta(X)' \oplus 1 \end{cases}$$
$$\Rightarrow \mathrm{MAC}(X) \oplus \mathrm{MAC}(Y) \oplus \mathrm{MAC}(Z) \oplus \mathrm{MAC}(T) = 0.$$

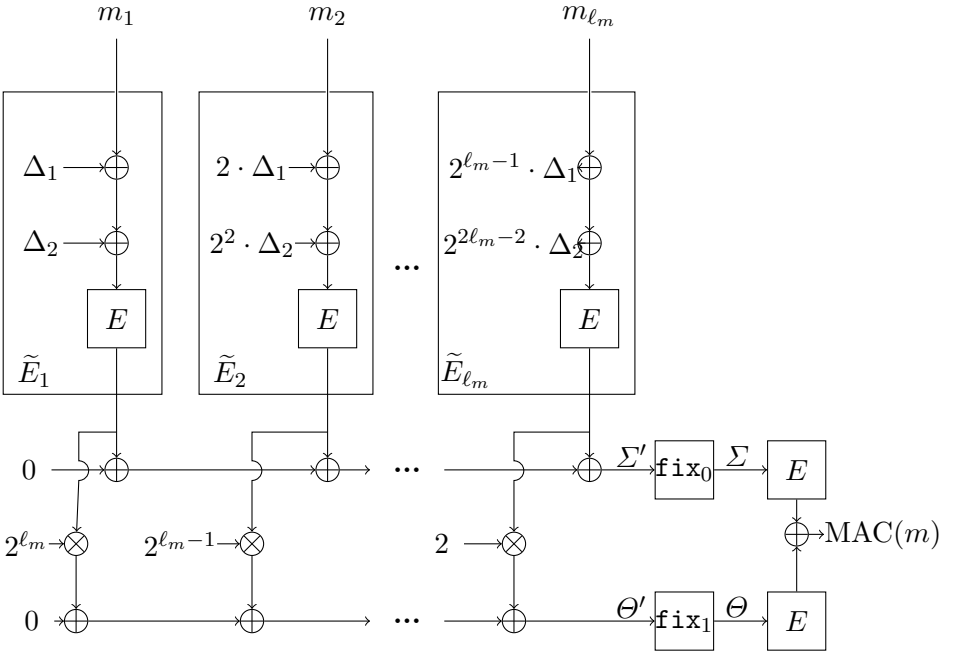


Figure 5.7: Diagram for `1kPMAC+` with a ℓ_m-block message where $\Delta_1 = E(1)$ and $\Delta_2 = E(2)$.

We can find quadruple of messages satisfying $\mathcal{R}$ using a single message injection function:

$$\begin{aligned} \phi_u(i) &= u \,\|\, i \\ X = \phi_u(x) &= u \,\|\, x \qquad Y = \phi_u(y) = u \,\|\, y \\ Z = \phi_u(z) &= u \,\|\, z \qquad T = \phi_u(t) = u \,\|\, t \end{aligned}$$

Indeed, we have

$$\mathrm{MAC}(\phi_u(i)) = E\Big(\mathtt{fix0}\big(\underbrace{\widetilde{E}_1(u) \oplus \widetilde{E}_2(x)}_{\Sigma'_u(i)}\big)\Big) \oplus E\Big(\mathtt{fix1}\big(\underbrace{4\widetilde{E}_1(u) \oplus 2\widetilde{E}_2(x)}_{\Theta'_u(i)}\big)\Big)$$

We observe that:

$$\mathcal{R}(x,y,z,t) \Leftrightarrow \begin{cases} \widetilde{E}_2(x) = \widetilde{E}_2(y) \oplus 1 \\ \widetilde{E}_2(z) = \widetilde{E}_2(t) \oplus 1 \\ 2\widetilde{E}_2(x) = 2\widetilde{E}_2(z) \oplus 1 \\ 2\widetilde{E}_2(y) = 2\widetilde{E}_2(t) \oplus 1 \end{cases}$$
$$\Leftrightarrow \begin{cases} \widetilde{E}_2(x) \oplus \widetilde{E}_2(y) \oplus \widetilde{E}_2(z) \oplus \widetilde{E}_2(t) = 0 \\ \widetilde{E}_2(x) = \widetilde{E}_2(y) \oplus 1 \\ \widetilde{E}_2(x) = \widetilde{E}_2(z) \oplus d \end{cases}$$

Therefore, $\mathcal{R}$ defines a $3n-$bit relation that is independent of the value u. This can be used for attacks in the same way as in the previous sections, using a single list

$$L = \left\{ \mathrm{MAC}(\phi_0(x)) \,\|\, \mathrm{MAC}(\phi_1(x)) \,\|\, \mathrm{MAC}(\phi_2(x)) \;:\; x < 2^{3n/4} \right\}$$

We can find a quadruple of four distinct values (x,y,z,t) such that $L[x] \oplus L[y] \oplus L[z] \oplus L[t] = 0$ with $\widetilde{\mathcal{O}}(2^{3n/2})$ operations, using a memory of size $\mathcal{O}(2^{3n/4})$, and this easily leads to forgeries.

5.3 Conclusion

5.3.1 Proof is hard

Flawed proof. We saw that the recent proof of [Nai18] for `LightMAC+` was directly invalidated by our attack.

Here we show how the security of withdrawn mode `1kf9` (Figure 5.6) is actually capped at $\mathcal{O}(2^{n/2})$ by giving a birthday bound attack. The proof was already known to be wrong, but this cryptanalysis shows that it is not possible to fix in order to get beyond-birthday-bound security.

5.3.2 Open Questions

Cryptanalysis and Proofs. The cryptanalysis techniques introduced in this chapter show how to attack Double-block Hash-then-Sum MACs in $\mathcal{O}(2^{3n/4})$ short queries. In particular, we show that this attack applies to `SUM-ECBC`, the authentication part of `GCM-SIV2`, `PMAC+`, `LightMAC+`, `1kPMAC+` and `3kf9`.

After the publication of this result, new proofs [KLL20] for the Double-block Hash-then-Sum construction showed that the attack is actually optimal in terms of short queries for `SUM-ECBC`, `PMAC+`, `LightMAC+` and `3kf9`. This is a great example of why we need to work on both better proofs and better cryptanalysis to better understand the security of certain modes of operation.

This leaves a gap for modes `GCM-SIV2` and `1kPMAC+`. For instance the proof cannot be directly applied to `1kPMAC+` due to the relation between the two parts of the internal state. For `GCM-SIV2` the issue is the relation between the two n-bit parts of the MAC, but we note that the proof of [KLL20] at least applies to both n-bit parts taken independently.

Other Attacks. Information theoretically, our attack shows that the bound of [KLL20] is optimal in terms of the number of queries. However, in the proof, the length of the queries is also an important parameter. An open question is to ask whether there are attacks exploiting long queries to further reduce the total number of queries and, by doing so, eventually match the best bound implied by the proof either given by [Dat+18] or [KLL20].

We also reduce the time complexity in the cases of `SUM-ECBC` and `GCM-SIV2`. However, there remain modes where we don't have an attack with a time complexity below 2^n. Further optimizing the attacks in terms of time and memory complexities while keeping a data complexity below 2^n is left as an open question.

Chapter 6
Release of Unverified Plaintext security of ANYDAE

Contributions brought forward in this chapter were published in ToSC Volume 2019, Issue 4 and are a joint work of Chang, Datta, Dutta, Mennink, Nandi, Sanadhya and I [Cha+19a]. It is a follow-up on the group discussion initiated in the Asian Symmetric Key Workshop, 2018.

Introduction

A secure authenticated encryption mode of operation is expected to provide both authenticity and confidentiality of the message which corresponds to the EUF (Section 2.2.1) and IND$-CPA (Section 2.1.1) security notions respectively. In Section 2.4.3 we've argued that there are subtle differences between achieving these notions in the random IV, nonce and deterministic model. Indeed, all are somehow equivalent as long as one respects the requirements of the model, but the fewer requirements the easiest it is to implement correctly and, thus, we say the mode is more robust. There are also advanced security notions aiming for a better robustness, and we've introduced the security under release of unverified plaintext or RUP security.

In this Chapter we'll show how the deterministic authenticated encryption mode SUNDAE [Ban+18] can be made more robust with only a slight, inexpensive modification. We'll see that the new construction MONDAE achieves authenticity and confidentiality in the RUP setting (INT-RUP and PA1 respectively). To do that, we introduce the AERUP security notion that is equivalent to the combination AE, PA1 and INT-RUP. Much like AE is equivalent to EUF-AE and IND$-CPA-AE combined, the AERUP notion will allow us to prove both conventional and RUP security within a single proof.

In fact, we prove the AERUP security of a generic mode ANYDAE and give two instantiations: MONDAE, that is SUNDAE with a twist, and TUESDAE, that optimizes the rate. The AERUP security of MONDAE and TUESDAE are directly implied by the one of ANYDAE.

6.1 RUP (In)Security of SUNDAE

6.1.1 RUP Security Notions

We've touched briefly on security under release of unverified plaintext in Section 2.4.3. Here we formally define the PA and the INT-RUP security notions first introduced by Andreeva et al. [And+14].

The concept applies to the context of authenticated encryption and it introduces a leakage function acting as a decryption oracle, $\mathbf{Dec}_k$, in addition to the usual encryption oracle $\mathbf{AEnc}_k$ and verification oracle V_k. Informally, the function $\mathbf{Dec}_k$ outputs the decryption of the ciphertext without checking its validity (so it never outputs $\perp$). Therefore, it has the same interface as V_k even though the associated data or the tag may be ignored depending on the mode. One may think of $\mathbf{Dec}_k$ as the decryption function of encryption-only modes but used in authenticated encryption context. It is often clear what the definition of $\mathbf{Dec}_k$ is from the definition of $\mathbf{AEnc}_k$.

Plaintext Awareness. To prove security under plaintext awareness we define a distinguishing game between the real and ideal worlds with two oracles. In the real world, the attacker has access to the authenticated encryption $\mathbf{AEnc}_k(\cdot)$ which takes a message as input and to the decryption oracle $\mathbf{Dec}_k(\cdot,\cdot)$ which takes a tag and ciphertext as inputs and outputs the resulting plaintext regardless of the validity of the tag. In the ideal world, the attacker has still access to $\mathbf{AEnc}_k(\cdot)$ and the decryption oracle is replaced by a simulator $\mathbf{Sim}(\cdot,\cdot)$. The mode is said to be PA secure if there exists a simulator that is undistinguishable from the decryption oracle. This simulator cannot query $\mathbf{AEnc}_k(\cdot)$ nor does he know the key. In other words the simulator could have been run by the adversary himself (with an independent source of randomness). This effectively captures the fact that future encryptions are not impacted by the adversary having access to a decryption oracle.

Concretely, the advantage of an adversary $\mathcal{A}$ for PA security of an authenticated encryption mode is defined as:

$$\mathbf{Adv}^{\mathsf{PA}}_{\cdot}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{\mathbf{AEnc}_k(\cdot),\mathbf{Sim}(\cdot,\cdot)} \to 1) \tag{6.1}$$

with randomness of k, **Sim** and $\mathcal{A}$. Then, the PA security is the maximum advantage over all $\mathcal{A}$ using the best efficient simulator that is

$$\mathbf{Adv}^{\mathsf{PA}}_{\cdot} = \min_{\mathbf{Sim}} \max_{\mathcal{A}} \mathbf{Adv}^{\mathsf{PA}}_{\cdot}(\mathcal{A}) \ .$$

In particular, the description of the simulator is left to the prover so that if he can upper-bound the advantage with a particular simulator, then this upper-bound is valid for the best efficient simulator.

Depending on the power of the simulator there are two distinct PA notions. If the simulator can record the attacker queries to $\mathbf{AEnc}(\cdot)$ and their outputs, we call this PA1; if it can't, we call this PA2. In this work we will mostly use the PA1 notion.

Authenticity under RUP. Similarly, having access to a decryption oracle shouldn't compromise the authenticity of the messages. The INT-RUP security notion is similar to the existential unforgeability (EUF) notion but with a stronger adversary that can decrypt any ciphertext.

In the INT-RUP security game the adversary must do a forgery while having access to $\mathbf{AEnc}_k$, $\mathbf{Dec}_k$ and the usual verification oracle V_k:

$$\mathbf{Adv}^{\mathsf{INT\text{-}RUP}}_{\cdot}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),V_k(\cdot,\cdot)} \text{ forges}) \tag{6.2}$$

with the randomness of k and $\mathcal{A}$. The definition of forgery remains the same: V_k has to output $\top$ for a (c, T) pairs that was not the result of an earlier query to $\mathbf{AEnc}_k(\cdot)$ (in the presence of associated data, it means that there is no m such that (c, T) was the result of an earlier (a, m) query).

6.1.2 RUP Attack on SUNDAE

SUNDAE (Figure 6.1) is a lightweight authenticated encryption mode of operation proposed by Banik et al. [Ban+18] with provable and tight birthday bound security.

It is a combination of a variant of CBC-MAC, close to mode GCBC1 [Nan09], for the authentication and the OFB mode for the encryption. SUNDAE combines them in a synthetic IV manner meaning that the tag produced is also used as an IV for encryption with the OFB mode. Hence, it is a deterministic AE mode secure up to $\mathcal{O}(2^{n/2})$ processed data which is tight since both the authentication and the encryption parts allow for an attack at birthday bound.

It has multiple good features making it fit for lightweight encryption. Indeed, SUNDAE is deterministic (no stateful counter nor source of randomness needed), it has an n-bit internal state and only uses a single secret key, and it computes the block cipher in the forward direction exclusively. Galois Field multiplications are limited to doubling which is fast. In the case of static associated data, SUNDAE doesn't necessarily reprocess them every time: one can remember the internal state value after processing the associated data and directly start from that point for future messages. A diagram and a formal algorithmic description of SUNDAE are given in Figure 6.1 and Algorithm 6.1 respectively where the padding function $\mathsf{pad}_n(\cdot)$ is an optional 10^* padding to n-bit as:

$$\mathsf{pad}_n(X) = \begin{cases} X \parallel 10^{n-|X|-1} & , \text{ if } |X| < n \\ X & , \text{ otherwise} \end{cases}$$

Although this mode is quite recent, it has been chosen as an AEAD mode for multiple submissions to the NIST Lightweight Cryptography Standardization Process including SIV-Rijndael256 [Bao+19a], SIV-TEM-PHOTON [Bao+19b], ESTATE [Cha+19a], SUNDAE-GIFT [Ban+19], and TRIFLE [Dat+19].

On the other hand, the authors of SUNDAE [Ban+18] claim that the mode "provides maximal robustness to a lack of proper randomness or secure state" while at the same time warning that "unverified plaintext from the decryption algorithm should not be released". This is somewhat contradictory as RUP security notions actually deal with robustness to a lack of secure state. Moreover, the synthetic IV construction of SUNDAE clearly forces the receiver to decrypt the message before verifying the validity of the tag as shown in Algorithm 6.1. This is what motivates our analysis of its security under release of unverified plaintext, especially in the context of lightweight cryptography.

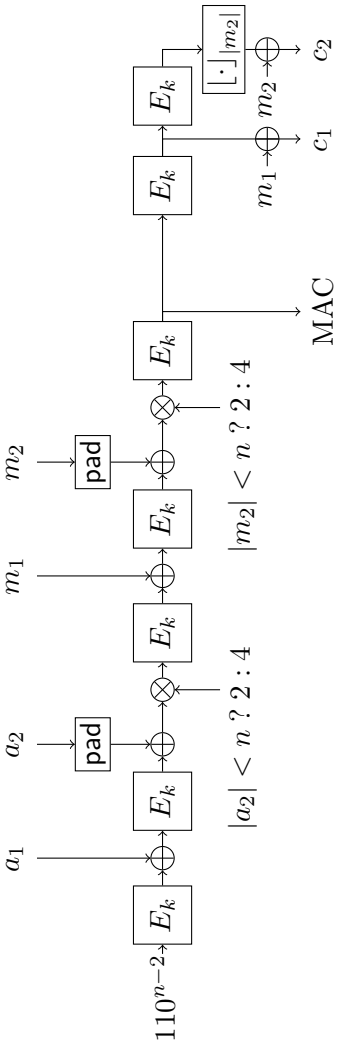


Figure 6.1: SUNDAE authenticated encryption mode of operations with two blocks of associated data $a = a_1 \| a_2$ and message $m = m_1 \| m_2$.

Algorithm 6.1 Authenticated Encryption, Decryption and Verification algorithms for SUNDAE oracles. pad_n is an optional 10^* padding.

Algorithm $\mathrm{MAC}(a, m)$

1. $b_1 \leftarrow |a| > 0 \ ? \ 1 \ : \ 0$
2. $b_2 \leftarrow |m| > 0 \ ? \ 1 \ : \ 0$
3. $T \leftarrow E_k(b_1 \,\|\, b_2 \,\|\, 0^{n-2})$
4. **if** $|a| > 0$ **then**
5. $\quad a_1 \,\|\, a_2 \,\|\, \ldots \,\|\, a_{\ell_a} \xleftarrow{n} a$
6. $\quad$ **for** $i = 1$ **to** $\ell_a - 1$
7. $\quad\quad T \leftarrow E_k(T \oplus a_i)$
8. $\quad X \leftarrow |a_{\ell_a}| < n \ ? \ 2 \ : \ 4$
9. $\quad T \leftarrow E_k\big(X \cdot (T \oplus \mathsf{pad}_n(a_{\ell_a}))\big)$
10. **if** $|m| > 0$ **then**
11. $\quad m_1 \,\|\, m_2 \,\|\, \ldots \,\|\, m_{\ell_m} \xleftarrow{n} m$
12. $\quad$ **for** $i = 1$ **to** $\ell_m - 1$
13. $\quad\quad T \leftarrow E_k(T \oplus m_i)$
14. $\quad X \leftarrow |m_{\ell_m}| < n \ ? \ 2 \ : \ 4$
15. $\quad T \leftarrow E_k\big(X \cdot (T \oplus \mathsf{pad}_n(m_{\ell_m}))\big)$
16. **return** T

Algorithm $\mathtt{OFB}(T, m)$

1. $m_1 \,\|\, m_2 \,\|\, \ldots \,\|\, m_{\ell_m} \xleftarrow{n} m$
2. $Z_0 \leftarrow T$
3. **for** $i = 1$ **to** m
4. $\quad Z_i \leftarrow E_k(Z_{i-1})$
5. $\quad c_i \leftarrow \lfloor Z_i \rfloor_{|m_i|} \oplus m_i$
6. **return** $c_1 \,\|\, c_2 \,\|\, \ldots \,\|\, c_{\ell_m}$

$\mathbf{AEnc}_k(a, m)$:

1. $T \leftarrow \mathrm{MAC}(a, m)$
2. $c \leftarrow \mathtt{OFB}(T, m)$
3. **return** (T, c)

$\mathbf{Dec}_k(a, c, T)$:

1. $m \leftarrow \mathtt{OFB}(T, c)$
2. **return** m

$V_k(a, c, T)$:

1. $m \leftarrow \mathtt{OFB}(T, c)$
2. $T' \leftarrow \mathrm{MAC}(a, m)$
3. **return** $T' \stackrel{?}{=} T \ ? \ \top \ : \ \bot$

RUP Attack on SUNDAE. While the mode is provably AE secure, an easy forgery attack shows that SUNDAE is actually not INT-RUP secure. Concretely, there exists an INT-RUP adversary $\mathcal{A}$ such that:

$$\mathbf{Adv}^{\mathsf{INT\text{-}RUP}}_{\texttt{SUNDAE}}(\mathcal{A}) = 1$$

where $\mathcal{A}$ makes 1 encryption query, 3 decryption queries and 1 verification query. The proof is simply the description of the attack.

Proof. We show a simple universal forgery attack for arbitrary associated data $a = a_1 \,\|\, a_2 \,\|\, \ldots \,\|\, a_{\ell_a}$ and message $m = m_1 \,\|\, m_2 \,\|\, \ldots \,\|\, m_{\ell_m}$ with $\ell_a \geq 2$ and $\ell_m \geq 1$ where ε is the empty string:

1. $\mathbf{Dec}_k(\varepsilon, T^1, c_1^1)$ with $T^1 = 110^{n-2}$. Get m_1^1;
2. $\mathbf{Dec}_k(\varepsilon, T^2, c_1^2)$ with $T^2 = m_1^1 \oplus c_1^1 \oplus a_1$. Get m_1^2;
3. $\mathbf{Dec}_k(\varepsilon, T^3, c_1^3)$ with $T^3 = m_1^1 \oplus c_1^1 \oplus a_1'$ for some $a_1' \neq a_1$. Get m_1^3;
4. Let $\Delta := m_1^2 \oplus c_1^2 \oplus m_1^3 \oplus c_1^3$;
5. $\mathbf{AEnc}_k(a_1' \,\|\, a_2 \oplus \Delta \,\|\, a_3 \,\|\, \cdots \,\|\, a_{\ell_a}, m)$. Get the tag T and ciphertext c;
6. $V_k(a, c, T)$ is a valid forgery for the data a and message m.

In the attack, c_1^1, c_1^2, and c_1^3 may take any n-bit value. □

This attack exploits the fact that a $\mathbf{Dec}_k$ query is a direct query to the underlying block cipher E_k. The tag generation process of SUNDAE can be reconstructed step by step by an attacker. In the first step we recover the value $E_k(110^{n-2}) = m_1^1 \oplus c_1^1$. The second and third queries are necessary to compute Δ which is the internal state difference after the second block cipher call between processing a_1 and a_1'. Therefore, processing $a_1' \,\|\, a_2 \oplus \Delta$ or processing $a_1 \,\|\, a_2$ will result in the same internal state meaning that this is a full internal state collision. Hence, the following states and outputs will be the same whenever the rest of the processed data and message are equal leading to an easy forgery.

The strategy is easily adapted when the target a or m is empty: simply adapt to the proper starting value $b_1 \,\|\, b_2 \,\|\, 0^{n-2}$ instead of 110^{n-2} and perform the same trick with the first two processed blocks. The case of a single block of data or message is also trivial: fully simulate the tag generation process with only 2 $\mathbf{Dec}_k$ queries.

This universal forgery attack is not in contradiction with the security claims of SUNDAE but the lack of security under the release of unverified plaintext is something to be aware of especially when using it in constrained environment.

6.2 RUP Security of ANYDAE

We showed in Section 6.1.2 that SUNDAE is not INT-RUP secure. In this section we show the condition that actually makes the mode more robust. The resulting generic mode is called ANYDAE. This preserves the lightweightness as well as the integrity and confidentiality security guarantees. We first introduce the AERUP security notion that ANYDAE satisfies. Then, we detail the generically defined ANYDAE mode of operation and two of its variants: MONDAE that only marginally differs from SUNDAE and TUESDAE that optimizes the rate of the mode.

6.2.1 AERUP Generalized Notion of Security

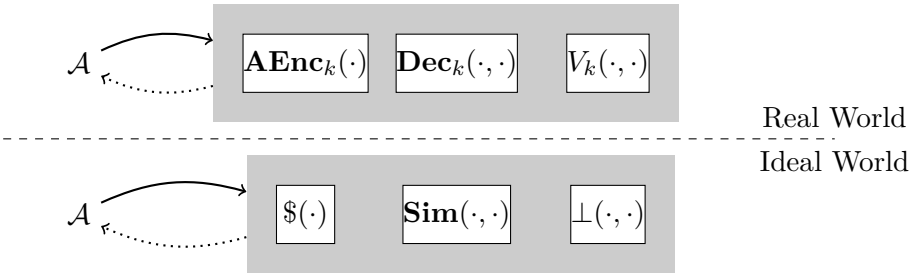


Figure 6.2: Distinguishing game for the AERUP security of an authenticated encryption mode where k is a random key value, $\$(\cdot)$ is a random function, $\mathbf{Sim}(\cdot,\cdot)$ is an efficient simulator with no access to $\$(\cdot)$ and $\perp(\cdot,\cdot)$ returns $\perp$ on every input.

The proof of ANYDAE is done using a security game for the new notion of AERUP security that we define as:

Definition 6.1 (AERUP advantage)**.** Let **Sim** be a simulator. The AERUP security of an authenticated encryption scheme against an adversary $\mathcal{A}$ is

defined as:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),V_k(\cdot,\cdot)} \to 1) \\ - \mathbf{Pr}(\mathcal{A}^{\$(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1)\,, \end{aligned} \tag{6.3}$$

with the randomness of $k \xleftarrow{\$} \{0,1\}^\kappa$, $\$$ a random function, and the random choices of **Sim** and $\mathcal{A}$. The simulator **Sim** has no access to the first oracle, but it has access to the query history of $\mathcal{A}$. The adversary is not allowed to relay an earlier response from the first oracle to the third oracle.

Then, the AERUP advantage $\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}$ is the maximum advantage over all $\mathcal{A}$ under the best efficient simulator. In other words, the AERUP advantage is $\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot} = \min_{\mathbf{Sim}} \max_{\mathcal{A}} \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A})$.

This will allow us to prove the robustness of ANYDAE in a single go.

Proving the Equivalence. Let us prove that AERUP is equivalent to AE, PA1 and INT-RUP security combined. We showed in Section 2.3.1 the equivalence of AE security with IND$-CPA-AE and EUF-AE combined. So now we proceed the same way with AERUP.

First, we prove that breaking AE breaks AERUP or, equivalently:

$$\mathbf{Adv}^{\mathsf{AE}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}$$

Indeed, for any AE adversary $\mathcal{A}_1$ there exists an AERUP adversary $\mathcal{A}_2$ with the same complexity with $\mathbf{Adv}^{\mathsf{AE}}_{\cdot}(\mathcal{A}_1) \leq \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_2)$ for any **Sim**. $\mathcal{A}_2$ simply runs $\mathcal{A}_1$ and answer all its queries with the first and third oracles and output the same conclusion. Indeed, the AERUP security game (Definition 6.1) falls back to the AE security game when ignoring the second oracle.

Now we wish to prove that breaking PA1 breaks AERUP. In fact, we'll show that breaking PA1 breaks either AERUP or IND$-CPA-AE or, equivalently:

$$\mathbf{Adv}^{\mathsf{PA1}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot} + \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot}$$

Let $\mathcal{A}_1$ be a PA1 adversary under some efficient simulator **Sim**, we define an AERUP adversary $\mathcal{A}_2$ that forwards $\mathcal{A}_1$ queries (without querying its

third oracle), and an IND$-CPA-AE adversary $\mathcal{A}_3$ with the same query complexity that is capable of simulating the simulator (as the simulator is efficient) and the $\perp$ oracle. Then:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{PA1}}_{\cdot}(\mathcal{A}_1) =& \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{Sim}(\cdot,\cdot)} \to 1) \\
=& \mathbf{Pr}(\mathcal{A}_2^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),V_k(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}_2^{\mathbf{AEnc}_k(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
=& \mathbf{Pr}(\mathcal{A}_2^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),V_k(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}_2^{\$(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
&+ \mathbf{Pr}(\mathcal{A}_2^{\$(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}_2^{\mathbf{AEnc}_k(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
=& \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_2) + \mathbf{Pr}(\mathcal{A}_3^{\$(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}_3^{\mathbf{AEnc}_k(\cdot)} \to 1) \\
\mathbf{Adv}^{\mathsf{PA1}}_{\cdot}(\mathcal{A}_1) =& \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_2) + \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot}(\mathcal{A}_3) \quad \square
\end{aligned}$$

Notice that $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{AE}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}$ therefore we've indeed proven that $\mathbf{Adv}^{\mathsf{PA1}}_{\cdot} \leq 2\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}$ meaning that breaking PA1 breaks AERUP with a significant advantage up to a factor 2.

Here we show that breaking INT-RUP breaks AERUP or, equivalently:

$$\mathbf{Adv}^{\mathsf{INT\text{-}RUP}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}$$

Indeed, for any INT-RUP adversary $\mathcal{A}_1$ there exists an AERUP adversary $\mathcal{A}_2$ with the same complexity such that $\mathbf{Adv}^{\mathsf{INT\text{-}RUP}}_{\cdot}(\mathcal{A}_1) = \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_2)$. $\mathcal{A}_2$ runs $\mathcal{A}_1$ answering all its queries with its corresponding oracles as both games have the same interface. If the verification oracle answers something else than $\perp$ then $\mathcal{A}_1$ has forged and $\mathcal{A}_2$ answers 1; otherwise it answers 0. The probability of $\mathcal{A}_1$ forging is thus the probability of $\mathcal{A}_2$ outputting 1. In the ideal world, both probability are thus 0. In the real world, both probability are the INT-RUP advantage of $\mathcal{A}_1$. Therefore, we have $\mathbf{Adv}^{\mathsf{INT\text{-}RUP}}_{\cdot}(\mathcal{A}_1) = \mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_2)$.

Finally we show that breaking AERUP breaks either INT-RUP, PA1 or AE or, equivalently:

$$\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{INT\text{-}RUP}}_{\cdot} + \mathbf{Adv}^{\mathsf{PA1}}_{\cdot} + \mathbf{Adv}^{\mathsf{AE}}_{\cdot}$$

In fact, we show that $\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{INT\text{-}RUP}}_{\cdot} + \mathbf{Adv}^{\mathsf{PA1}}_{\cdot} + \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot}$ which implies what we wish to prove as $\mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot} \leq \mathbf{Adv}^{\mathsf{AE}}_{\cdot}$.

Indeed, for any AERUP adversary $\mathcal{A}_1$ with some efficient simulator **Sim** we define an INT-RUP adversary $\mathcal{A}_2$ with the same query complexity

that runs $\mathcal{A}_1$ and forwards its queries with its corresponding oracles hoping to forge that way. Moreover, we define a PA1 adversary $\mathcal{A}_3$ that runs $\mathcal{A}_1$ and forwards its queries to its first and second oracle while simulating the third oracle as a $\perp$ oracle; then it also forwards the output. Finally, we have an IND$-CPA-AE adversary $\mathcal{A}_4$ that runs $\mathcal{A}_1$ and forwards its queries to its first oracle while simulating the second and third oracles as the efficient simulator **Sim** and the $\perp$ oracle respectively; then it also forwards the output. Following this, we have:

$$
\begin{aligned}
\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_1) =& \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),V_k(\cdot,\cdot)} \to 1) \\
&\quad - \mathbf{Pr}(\mathcal{A}_1^{\$(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_1) =& \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),V_k} \to 1) \\
&\quad - \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
&+ \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
&\quad - \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
&+ \mathbf{Pr}(\mathcal{A}_1^{\mathbf{AEnc}_k(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
&\quad - \mathbf{Pr}(\mathcal{A}_1^{\$(\cdot),\mathbf{Sim}(\cdot,\cdot),\perp(\cdot,\cdot)} \to 1) \\
\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_1) \leq& \mathbf{Pr}(\mathcal{A}_2^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot),V_k(\cdot,\cdot)} \text{ forges}) \\
&+ \mathbf{Pr}(\mathcal{A}_3^{\mathbf{AEnc}_k(\cdot),\mathbf{Dec}_k(\cdot,\cdot)} \to 1) \\
&\quad - \mathbf{Pr}(\mathcal{A}_3^{\mathbf{AEnc}_k(\cdot),\mathbf{Sim}(\cdot,\cdot)} \to 1) \\
&+ \mathbf{Pr}(\mathcal{A}_4^{\mathbf{AEnc}_k(\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}_4^{\$(\cdot)} \to 1) \\
\mathbf{Adv}^{\mathsf{AERUP}}_{\cdot}(\mathcal{A}_1) \leq& \mathbf{Adv}^{\mathsf{INT\text{-}RUP}}_{\cdot}(\mathcal{A}_2) + \mathbf{Adv}^{\mathsf{PA1}}_{\cdot}(\mathcal{A}_3) + \mathbf{Adv}^{\mathsf{IND\$\text{-}CPA\text{-}AE}}_{\cdot}(\mathcal{A}_4)
\end{aligned}
$$

□

Comparison with other security notions. We've just seen that the AERUP security notion is equivalent to the combination of INT-RUP, PA1 and AE security. In the literature there has been a few other notions summarizing a robust security definition.

For instance, Hoang et al. [HKR15] introduced the notion of robust authenticated encryption, RAE security. RAE deals with the nonce misuse case by introducing a parameter limiting the number of nonce reuse allowed. It also generalizes the authenticity goal of an authenticated encryption scheme with a ciphertext expansion parameter λ instead of a separate tag.

Moreover, they also considered security under decryption leakage with the $\mathsf{RAE}_{\mathsf{sim}}$ security notion. The simulator for $\mathsf{RAE}_{\mathsf{sim}}$ has no access to the query history of the attacker therefore it is stronger than the PA1 notion. AERUP can be seen as a variant of $\mathsf{RAE}_{\mathsf{sim}}$ where the simulator records the interactions with the encryption oracle. In particular, $\mathsf{RAE}_{\mathsf{sim}}$ security implies AERUP security but is not equivalent.

Another example is the alternative notion of RUP security, RUPAE, by Ashur et al. [ADL17]. RUPAE combines the PA1 and INT-RUP notions for nonce-based schemes and imposes that the ideal model decryption oracle is simply a random function. For comparison, AERUP is not relying on a nonce, but the simulator cannot be reduced to a random function.

$\mathsf{RAE}_{\mathsf{sim}}$ and RUPAE are therefore stronger notions than AERUP. However, the goal of the AERUP security notion is not to describe a more robust security notion. Indeed, we showed that it is equivalent to the combination of already known security notions and thus cannot be stronger. On the other hand, the AERUP security game will allow us to prove the security of ANYDAE in a single go while stronger security notions wouldn't be a fit. Ultimately, concise and short proofs increase the trust we have in them.

6.2.2 ANYDAE Mode of Operation

Remember that the INT-RUP attack on SUNDAE (Section 6.1.2) directly query the block cipher through the decryption oracle to reconstruct the internal state in the authentication part. There are various solutions to counter this problem by adjusting the way the tag T is generated or the way T is used to generate the key stream.

Solutions like masking or transforming T through a block cipher are undesirable: it would increase the state size (for the mask) or implementation size (to make use of the block cipher inverse). So we rather focus on the way the tag is used to generate the key stream. ANYDAE is a generalized construction solving this issue and just concrete enough to allow for an AERUP security proof to be made.

Specification. ANYDAE is, just like SUNDAE, an AEAD scheme built on top of a block cipher E and parametrized by a single key k. In addition, it uses a formatting function Fmt to parse the data and mixing functions ρ_1, ρ_2, ρ_3 to process the state. Concretely, let $\mathcal{T}$ be a (possibly empty) finite set. Then $\mathsf{Fmt} : \{0,1\}^* \to (\{0,1\}^n)^\ell \times \mathcal{T}^{\ell-1}$ for any $\ell > 0$ is a formatting function that takes an arbitrarily long bit string and generates

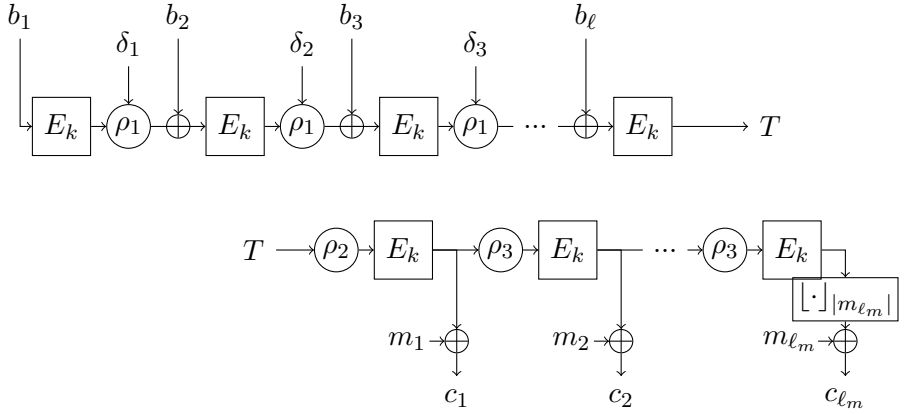


Figure 6.3: ANYDAE authenticated encryption mode of operations with formatting function preprocessing $(b, \delta) \leftarrow \mathsf{Fmt}(a, m)$.

a sequence of n-bit blocks along with a sequence of elements of $\mathcal{T}$ of the same length minus one. Furthermore, the domain of the three state processing functions are:

$$\rho_1 \colon \{0,1\}^n \times \mathcal{T} \to \{0,1\}^n, \rho_2 \colon \{0,1\}^n \to \{0,1\}^n, \rho_3 \colon \{0,1\}^n \to \{0,1\}^n .$$

ANYDAE is defined as in the diagram of Figure 6.3 and its oracles are formally described in Algorithm 6.2. By construction, it has many similarities with SUNDAE namely it has an n-bit state, is length preserving and only requires a single key.

Before we state the security of ANYDAE, we remind the definition of almost XOR-universal (AXU) and almost uniform functions:

Definition 6.2. Let $\epsilon > 0$, $n \in \mathbb{N}$ and a function $\rho : \{0,1\}^n \times \mathcal{T} \to \{0,1\}^n$ for a non-empty set $\mathcal{T}$.

- $\rho(X, t)$ is said to be ϵ-almost uniform if for any $t \in \mathcal{T}$ and any $Y \in \{0,1\}^n$,

$$\mathbf{Pr}(X \xleftarrow{\$} \{0,1\}^n : \rho(X, t) = Y) \leq \epsilon .$$

Algorithm 6.2 Authenticated Encryption, Decryption and Verification algorithms for ANYDAE oracles.

Algorithm $\mathrm{MAC}(a, m)$

1. $(b_1 \| \ldots \| b_\ell, \delta_1 \| \ldots \| \delta_{\ell-1}) \leftarrow \mathsf{Fmt}(a, m)$
2. $X_1 \leftarrow b_1$
3. **for** $i = 1$ **to** $\ell - 1$
4. $\quad Y_i \leftarrow E_k(X_i)$
5. $\quad X_{i+1} \leftarrow \rho_1(Y_i, \delta_i) \oplus b_{i+1}$
6. $T \leftarrow E_k(X_\ell)$
7. **return** T

Algorithm $\mathtt{OFB}(T, m)$

1. $m_1 \| \ldots \| m_{\ell_m} \xleftarrow{n} m$
2. $U_1 \leftarrow \rho_2(T)$
3. **for** $i = 1$ **to** m
4. $\quad V_i \leftarrow E_k(U_i)$
5. $\quad c_i \leftarrow \lfloor V_i \rfloor_{|m_i|} \oplus m_i$
6. $\quad U_{i+1} \leftarrow \rho_3(V_i)$
7. **return** $c_1 \| \ldots \| c_{\ell_m}$

$V_k(a, c, T)$

1. $m \leftarrow \mathtt{OFB}(T, c)$
2. $T' \leftarrow \mathrm{MAC}(a, m)$
3. **return** $T' \stackrel{?}{=} T \ ? \ \top \ : \ \bot$

AEnc$_k(a, m)$

1. $T \leftarrow \mathrm{MAC}(a, m)$
2. $c \leftarrow \mathtt{OFB}(T, m)$
3. **return** (T, c)

Dec$_k(a, c, T)$

1. $m \leftarrow \mathtt{OFB}(T, c)$
2. **return** m

- $\rho(X,t)$ is said to be ϵ-almost XOR-universal (ϵ-AXU) if for any distinct t and $t' \in \mathcal{T}$ and any $Y \in \{0,1\}^n$,

$$\mathbf{Pr}(X \xleftarrow{\$} \{0,1\}^n : \rho(X,t) \oplus \rho(X,t') = Y) \leq \epsilon \ .$$

Based on Definition 6.2, we obtain the following corollary:

Corollary 6.1. *Let $\epsilon > 0$, $n \in \mathbb{N}$ and a finite set $\mathcal{T}$. Consider an ϵ-almost uniform function $\rho\colon \{0,1\}^n \times \mathcal{T} \to \{0,1\}^n$. Then, for any $t, t' \in \mathcal{T}$, $Y \in \{0,1\}^n$ we have*

$$\mathbf{Pr}((X,X') \xleftarrow{\$} \{0,1\}^n \times \{0,1\}^n : \rho(X,t) \oplus \rho(X',t') = Y) \leq \epsilon \, .$$

Then, the AERUP security of ANYDAE is stated in Theorem 6.1:

Theorem 6.1 (AERUP security of ANYDAE)**.** *We consider* ANYDAE *with formatting and processing functions* Fmt, ρ_1, ρ_2, ρ_3 *and based on the block cipher $E\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$.*
Denote by $\mathcal{F}_1$ the set of first block outputs of Fmt*. If*

1. Fmt *is injective and prefix-free;* [1]
2. *ρ_1 is ϵ_1-AXU and γ_1-uniform;*
3. *ρ_2 is γ_2-uniform;*
4. *ρ_3 is γ_3-uniform;*
5. *$|\mathcal{F}_1 \cap \mathbf{Im}(\rho_2)| = 0$ and $|\mathcal{F}_1 \cap \mathbf{Im}(\rho_3)| = \Omega$,*

then

$$\mathbf{Adv}^{\mathit{AERUP}}_{\texttt{ANYDAE}}(\sigma, q_v, t) \leq \mathbf{Adv}^{\mathit{PRP}}_{E}(\sigma, t') + \binom{\sigma}{2}\left(\frac{1}{2^n} + \max\{\epsilon_1, \gamma_1, \gamma_2, \gamma_3\}\right) + \Omega\sigma \cdot \gamma_3 + \frac{q_v}{2^n} \, ,$$

against adversaries limited to σ total block cipher calls, q_v verification queries and running in time t and $t' \approx t$ respectively.

The proof details are given in Section 6.3.

[1] For Fmt, prefix-freeness means that for any two elements $\big(b_1 \parallel \ldots \parallel b_\ell, \delta_1 \parallel \ldots \parallel \delta_{\ell-1}\big), \big(b'_1 \parallel \ldots \parallel b'_{\ell'}, \delta'_1 \parallel \ldots \parallel \delta'_{\ell'-1}\big) \in \mathbf{Im}(\mathsf{Fmt})$ with $\ell < \ell'$, $\big(b_1 \parallel \ldots \parallel b_\ell, \delta_1 \parallel \ldots \parallel \delta_{\ell-1}\big) \neq \big(b'_1 \parallel \ldots \parallel b'_\ell, \delta'_1 \parallel \ldots \parallel \delta'_{\ell-1}\big)$.

6.2.3 MONDAE and TUESDAE Mode of Operation

By choosing the parameters of ANYDAE, we propose two AERUP secure modes: MONDAE and TUESDAE.

The MONDAE Robust Authenticated Encryption. With MONDAE we propose a minimal fix of SUNDAE to achieve AERUP security. The formatting function and processing functions ρ_1 and ρ_3 are kept as in SUNDAE, the only difference is to introduce the $\texttt{fix}_1$ as ρ_2. Concretely, in MONDAE ρ_2 fixes the least significant bit to 1 and leave the rest untouched. This ensures that one cannot reconstruct the internal state of the authentication part as the starting value is always set to 0 (control bits are the most significant ones).

Furthermore, MONDAE is a specific instantiation of ANYDAE with:

$$\rho_1(S, X) = X \cdot S\,, \qquad \rho_2(S) = \texttt{fix}_1(S) = \lfloor S \rfloor_{n-1} \,\|\, 1\,, \qquad \rho_3(S) = S\,,$$

and the formatting function of SUNDAE described in Algorithm 6.3.

The AERUP security of MONDAE is a direct corollary of the security statement of ANYDAE, Theorem 6.1. We simply have to note that $\epsilon_1 = \gamma_1 = \gamma_3 = 1/2^n$, $\gamma_2 = 2/2^n$, and $\Omega = 4$ (since $\mathcal{F}_1 = \{0^n, 10^{n-1}, 010^{n-2}, 110^{n-2}\}$ and $\mathbf{Im}(\rho_3) = \{0,1\}^n$). Hence:

Corollary 6.2 (AERUP security of MONDAE)**.** *Let be the authenticated encryption scheme* MONDAE *based on a block cipher E. Then*

$$\mathbf{Adv}^{\mathsf{AERUP}}_{\texttt{MONDAE}}(\sigma, q_v, t) \leq \mathbf{Adv}^{\mathsf{PRP}}_{E}(\sigma, t') + \frac{1.5\sigma^2}{2^n} + \frac{2.5\sigma}{2^n} + \frac{q_v}{2^n}\,,$$

against adversaries limited to σ total block cipher calls, q_v verification queries and running in time t and $t' \approx t$ respectively.

The TUESDAE Mode of Operation. The goal of TUESDAE is to optimize the functions of ANYDAE so that the number of block cipher calls becomes optimal for most inputs. Indeed, for ℓ_a blocks of associated data and ℓ_m block of message with $\ell_a + \ell_m > 1$ then one encryption using TUESDAE use exactly $\ell_a + 2\ell_m$ block cipher calls. Moreover, when (ℓ_a, ℓ_m) is $(1, 0)$ or $(0, 1)$ and the total length is less than $n - 4$ bits then TUESDAE requires a single block cipher call for authentication (and another to encrypt in the latter case). Notice that, as we need to distinguish between the cases of associated data and message, and between full and partial block, it is

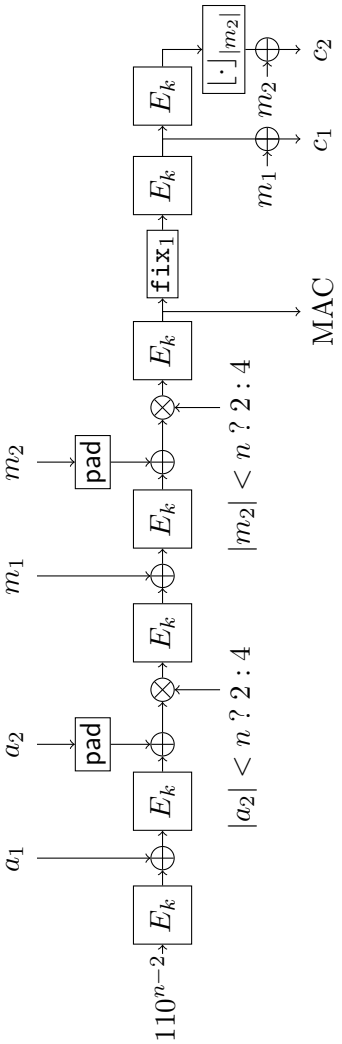


Figure 6.4: MONDAE authenticated encryption mode of operations with two blocks of associated data $a = a_1 \| a_2$ and message $m = m_1 \| m_2$.

Algorithm 6.3 The formatting function Fmt used in SUNDAE and MONDAE with $\delta \in \{1, 2, 4\}$ multiplied is a characteristic 2 Galois field. pad_n is an optional 10^* padding.

Function $\mathsf{Fmt}(a, m)$

1. $r_1 \leftarrow |a| > 0 \ ? \ 1 \ : \ 0$
2. $r_2 \leftarrow |m| > 0 \ ? \ 1 \ : \ 0$
3. $\ell \leftarrow \ell_a + \ell_m + 1$
4. $b_1 \leftarrow r_1 \,\|\, r_2 \,\|\, 0^{n-2}$
5. **for** $i = 2$ **to** ℓ_a
6. $\quad \delta_{i-1} \leftarrow 1$
7. $\quad b_i \leftarrow a_{i-1}$
8. $\delta_{\ell_a} \leftarrow |a_{\ell_a}| < n \ ? \ 2 \ : \ 4$
9. $b_{\ell_a+1} \leftarrow \delta_{\ell_a} \cdot \mathsf{pad}_n(a_{\ell_a})$
10. **for** $i = 2$ **to** m
11. $\quad \delta_{\ell_a+i-1} \leftarrow 1$
12. $\quad b_{\ell_a+i} \leftarrow m_{i-1}$
13. $\delta_{\ell-1} \leftarrow |m_{\ell_m}| < n \ ? \ 2 \ : \ 4$
14. $b_\ell \leftarrow \delta_{\ell-1} \cdot \mathsf{pad}_n(m_{\ell_m})$
15. **return** $(b_1 \,\|\, \ldots \,\|\, b_\ell, \delta_1 \,\|\, \ldots \,\|\, \delta_{\ell-1})$

impossible for any ANYDAE instantiation to have a single block cipher call for a single full data block. Also, the choice of ρ_1 requires that $n-1$ be a prime number therefore it is most well suited to be built on $n = 128$-bit block cipher.

TUESDAE is an instantiation of ANYDAE with the formatting function Fmt of Algorithm 6.4 and:

$$\rho_1(S, X) = \lfloor S \rfloor_1 \,\|\, (\lceil S \rceil_{n-1} \ggg X)\,, \rho_2(S) = \mathtt{fix}_{10}(S)\,, \rho_3(S) = \mathtt{fix}_{10}(S)\,,$$

where $X \in \{0,1\}^5$, $\ggg$ is the circular shift function and $\mathtt{fix}_{10}$ is the fix function that puts the two least significant bits to 10 and leaves the rest untouched.

It is easy to verify that TUESDAE's Fmt is prefix-free: the three rightmost bits of b_1 are 000 in case A, 100 in case B, and $**1$ in cases C, D, and E. For the last three cases, difference is in δ_1: it equals $*011*$ for case C, either of $\{00*0*, 01**, 100**\}$ for case D, and either of $\{1010*, 110**, 111**, 0001*\}$ for case E. Here, for case E, distinction is made using $\mathsf{empty}_i \,\|\, \mathsf{final}_{D[i+1]} \,\|\, \mathsf{full}_{D[i+1]}$.

Moreover, in every case $\lceil b_1 \rceil_2 \neq 10$ which directly implies that $\mathcal{F}_1 \cap \mathbf{Im}(\rho_3)| = 0$, and the almost XOR-universal property of ρ_1 was shown by Contini and Yin [CY99] who proved that if $|S|$ is prime and $t \leq |S|$, $S \ggg t$ is $(2^{|S|-1})^{-1}$-AXU. Therefore, the security of TUESDAE is a corollary of the security of ANYDAE (Theorem 6.1) with $\epsilon_1 = \gamma_2 = \gamma_3 = 4/2^n$ whenever $n-1$ is prime, $\gamma_1 = 1/2^n$, and $\Omega = 0$.

Corollary 6.3 (AERUP security of TUESDAE)**.** *Let be the authenticated encryption scheme* TUESDAE *based on an n-bit block cipher E with $n-1$ prime. Then*

$$\mathbf{Adv}^{\mathit{AERUP}}_{\mathtt{TUESDAE}}(\sigma, q_v, t) \leq \mathbf{Adv}^{\mathit{PRP}}_{E}(\sigma, t') + \frac{2.5\sigma^2}{2^n} + \frac{q_v}{2^n}\,,$$

against adversaries limited to σ total block cipher calls, q_v verification queries and running in time t and $t' \approx t$ respectively.

This result only applies when $n-1$ is prime, but the choice of ρ_1 is not absolute; any good almost XOR-universal function will suit. In particular, Corollary 6.3 holds for popular $n = 128$-bit block ciphers such as the AES. Notice that using a smaller block cipher is not recommended anyway due to the inherent birthday-bound security of the ANYDAE construction.

Algorithm 6.4 The formatting function Fmt of TUESDAE where $\delta_i \in \{0,1\}^5$ with control bits type, that indicates whether the current block is associated data ($\mathsf{type} = 0$) or message ($\mathsf{type} = 1$), full, that indicates whether X is n-bit ($\mathsf{full}_X = 1$) or less ($\mathsf{full}_X = 0$), and final, that indicates whether X is the last of its type ($\mathsf{final}_X = 1$) or not ($\mathsf{final}_X = 0$). $\mathsf{bin}(\ell)_i$ is the binary encoding of the integer ℓ on i bits. pad_n is an optional 10^* padding.

Function $\mathsf{Fmt}(a, m)$

$D \xleftarrow{n} a \,\|\, m$
if $\ell_a = \ell_m = 0$ **then**
  $b_1 \leftarrow 0^n$
  return b_1 ▷ **Case A**
if $\ell_a + \ell_m = 1$ and $|D| \leq n - 5$ **then**
  $b_1 \leftarrow \mathsf{pad}_{n-4}(D_1) \,\|\, \mathsf{type} \,\|\, 100$
  return b_1 ▷ **Case B**
$b_1 \leftarrow \mathtt{fix}_1(\mathsf{pad}_n(D_1))$
$r \leftarrow \lfloor \mathsf{pad}_n(D_1) \rfloor_1$
if $\ell_a + \ell_m = 1$ and $|D| \geq n - 4$ **then**
  $\delta_1 \leftarrow \mathsf{full}_{D_1} \,\|\, 011 \,\|\, r$
  $b_2 \leftarrow 0^* \,\|\, \mathsf{type} \,\|\, 010$
  return $(b_1 \,\|\, b_2, \delta_1)$ ▷ **Case C**
if $\ell_a + \ell_m = 2$ **then**
  $\delta_1 \leftarrow \mathsf{bin}(\ell_a)_2 \,\|\, \mathsf{full}_{m_{\ell_m}} \,\|\, \mathsf{full}_{a_{\ell_a}} \,\|\, r$
  $b_2 \leftarrow \mathsf{pad}_n(D_2)$
  return $(b_1 \,\|\, b_2, \delta_1)$ ▷ **Case D**
if $\ell_a + \ell_m > 2$ **then**
  if $\ell_a < 3$ **then**
    $\delta_1 \leftarrow \mathsf{bin}(\ell_a + 5)_3 \,\|\, \mathsf{full}_{a_{\ell_a}} \,\|\, r$
  else
    $\delta_1 \leftarrow 0001 \,\|\, b$
  $\ell \leftarrow \ell_a + \ell_m$
  for $i = 2$ **to** ℓ
    $b_i \leftarrow \mathsf{pad}_n(D_i)$
  for $i = 2$ **to** $\ell - 1$
    $\delta[i] \leftarrow 00 \,\|\, \mathsf{empty}_i \,\|\, \mathsf{final}_{D_{i+1}} \,\|\, \mathsf{full}_{D_{i+1}}$
  return $(b_1 \,\|\, \ldots \,\|\, b_\ell, \delta_1 \,\|\, \ldots \,\|\, \delta_{\ell-1})$
  ▷ **Case E**

6.3 Proving AERUP Security of ANYDAE

In this Section we take up the task of proving Theorem 6.1. Thanks to the generalized AERUP security notion, this will only require a single proof.

6.3.1 H-Coefficient Technique and Proof Strategy

The broad idea of this proof follows a widely used strategy: we first replace the block cipher by a random function and use Patarin's H-Coefficient Technique to bound the adversary advantage in the information theoretic setting.

Patarin's H-Coefficient Technique. Consider a computationally unbounded deterministic adaptive adversary $\mathcal{A}$ for a distinguishing game between a real and an ideal world. As usual, $\mathcal{A}$ must output a decision bit at the end of its interaction with its oracles. We call τ the transcript all queries-responses made by $\mathcal{A}$ to its oracles. The transcript may also contain additional information revealed to $\mathcal{A}$ at the end of its interactions but before its decision. Indeed, additional information can only give more distinguishing power to the attacker and thus the derived advantage upper-bound will stand.

Let X_{re} and X_{id} be random variables denoting the transcript in the real and ideal worlds respectively. Therefore, $\mathbf{Pr}(X_{\text{re}} = \tau)$ is the probability that the transcript τ is realized in the real world and $\mathbf{Pr}(X_{\text{id}} = \tau)$ the probability that the transcript τ is realized in the ideal world. We say a transcript τ is attainable if $\mathbf{Pr}(X_{\text{id}} = \tau) \neq 0$. The set of attainable transcript is noted Θ and the main theorem of the H-coefficient technique [Pat09; CS14] is as follows.

Theorem 6.2 (H-coefficient technique)**.** *Let $\mathcal{A}$ be a fixed computationally unbounded deterministic adversary that has access to either the real world oracle $\mathcal{O}_{\text{re}}$ or the ideal world oracle $\mathcal{O}_{\text{id}}$. Let $\Theta = \Theta_{\text{g}} \sqcup \Theta_{\text{b}}$ be some partition of the set of all attainable transcripts into* good *and* bad *transcripts. Suppose there exists $\epsilon_{\text{ratio}} \geq 0$ such that for any $\tau \in \Theta_{\text{g}}$,*

$$\frac{\mathbf{Pr}(X_{\text{re}} = \tau)}{\mathbf{Pr}(X_{\text{id}} = \tau)} \geq 1 - \epsilon_{\text{ratio}} \,,$$

and there exists $\epsilon_{\text{bad}} \geq 0$ such that $\mathbf{Pr}(X_{\text{id}} \in \Theta_{\text{b}}) \leq \epsilon_{\text{bad}}$. Then,

$$\mathbf{Pr}(\mathcal{A}^{\mathcal{O}_{\text{re}}} \to 1) - \mathbf{Pr}(\mathcal{A}^{\mathcal{O}_{\text{id}}} \to 1) \leq \epsilon_{\text{ratio}} + \epsilon_{\text{bad}} \,. \tag{6.4}$$

Notice that the best strategy (or one of the best) of a computationally unbounded adversary is necessarily deterministic so the upper-bound on the advantage of deterministic adversaries given by the H-Coefficient technique also applies to probabilistic adversaries.

Security Game. We consider any adversary $\mathcal{A}$ in the AERUP security game, Definition 6.1. As it is common in security proofs of birthday-bound secure schemes we first use the PRP/PRF switch (Lemma 1.1) and consider the ANYDAE construction instantiated by a random function instead of a block cipher.

Concretely, we have:

$$\mathbf{Adv}^{\mathsf{AERUP}}_{\mathtt{ANYDAE}-E}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{PRP}}_{E}(\mathcal{B}) + \frac{\sigma(\sigma-1)}{2^{n+1}} + \mathbf{Adv}^{\mathsf{AERUP}}_{\mathtt{ANYDAE}-f}(\mathcal{A})$$

where f is an n to n bit random function and σ is the number of block cipher calls (which depends on the query complexity, the queries length and the Fmt specification). To upper-bound $\mathbf{Adv}^{\mathsf{AERUP}}_{\mathtt{ANYDAE}-f}(\mathcal{A})$ we use the H-coefficient technique for any adversary $\mathcal{A}$ making q_e, q_d and q_v encryption, decryptions and verification queries respectively.

6.3.2 Oracles Definition for AERUP Security

First, a few notations:
Let (a^{i+}, m^{i+}) be the ith encryption query $(1 \leq i \leq q_e)$ where the associated data a^{i+} and the message m^{i+} are of block lengths ℓ_a^{i+} and ℓ_m^{i+} respectively.
Let (a^{i-}, c^{i-}, T^{i-}) be the ith decryption query $(1 \leq i \leq q_d)$ where the associated data a^{i-} and the ciphertext c^{i-} are of block lengths ℓ_a^{i-} and ℓ_c^{i-} respectively.
Let $(a^{i\star}, c^{i\star}, T^{i\star})$ be the ith verification query $(1 \leq i \leq q_v)$ where the associated data $a^{i\star}$ and the ciphertext $c^{i\star}$ are of block lengths $\ell_a^{i\star}$ and $\ell_c^{i\star}$ respectively. We assume a non-trivial, non-repeating $\mathcal{A}$ so that all queries are distinct and no $(a^{i\star}, c^{i\star}, T^{i\star})$ is an answer of an earlier encryption query.
Let $(i, *)$ be the ith message of type $* \in \{+, -, \star\}$, we note $(j, \circledast) \prec (i, *)$ the fact that the jth message of type $\circledast$ was queried before the ith message of type $*$.
Let $B^{i*} = (b_1^{i*} \parallel ... \parallel b_{\ell_b^{i*}}^{i*}, \delta_1^{i*} \parallel ... \parallel \delta_{\ell_b^{i*}-1}^{i*}) \leftarrow \mathsf{Fmt}(a^{i*}, m^{i*})$ for $* \in \{+, \star\}$ as

the formatting function is not used in decryption queries.
We say B^{i*} and $B^{j\circledast}$ have a common prefix of size p if and only if $(b_1^{i*} \parallel ... \parallel b_p^{i*}, \delta_1^{i*} \parallel ... \parallel \delta_{p-1}^{i*}) = (b_1^{j\circledast} \parallel ... \parallel b_p^{j\circledast}, \delta_1^{j\circledast} \parallel ... \parallel \delta_{p-1}^{j\circledast})$.
We assume that $\mathcal{A}$ is non repeating and Fmt is prefix-free so having $B^{i*} = B^{j\circledast}$ is impossible except for some $(i, \star) \prec (j, +)$ (a verification followed by an encryption query of the same message). We call $\mathcal{Q}$ the set of all such $((i, \star), (j, +))$. Otherwise, as Fmt is a public function, the values of B^{i*} can be considered as under the control of the adversary but for two different queries $(i, *) \neq (j, \circledast)$ with $((i, *), (j, \circledast)) \notin \mathcal{Q}$ we necessarily have $p < \min(\ell_b^{i*}, \ell_b^{j\circledast})$.

Now let us define the oracles for both the real and ideal worlds. In particular, we are free to define any simulator **Sim** we like to get a proper upper-bound as long as it only accesses the query history and do not directly query the other oracles.

Real World Oracles. At the start of the game the real world oracles draw a random n to n bits function f. The encryption, decryption and verification oracles are $\mathbf{AEnc}(\cdot,\cdot)$, $\mathbf{Dec}(\cdot,\cdot,\cdot)$ and $V(\cdot,\cdot,\cdot)$ as described for ANYDAE in Algorithm 6.2 where E_k is replaced by f. At the end of the interaction, the real world oracles reveals all intermediate values to the attacker as supplementary information to be included in the transcript. In Algorithm 6.2 those values are noted Y_i and X_i in the authentication part (MAC), and V_i and U_i in the encryption part (OFB). They correspond to all inputs/outputs of f.

Ideal World Oracles. Following the AERUP security game, the ideal world consists of three oracles $(\$(\cdot,\cdot), \mathbf{Sim}(\cdot,\cdot,\cdot), \bot(\cdot,\cdot,\cdot))$. $\bot(\cdot,\cdot,\cdot)$ simply returns $\bot$ on all $(a^{i\star}, c^{i\star}, T^{i\star})$ inputs. As the simulator has access to the query history to the $ oracle, we denote an initially empty table $\mathcal{L}$ that the simulator will use to store (U, V)-tuples.

The encryption oracle $\$(\cdot,\cdot)$ is a random function that answers any query by a random ciphertext of corresponding length and a random tag. As we consider non repeating attacker, the $ oracle randomly samples new values (T^{i+}, c^{i+}) for every query (a^{i+}, m^{i+}) as:

$$c^{i+} = c_1^{i+} \parallel \ldots \parallel c_{\ell_m^{i+}}^{i+} \xleftarrow{\$} \{0,1\}^{|m^{i+}|},$$
$$T^{i+} \xleftarrow{\$} \{0,1\}^n.$$

Then, as the simulator observes the query (a^{i+}, m^{i+}) and its response (T^{i+}, c^{i+}) it stores in table $\mathcal{L}$ new (U^{i+}, V^{i+}) tuples as:

$$(U_k^{i+}, V_k^{i+}) \leftarrow \begin{cases} (\rho_2(T^{i+}), & m_1^{i+} \oplus c_1^{i+}), \text{ for } k = 1, \\ (\rho_3(V_{k-1}^{i+}), & m_k^{i+} \oplus c_k^{i+}), \text{ for } k = 2, \ldots, \ell_m^{i+}. \end{cases}$$

In case of a collision where $U_k^{i+} \in \mathcal{L}$, the simulator will override the old V_k^{i+} value. This event anyway provokes a bad transcript, and so we bound its probability in Section 6.3.3.

Finally, the ideal world decryption oracle is a simulator $\mathbf{Sim}(\cdot,\cdot,\cdot)$ which takes $(a^{i-}, c^{i-}, T^{i-}) = (a_1^{i-} \parallel \ldots, \| a_{\ell_{a^{i-}}}^{i-}, c_1^{i-} \parallel \ldots, \| c_{\ell_c^{i-}}^{i-}, T^{i-})$ as queries and answers with $m^{i-} = m_1^{i-} \parallel \ldots \parallel m_{\ell_c^{i-}}^{i-}$ computed by:

1. $k \leftarrow 1$
2. $U_1^{i-} \leftarrow \rho_2(T^{i-})$
3. **while** $U_k^{i-} \in \mathcal{L}$ **do**
4. $\quad V_k^{i-} \leftarrow \mathcal{L}(U_k^{i-})$
5. $\quad m_k^{i-} \leftarrow V_k^{i-} \oplus c_k^{i-}$
6. $\quad U_k^{i-} \leftarrow \rho_3(V_k^{i-})$
7. $\quad k \leftarrow k + 1$
8. **for** $j = k$ **to** ℓ_c^{i-}
9. $\quad m_j^{i-} \xleftarrow{\$} \{0,1\}^n$
10. $\quad V_j^{i-} \leftarrow m_j^{i-} \oplus c_j^{i-}$
11. $\quad U_j^{i-} \leftarrow \rho_3(V_{j-1}^{i-})$
12. $\quad$ add (U_j^{i-}, V_j^{i-}) to $\mathcal{L}$
13. **return** $m_1^{i-} \| \ldots \| m_{\ell_c^{i-}}^{i-}$

Once the adversary finishes its interactions with the oracles, additional information is revealed to him before the decision. Just like in the real world, he will be given internal values (X, Y) and (U, V). For encryption and decryption queries, the (U, V) values have already been defined as stored in the table $\mathcal{L}$. For verification queries $(a^{i\star}, c^{i\star}, T^{i\star})$, the corresponding (U, V) values are defined in the same way the simulator does for decryption queries (a^{i-}, c^{i-}, T^{i-}) and so does the underlying message $m^{i\star}$.

Notice that the (X, Y) values only exist for encryption and verification queries as the tag is not verified in decryption queries. To sample those (X, Y) values we first compute $B^{i*} \leftarrow \mathsf{Fmt}(a^{i*}, m^{i*})$ for $* \in \{+, \star\}$. For encryption queries, we first set $Y_{\ell_b^{i+}}^{i+} = T^{i+}$. Then, we go in reverse chronological order for all queries $(i, *)$ $(* \in \{+, \star\})$ and look for the

later query $(j, \circledast)$ ($\circledast \in \{+, \star\}$) with $(i, *) \prec (j, \circledast)$ such that B^{i*} has the longest common prefix with $B^{j\circledast}$. If $(i, *)$ is a verification query such that $((i, \star), (j, +)) \in \mathcal{Q}$ (the longest common prefix query is necessarily be the corresponding $(j, +)$), we set $Y_k^{i\star} = Y_k^{j+}$ for all $1 \leq k \leq \ell_b^{i\star}$. Otherwise, let $p < \ell_b^{i*}$ the length of the longest common prefix between B^{i*} and $B^{j\circledast}$. We set $Y_k^{i*} = Y_k^{j\circledast}$ for $1 \leq k \leq p$ and randomly sample $Y_k^{i*} \xleftarrow{\$} \{0,1\}^n$ for $p+1 \leq k \leq \ell_b^{i\star}$ for verification queries and for $p+1 \leq k \leq \ell_b^{i+} - 1$ for encryption queries where we additionally set $Y_{\ell_b^{i+}}^{i+} = T^{i+}$. Finally, the corresponding X values are computed as $X_1^{i*} \leftarrow b_1^{i*}$ and $X_k^{i*} \leftarrow \rho_1(Y_{k-1}^{i*}, \delta_{k-1}^{i*}) \oplus b_k^{i*}$ for $2 \leq k \leq \ell_b^{i*}$.

Attainable Transcripts. We've defined all values that are either exchanged during the interactions or revealed afterward. A transcript of the attack $\tau = (\tau_e, \tau_d, \tau_v)$ is thus written as:

$$\begin{aligned}
\tau_e &= \{(a^{i+}, m^{i+}, c^{i+}, T^{i+}, X^{i+}, Y^{i+}, U^{i+}, V^{i+}) : 1 \leq i \leq q_e\}, \\
\tau_d &= \{(a^{i-}, m^{i-}, c^{i-}, T^{i-}, U^{i-}, V^{i-}) : 1 \leq i \leq q_d\}, \\
\tau_v &= \{(a^{i\star}, m^{i\star}, c^{i\star}, T^{i\star}, X^{i\star}, Y^{i\star}, U^{i\star}, V^{i\star}, \top/\bot) : 1 \leq i \leq q_v\}.
\end{aligned}$$

A transcript is said to be attainable (with respect to $\mathcal{A}$) if the probability to realize this transcript in the ideal world is not zero. For instance, the verification queries in the ideal world only answers $\bot$ so the last value of all elements of τ_v must be $\bot$ whenever $\tau = (\tau_e, \tau_d, \tau_v)$is attainable.

Following the H-coefficient technique (Theorem 6.2), we call Θ the set of all attainable transcripts and X_{re} and X_{id} the random variables following the probability distributions of the transcript τ induced by the real and ideal worlds respectively.

6.3.3 Analysis of Bad Transcripts

Definition of Bad Transcripts. A bad transcript happens when a bad event occurs so let us define what a bad event is. We distinguish four

types of bad events: $\mathsf{Coll}_{\mathsf{XX}}$, $\mathsf{Coll}_{\mathsf{XU}}$, $\mathsf{Coll}_{\mathsf{UU}}$ and Forge defined as:

$$\begin{aligned}
\mathsf{Coll}_{\mathsf{XX}}\colon &\exists (j,\circledast) \preceq (i,*), k, k' \text{with } (k \neq k' \text{ or } B^{i*} \text{ and } B^{j\circledast} \text{ do not have} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{a common prefix of size } k) \\
&\qquad\qquad\qquad\qquad\qquad \text{such that } X^{i*}_{k'} = X^{j\circledast}_{k}\,, \\
\mathsf{Coll}_{\mathsf{XU}}\colon &\exists (j,\circledast), (i,*), k, k' \text{ such that } U^{i*}_{k'} = X^{j\circledast}_{k}\,, \\
\mathsf{Coll}_{\mathsf{UU}}\colon &\exists (j,\circledast) \preceq (i,*), k, k' \text{with } \left(* = + \text{ or } U^{i*}_{1} \neq U^{j\circledast}_{k-k'+1}\right) \\
&\qquad\qquad\qquad\qquad\qquad \text{such that } U^{i*}_{k'} = U^{j\circledast}_{k}\,, \\
\mathsf{Forge}\colon &\exists i \qquad\qquad\qquad\qquad\quad \text{such that } Y^{i\star}_{\ell^{i\star}_b} = T^{i\star}\,.
\end{aligned}$$

In other words, $\mathsf{Coll}_{\mathsf{XX}}$ denotes an accidental collision between two inputs to f in the authentication part excluding trivial collisions due to common prefix. $\mathsf{Coll}_{\mathsf{XU}}$ denotes an accidental collision between an input to f in the authentication part and one in the encryption part. $\mathsf{Coll}_{\mathsf{UU}}$ denotes an accidental collision between two inputs to f in the encryption part excluding trivial collisions due to a decryption or verification query starting with a previously known value. And Forge corresponds to the event that the last output of f in the authentication part of a verification query actually corresponds to the given tag value.

Again following the H-coefficient technique (Theorem 6.2), we note Θ_{b} the set of all bad attainable transcripts.

Probability of Bad Transcripts. In order to bound the probability of a bad transcript $\mathbf{Pr}(X_{\mathrm{id}} \in \Theta_{\mathrm{b}})$, we bound the probability of every bad events to occur in the ideal world and prove Lemma 6.1:

Lemma 6.1. *Let X_{id} and Θ_{b} be as defined as above. Then,*

$$\mathbf{Pr}(X_{\mathrm{id}} \in \Theta_{\mathrm{b}}) \leq \binom{\sigma}{2} \cdot \max\{\epsilon_1, \gamma_1, \gamma_2, \gamma_3\} + \Omega\sigma \cdot \gamma_3 + \frac{q_v}{2^n}\,.$$

Let us bound the probability of $\mathsf{Coll}_{\mathsf{XX}}$. Remember that $X^{i*}_1 = b^{i*}_1$ and $X^{i*}_{k'} = \rho_1(Y^{i*}_{k'-1}, \delta^{i*}_{k'-1}) \oplus b^{i*}_{k'}$ for $k' \neq 1$ and for Y values randomly sampled after the adversary interactions. We consider the following cases:

(i) $k = k' = 1$. The event is a contradiction as we require $b^{i*}_1 \neq b^{j\circledast}_1$. The event is set with probability 0;

(ii) $k = 1, k' \neq 1$. The event implies that $\rho_1(Y^{i*}_{k'-1}, \delta^{i*}_{k'-1}) = b^{j\circledast}_1 \oplus b^{i*}_{k'}$. As ρ_1 is γ_1-uniform, the probability of this event is at most γ_1;

(iii) $k \neq 1, k' = 1$. The event implies that $\rho_1(Y^{j\circledast}_{k-1}, \delta^{j\circledast}_{k-1}) = b^{i*}_1 \oplus b^{j\circledast}_k$. As ρ_1 is γ_1-uniform, the probability of this event is at most γ_1;

(iv) $k \neq 1, k' \neq 1$. The event implies that

$$\rho_1(Y^{i*}_{k'-1}, \delta^{i*}_{k'-1}) \oplus \rho_1(Y^{j\circledast}_{k-1}, \delta^{j\circledast}_{k-1}) = b^{i*}_{k'} \oplus b^{j\circledast}_k \,.$$

To bound the above event, we split it into two different subcases:

Case (a): When $k = k'$ and B^{i*} and $B^{j\circledast}$ have a common prefix of size $k-1$ (but not k). This implies that $Y^{i*}_{k-1} = Y^{j\circledast}_{k-1}$. Now there are two different subcases. If $\delta^{i*}_{k-1} = \delta^{j\circledast}_{k-1}$, then $b^{i*}_k \neq b^{j\circledast}_k$ and the probability of the above event is zero. Otherwise, the above event boils down to

$$\rho_1(Y^{i*}_{k-1}, \delta^{i*}_{k-1}) \oplus \rho_1(Y^{j\circledast}_{k-1}, \delta^{j\circledast}_{k-1}) = b^{i*}_k \oplus b^{j\circledast}_k \,.$$

This event is bounded by the ϵ_1-AXU property of ρ_1, where the probability is calculated over the random sampling of Y^{j*}_{k-1} as $(j, \circledast) \prec (i, *)$.

Case (b): Otherwise, Y^{i*}_{k-1} and $Y^{j\circledast}_{k-1}$ are independent and the above event is bounded by the uniform probability γ_1 of the ρ_1 function that directly follows from Corollary 6.1.

Combining all the four cases, we obtain

$$\mathbf{Pr}(\mathsf{Coll}_{\mathsf{XX}} \text{ occurs}) \leq \binom{\#X}{2} \cdot \max\{\epsilon_1, \gamma_1\} \,.$$

where $\#X$ is the number of X values in the transcript.

Let us bound the probability of $\mathsf{Coll}_{\mathsf{XU}}$. We consider the following cases:

(i) $k = k' = 1$. We have $|\mathcal{F}_1 \cap \mathbf{Im}(\rho_2)| = 0$, and hence, the probability of $\mathsf{Coll}_{\mathsf{XU}}$ occurring is 0;

(ii) $k' \neq 1, k = 1$. The event implies that $\rho_3(c_{k'}^{i*} \oplus m_{k'}^{i*}) = b_1^{j\circledast}$ where $b_1^{j\circledast}$ can be adaptively chosen. Thus, we suppose that it happens whenever $\rho_3(c_{k'}^{i*} \oplus m_{k'}^{i*}) \in \mathcal{F}_1$. As $|\mathcal{F}_1 \cap \mathbf{Im}(\rho_3)| = \Omega$, and as ρ_3 is γ_3-uniform, the probability of this event is at most $\Omega \# U \cdot \gamma_3$, where we have already summed over all possible query choices;

(iii) $k' = 1, k \neq 1$. The event implies that $\rho_1(Y_{k-1}^{j\circledast}, \delta_{k-1}^{j\circledast}) = U_1^{i*} \oplus b_k^{j\circledast}$. As ρ_1 is γ_1-uniform, the probability of this event is at most γ_1;

(iv) $k \neq 1, k' \neq 1$. The event implies $\rho_3(c_{k'}^{i*} \oplus m_{k'}^{i*}) = \rho_1(Y_{k-1}^{j\circledast}, \delta_{k-1}^{j\circledast}) \oplus b_k^{j\circledast}$. If $(j, \circledast) \prec (i, *)$, we bound this event by γ_3 due to the random sampling of $c_{k'}^{i*}$ or $m_{k'}^{i*}$ and ρ_3 being γ_3-uniform. Otherwise, we bound the event by γ_1 as ρ_1 is γ_1-uniform.

For the third case, we have already summed over all possible occurrences of the case. The second and fourth case together occur at most $\#X \cdot \#U$ times. We therefore obtain

$$\mathbf{Pr}(\mathsf{Coll}_{\mathsf{XU}} \text{ occurs}) \leq (\#X \cdot \#U) \cdot \max\{\gamma_1, \gamma_3\} + \Omega \# U \cdot \gamma_3 \,.$$

where $\#U$ is the number of U values in the transcript.

Let us bound the probability of $\mathsf{Coll}_{\mathsf{UU}}$. We consider the following cases:

(i) $k' = 1$.

a) $* \neq +$. The event is a contradiction as we require $U_1^{i*} \neq U_k^{j\circledast}$. The event is set with probability 0;

b) $* = +$. The event implies $\rho_2(T^{i+}) = U_k^{j\circledast}$ for some previous request $(j, \circledast)$ and some k. Since T^{i+} is always sampled uniformly at random and ρ_2 is γ_2-uniform, the probability of this event is at most γ_2;

(ii) $k' \neq 1$.

a) $* \neq +$. The event implies $U_1^{i*} \neq U_{k-k'+1}^{j\circledast}$. This implies that there is an index $h \leq k'$ where the sequence merges, that is $U_{k'-h-1}^{i*} \neq U_{k-h-1}^{j\circledast}$ and $U_{k'-h}^{i*} = U_{k-h}^{j\circledast}$. If $U_{k'-h-1}^{i*} \in \mathcal{L}$, the event already happened before and this was already a bad transcript. Else, if $U_{k'-h-1}^{i*} \notin \mathcal{L}$, the event occurs when $\rho_3(V_{k'-h-1}^{i*}) = U_{k-h}^{j\circledast}$ with $V_{k'-h-1}^{i*}$ sampled uniformly at random. As ρ_3 is γ_3-uniform, the probability of this event is at most γ_3;

b) $* = +$. The event implies $\rho_3(V^{i+}_{k'-1}) = U^{j\circledast}_k$ for some previous request $(j, \circledast)$ and some k. Since $c^{i+}_{k'-1}$ is always sampled uniformly at random, so is $V^{i+}_{k'-1}$. As ρ_3 is γ_3-uniform, the probability of this event is at most γ_3.

We obtain

$$\mathbf{Pr}(\mathsf{Coll}_{\mathsf{UU}}) \leq \binom{\#U}{2} \cdot \max\{\gamma_2, \gamma_3\}\,.$$

As for the probability of the event Forge, it is trivially bounded by 2^{-n} for every verification query as the value $T^{i\star}$ is always chosen before the sampling of $Y^{i\star}_{\ell^{i\star}_c}$. Indeed, $Y^{i\star}_{\ell^{i\star}_c}$ is either uniformly sampled after interactions with the oracles or, if there is a $(j,+)$ such that $((i,\star),(j,+)) \in \mathcal{Q}$, it is set to T^{j+} sampled at a later query $(i,\star) \prec (j,+)$. Thus, we obtain

$$\mathbf{Pr}(\mathsf{Forge}) \leq \frac{q_v}{2^n}\,.$$

Summing over everything we get:

$$\mathbf{Pr}X_{\mathrm{id}} \in \Theta_{\mathrm{b}} \leq$$
$$\left(\binom{\#X}{2} + (\#X \cdot \#U) + \binom{\#U}{2}\right) \cdot \max\{\epsilon_1, \gamma_1, \gamma_2, \gamma_3\} + \Omega \#U \cdot \gamma_3 + \frac{q_v}{2^n}\,.$$

The real world equivalent of $\#X$ is the number of inputs to f in the authentication part and $\#U$ the number of inputs to f in the encryption part, so we let $\#X + \#U = \sigma$ the total number of calls to f. Moreover, we have

$$\binom{\#X}{2} + (\#X \cdot \#U) + \binom{\#U}{2} = \binom{\#X + \#U}{2} = \binom{\sigma}{2}\,,$$

and we use $\#U \leq \sigma$ to get to the formula of Lemma 6.1. □

6.3.4 Analysis of Good Transcripts

Since we've defined the set of bad transcript Θ_{b} we define the set of good transcript as $\Theta_{\mathrm{g}} = \Theta \backslash \Theta_{\mathrm{b}}$. We show that for any good transcript $\tau \in \Theta_{\mathrm{g}}$ the probability of it happening is the same in the real and ideal worlds as stated in Lemma 6.2:

Lemma 6.2. *Let X_{re}, X_{id}, and Θ_{g} be as defined as above. For any good transcript $\tau = (\tau_e, \tau_v, \tau_d) \in \Theta_{\mathrm{g}}$,*

$$\frac{\mathbf{Pr}(X_{\mathrm{re}} = \tau)}{\mathbf{Pr}(X_{\mathrm{id}} = \tau)} = 1\,.$$

Probability in the Ideal World. Let $\tau = (\tau_e, \tau_v, \tau_d)$ be a good transcript. We note s_e the number of distinct X values among $\{X^{1+}, \ldots, X^{q_e+}\}$ and s_v the number of distinct X values among $\{X^{1\star}, \ldots, X^{q_v\star}\}$. Thus, there are $s_e + s_v$ values Y sampled after interaction.

For the encryption part, let C_e the number of ciphertext blocks randomly sampled during encryption queries, M_d the number of message blocks randomly sampled by the simulator during interaction and M_v for the ones sampled by the verification oracle after interaction. So we have:

$$\mathbf{Pr}(X_{\mathrm{id}} = \tau) = \left(\frac{1}{2^n}\right)^{C_e} \cdot \left(\frac{1}{2^n}\right)^{s_e+s_v} \cdot \left(\frac{1}{2^n}\right)^{M_d+M_v}\,.$$

Probability in the Real World. In the real world, the equivalent of a random sampling is to compute the random function f on a fresh input to get a uniformly distributed output. Therefore, all unique X values and U values are a unique input to the function f and thus also incurs a $1/2^n$ probability of observing the corresponding output.

For instance the fact that all $\{U^{1+}, \ldots, U^{q_e+}\}$ are unique also means that all C_e block of ciphertext are uniformly distributed. Moreover, it is clear that the decryption and verification oracles will compute f on $M_d + M_v$ fresh inputs to reconstruct the plaintext. Finally, since there are $s_e + s_v$ distinct X values in the transcript the corresponding Y values will also be uniformly distributed, so we have:

$$\mathbf{Pr}(X_{\mathrm{re}} = \tau) = \left(\frac{1}{2^n}\right)^{C_e} \cdot \left(\frac{1}{2^n}\right)^{s_e+s_v} \cdot \left(\frac{1}{2^n}\right)^{M_d+M_v}\,.$$

Therefore, we can directly conclude that the two probabilities are equals which prove Lemma 6.2.

6.3.5 AERUP Security of ANYDAE

We apply the H-coefficient technique (Theorem 6.2) with $\epsilon_{\text{ratio}} = 0$ following Lemma 6.2 and ϵ_{bad} as the bound in Lemma 6.1 to get the bound:

$$\mathbf{Adv}^{\mathsf{AERUP}}_{\mathtt{ANYDAE}-f}(\mathcal{A}) \leq \mathbf{Pr}(X_{\text{id}} \in \Theta_{\text{b}}) \leq \binom{\sigma}{2} \cdot \max\{\epsilon_1, \gamma_1, \gamma_2, \gamma_3\} + \Omega\sigma \cdot \gamma_3 + \frac{q_v}{2^n} \ .$$

Putting it together with the bound obtained with the PRP/PRF switch (Lemma 1.1), that is:

$$\mathbf{Adv}^{\mathsf{AERUP}}_{\mathtt{ANYDAE}-E}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{PRP}}_{E}(\mathcal{B}) + \binom{\sigma}{2} \cdot \frac{1}{2^n} + \mathbf{Adv}^{\mathsf{AERUP}}_{\mathtt{ANYDAE}-f}(\mathcal{A})$$

and taking the maximum over all adversaries $\mathcal{A}$ and $\mathcal{B}$ making queries with at most σ calls to the underlying block cipher, q_v verification queries (for $\mathcal{A}$) and running in time t and $t' \approx t$ respectively, we obtain the AERUP security bound of Theorem 6.1. □

Conclusion. We've thus proved the robustness of ANYDAE in the AERUP security model. With a single proof we showed that the deterministic AEAD mode ANYDAE is AE, PA1 and INT-RUP secure. Moreover, ANYDAE is described quite generically following the requirement of the proof. There are multiple possible instances of ANYDAE whose security is directly implied by its proof, and we've shown two of them: MONDAE that only slightly modifies SUNDAE to achieve AERUP security and TUESDAE that aim at being optimal in the number of block cipher queries for most inputs.

Part II
Idealized Designs

Chapter 7
Introduction to Idealized Designs

Idealized designs introduced in this chapter build new primitives upon existing primitives. Just like modes of operation, they rely on the security of an underlying cryptographic primitive. Therefore, the techniques we've seen for modes such as security games, proofs and even cryptanalysis will remain relevant when talking about idealized designs.

Idealizing existing constructions are introduced to provide a formal analysis of pre-existing designs strategy. For instance the DES block cipher follows the older Feistel Network strategy and, later, this strategy was idealized and formally studied by Luby and Rackoff [LR88] (Section 7.1.1). The FX construction (Section 7.1.3) was introduced by [KR96] to formally analyze DESX, the iterated Even-Mansour construction (Section 7.1.2) allowed [Bog+12] to formally analyze key-alternating ciphers that is the design strategy of the AES and tweakable block ciphers (Section 2.3.4) were formalized by [LRW11] with the goal of simplifying the analysis of the `OCB` mode.

7.1 Building Block Ciphers

7.1.1 Feistel Network

A Feistel network builds a block cipher using a family of pseudo-random functions. Let $F_{k_i}(\cdot)$ be a family of n to n bits functions indexed by a key and $k = k_1 \| \ldots \| k_r$. The $2n$-bit input message is first split into two n bit half to initialize the internal state $s_L^0 \| s_R^0 = m_L \| m_R$. A round is defined as $s_L^i \| s_R^i = s_R^{i-1} \| s_L^{i-1} \oplus F_{k_i}(s_R^{i-1})$ and the output after r rounds is $E_k(m) = c_L \| c_R = s_L^r \| s_R^r$.

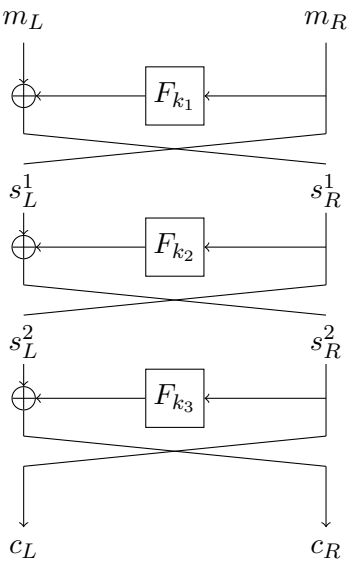


Figure 7.1: A 3-round Feistel network with input $m = m_L \parallel m_R$ and output $c = c_L \parallel c_R$.

As it is always invertible regardless of the underlying functions, a Feistel network define a family of a $2n$-bit permutations which is, indeed, a block cipher. To see this, we simply need to write the inverse round function: $s_L^{i-1} \parallel s_R^{i-1} = s_R^i \oplus F_{k_i}(s_L^i) \parallel s_L^i$.

Provable Security. This construction was first studied by Luby and Rackoff [LR88] as an idealization of the design strategy of the DES block cipher [DES77]. When all F_{k_i} functions are replaced by independent and random functions, they showed that 3 rounds (Figure 7.1) are sufficient to be prp secure (Definition 1.1) and 4 rounds to be sprp secure (Definition 1.2) both up to $2^{n/2}$ queries.

That bound can be interpreted as a birthday bound on the F_{k_i} functions and there are matching distinguishers for 3 and 4 rounds of the respective types exploiting collisions. Let us show a prp distinguisher on 3 rounds of Feistel:

1. For a fixed m_L, query $E_k(m)$ to find a collision $m_R \oplus c_L = m'_R \oplus c'_L$;

2. Take $m'_L \neq m_L$ and query $c^\star_L \parallel c^\star_R = E_k(m'_L \parallel m_R)$ and $c^\circ_L \parallel c^\circ_R = E_k(m'_L \parallel m'_R)$;

3. Check that the collision still holds: if $m_R \oplus c^\star_L \stackrel{?}{=} m'_R \oplus c^\circ_L$ return 1 else 0;

The first step is the most costly one as it looks for an n-bit collision. Its data and time complexity are thus the birthday bound $\mathcal{O}(2^{n/2})$. The idea of the attack is to look for a collision in $s^1_R = m_L \oplus F_{k_1}(m_R)$ (see Figure 7.1). Indeed, we have $c_L = m_R \oplus F_{k_2}(m_L \oplus F_{k_1}(m_R))$ so whenever there are two inputs m_R and m'_R such that we have a collision $F_{k_1}(m_R) = F_{k_1}(m'_R)$, we will observe, for a fixed m_L, that $c_L \oplus m_R = c'_L \oplus m'_R$. This relation is independent of the value m_L so it should still hold with a different one. Notice that it is also possible to observe $c_L \oplus m_R = c'_L \oplus m'_R$ because of a collision in F_{k_2} for two different values of s^1_R; in which case Step 3 will fail. However, the probability of success is already constant $\Omega(1)$ and can be arbitrarily improved by repeating the attack. As we only queried the construction in the forward direction, it is indeed a birthday bound distinguisher on the prp security of the 3 rounds construction.

Interestingly, the security of the 3 rounds Feistel breaks when we allow the adversary to query in the inverse direction. This makes it very easy to provoke a collision in the s^1_R internal state value. Let us describe the sprp distinguisher given in [LR88] using only 3 queries:

1. Query $E_k(m)$ for some $m = m_R \parallel m_L$. Get $c = c_L \parallel c_R$;

2. Query $E_k(m_L \oplus \delta \parallel m_R)$ for some $\delta \neq 0$. Get $c' = c'_L \parallel c'_R$;

3. Query the inverse $E^{-1}_k(c_L \parallel c_R \oplus \delta)$. Get $m' = m'_L \parallel m'_R$;

4. If $m_R \oplus c'_L \stackrel{?}{=} m'_R \oplus c_L$ return 1 else 0;

Indeed, with a 3-round Feistel network we observe that $m_R \oplus c'_L = m'_R \oplus c_L$ which is unlikely to happen with a truly random permutation. The attack exploits the fact that, for a given m_R, any difference in m_L propagates to s^1_R and, in the inverse direction, for a given c_R any difference to c_L also propagates backward to s^1_R. In the first query $E_k(m)$, the internal state value s^1_R is unknown, but in the second and third queries we know that it will become $s'^1_R = s^1_R \oplus \delta$ and, hence, be equal. We observe that this indeed happens as $m_R \oplus c'_L = F_{k_2}(s'^1_R) = m'_R \oplus c_L$.

7.1.2 Even-Mansour Construction

Even and Mansour [EM93] proposed an idealized design that builds a block cipher using a public permutation as $E_k(x) = P(x \oplus k_1) \oplus k_2$ parametrized by a $2n$-bit key $k = k_1 \parallel k_2$. They proved its security and it later became known as the Even-Mansour construction.

Their security analysis is equivalent to the sprp notion and thus satisfies the sPRP security game (Definition 1.2) but with a random permutation. Concretely, consider the Even-Mansour construction $E_k(x) = P_0(x \oplus k_1) \oplus k_2$ with a public permutation P_0. The sprp advantage of $E_k(\cdot)$ is:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{sprp}}_{E}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{E_k(\cdot),E_k^{-1}(\cdot),P(\cdot),P^{-1}(\cdot)} \to 1) \\ - \mathbf{Pr}(\mathcal{A}^{p(\cdot),p^{-1}(\cdot),P(\cdot),P^{-1}(\cdot)} \to 1)\ .\end{aligned}$$

where $P_0(\cdot)$ is replaced by $P(\cdot)$ in $E_k(\cdot)$, $k \xleftarrow{\$} \{0,1\}^{2n}$, P and p independently and uniformly drawn among all n to n bits permutations.

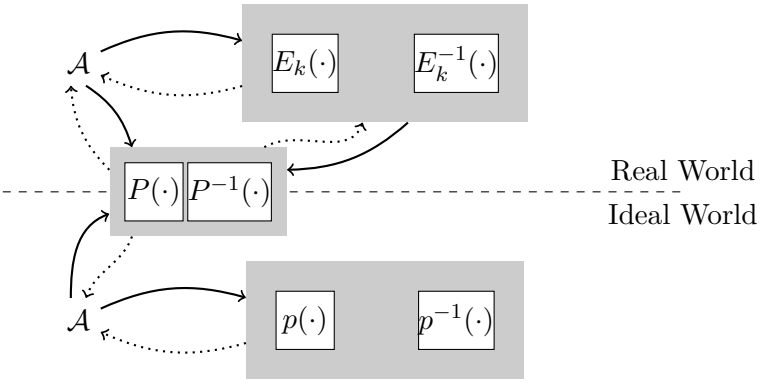


Figure 7.2: Distinguishing game for the sprp security of a block cipher E based on a public permutation where P and p are independent random permutations, k is a random key value.

Notice that the public permutation P_0 does not appear in the game and is replaced by a truly random permutation. As discussed in Section 2.3.4, this is impossible to formally define the security notion for a public permutation. Nevertheless, a low sprp advantage is a good indicator that the construction is secure given a "good enough" public permutation. In other words, this formalization captures the advantage of all attacker $\mathcal{A}$ that don't exploit any particular property of P_0.

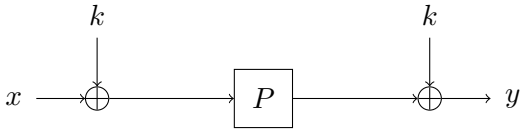


Figure 7.3: Single key Even-Mansour scheme $y = k \oplus P(k \oplus x)$.

So, let D be the number of query to the keyed construction and Q be the number of query to the public permutation, then the main result of [EM93] is that $\mathbf{Adv}_E^{\mathsf{sprp}}(\mathcal{A}) \leq \mathcal{O}(D \cdot Q/2^n)$. In particular, any attack with an $\Omega(1)$ success probability requires $D \cdot Q \geq \Omega(2^n)$. As P is public, query to its oracle are actually offline computations while query to E_k are online data. Therefore, the result is often stated as $D \cdot T \geq \Omega(2^n)$ with T the time complexity. This is also birthday-bound security as $D = T = 2^{n/2}$ is enough to have $D \cdot T = 2^n$ and there are matching attacks within those parameters.

A single-key version of the Even-Mansour construction (Figure 7.3) has also been proposed by Dunkelman, Keller and Shamir [DKS12], defined as $E_k(x) = P(x \oplus k) \oplus k$. They showed that the security remained the same even with this simplification. The single-key Even-Mansour construction is probably the simplest way to securely build a block cipher from a public permutation.

Cryptanalysis. There are multiple matching cryptanalyses of the Even-Mansour scheme in chosen plaintext [Dae93] and known plaintext [DKS15] that work for both the single key and the original versions. The main idea of those attacks is to exploit the fact that a difference Δ in the input of the construction will propagate to the input of the permutation. Concretely, by construction we have for any n-bit values x, y, Δ:

$$x \oplus y = k_1 \implies E_k(x) \oplus E_k(x \oplus \Delta) = P(y) \oplus P(y \oplus \Delta) \tag{7.1}$$

and the converse holds true with good probability. Therefore, if we randomly find x, y and Δ such that $E_k(x) \oplus E_k(x \oplus \Delta) = P(y) \oplus P(y \oplus \Delta)$, it is most likely the case that $x \oplus y = k_1$ and $E_k(x) \oplus P(y) = k_2$.

The slide attack of Daemen [Dae93] works by fixing a value for Δ and looking for a collision satisfying (7.1). See Algorithm 7.1.

Algorithm 7.1 Slide attack [Dae93] on Even-Mansour.

1: **input:** $E(x) = k_2 \oplus P(k_1 \oplus x)$.
2: **output:** (k_1, k_2) .
3: **procedure** SLIDEATTACK($E(\cdot), P(\cdot)$)
4: $\Delta \leftarrow \delta$ ▷ For any $\delta \neq 0$.
5: $f(x) \leftarrow E(x) \oplus E(x \oplus \Delta)$
6: $g(y) \leftarrow P(y) \oplus P(y \oplus \Delta)$
7: Let $\mathcal{X}$ and $\mathcal{Y}$ such that $\{x \oplus y : (x, y) \in \mathcal{X} \times \mathcal{Y}\} = \{0,1\}^n$
8: $(x, y) \leftarrow$ COLLISIONFUN($f(\cdot), g(\cdot), \mathcal{X}, \mathcal{Y}$)
9: **return** $(x \oplus y, E(x) \oplus P(y))$

The Step 8 of Algorithm 7.1 is about finding a collision and thus works in data and time complexity $\mathcal{O}(2^{n/2})$. More precisely, this attack works as soon as there exists x, y such that $x \oplus y = k_1$ to provoke Equation (7.1) which how we define the sets $\mathcal{X}$ and $\mathcal{Y}$ in Step 7. Therefore, after querying $D = |\mathcal{X}|$ different values x (it is easy to choose $\mathcal{X}$ such that $\forall x \in \mathcal{X} : x \oplus \Delta \in \mathcal{X}$), the attack requires $|\mathcal{Y}| = Q/2 = 2^n/D$ different values y. Hence, we indeed have $D \cdot Q = 2^{n+1}$ for a sure success and, if $D \leq Q$, $D \cdot T = 2^{n+1}$ as well. Notice that while we don't need to choose x, we need to query $x \oplus \Delta$ for a fixed Δ so this is indeed a chosen plaintext attack.

The slidex attack [DKS15], however, works with known plaintexts and the same trade-off complexity. The trick is to rewrite Δ as $k_1 \oplus \Delta$ since it is an arbitrary value and Equation (7.1) becomes:

$$x \oplus y = k_1 \implies E_k(x) \oplus P(x \oplus \Delta) = E_k(y \oplus \Delta) \oplus P(y) \tag{7.2}$$

The slidex attack is shown in Algorithm 7.2. It is a known plaintext attack where the set $\mathcal{X}$ is known but imposed. The data complexity is simply $D = |\mathcal{X}|$. The attack builds Γ lists of size D so the total number of queries to the permutation done in Step 3.1 is $Q = D \cdot \Gamma$. The expected number of collisions is $\Gamma \cdot \frac{D(D-1)}{2^{n+1}} = \frac{Q(D-1)}{2^{n+1}}$ so we have a good probability of success with $D \cdot Q = 2^{n+1}$ and, when $D \leq Q$, $D \cdot T = 2^{n+1}$.

In [DKS15] they notice that for a fixed Δ in Equation (7.2) we can look for a collision in $f(x) = E_k(x) \oplus P(x \oplus \Delta)$ using Algorithm 3.3 for memoryless collision search. The slide attack can also be done in

Algorithm 7.2 Slidex attack [DKS15] on Even-Mansour.

input: $E(x) = k_2 \oplus P(k_1 \oplus x)$.
output: (k_1, k_2) .
procedure SLIDEXATTACK($E(\cdot), P(\cdot)$)
    $L \leftarrow \{(x, E(x)) : x \in \mathcal{X}\}$ ▷ For an obersvable set $\mathcal{X}$ of size D.
    for all $\Delta \in \{\Delta_1, ..., \Delta_\Gamma\}$ **do** ▷ Arbitrary values $\Delta \neq 0$.
        $(x, y) \leftarrow$ COLLISION($\{E(x) \oplus P(x \oplus \Delta) : x \in \mathcal{X}\}$)
        if $(x, y) \neq \emptyset$ **then**
            return $(x \oplus y \oplus \Delta, E_k(x) \oplus P(y \oplus \Delta))$
    return $\emptyset$

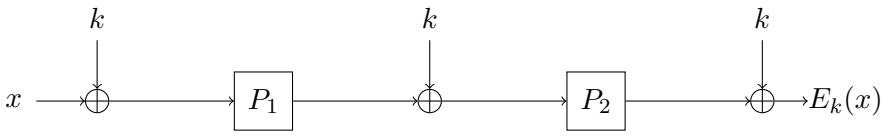


Figure 7.4: Single key two-round Even-Mansour scheme (2EM) with two independent permutations $E_k(x) = k \oplus P_1(k \oplus P_2(k \oplus x))$.

a memoryless manner to look for a collision. The complexity is still in $\mathcal{O}(2^{n/2})$ and this shows that the memory cannot be lower bounded in the way the time complexity is bounded by Q. These memoryless versions of the cryptanalysis are adaptively chosen plaintext attacks as not only do we need to choose the values of the queries, but they depend on the previous query.

Iterating Even-Mansour. Bogdanov et al. [Bog+12] proposed to generalize the simple Even-Mansour scheme by iterating it over multiple rounds. An r-round Even-Mansour scheme is based on r public permutations. First, a key is directly added to the input $s^0 = x \oplus k_0$ and a round is defined as $s^i = P_i(s^{i-1}) \oplus k_i$, the output after r rounds is s^r. This is an idealization of the design strategy of key alternating ciphers which interleaves key additions with known (and, in practice, simple) permutations. This design strategy is notably used in the AES [AES].

The iterated construction was first proven to be secure beyond the birthday bound and up to $\mathcal{O}(2^{2n/3})$ queries for $r \geq 2$ [Bog+12], and later improved to $\mathcal{O}(2^{nr/(r+1)})$ queries [LPS12; CS14] or, more generally, $D \cdot Q^r \geq \mathcal{O}(2^{rn})$. The proof holds even when a single key is reused at

every rounds like illustrated in Figure 7.4.

There is also a known information theoretic key recovery by Bogdanov et al.[Bog+12] that matches the best provable bound for all r. Algorithm 7.3 describes the attack on r-round Even-Mansour interleaving n-bit keys $k_1 \| \ldots \| k_{r+1}$ with permutations $P_1 \| \ldots \| P_r$. It works by

Algorithm 7.3 Generic attack [Bog+12] on r-round Even-Mansour.

1: **input:** $E(\cdot)$ is Even-Mansour with key $k_1 \| \ldots \| k_{r+1}$.
2: **output:** $k_1 \| \ldots \| k_{r+1}$.
3: **procedure** AttackrEM($E(\cdot), P_1(\cdot), \ldots, P_r(\cdot)$)
4: $L \leftarrow \{(x, E(x)) : x \in \mathcal{X}\}$ ▷ For an obersvable set $\mathcal{X}$ of size D.
5: **for** $i = 1$ to r **do**
6: $L_i = \{(x, P_i(x)) : x \in \mathcal{X}_i\}$ ▷ For a random set $\mathcal{X}_i$ of size Q.
7: **for all** $k \in \{0,1\}^{(r+1)n}$ **do**
8: $k_1 \| \ldots \| k_{r+1} \leftarrow k$
9: $b \leftarrow$ False ▷ Control whether we could reconstruct a path.
10: **for all** $x \in \mathcal{X}$ **do**
11: $s \leftarrow x$
12: **for** $i = 1$ to r **do**
13: **if** $(s \oplus k_i) \in \mathcal{X}_i$ **then**
14: $s \leftarrow P_i(s \oplus k_i)$ ▷ Value from L_i.
15: **else**
16: Go to next x (Step 10)
17: **if** $(s \oplus k_{r+1}) \neq E(x)$ **then** ▷ Value from L.
18: Go to next k (Step 7)
19: **else**
20: $b \leftarrow$ True ▷ Path successfully reconstructed.
21: **if** b **then**
22: **return** k
23: **return** $\emptyset$

guessing the key and reconstructing the internal state part (the value s in Algorithm 7.3) in order to test whether it is consistent with the queries made to the keyed construction. All the wrong key guesses will thus be discarded given that we have enough data to build a few paths for all guesses. To build a path we start from D possible values and proceed to keep values that belong to a set of size Q after a key addition. We iterate

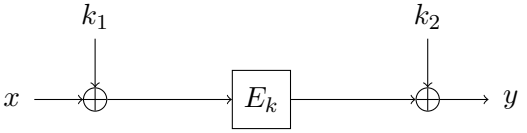


Figure 7.5: The FX construction $E'_{k'}(x) = k_2 \oplus E_k(k_1 \oplus x)$ with a $2n+\kappa$-bit key $k' = k \parallel k_1 \parallel k_2$.

this for every round, and, in expectation, we can build $D \cdot Q^r/2^{rn}$ paths. Therefore, we need a data-query complexity trade-off of $D \cdot Q^r = \Omega(2^{rn})$. The balanced case requires $D = Q = \Omega(2^{nr/(r+1)})$ matching the proof of [LPS12; CS14].

However, this attack requires a much larger number of computations than Q. For two and three rounds the best attacks run in $T = 2^n/n$ and for $r \geq 4$ there's no known attack with less that 2^n computations even in the single key variant. The gap between the information theoretic complexity and the computational complexity is arguably the largest for $r = 2$ rounds. Indeed, for two rounds an attack is possible with $D = Q = 2^{2n/3}$ but the best attack uses $T = 2^n/n$ computations. In Chapter 8 we devise new attacks on the 2-round Even-Mansour with a single key (Figure 7.4) by linking it to the 3-XOR problem (Section 3.2.2). Our approach optimizes both the data D and the memory while keeping $T = 2^n/n$. The link to the 3-XOR problem might give some insight on why such a gap exists. In particular, reducing the time complexity of solving the 3-XOR problem would reduce the time complexity of our two rounds Even-Mansour cryptanalysis.

7.1.3 FX Construction

The FX construction (Figure 7.5) builds a block cipher E' with a $2n+\kappa$-bit key based on another secure block cipher E with a κ-bit key as $E'_{k'}(x) = k_2 \oplus E_k(k_1 \oplus x)$. It has been first studied Killian and Rogaway [KR96] to analyze the DESX construction, a suggested solution by Rivest (according to [KR96]) that aimed at increasing the security of the DES [DES77] block cipher against brute-force attacks. Indeed, a generic brute-force key recovery attack requires $T = \mathcal{O}(2^\kappa)$ computations and [KR96] proved that the FX construction increases the complexity of the generic cryptanalysis to $T = \mathcal{O}(2^{\kappa+n}/D)$ when the attacker has access to D input/output pairs. In the case of DES, it has $\kappa = 56$ and $n = 64$ so this makes a

substantial difference. The FX construction has since be notably used in PRINCE [Bor+12] and PRIDE [Alb+14].

To prove such a security result, we need to use a stronger assumption than the sprp security of the underlying block cipher E. Indeed, $\mathbf{Adv}^{\mathsf{prp}}_{E}(t)$ necessarily tends to 1 as t tends to 2^{κ} since this is the complexity of an exhaustive key search. The assumption used here is called the ideal cipher model. In the ideal cipher model, the actual block cipher is replaced by a family of independently and uniformly drawn random permutations. In particular, the actual block cipher is implicitly required to resist cryptanalysis such as related key attacks where a relation between two keys implies a relation between the two induced permutations. On the other hand, the lack of randomness makes it impossible to formally define such a security notion within a distinguishing game hence the actual block cipher is altogether ignored for the proof. Notice that we couldn't formally define the security of a public permutation for the same subtle reasons.

Algorithm 7.4 Key recovery on FX construction.

1: **input:** $E'(x) = k_2 \oplus E_k(k_1 \oplus x)$.
2: **output:** (k, k_1, k_2) .
3: **procedure** AttackFX($E'(\cdot), E_{\cdot}(\cdot)$)
4: $L \leftarrow \{(x, E(x)) : x \in \mathcal{X}\}$ ▷ For an obersvable set $\mathcal{X}$ of size D.
5: **for all** $k \in \{0,1\}^{\kappa}$ **do**
6: $(k_1, k_2) \leftarrow$ SlidexAttack($E'(\cdot), E_k(\cdot)$) ▷ Provide the set L.
7: **if** $(k_1, k_2) \neq \emptyset$ **then**
8: **return** (k, k_1, k_2)
9: **return** $\emptyset$

Cryptanalysis. There is a simple matching key recovery working with known plaintexts given in Algorithm 7.4. The attack exploits the fact that once the right key k is guessed, the FX construction is reduced to an Even-Mansour scheme as shown in Section 7.1.2. Step 6 will always reuse the initially observed values and, in addition, will require $Q = \mathcal{O}(2^n/D)$ computations to perform the slidex attack for each guess of k. Therefore, the total time complexity is indeed $T = \mathcal{O}(2^{\kappa} \cdot Q) = \mathcal{O}(2^{\kappa+n}/D)$ matching the lower bound of [KR96].

Algorithm 7.5 Key recovery on r-round FX construction [Gaž13].

input: E' is r rounds FX with $k_1, k_2, ..., k_{r+1}$ and $E_{k_1^\star}, E_{k_2^\star}, ..., E_{k_r^\star}$.
output: $(k_1 \parallel \ldots \parallel k_{r+1}, k_1^\star \parallel \ldots \parallel k_r^\star)$.
procedure AttackrFX($E'(\cdot), E.(\cdot)$)
    $L \leftarrow \{(x, E(x)) : x \in \{0,1\}^n\}$ ▷ Query the whole codebook.
    for all $i \in \{0,1\}^\kappa$ **do**
        $L_i \leftarrow \{(x, E_i(x)) : x \in \mathcal{X}_i\}$ ▷ For a random set $\mathcal{X}_i$ of size $Q/2^\kappa$.
    for all $k^\star \in \{0,1\}^{r\kappa}$ **do**
        $k_1^\star \parallel k_2^\star \parallel \ldots \parallel k_r^\star \leftarrow k^\star$
        $k \leftarrow$ AttackrEM($E'(\cdot), E_{k_1^\star}(\cdot), \ldots, E_{k_r^\star}(\cdot)$)
                ▷ Provide the sets $L, L_{k_1^\star}, \ldots, L_{k_r^\star}$.
        if $k \neq \emptyset$ **then**
            return $(k, k^\star)$
    return $\emptyset$

Iterating FX. We can iterate r rounds of the FX construction to build a block cipher with a $(r+1)n + r\kappa$-bit key upon a κ-bit block cipher (or r different block ciphers which is equivalent in the ideal cipher model). The best information theoretic key recovery on r-round iterated FX construction is due to Gaži [Gaž13] and makes $Q = \mathcal{O}(2^{\frac{r-1}{r}n+\kappa})$ queries. Algorithm 7.5 gives the attack on an r-round iterated FX $E'_k(x)$ interleaving $k_1, k_2, ..., k_{r+1}$ n-bit keys with $E_{k_1^\star}, E_{k_2^\star}, ..., E_{k_r^\star}$ block ciphers. All the queries are done beforehand so that Step 9 does not make any additional query. To optimize the complexity the attack queries the construction for all possible inputs so $D = 2^n$. Algorithm 7.5 succeeds if the sets L_i are sufficiently big so that Algorithm 7.3 succeeds for the right guess of $k^\star$. The required trade-off is thus $D \cdot (Q/2^\kappa)^r = \mathcal{O}(2^{rn})$. Since we have $D = 2^n$, this implies a query complexity of $Q = \mathcal{O}(2^{\frac{r-1}{r}n+\kappa})$.

7.2 Other Designs

Block ciphers are not the only idealized designs that are provable schemes. In this section we'll show how one can build hash functions and tweakable block ciphers based on standard block ciphers or public permutations.

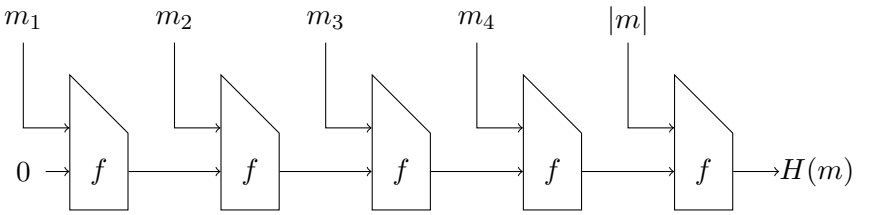


Figure 7.6: Merkle-Damgård construction with a compression function f.

7.2.1 Building Hash Function

Ideally, a hash function is a public function that produces a random looking output (of fixed or arbitrary length) from an arbitrary long input. However, a hash is not a PRF since it is a keyless construction. In practice, the security requirement of a hash depends on the usage. One typical usage is to guarantee the integrity of large volume of data by only comparing a relatively short hash value, for that we need our hash function to be collision resistant.

Merkle-Damgård construction. A collision resistant hash function can be built by iterating a compression function f with fixed sized input and output α and β respectively with $\alpha > \beta$. This is the Merkle-Damgård construction as shown in Figure 7.6 that outputs a β-bit hash value. Given a padding scheme that appends the bit length of the input, Merkle and Damgård [Mer79; Dam90] independently proved that finding a collision on the hash construction $H(m)$ is as hard as finding a collision on the underlying compression function $f(x_1, x_2)$. In other word, the Merkle-Damgård construction is collision resistant as long as f is collision resistant.

Interestingly, Joux [Jou04] showed that finding a multi-collision (finding multiple inputs matching the same output) on the Merkle-Damgård construction is much easier than in the generic case independently of the compression function f. Indeed, [Jou04] exhibits a 2^u multi-collision attack on Merkle-Damgård for the complexity of looking for u collisions in f. We explicitly describe the attack in Algorithm 7.6. For instance, a generic 4-collision algorithm on a β-bit output requires at least $\Omega(2^{3\beta/4})$ data, but on Merkle-Damgård it can always be done in $\Omega(2^{\beta/2})$ data that is the cost of looking for 2 collisions.

Nevertheless, it is a widely spread construction notably used in the hash functions MD5, SHA-1 and SHA-2.

Algorithm 7.6 Multi-collision attack on Merkle-Damgård [Jou04].

input: f is a α to β-bit function for the MD construction H.
output: $\mathcal{M}$ of size 2^u such that $H(p \,\|\, m_i) = H(p \,\|\, m_j) \forall m_i, m_j \in \mathcal{M}$.
procedure MULTICOLLISIONMD($f(\cdot,\cdot), p, u$)
    $p_1 \,\|\, \ldots \,\|\, p_\ell \leftarrow p$
    $s \leftarrow 0$
    for $i = 1$ to ℓ **do**
        $s \leftarrow f(s, p_i)$ ▷ Internal state after processing p.
    for $i = 1$ to u **do**
        $(m_0^i, m_1^i) \leftarrow$ COLLISION($f(s,\cdot)$)
        $s \leftarrow f(s, m_0^i)$
    return $\{m_{b_1}^1 \,\|\, m_{b_2}^2 \,\|\, \ldots \,\|\, m_{b_u}^u : (b_1, b_2, \ldots, b_u) \in \{0,1\}^u\}$

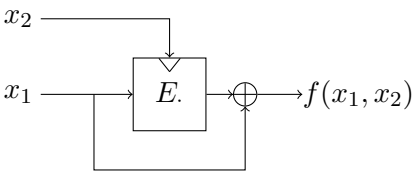


Figure 7.7: Davies-Meyer compression function $f(x_1, x_2) = E_{x_2}(x_1) \oplus x_1$.

Davies-Meyer. With the Merkle-Damgård construction, the problem of building a collision resistant hash function is reduced to building a collision resistant compression function. It turns out that one can build such a function with a block cipher as $f(x_1, x_2) = E_{x_2}(x_1) \oplus x_1$. This is the Davies-Meyer compression function. Winternitz [Win83] proposed this construction while attributing the idea to Davies who denied it and attributed the idea to Meyer (According to [PGV94]). Later, Winternitz himself proved the construction secure as a collision resistant one-way compression function under the ideal cipher model [Win84]. The provable security is optimum meaning that if the cipher is an ideal cipher, then there is no collision attack on the Davies-Meyer construction faster than the generic birthday bound attack. Note that there are other secure

compression functions based on block ciphers that are notably discussed in [PGV94], but the Davies-Meyer construction is the most popular one.

A block cipher can thus serve as a basis to build a collision resistant hash function in addition to authenticated encryption modes. This is especially interesting in restricted environments: one only needs to implement a good block cipher to have access to many cryptographically secure constructions. On the downside, the Davies-Meyer construction requires to remember the input value in order to XOR it with the output which increases the internal state size.

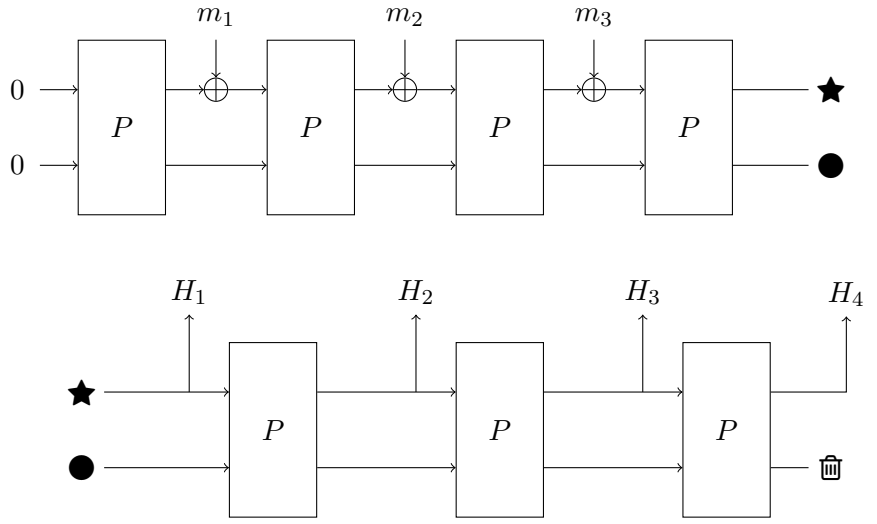


Figure 7.8: The sponge construction for an arbitrary long hash based on a public permutation P. The diagram shows $H(m_1 \parallel m_2 \parallel m_3) = H_1 \parallel H_2 \parallel H_3 \parallel H_4$.

Sponge construction. We've shown in Section 2.3.4 how to build an authenticated encryption schemes out of a public permutation with the `SpongeWrap` mode. In fact, the security of such a mode is implied by the security of the sponge construction for a hash function as shown in Figure 7.8 (each r-bit block of key stream is seen as the output of a different hash from a random permutation). One particular aspect of a sponge based hash function is that the output is also of arbitrary length: it is easy to keep unrolling the construction to output as many bits as we want. The construction is based on an n to n bit permutation P separated

into an α-bit rate (or outer-part) and a β-bit capacity (or inner-part). Bertoni, Daemen, Peeters and Van Assche [Gui+11] proved that a sponge construction instantiated with a random permutation is behaving like a random function as long as there is no collision in the capacity part. This means that there are no better attack than generic attacks up to $\mathcal{O}(2^{\beta/2})$ computations of P that is not exploiting some property of P. Indeed, the proof replaces P by a random permutation.

The sponge construction therefore offers another way of building a variety of provably secure cryptographic functions from a single secure primitive. At the time of this writing, the sponge construction is mainly used for hashing as the SHA-3 standard [Dwo15] is a sponge using a `Keccak-`f permutation. However, we've lately seen many proposals for authenticated encryption schemes relying on the sponge construction, notably as part of the NIST lightweight competition and the CAESAR competition.

7.2.2 Building Tweakable Block Ciphers

Tweakable block ciphers have been formalized by Liskov, Rivest and Wagner [LRW11] as a family of permutation indexed by both a secret key and a public tweak that is an application $\widetilde{E} : \{0,1\}^\kappa \times \{0,1\}^\tau \times \{0,1\}^n \to \{0,1\}^n$. Since it is a keyed construction, its security notion can be made into a distinguishing game. We define the $\widetilde{\mathsf{prp}}$ and $\widetilde{\mathsf{sprp}}$ security advantage for a tweakable block cipher $\widetilde{E}_k(t,x)$ in an analogous way to the prp and sprp notions for block ciphers, that is:

$$\mathbf{Adv}^{\widetilde{\mathsf{prp}}}_{\widetilde{E}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\widetilde{E}_k(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{p(\cdot,\cdot)} \to 1)\ ,$$

and

$$\mathbf{Adv}^{\widetilde{\mathsf{sprp}}}_{\widetilde{E}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\widetilde{E}_k(\cdot,\cdot),\widetilde{E}_k^{-1}(\cdot,\cdot)} \to 1) - \mathbf{Pr}(\mathcal{A}^{p(\cdot,\cdot),p^{-1}(\cdot,\cdot)} \to 1)$$

with $k \xleftarrow{\$} \{0,1\}^\kappa$ and $p(t,\cdot)$ an independent random n to n-bit permutation for all $t \in \{0,1\}^\tau$.

Liskov et al. [LRW11] proposed two constructions of tweakable block ciphers from block ciphers, known as `LRW1` and `LRW2` and described as $\widetilde{E}_k(t,m) = E_k(t \oplus E_k(m))$ and $\widetilde{E}_k(t,m) = E_k(m \oplus h(t)) \oplus h(t)$ respectively with the requirement that h be an almost XOR-universal function (Definition 9.1). Those constructions are proven $\widetilde{\mathsf{prp}}$ and $\widetilde{\mathsf{sprp}}$ secure up

to the birthday bound assuming an underlying prp and sprp secure block cipher respectively. There are simple matching attacks with complexity $\mathcal{O}(2^{n/2})$ at the first collision.

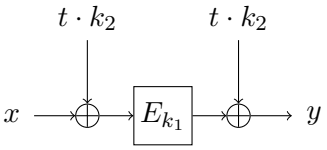


Figure 7.9: The Xor-Encrypt-Xor (XEX) tweakable block cipher with a key $k = k_1 \parallel k_2$, a tweak t and where $y = \widetilde{E}_k^t(x) = E_{k_1}(x \oplus t \cdot k_2) \oplus t \cdot k_2$ with $t \cdot k_2$ a Galois field multiplication.

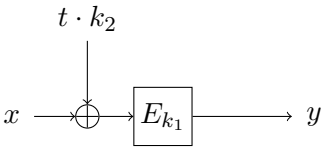


Figure 7.10: The Xor-Encrypt (XE) tweakable block cipher with a key $k = k_1 \parallel k_2$, a tweak t and where $y = \widetilde{E}_k^t(x) = E_{k_1}(x \oplus t \cdot k_2)$ with $t \cdot k_2$ a Galois field multiplication.

XEX and XE constructions. The original motivation of [LRW11] was to simplify the proof of the mode OCB. In Section 2.3.4, the OCB mode is shown as a mode relying on a tweakable block cipher while the full specification of OCB [KR14] relies on the XEX and XE construction to build a tweakable block cipher out of a standard block cipher as shown in Figure 7.9 and 7.10 respectively.

The exact way the XEX and XE constructions are used in OCB is extensively analyzed in [Rog04] and so is the security as OCB uses both constructions with the same key k_1 and derive the secret value k_2 from the nonce. However, the proof of security for XEX and XE, as shown in Figures 7.9 and 7.10 respectively, is derived from [LRW11]. Indeed, XEX is an instantiation of LRW2 with the AXU function $h(t) = t \cdot k_2$ where multiplication is done in a Galois Field. Therefore, the proof of [LRW11] applies showing that XEX is $\widetilde{\text{sprp}}$ secure up to the birthday bound assuming a sprp secure E. It is also easy to adapt the proof to show the $\widetilde{\text{prp}}$ security of the XE construction assuming a prp secure E.

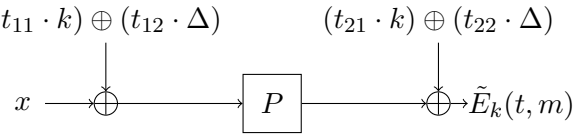


Figure 7.11: The XPX construction with key k and a secret value $\Delta = P(k)$. The tweak is $t = t_{11} \parallel t_{12} \parallel t_{21} \parallel t_{22}$ and is multiplied in a Galois Field.

The XEX constructions is also notably used for disk encryption within the XTS mode of operation [Dwo10].

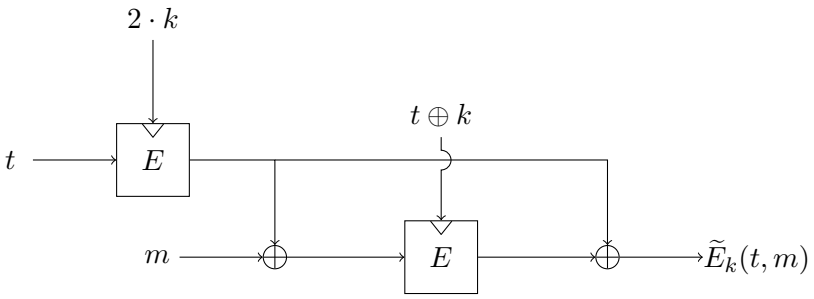


Figure 7.12: The $\widetilde{F}[2]$ construction. Multiplications are done in a Galois Field.

Other Constructions. There are several other constructions of tweakable block ciphers. For instance the XPX construction by Mennink [Men16] (Figure 7.12) that reaches a birthday bound $\mathcal{O}(2^{n/2})$ security by building a tweakable block cipher based on a public permutation. There are also constructions where the tweak influences the key of the block cipher. Those constructions need to be proven secure in the ideal cipher model. This is the case of the $\widetilde{F}[2]$ construction by Mennink [Men15] (Figure 7.12) that reaches a $\mathcal{O}(2^n)$ $\widetilde{\mathsf{sprp}}$ security. There is also the XHX construction by Jha, List, Minematsu, Mishra and Nandi [Jha+17] (Figure 7.13) who is $\widetilde{\mathsf{sprp}}$-secure up to $\mathcal{O}(2^{(n+\kappa)/2})$ queries. This is interesting as it shows that security beyond 2^n is achievable in the ideal cipher setting.

Exploring in that direction, Lee and Lee proposed to iterate two independent XHX with the XHX2 construction [LL18] (Figure 7.14). They proved that it is $\widetilde{\mathsf{sprp}}$ secure up to $\min\{2^{\frac{2}{3}(n+\kappa)}, 2^{n+\kappa/2}\}$ queries and left

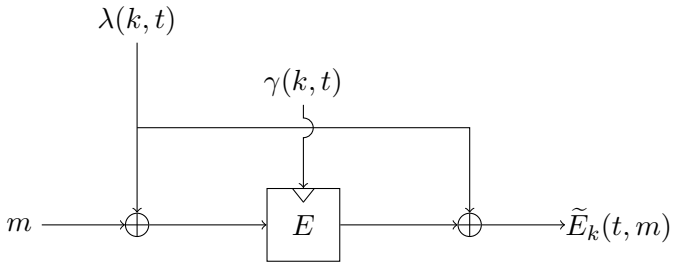


Figure 7.13: The `XHX` construction with a master key k, an almost universal function $\gamma(\cdot,\cdot)$ and an almost XOR-universal $\lambda(\cdot,\cdot)$.

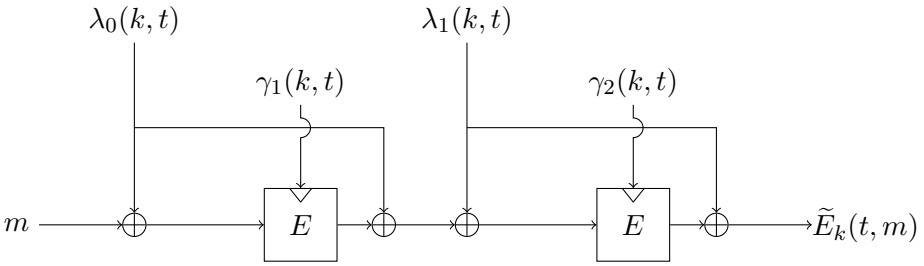


Figure 7.14: The `XHX2` construction is the cascade of two independent `XHX` with almost universal functions $\gamma 1(\cdot,\cdot), \gamma 2(\cdot,\cdot)$ and almost XOR-universal functions $\lambda_0(\cdot,\cdot), \lambda_1(\cdot,\cdot)$.

the tightness of the bound as an open question. In Chapter 9 we present a cryptanalysis of `XHX2` running in $\mathcal{O}(2^{\frac{2}{3}(n+\kappa)})$ showing the tightness the bound of Lee and Lee for $\kappa \leq 2n$. In fact, we generalize this design strategy by interpreting it as follows: First, expand the key space with an r-round iterated FX construction (seen in Section 7.1.3), and then compute the subkey by blending the master key with the tweak. Each subkeys become a function of the tweak and the master key, and we call the resulting scheme the iterated tweakable FX construction. Therefore, we also generalize the cryptanalysis to attack r rounds of the generic iterated tweakable FX construction with a query complexity of $\mathcal{O}(2^{\frac{r}{r+1}(n+\kappa)})$.

Chapter 8
Low-Memory Attack on 2-round Even-Mansour

Contributions brought forward in this chapter were published in Crypto 2019 and are a joint work of Leurent and I [LS19].

Introduction

We've seen in Section 7.1.2 that iterated Even-Mansour is an idealization of the SPN strategy to build block ciphers notably used for the AES. Although tight bounds for information theoretic is now known for all r-round iterated Even-Mansour, attacks in the computational setting may not match this bound. The gap is arguably the largest for the 2-round version. Indeed, while there is a tight information security proof and a matching attack in $\mathcal{O}(2^{2n/3})$ queries, the best attacks on single keyed 2-round Even-Mansour have a time complexity of $\mathcal{O}(2^n/n)$. Even worse, the best strategies also have either the data or the memory complexity in the order of $\mathcal{O}(2^n/n)$ making them hardly better than a brute-force attack that has negligible data and memory usages.

In this chapter we show how to use algorithms solving the 3-XOR problem seen in Section 3.2.2 in order to perform novel cryptanalyses of 2-round Even-Mansour. Those cryptanalyses reduce for the first time both the data and memory usages to way below 2^n while keeping a time complexity of $\mathcal{O}(2^n/n)$. In addition, we find the same kind of gap we explored in Section 3.2.2 hinting that it may be really hard to get a better time complexity. More precisely, we show a reduction that implies that an improved 3-XOR algorithm would lead to an improved 2-round Even-Mansour attack.

Our results. The main contributions of the chapter are key-recovery attacks on single key 2-round Even-Mansour (2EM). Those attacks are

given in Section 8.3 and their complexities are summarized in Tables 8.1 and 8.2. These are the first attacks on 2EM to significantly reduce simultaneously the data and the memory complexities below 2^n. In Section 8.3.2 we show how to use a 3-XOR solver in a black-box manner to attack 2EM and propose some instantiations of the solver with known 3-XOR algorithms seen in Section 3.2.2. In particular, when using the generalized approach by Bouillaguet, Delaplace and Fouque [BDF18], this shows that we can achieve the best computational time complexity known so far, that is $\mathcal{O}(2^n/n)$, while using just as much data and queries as the best known distinguisher which is optimal in the balanced case ($2^{2n/3}$ calls to E, P_1 and P_2) with a memory usage not exceeding the number of queries. We can also use the Baran, Demaine and Pătraşcu [BDP08] algorithm to further improve the asymptotic time complexity to $\mathcal{O}(2^n \ln^2 n/n^2)$ which beats the best one known so far. Unfortunately, this 3-XOR algorithm is impractical for realistic block sizes, notably for $n \leq 10^6$. Our last attack shown in Section 8.3.3 exploits the specific properties of the 3-XOR problem derived from 2EM to improve those generic results. This results in a very low data attack, λn online queries, with low memory, $2^{\lambda n}$, for some $\lambda < 1$ while keeping a competitive asymptotic time complexity of $\mathcal{O}(2^n/\lambda n)$.

We also present some security reduction in Section 8.2 notably showing that adding a linear key schedule does not protect against generic attacks. This effectively extends the scope of our attacks in particular showing they can also be applied to various variants. Then, we exhibit a symmetry in the Even-Mansour construction that shows how, in the sprp security game, an attacker can always swap the number of queries he is making to E, P_1 and P_2 to optimize on the most available resources. This result implicitly extends our and previous attacks to adapt to many data and query complexity profiles.

Lastly, we generalize our approach in Section 8.4.1 to show that a single key r-round Even-Mansour scheme can be rewritten as a structured $(r+1)$-XOR problem with words of size rn. Interestingly, both the single key r-round Even-Mansour and the $(r+1)$-XOR problem with words of size rn have a simple information theoretic solver using $2^{\frac{r \cdot n}{r+1}}$ queries though solving these uses more computations than a brute-force solution for $r \geq 4$.

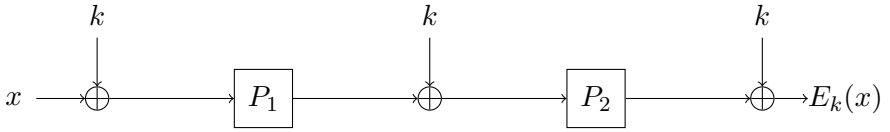


Figure 8.1: Single key two-round Even-Mansour scheme (2EM) with two independent permutations $E_k(x) = k \oplus P_1(k \oplus P_2(k \oplus x))$.

8.1 Previous Results

There are two kinds of single-key 2-round Even-Mansour schemes with a security proof, one with two independent permutations (EMIP) and one with a single permutation and a fixed linear orthomorphism (that is a linear operation such that $x \mapsto \pi(x)$ and $x \mapsto x \oplus \pi(x)$ are invertible) such as a doubling in a $\mathbf{GF}(2^n)$ Galois Field (EMSP):

$$\begin{aligned} \text{EMIP}: &\quad E_k(x) = P_2\big(P_1(x \oplus k) \oplus k\big) \oplus k \\ \text{EMSP}: &\quad E_k(x) = P\big(P(x \oplus k) \oplus \pi(k)\big) \oplus k, \quad \pi \text{ a linear orthomorphism.} \end{aligned}$$

In this chapter we'll focus on the EMIP variant as we'll see in Section 8.2.1 that any attack on EMIP is easily translated to an attack on EMSP. We refer to single key 2-round Even-Mansour EMIP as simply 2EM.

Multi-collision based Cryptanalysis. The first non-trivial attack against 2EM was proposed by Nikolic, Wang, and Wu [NWW14] using multi-collisions. They consider the function $\phi : u \mapsto P_1(u) \oplus u$ and evaluate it on many points to find a value v that occurs multiple times (say ν times). Then, for each known plaintext pair $(x, E(x))$ they assume that $\phi(x \oplus k) = v$ implying that $P_1(x \oplus k) \oplus k = x \oplus v$ and thus they guess $k \stackrel{?}{=} P_2(x \oplus v) \oplus E(x)$. Since $\phi(\cdot)$ has at least $\nu/2^n$ chances to be equal to v, the expected complexity is $2^n/\nu$.

Following the asymptotic analysis of [NS15], the algorithm is optimal when $\nu = \theta(n/\ln n)$. Indeed, a value v repeating $\theta(n/\ln n)$ times is expected to be found after evaluating $\phi(\cdot)$ about $2^n/n$ times so that the total complexity of this attack is $2^n \cdot \ln n/n$ which is asymptotically smaller than 2^n.

Dinur, Dunkelman, Keller and Shamir [Din+13] later improved this attack reducing its data complexity. Their variant looks for N_v different values v_i appearing at least ν times each for a smaller value of ν. Thus,

they make N_v guesses of the key for each known plaintext pair $(x, E(x))$ which reduce the data complexity to $2^n/(N_v \cdot \nu)$. They computed that after evaluating the function on $\mu 2^n$ points then $N_v = 2^n \mu^\nu e^{-\nu}/\nu!$ multi-collisions should be found. In particular, setting $\mu = 1/n$ and $\nu = o(n/\ln n)$, an upper bound on the data complexity is given by $2^n/N_v \leq n^{2\nu} = e^{2\nu \ln n}$ which is $2^{o(n)}$. The time complexity remains at $2^n/\nu$.

Dinur *et al.* also proposed attacks against the variant construction with 3 independent keys, using multi-collisions to find differential properties of the random permutation. However, this attack only reaches time complexity $\mathcal{O}(2^n/\sqrt{n/\ln n})$.

Overall, multi-collisions attacks require large pre-processing to locate such a ν-collision that is only expected after $2^{n(\nu-1)/\nu}$ evaluations of $\phi(\cdot)$. Moreover, the best known multi-collision algorithm requires $2^{n(\nu-2)/\nu}$ memory [JL09]. Therefore, those techniques intrinsically require time and memory close to 2^n (asymptotically, we need to have ν approaching infinity in order to gain a non-constant advantage over brute-force attacks).

Other Approaches. In the journal version of their paper, Dinur *et al.* show an interesting side-result on EMIP. They describe an alternative attack with low memory using linear algebra [Din+16, Section 4.2]. In this attack, they evaluate $\phi : u \mapsto P_1(u) \oplus u$ on a small set of λn values $(0 < \lambda < 1/3)$, and they look for linear relations that are satisfied by all $\phi(u)$ in the set: $L(\phi(u)) = 0$ with $n - \lambda n$ equations. Then, for a given plaintext pair $(x, E(x))$, if $x \oplus k$ is in the set, this implies linear relations on $z = k \oplus P_1(x \oplus k)$, the input of P_2: $L(z) = L(x)$. Finally, using structures for x and z, a match can be identified using linear relations on the key (following from the assumption that $x \oplus k$ is in the set), using $k = P_2(z) \oplus E(x)$. The full details of the attack are given in [Din+16]. This attack only requires a memory of size $2^{\lambda n}$ to store the structures, but it requires $2^n/\lambda n$ chosen plaintext pairs. In fact, our low data memory attack can be seen as an improved version of their attack. We comment on the similarities and improvements at the end of Section 8.3.3.

Another approach by Isobe and Shibutani [IS17] introduced Meet-in-the-Middle techniques to attack the 2-round Even-Mansour construction. The basic variant of their attack uses a function f depending on a bits of the key k_f (with a in the order of $\ln n$), and a function g depending on the remaining $n - a$ bits k_g. Furthermore, they use a starting point such that a output bits of f are actually independent of the key k_f. This

allows them to do the matching over P_2 using just k_g. The attack works with chosen plaintexts and has a time and data complexity of 2^{n-a}.

Actually, the function f is equivalent to looking for partial multi-collisions in $\phi(\cdot)$ while imposing a structure on the inputs: they fix $n-a$ bits of u and hope that a outputs bits of $\phi(u)$ will be independent of the remaining a bits of u. The parameter a must satisfy $a \cdot (2^a - 1) \leq n - a$, and Isobe and Shibutani [IS17] only give concrete parameters for some values of n. Asymptotically, the maximal value of a can be found by solving $a \cdot (2^a - 1) = n - a$; since $a \lll n$ and $1 \lll 2^a$, we have $a \approx W(n \ln 2)/\ln 2 \approx \log n - \log\log n$, using the Lambert W function (remember that ln and log are the natural and base 2 logarithms, respectively).

They also describe a low data-complexity variant of the attack where the starting point is dynamically chosen so that $a + d$ bits of the plaintext are fixed. This reduces the data complexity to 2^{n-d-a}, while the time complexity is still 2^{n-a}. The parameters are more constrained and must satisfy $a \cdot 2^a + d \leq n - a$. If we want to achieve a data complexity of $2^{\lambda n}$ for a constant $0 < \lambda < 1$, we can set $d = n - \lambda n$, and $a = \log \lambda + \log n - \log\log n$. This gives a time complexity of $2^n \log n / \lambda n$.

Finally, they give a time-optimized attack where $b = a + c$ output bits of f are independent of k_f (instead of just a). This reduces the number of queries and memory needed for the matching to 2^{n-b}, but the attack still requires 2^{n-a} memory accesses and chosen plaintext. The parameters must satisfy $b \cdot 2^a + b - a \leq n - b$, but the authors only give concrete values for some choices of n, and no asymptotic analysis. However, we can observe that we must have $b \cdot 2^a \leq n$; in particular, if we want an attack with an advantage that is not asymptotically bounded, we need to have a approaching infinity and therefore b/n approaching zero (this attack cannot reduce the memory to $2^{\lambda n}$ with $\lambda < 1$). In particular, the optimal parameters satisfy $b \cdot 2^a + b - a = n - b$, with $b \lll n$ and $a \lll 2^a$, hence $b \cdot 2^a \approx n$. Therefore, we have a complexity of roughly 2^{n-b} in queries and memory, and $b2^n/n$ in time and data, with $\log n \leq b \lll n$.

All those attacks are summarized in Tables 8.1 and 8.2. We point out that the complexity reported in [IS17] are lower than listed here, because the authors assume that a memory access to a large table is significantly cheaper than the evaluation of the public permutations P_i. Given that a public permutation can obviously be implemented with a table lookup if memory is fast and cheap, we assume that a memory access to a table of size roughly 2^n cannot be faster than the evaluation of the P_i permutations.

Ref	Data	Queries	Time	Memory	Comment
[NWW14]	KP $2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	Multi-collisions
[Din+13]	CP $2^n\sqrt{\ln n/n}$	$2^n\sqrt{\ln n/n}$	$2^n\sqrt{\ln n/n}$	$2^n\sqrt{\ln n/n}$	Diff. m-c (indep. keys)
[Din+13]	KP $2^{\lambda n}$	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	Multi-collisions
[Din+16]	CP $2^n/\lambda n$	$2^n/\lambda n$	$2^n/\lambda n$	$2^{\lambda n}$	Linear algebra
[IS17]	CP $2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	MitM
	CP $2^{\lambda n}$	$2^n \ln n/n$	$2^n \ln n/n$	$2^n \ln n/n$	MitM
	CP $2^n\beta/n$	$2^n/2^\beta$	$2^n\beta/n$	$2^n/2^\beta$	MitM
S. 8.3.1	KP n	$2^n/\sqrt{n}$	$2^n/\sqrt{n}$	$2^n/\sqrt{n}$	3XOR [Jou09]
S. 8.3.2	KP 2^d	$2^{n-d/2}$	$2^n/n$	$2^{n-d/2}$	3XOR [BDF18]
S. 8.3.2	KP 2^d	$2^{n-d/2}$	$2^n \ln^2 n/n^2$	$2^{n-d/2}$	3XOR [BDP08]
S. 8.3.3	KP λn	$2^n/\lambda n$	$2^n/\lambda n$	$2^{\lambda n}$	Low Data Filter

Table 8.1: Comparison of attacks against 2EM. Asymptotic complexity, up to constants. "Data" denotes encryption queries, while "Queries" denotes calls to the public permutations P_i.
$0 < \lambda < 1$; $\log n \leq \beta \lll n$; KP: Known plaintext; CP: Chosen plaintext.

Ref	Data	Queries	Time	Memory	Comment
[NWW14]	KP $2^{58.7}$	$2^{60.5}$	$2^{60.9}$	2^{60}	Multi-collisions
[Din+13]	KP 2^{45}	$2^{60.7}$	$2^{60.7}$	2^{60}	Multi-collisions
[Din+16]	CP 2^{60}	2^{59}	$2^{60.6}$	2^{16}	Linear algebra
[IS17]	CP 2^{60}	2^{60}	$2^{61.3}$	2^{60}	MitM
	CP 2^{8}	2^{62}	$2^{62.6}$	2^{62}	MitM
	CP 2^{61}	2^{57}	$2^{61.7}$	2^{58}	MitM
Sec. 8.3.1	KP 2^{6}	2^{61}	2^{62}	2^{61}	3XOR
Sec. 8.3.2	KP 2^{42}	2^{43}	2^{58}	2^{42}	Clamping + 3XOR [BDF18], bal. case
	KP 2^{12}	2^{58}	2^{59}	2^{58}	optim. data
Sec. 8.3.2	CP 2^{35}	2^{57}	$2^{58.6}$	2^{35}	optim. memory & swap $E \leftrightarrow P_1$
Sec. 8.3.2	KP 2^{42}	2^{43}	N.A.	N.A.	Clamping + 3XOR [BDP08], bal. case
Sec. 8.3.3	KP 2^{5}	2^{59}	2^{60}	2^{32}	Low Data Filter $\lambda = 1/2$
	KP 2^{4}	2^{60}	2^{61}	2^{16}	$\lambda = 1/4$

Table 8.2: Comparison of attacks against 2EM with $n = 64$. The complexity unit is one evaluation of the cipher; we assume that computing P_1 or P_2 costs $1/2$, and that a memory access to a large table also costs $1/2$. The time complexity also includes the time necessary to generate the data.

Practical considerations. In the computational setting, the data and memory complexity are important considerations. In particular, an attack with time complexity $2^n/n$ is unlikely to be more efficient than a brute-force attack if it requires in addition almost 2^n data, or almost 2^n memory. As mentioned above, some previous attacks can reduce the data complexity to $2^{\lambda n}$ for an arbitrary $\lambda > 0$, and the attack from [Din+16, Section 4.2] can reduce the memory to $2^{\lambda n}$, but so far none of them can simultaneously reduce the data and memory complexity below $2^{\lambda n}$ for $\lambda < 1$.

Besides, multi-collision based attacks can use a sequential memory (such as a hard drive) and sort values to locate collisions whereas the Meet-in-the-Middle attacks require random access memory, with $\Theta(2^n \ln n/n)$ accesses to a table of size $\Theta(2^n \ln n/n)$.

On the other hand, the linear algebra techniques we use in our attacks will require algorithmic tricks very close to what was done by Bouillaguet, Delaplace and Fouque [BDF18] for the 3-XOR problem. In particular, the values we deal with are sufficiently random to be sorted linearly and the right matrix multiplication LM in a $\mathbf{GF}(2)$ field for an exponentially long matrix L can be computed with a number of operations linear in the size of L. Many constant time optimizations are therefore omitted in this work which justifies that performing indepently a right multiplication, sorting and merging two big lists L_1 and L_2 take time and space $\mathcal{O}(|L_1| + |L_2|)$. This is consistent with previous cryptanalysis on 2EM.

For the cost of queries to the oracles E, P_1 and P_2 we mainly follow the convention established by Dinur et al. [Din+16] which states that an online query to E costs 1 unit of computation implying that P_1 and P_2 cost $1/2$. The main advantage is that it makes it easy to compare with the brute-force solution whose complexity is $\mathcal{O}(2^n)$ computations. The disadvantage is that it makes it hard to combine with the computations used for simple operations: an evaluation of a cryptographically secure permutation should cost more than a XOR operation.

We give concrete complexity values for $n = 64$ in Table 8.2 with the assumption that a combination of some linear time operations does not exceed the cost of computing a permutation that is $1/2$ time unit. Concretely, iteratively right multiplying, sorting and merging two lists L_1, L_2 costs $|L_1|/2 + |L_2|/2$. We believe this makes an honest comparison with previous works though they may use other assumptions.

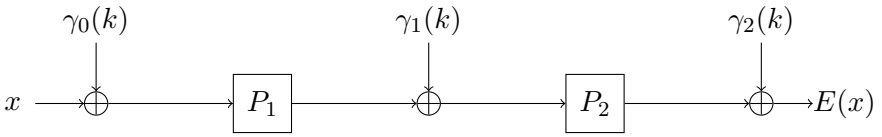


Figure 8.2: Linear key-schedule 2-round Even-Mansour.

8.2 Security Reductions

We start with some general observations about the security of iterated Even-Mansour schemes. In particular, we show that we can focus on the EMIP construction without loss of generality, how to reduce the security of this construction to an instance of the 3-XOR problem, and how to reorder the oracles to achieve many trade-offs.

Some previous works already implicitly took advantage of such reductions. For example Isobe and Shibutani [IS17] realised that their recent attack on EMIP is also applicable to EMSP and Dinur et al. [Din+16] realised that they could reorder the oracles for their cryptanalysis of reduced round LED. We formally show here that these tricks are in fact real security reductions and do not depend on the approach used.

8.2.1 Taking care of Linear Key Schedules

Several variants of single key 2-round Even-Masour were studied in the literature. The most general form uses two independent permutations and a fixed key schedule (see Figure 8.2):

$$E_k(x) = P_2\big(P_1(x \oplus \gamma_0(k)) \oplus \gamma_1(k)\big) \oplus \gamma_2(k).$$

Following the analysis of [Che+14], the construction is secure when the key schedules $\gamma_i(\cdot)$ are public linear bijective n to n bits functions. Actually, when the $\gamma_i(\cdot)$'s are such a public linear bijective functions, the security of this construction reduce to the single key EMIP construction without any key schedule.

The main trick is to rewrite the addition of the subkey $\gamma_i(k)$ as the application of the inverse $\gamma_i^{-1}(\cdot)$, the addition of k and the application of the forward $\gamma_i(\cdot)$:

$$\begin{aligned} x \oplus \gamma_i(k) &= \gamma_i\big(\gamma_i^{-1}(x \oplus \gamma_i(k))\big) \\ &= \gamma_i\big(\gamma_i^{-1}(x) \oplus k\big) \end{aligned}$$

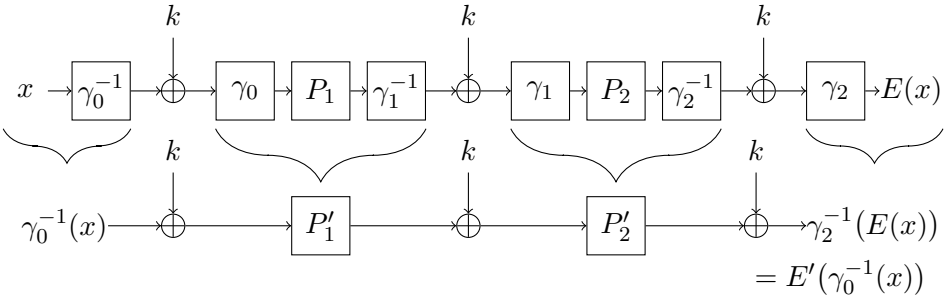


Figure 8.3: Reduction of linear key schedule 2EM to EMIP.

which works thanks to $\gamma_i(\cdot)$ being linear. Then, we define E', P_1', P_2' as follows:

$$P_1'(x) = \gamma_1^{-1}(P_1(\gamma_0(x)))$$
$$P_2'(x) = \gamma_2^{-1}(P_2(\gamma_1(x)))$$
$$E'(x) = \gamma_2^{-1}(E(\gamma_0(x)))$$

Thanks to the previous relation, E', P_1', P_2' is actually an instance of EMIP with the same key k (see Figure 8.3):

$$E'(x) = P_2'(P_1'(x \oplus k) \oplus k) \oplus k.$$

Therefore, any attack against EMIP can be used on E', P_1', P_2', and break the initial construction with a linear key schedule. In particular, a key-recovery attack against EMIP will recover the key of the more general scheme of 2EM.

In the following we only consider the EMIP variant without a key schedule, but thanks to this reduction our attacks can be applied to many other 2EM variants, including the EMSP construction of [Che+14].

8.2.2 From a Key Recovery to a 3-XOR Problem

Instead of directly focusing on a key-recovery attack, we focus on locating a triplet of values x, y, z such that the encryption of x is evaluated with permutation call $P_1(y)$ and $P_2(z)$. Formally, we say that x, y, z is a *right* triplet when $y = x \oplus k$ and $z = P_1(y) \oplus k$. A right triplet corresponds to a sequence of intermediate values in the Even-Mansour encryption, as shown

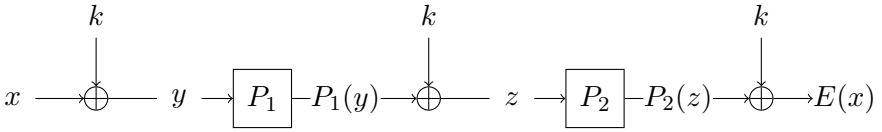


Figure 8.4: A right (x, y, z) triplet forms a path of EMIP style 2-round Even-Mansour.

in Figure 8.4: $(x, y = x \oplus k, P_1(y), z = P_1(y) \oplus k, P_2(z), E(x) = P_2(z) \oplus k)$; we call this sequence a *path*.

Since the permutations P_1 and P_2 are public, it is easy to compute a path given the key. Recovering the key from a path is also easy (we have $k = x \oplus y$), but it is hard to identify a right triplet corresponding to a path without the key. By definition, a triplet is right when it follows the relation $\mathcal{R}$ defined as:

$$\mathcal{R}(x, y, z) := \begin{cases} x \oplus y = k \\ P_1(y) \oplus z = k \\ P_2(z) \oplus E(x) = k \end{cases} \tag{8.1}$$

$$\Rightarrow \begin{cases} x \oplus y = P_1(y) \oplus z \\ x \oplus y = P_2(z) \oplus E(x) \end{cases} \tag{8.2}$$

Notice that we can't directly observe (8.1) since we don't know k, but we can easily verify the implied relation (8.2).

We claim that if one takes a random triplet and observes that it respects (8.2), then it is a right triplet with good probability. Indeed, there are 2^n possible paths (one for every possible input x) implying as many right triplets and 2^{3n} possible triplet combinations; thus a random triplet will be right with probability 2^{-2n}. Since (8.2) is a $2n$-bit relation, a random but false triplet respects (8.2) also with probability 2^{-2n}. Therefore, we can expect roughly as many right triplets than false triplets that respect (8.2), implying that the first one we find is right with probability $\Omega(1)$.

So from now on and for simplicity we focus on filtering and recovering a triplet that simply respects (8.2). This means that our algorithms fails to recover the key on some instances, but they have a constant (non-zero) probability of success. In order to improve the success probability arbitrarily close to one, it is easy to test the triplets and continue the attack until we find a right triplet (alternatively, the whole attack can just be repeated).

In order to look for a triplet, the condition (8.2) is rewritten as:

$$\begin{cases} (x \qquad\quad) \oplus (y \oplus P_1(y)) \oplus (z \qquad) = 0 \\ (x \oplus E(x)) \oplus (y \qquad\quad) \oplus (P_2(z)) = 0 \end{cases}$$

Therefore, finding a triplet satisfying (8.2) is equivalent to solving an instance of the 3-XOR problem with the functions defined as:

$$\begin{aligned} f_0(x) &:= x \qquad\quad \| \; x \oplus E(x) \\ f_1(y) &:= y \oplus P_1(y) \| \; y \\ f_2(z) &:= z \qquad\quad \| \; P_2(z) \end{aligned} \tag{8.3}$$

Notice that the number of evaluations of $f_0(x)$ is the data complexity D as it requires to query the keyed construction. Conversely, the sum of the number of evaluations of $f_1(y)$ and $f_2(z)$ is the query complexity Q which can be seen as offline computations in the computational setting.

8.2.3 Permuting Oracle Calls

In the random 3-XOR problem the three functions behave essentially in the same way; if one has a distinguisher using a few evaluations f_0 and lots of evaluations of f_1 and f_2, then the same distinguisher can aritraryly decide to use lots of queries to f_0 and f_1 and use fewer f_2 queries (just by permuting the functions). In our case, a natural choice is to minimize the number of evaluations of f_0 in order to optimize for the data complexity. This ensures that we have $D \leq Q$. While this is easy to do with a 3-XOR approach, it is not obvious whether this can be done in general for a 2EM key recovery. Nevertheless, this is indeed the case: in the sprp security distinguisher, an attacker is free to permute the functions E_k, P_1 and P_2 and, doing so, minimize the amount of online queries.

Consider an EMIP instance $E_k(x) = k \oplus P_2(k \oplus P_1(k \oplus x))$ of 2EM based on P_1, P_2 in the sprp security game, so that we have forward oracle access to E_k, P_1, P_2, and backward oracle access E^{-1}, P_1^{-1}, P_2^{-1}. We use a black-box distinguisher $\mathcal{A}$ that distinguishes the construction $E_k[P_1, P_2]$ from a third independent random permutation P:

$$\begin{aligned} \mathbf{Adv}^{\mathsf{sprp}}_{\mathrm{2EM}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{E_k, E_k^{-1}, P_1, P_1^{-1}, P_2, P_2^{-1}} \to 1) \\ - \mathbf{Pr}(\mathcal{A}^{P, P^{-1}, P_1, P_1^{-1}, P_2, P_2^{-1}} \to 1) \end{aligned}$$

where P, P_1, P_2 are independent random permutations, E_k answer query x with $k \oplus P_2(k \oplus P_1(k \oplus x))$, and $k \xleftarrow{\$} \{0,1\}^n$.
We assume $\mathcal{A}^{E_k,E_k^{-1},P_1,P_1^{-1},P_2,P_2^{-1}}$ uses α calls to E_k/E_k^{-1} (online queries), β calls to P_1/P_1^{-1} and γ calls to P_2/P_2^{-1} and outputs its bit decision.

The trick is that we can rewrite the 2EM instance E_k, P_1, P_2, by permuting the oracles. For instance, we know that $P_1(x) = k \oplus P_2^{-1}(k \oplus E_k(k \oplus x))$ (directly from the definition of E_k), which gives the following 2EM instance parametrized by the same secret key k:

$$E'_k = P_1 \qquad P'_1 = E_k \qquad P'_2 = P_2^{-1}.$$

Therefore, we can use the same distinguisher but permuting the oracles as $\mathcal{A}^{P_1,P_1^{-1},E_k,E_k^{-1},P_2^{-1},P_2}$ using β online queries with the same advantage that is:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{sprp}}_{\mathrm{2EM}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{P_1,P_1^{-1},E_k,E_k^{-1},P_2^{-1},P_2} \to 1) \\ - \mathbf{Pr}(\mathcal{A}^{P_1,P_1^{-1},P,P^{-1},P_2^{-1},P_2} \to 1)\end{aligned}$$

as E_k and P_2^{-1} will behave exactly like independent random permutations and P_1 like the keyed construction with the random key k. Similarly, we can write $P_2(x) = k \oplus E_k(k \oplus P_1^{-1}(k \oplus x))$; therefore, we can use the distinguisher as $\mathcal{A}^{P_2,P_2^{-1},P_1^{-1},P_1,E_k,E_k^{-1}}$ with the same advantage using γ online queries.

It is easy to see that the result also applies for key recovery attacks.

8.3 2EM Cryptanalysis

Following the security reductions shown in Section 8.2, we focus on cryptanalysis of the EMIP variant of 2EM by looking for a triplet that is a solution to the 3-XOR problem with functions f_0, f_1, f_2 as described in Equation (8.3).

3-XOR Algorithms. Let us first recall the different known techniques to solve the 3-XOR problem for w-bit words we've seen in Section 3.2.2. There are three main techniques, the one using multi-collisions by Nikolic and Sasaki [NS15] running in time and memory $\mathcal{O}(2^{w/2}/\sqrt{w/\ln(w)})$ and performing as many queries to all three functions. The techniques exploiting linear algebra by Joux [Jou09] that reaches a time and memory

complexity of $\mathcal{O}(2^{w/2}/\sqrt{w})$ and as many queries to f_1 and f_2 but with only $w/2$ evaluations of f_0. The generalization of this technique by Bouillaguet, Delaplace and Fouque [BDF18] builds lists L_0, L_1, L_2 from the respective functions so that $|L_0| \cdot |L_1| \cdot |L_2| = 2^w$ and solves the 3-XOR problem with a time complexity of $\mathcal{O}(|L_0| \cdot (|L_1| + |L_2|)/w)$. The last algorithm comes from Baran, Demaine and Pătraşcu [BDP08] initially for the 3-SUM problem and it was adapted to the 3-XOR by Bouillaguet et al. [BDF18]. It has an asymptotic time complexity of $\mathcal{O}(2^{w/2} \cdot \ln^2(w)/w^2)$ but relies on heavy precomputations making it impracticable for most w.

Notations. The ith element of a list L is denoted as $L[i]$, a list L of w-bit words is often seen as a $|L| \times w$ matrix with bit coefficients in $\mathbf{GF}(2)$ where the w bits of the element $L[i]$ form the ith row. Moreover, $0_{a\times b}$ is the $a \times b$ zero $\mathbf{GF}(2)$ matrix and I_a is the $a \times a$ identity $\mathbf{GF}(2)$ matrix.

8.3.1 Direct Applications

We start to describe a key recovery algorithm by directly using the linear algebra techniques of Joux [Jou09]. In our case, $w = 2n$ so the attack has a time and memory complexity of $\mathcal{O}(2^n/\sqrt{n})$.

See Algorithm 8.1 for a description of the attack. Step 7 implicitly requires that L_0 contains n linearly independent $2n$-bit rows which is true with very high probability. As we only need to observe n input/output pairs of the keyed construction, this is a known plaintext attack with a data complexity of only $D = n$.

The complexity analysis follows the one by Joux [Jou09] and so we have a constant probability of success with a query complexity $Q = 2^n/\sqrt{n}$ which is also the memory complexity. As we assume that looking for partial collision also costs Q operation, the total time complexity is $T = 2Q = 2 \cdot 2^n/\sqrt{n}$. Computations of the permutations in Steps 4 and 5 can be done as precomputations.

Using Multi-collisions. If, instead, we use the multi-collision based approach of [NS15] to solve the 3-XOR problem with $w = 2n$, we get a cryptanalysis of 2EM running in time and memory $2^n/\sqrt{n/\ln(n)}$. In fact, with a simple optimization to the algorithm we can get to a complexity of $2^n \cdot \ln n/n$ as the attack will be equivalent to the one of [NWW14].

Algorithm 8.1 Key recovery on 2EM using linear algebra [Jou09].

input: E, P_1, P_2 is EMIP.
output: the key k.
procedure AttackEMJoux($E(\cdot), P_1(\cdot), P_2(\cdot)$)
  $L_1 \leftarrow \{(y \oplus P_1(y)) \,\|\, y : y \in \mathcal{Y}\}$ ▷ For a random set $\mathcal{Y}$ of size Q.
  $L_2 \leftarrow \{z \,\|\, P_2(z) : z \in \mathcal{Z}\}$ ▷ For a random set $\mathcal{Z}$ of size Q.
  $L_0 \leftarrow \{x \,\|\, x \oplus E(x) : x \in \mathcal{X}\}$ ▷ For an obersvable set $\mathcal{X}$ of size n.
    ▷ See sets L_i as $|L_i| \times 2n$ matrices in $\mathbf{GF}(2)$.
  Find M s.t. $L_0 M = [0_{n\times n} \,\|\, I_n]$.[a]
  $L'_0 \leftarrow L_0 M$
  $L'_1 \leftarrow L_1 M$
  $L'_2 \leftarrow L_2 M$
  for all (i, j) s.t. $\lfloor L'_1[i] \rfloor_n = \lfloor L'_2[j] \rfloor_n$ **do** ▷ n-bit partial collisions.
    if $L'_1[i] \oplus L'_2[j] \in L'_0$ **then**
      Let h such that $L'_1[i] \oplus L'_2[j] = L'_0[h]$.
      return $\lfloor L_0[h] \rfloor_n \oplus \lceil L_1[i] \rceil_n$ ▷ Corresponds to $x \oplus y$.
  return $\emptyset$ ▷ No solution found.

[a] We write $L_0 = \begin{bmatrix} A & B \end{bmatrix}$. If B is non-singular, we can use $M = \begin{bmatrix} I & 0 \\ B^{-1}A & B^{-1} \end{bmatrix}$

Indeed, looking for an n-bit partial multi-collision in $f_1(y) = y \oplus P_1(y) \,\|\, y$ is the same as looking for a multi-collision in $\phi(u) = u \oplus P_1(u)$ (it is impossible to have a multi-collision on the input y). We expect to find a value $v = y \oplus P_1(y)$ appearing for $n/\ln n$ different values y. However, in our case, for every observed $f_0(x) = x \,\|\, x \oplus E(x)$ we can compute $f_2(x \oplus v) = x \oplus v \,\|\, P_2(x \oplus v)$ by setting $z = x \oplus v$ which ensures that the triplet match on the first part.

Choosing z is something we cannot do in the random 3-XOR but is possible in our case. As a result, for every plaintext/ciphertext pairs observed we can build $n/\ln n$ triplets that sum to 0 on the first n-bit half and one of them is a solution with probability $n/\ln n/2^n$. Hence, we expect a data and query complexity of $2^n \cdot \ln n/n$. Interestingly, the strategy and the complexity correspond to [NWW14] even though they didn't use the 3-XOR problem to describe their attack.

Algorithm 8.2 Key recovery on 2EM using multicollision techniques for 3-XOR [NS15] equivalent to [NWW14].

input: E, P_1, P_2 is EMIP.
output: the key k.
procedure AttackEMJoux($E(\cdot), P_1(\cdot), P_2(\cdot)$)
  $v \leftarrow \text{argmax}_v \, |\{y : y \oplus P_1(y) = v\}|$ ▷ Any multi-collision algorithm.
  $L_1 \leftarrow \{v \,\|\, y : y \oplus P_1(y) = v\}$
  $L_0 \leftarrow \{x \,\|\, x \oplus E(x) : x \in \mathcal{X}\}$ ▷ For an obersvable set $\mathcal{X}$ of size n.
  for all $e_0 \in L_0$ **do**
    $e_2 \leftarrow x \oplus v \,\|\, P_2(x \oplus v)$ ▷ Fix $z = x \oplus v$.
    if $e_0 \oplus e_2 \in L_1$ **then**
      $e_1 \leftarrow e_0 \oplus e_2$
      return $\lfloor e_0 \rfloor_n \oplus \lceil e_1 \rceil_n$ ▷ Corresponds to $x \oplus y$.
  return $\emptyset$ ▷ No solution found.

8.3.2 Using Black-box 3-XOR Algorithms

As seen with the multi-collision attack, the 3-XOR problem (Equation 8.3) we wish to solve is strictly easier than a random 3-XOR problem as we can arbitrarily choose part of the values. Typically, the inputs to the permutations y and z are under our control and, in a chosen plaintext attack, also x. Thus, a variant of the clamping trick of Bernstein [Ber07] can be used to simplify the 3-XOR instance before using generic 3-XOR solvers.

Let us assume a known plaintext attack so that we can't choose the x values. We first align the values we can control, so we equivalently rewrite the Equation 8.3 as:

$$\begin{aligned} f_0(x) &:= x && \| \; x \oplus E(x) \\ f_1(y) &:= y \oplus P_1(y) && \| \; y \\ f_2(z') &:= P_2^{-1}(z') && \| \; z' \end{aligned} \tag{8.4}$$

As we set the data complexity $D = 2^d$, we require at least $D \cdot Q^2 = 2^n$ to have a solution and the query complexity is $Q = 2^{n-d/2}$. We force the equality by choosing y and z' such that they end by $d/2$ zeros and, additionally, filter the known plaintext/ciphertext pairs to keep them only if $(x \oplus E(x))$ also ends by $d/2$ zeros. This is a $d/2$-bit filter on 2^d elements, so we expect L_0 to contain $2^{d/2}$ elements.

This works as a clamping strategy and effectively reduces the bit size of the elements from $2n$ to $2n - d/2$. We then apply any 3-XOR solver we like on the lists L_0, L_1, L_2 with word's size $w = 2n - d/2$ and lists size $2^{d/2}, 2^{n-d/2}, 2^{n-d/2}$ respectively (notice that the lists size multiply to $2^{2n-d/2} = 2^w$ as expected). We describe this in Algorithm 8.3. There is no obvious way to further clamp down the data using chosen plaintext or even chosen ciphertext.

Algorithm 8.3 Key recovery on 2EM using black-box solver.

input: E, P_1, P_2 is EMIP, data complexity is 2^d and Solver3XOR is a 3-XOR solver.
output: the key k.
procedure ClampSolveEM($E(\cdot), P_1(\cdot), P_2(\cdot), d,$ Solver3XOR)
    $L_1 \leftarrow \{(u \,\|\, 0^{d/2} \oplus P_1(u \,\|\, 0^{d/2})) \,\|\, u : u \in \{0,1\}^{n-d/2}\}$
    $L_2 \leftarrow \{P_2^{-1}(u \,\|\, 0^{d/2}) \,\|\, u : u \in \{0,1\}^{n-d/2}\}$
    Let $\mathcal{X}$ be an observable set of size 2^d.
    $L_0 \leftarrow \emptyset$
    for all $x \in \mathcal{X}$ **do**
        if $\lceil x \oplus E(x) \rceil_{d/2} \stackrel{?}{=} 0$ **then**
            $L_0 \leftarrow L_0 \cup \{x \,\|\, \lfloor x \oplus E(x) \rfloor_{n-d/2}\}$
    $(e_0, e_1, e_2) \leftarrow$ Solver3XOR(L_0, L_1, L_2)
        ▷ Black-box 3XOR solver with $w = 2n - d/2$.
    return $\lfloor e_0 \rfloor_n \oplus (\lceil e_1 \rceil_{n-d/2} \,\|\, 0^{d/2})$ ▷ Corresponds to $x \oplus y$.

Using Known Solvers. For instance let us use the algorithm of Bouillaguet, Delaplace and Fouque [BDF18] using linear algebra to solve the 3-XOR with list of arbitrary sizes in $\mathcal{O}(|L_0| \cdot (|L_1| + |L_2|)/w)$ using no more memory than the lists require. In our case this yields a time complexity of $\mathcal{O}(2^n/(2n - d/2))$ matching the best known cryptanalysis of 2EM. The memory needed to store L_1 and L_2 is $\mathcal{O}(2^{n-d/2})$, and the total time complexity stays $\mathcal{O}(2^n/n)$ for all values of D as long as the lists size does not exceed $\mathcal{O}(2^n/n)$, that is as long as $D \geq n^2$. Therefore, a data optimized attack in this setting can set the data complexity to as low as $D = n^2$.

In the balanced case $D = Q = 2^{2n/3}$, the attack is optimal in the information theoretic model [Che+14] and both the data and the memory are significantly below 2^n while having a time complexity competitive

with the best known cryptanalysis. Notice also that evaluations of the cipher and permutations do not dominate the time complexity. Thus, the attack becomes even more competitive if we assume an evaluation of a permutation is much more costly than n-bit words operations and memory access to big lists as it was done in previous works [IS17].

To optimize the memory complexity, we need to choose a fairly high value d. The data complexity $D = 2^d$ may become problematic, so we simply swap the number of query to E with the number of offline query to P_1 as shown in Section 8.2.3. This effectively swaps f_0 and f_1. Thus, we can have a data and memory complexity of $2^{n-d/2}$, a query complexity $Q = 2^{n-d/2-1} + 2^{d-1}$ while the time is still $\mathcal{O}(2^n/n)$. As we need data with special values of x, this becomes a chosen plaintext attack. Examples of concrete trade-offs for $n = 64$ using [BDF18] algorithm are given in Table 8.2.

Another interesting solver using arbitrary sized lists is the BDP Algorithm [BDP08] which runs in $\mathcal{O}\left(|L_0| \cdot (|L_1| + |L_2|) \cdot \ln^2(w)/w^2\right)$ and has an asymptotic memory of the size of the lists. In our case, this yields the best asymptotic complexity known so far $T = \mathcal{O}(2^n \cdot \ln^2(n)/n^2)$. However, this approach is hardly relevant for any realistic size of n. Indeed, following the analysis of [BDF18] the complexity of the BDP algorithm is dominated by $|L_0|.|L_1|/m^2$ where $m \simeq n/(112\ln(n))$. Therefore, we expect this approach to be competitive when $m^2 > n$ that implies $n > 2.75 \times 10^6$, an absurdly big state size.

8.3.3 Using Very Low Data

The previous algorithm can reach a low data complexity for a small parameter d and a relatively low memory complexity close to $2^{n/2}$ for large d. Having both the data and memory complexity close to $2^{n/2}$ requires a chosen plaintext attack.

In this section we show a cryptanalysis with an online or data complexity $D < n$ and a memory complexity of 2^D possibly below $2^{n/2}$. Concretely, it uses a fraction $0 < \lambda < 1$ of n that is $D = \lambda n$ known plaintext/ciphertext pairs along with $2^{\lambda n}$ memory and a time and query complexity $T = Q = \mathcal{O}(2^n/\lambda n)$. Notice that we have $D \cdot T = 2^n$ and $D \cdot T^2 = \mathcal{O}(2^{2n}/\lambda n)$ which makes for a better trade-off than the best information theoretic attack known so far for low values of D. In fact, it is

information theoretically optimal since it matches the $D \cdot T \geq 2^n$ bound of the original Even-Mansour scheme which obviously applies for two rounds.

Algorithm 8.4 Key recovery on 2EM for low data.

input: E, P_1, P_2 is EMIP, data complexity is λn with $0 < \lambda \leq \frac{W(2^n \ln 2)}{n \ln 2}$.
output: the key k.
procedure LowDataEM($E(\cdot), P_1(\cdot), P_2(\cdot), \lambda$)
    $L_0 \leftarrow \{x \,\|\, x \oplus E(x) : x \in \mathcal{X}\}$ ▷ For an obersvable set $\mathcal{X}$ of size λn.
    $[\underbrace{A}_{n} \| \underbrace{B}_{n-\lambda n} \| \underbrace{C}_{\lambda n}] \leftarrow L_0$ ▷ See L_0 as three concatenated λn-line matrices.
    $M_s \leftarrow \begin{bmatrix} I & 0 \\ C^{-1}B & C^{-1} \end{bmatrix}$
    $M_s^{-1} \leftarrow \begin{bmatrix} I & 0 \\ B & C \end{bmatrix}$
    $M \leftarrow \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ C^{-1}A & C^{-1}B & C^{-1} \end{bmatrix}$
    $L_0' \leftarrow L_0 M$ ▷ Notice that $L_0' = [\underbrace{0}_{n} \| \underbrace{0}_{n-\lambda n} \| \underbrace{I}_{\lambda n}]$.
    for all $\alpha \in \{0,1\}^{n-\lambda n}$ **do**
        $\mathcal{V} \leftarrow \{[\alpha \,\|\, u] \cdot M_s^{-1} : u \in \{0,1\}^{\lambda n}\}$
        $L_1 \leftarrow \{v \oplus P_1(v) \,\|\, v : v \in \mathcal{V}\}$
        $L_2 \leftarrow \{P_2^{-1}(v) \,\|\, v : v \in \mathcal{V}\}$
        $L_1' \leftarrow L_1 M$
        $L_2' \leftarrow L_2 M$ ▷ Note that $e_{[n:2n-\lambda n]} = \alpha \; \forall e \in L_1' \cup L_2'$.
        for all (i,j) s.t. $\lfloor L_1'[i] \rfloor_n = \lfloor L_2'[j] \rfloor_n$ **do** ▷ n-bit partial collisions.
            if $L_1'[i] \oplus L_2'[j] \in L_0'$ **then**
                Let h such that $L_1'[i] \oplus L_2'[j] = L_0'[h]$.
                return $\lfloor L_0[h] \rfloor_n \oplus \lceil L_1[i] \rceil_n$ ▷ Corresponds to $x \oplus y$.

We describe the attack in Algorithm 8.4. The broad strategy is again to look for a solution of the 3-XOR problem with functions as defined in Equation (8.4). However, since we have very few data we perform Gaussian elimination on L_0 to get words starting with $2n - \lambda n$ zeros. In fact, we

define a small $n \times n$ transformation matrix M_s that only deals with the right-hand side of L_0. Decompose L_0 as $L_0 = [\underbrace{A}_{n} \| \underbrace{B}_{n-\lambda n} \| \underbrace{C}_{\lambda n}]$ then the small transformation matrix is such that $[B \| C]M_s = [0_{\lambda n \times (n-\lambda n)} \| I_{\lambda n}]$. And the big $2n \times n$ transformation matrix M deals with the whole L_0 and is defined as:

$$M = \begin{bmatrix} I & 0 \\ \left(M_s \begin{bmatrix} 0 \\ A \end{bmatrix}\right) & M_s \end{bmatrix}$$

After we fix an $(n - \lambda n)$-bit value α, the small transformation matrix is used to compute the y and z' values for L_1 and L_2. In Steps 12 and 13 of Algorithm 8.4 we choose y and z' to be of the form $[\alpha \| u] \cdot M_s^{-1}$ for a fixed α and all λn-bit values u. That way, when applying M to get the transformed problem, the right-hand side values of L_1' and L_2' revert to the form $[\alpha \| u]$. Let $h_1(u) = [\alpha \| u]M_s^{-1} \oplus P_1([\alpha \| u]M_s^{-1}) \oplus uA$ and $h_2(u) = P_2^{-1}([\alpha \| u]M_s^{-1}) \oplus uA$, the three transformed lists are thus:

$$\begin{aligned} L_0' &= [0_{\lambda n \times n} \| 0_{\lambda n \times n - \lambda n} \| I_{\lambda n}] \\ L_1' &= \{h_1(u) \| \quad \alpha \quad \| \; u \quad : \forall u \in \{0,1\}^{\lambda n}\} \\ L_2' &= \{h_2(u) \| \quad \alpha \quad \| \; u \quad : \forall u \in \{0,1\}^{\lambda n}\} \end{aligned}$$

This forces an $(n - \lambda n)$-bit collision at the beginning of the right-hand side where the fixed α values will always sum up to match the corresponding zeros in L_0'.

Complexity Analysis. In Step 4 we require $D = \lambda n$ known plaintext/ciphertext pairs with makes for the data complexity. To build M_s we implicitly assume that the matrix C is invertible which is true with constant probability for a random square matrix.

In each loop a new value for α is chosen and new lists L_1, L_2 of size $2^{\lambda n}$ are built. A solution exists if one pair among the $2^{2\lambda n}$ pairs XORs to one of the λn values of L_0. As the lists are built so that it forces an $(n - \lambda n)$-bit collision, a pair will match a value in L_0 with probability $\lambda n \cdot 2^{-(n+\lambda n)}$. Thus, each loop yields a solution with probability $2^{2\lambda n} \cdot \lambda n \cdot 2^{-(n+\lambda n)}$. Therefore, Algorithm 8.4 has a constant probability of success after $\frac{2^{n-\lambda n}}{\lambda n}$ iterations.

The memory complexity is dominated by storing the lists L_1 and L_2 of size $2^{\lambda n}$. As the algorithm loops over values of α those lists are reused so that the total memory complexity is indeed $\mathcal{O}(2^{\lambda n})$.

The expected query complexity is simply $2^{\lambda n}$ times the expected number of loops that is

$$Q = 2^{\lambda n} \cdot \frac{2^{n-\lambda n}}{\lambda n} = \frac{2^n}{\lambda n} .$$

The time complexity is essentially the query complexity. Indeed, computations of the matrices M_s and M are polynomial in n and so negligible while building the lists L_1 and L_2, computing L'_1 and L'_2 and looking for collisions are linear in the number of elements so it takes $\mathcal{O}(2^{\lambda n})$ time per loop. Therefore, $T = \mathcal{O}(\frac{2^n}{\lambda n})$.

The reasoning to derive the query and time complexity implicitly assumes that one needs at least one loop to finish the algorithm as it makes no sense to finish after half-a-round. Therefore, those trade-offs depending on λ are constraints by:

$$\frac{2^{n-\lambda n}}{\lambda n} \geq 1 \Leftrightarrow \lambda \leq \frac{W(2^n \ln 2)}{n \ln 2} = 1 - \frac{\ln(n \ln 2)}{n \ln 2} + o(1)$$

using the Lambert W function.

Discussion. Algorithm 8.4 thus describes a known plaintext attack using only λn pairs and $\mathcal{O}(2^{\lambda n})$ memory while being on par with the time complexity of the best known cryptanalysis of 2EM.

In the information theoretic setting, the best distinguisher by Gaži has a trade-off $DQ^2 = 2^{2n}$ [Gaž13]. Our attack has a trade-off $DQ^2 = 2^{2n}/\lambda n$ thus being the best distinguisher as well as the best key recovery for very low data D. In fact, the proof of security of 2EM by Chen et al. [Che+14] says nothing for low data range $D \leq 2^{n/4}$. The best proof of security of 2EM in this case is inherited from the original Even-Mansour proof which state $D \cdot T \geq 2^n$. Therefore, Algorithm 8.4, where $D \cdot T = 2^n$, shows the tightness of the original proof for low data range $1 \leq D \leq \frac{W(2^n \ln 2)}{\ln 2} \simeq n - \frac{\ln(n \ln 2)}{\ln 2}$.

Overall, this cryptanalysis can be seen as an advanced version of the attack by Dinur et al.using linear algebra [Din+16, Section 4.2]. However, there are three main differences that make this attack an improvement over the previous one. First, we use the symmetry between E, P_1, P_2 (as shown in Section 8.2.3) to reduce the data complexity from $2^n/\lambda n$ to λn. Then, the use of the big transformation matrix M, that essentially

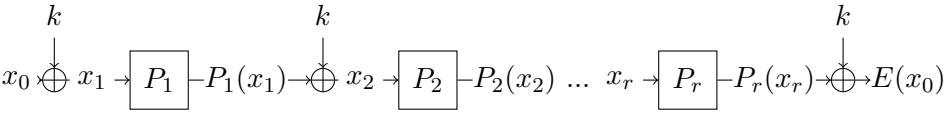


Figure 8.5: A right $(x_0, x_1, ..., x_r)$ tuple forms a path of r-round Even-Mansour.

performs a Gaussian elimination over the whole $2n$-bit words of L_0, makes the attack works with known plaintexts while Dinur et al. required chosen plaintexts (even after applying the symmetry trick). Finally, the resulting n-bit filter of Step 16 allows for a larger acceptable range of λ than their attack that required $\lambda < 1/3$ to limit the number of partial collisions.

8.4 Going Further

8.4.1 Extending to More than 2 Rounds

The idea of the attack can also be used for longer iteration of Even-Mansour. In general, a single key r-round Even-Mansour key recovery attack can be done by solving a particular $r+1$-XOR problem with words of size rn. While generic solvers won't be of much help as soon as $r \geq 4$, this elegantly rewrites the known generic distinguisher on rEM shown in Section 7.1.2, Algorithm 7.3.

Generic Reduction. We follow the same reasoning as in Section 8.2.2 and apply it to r-round Even-Mansour (Figure 8.5) so that we look for an $(r+1)$-tuple $(x_0, x_1, \ldots, x_r)$ satisfying the relation $\mathcal{R}$:

$$\mathcal{R}(x_0, x_1, x_2, ..., x_r) := \begin{cases} x_0 \oplus x_1 = k \\ P_i(x_i) \oplus x_{i+1} = k, \quad 1 \leq i \leq r-1 \\ P_r(x_r) \oplus E(x_0) = k \end{cases} \tag{8.5}$$

$$\implies \begin{cases} x_0 \oplus x_1 = P_1(x_1) \oplus x_2 \\ P_i(x_i) \oplus x_{i+1} = P_{i+1}(x_{i+1}) \oplus x_{i+2}, \quad 1 \leq i \leq r-2 \\ P_{r-1}(x_{r-1}) \oplus x_r = P_r(x_r) \oplus E(x_0) \end{cases} \tag{8.6}$$

Again, while $\mathcal{R}$ cannot be observed it is easy to verify the implied relation (8.6) which forms an rn-bit filter. Such a filter is enough so that a randomly $(r+1)$-tuple satisfying (8.6) also satisfies $\mathcal{R}$ with good probability.

From the filter (8.6) we define $r+1$ functions whose words are the concatenation of r entries of bit-size n:

$$f_0(x_0)_{[hn-n:hn]} := \begin{cases} x_0 & ,h=1 \\ 0 & ,2 \le h \le r-1 \\ E(x_0) & ,h=r \end{cases}$$

$$f_1(x_1)_{[hn-n:hn]} := \begin{cases} x_1 \oplus P_1(x_1) & ,h=1 \\ P_1(x_1) & ,h=2 \\ 0 & ,h \ge 3 \end{cases}$$

$$f_i(x_i)_{\substack{[hn-n:hn] \\ 2 \le i \le r-1}} := \begin{cases} 0 & ,h \le i-2 \\ x_i & ,h=i-1 \\ x_i \oplus P_i(x_i) & ,h=i \\ P_i(x_i) & ,h=i+1 \\ 0 & ,h \ge i+2 \end{cases}$$

$$f_r(x_r)_{[hn-n:hn]} := \begin{cases} 0 & ,h \le r-2 \\ x_r & ,h=r-1 \\ x_r \oplus P_r(x_r) & ,h=r \end{cases}$$

An example in lists form for $r=5$ is given in Table 8.3. This indeed defines an $(r+1)$-XOR problem with rn-bit words even though it is way more structured than a purely random $(r+1)$-XOR problem. Upon its resolution we make a key guess $k \stackrel{?}{=} x_0 \oplus x_1$ that succeeds with high probability.

Generic Solvers. Generic algorithms for the k-XOR problem can typically be applied to this structured variant to perform a key recovery attack. The generic query lower-bound seen in Section 3.2 for the random $(r+1)$-XOR problem with words of size $w=rn$ is $\mathcal{O}(2^{\frac{rn}{r+1}})$. Interestingly this exactly coincides with the lower bound on the queries for the single key r-round Even-Mansour scheme proved in [Bog+12]. Generic solvers requiring $D=Q=\mathcal{O}(2^{\frac{rn}{r+1}})$ are thus optimal with respect to the data/query complexity trade-off. This results in a distinguisher quite similar to the

Lists' construction for a cryptanalysis using the 6-XOR problem.					
$L_0 \ni \{$	x_0	.	.	.	$E(x_0)\}$
$L_1 \ni \{$	$x_1 \oplus P_1(x_1)$	$P_1(x_1)$	.	.	. $\}$
$L_2 \ni \{$	x_2	$x_2 \oplus P_2(x_2)$	$P_2(x_2)$	.	. $\}$
$L_3 \ni \{$	.	x_3	$x_3 \oplus P_3(x_3)$	$P_3(x_3)$	. $\}$
$L_4 \ni \{$	.	.	x_4	$x_4 \oplus P_4(x_4)$	$P_4(x_4)\}$
$L_5 \ni \{$	.	.	.	x_5	$x_5 \oplus P_5(x_5)\}$

Table 8.3: Cryptanalysis of 5EM.

one shown in Section 7.1.2, Algorithm 7.3 but, instead of looking for contradictory paths, it directly looks for a correct path implying a right tuple and guesses the key.

In the computational setting, Wagner's algorithm [Wag02] can be applied to our case with a time complexity of $T = \mathcal{O}(r \cdot 2^{\frac{rn}{\lfloor \log(r+1) \rfloor + 1}})$. For $r = 2$ and 3 rounds this becomes $\mathcal{O}(2^n)$ and in the case of 2EM we could use more recent techniques to get slightly below $\mathcal{O}(2^n)$. In the case of 3EM, Dinur et al. [Din+16] described a cryptanalysis with a complexity of $\mathcal{O}(2^n \cdot \frac{\ln n}{n})$ using multicollisions and while it is fairly straightforward to rewrite the same attack in the 4-XOR context it is also non-trivial to improve this.

On the other hand, the time complexity of the generic solver becomes significantly higher than 2^n for $r \geq 4$ while there is no known cryptanalysis with a time complexity below the 2^n brute-force attack. However, as the number of rounds grows the lists get a strong structure as illustrated in Table 8.3 with many bits fixed to 0. This opens the question of a dedicated algorithm with competitive computational time/memory trade-off for $r \geq 4$.

8.4.2 Conclusion

Iterated Even-Mansour schemes are an idealization of SPN networks and understanding their security is important because many block ciphers,

including the AES, are based on this design. In this chapter we focused on the two-round construction linking it to the 3-XOR problem. Using linear algebra techniques initially developed for the random 3-XOR problem we devised novel and competitive key recovery attacks with a particularly competitive data and memory complexity. In particular, we give the first attacks where both the data and memory complexity are below $\mathcal{O}(2^{n-\varepsilon})$ for $\varepsilon > 0$, while achieving the best known time complexity of $\mathcal{O}(2^n/n)$. Previous attacks with a similar time complexity required either a very large memory or very large data, making them unlikely to be useful in practice. We also give an attack that improves the asymptotic time complexity to $\mathcal{O}(2^n \cdot \ln^2(n)/n^2)$, although it is not applicable for practical values of n. Additionally, our low data attack Algorithm 8.4 reaches a trade-off $DT = 2^n$ for small D beating the best known distinguisher and proving the optimality of the original lower bound given for the single round.

As Algorithm 8.3 is a black-box construction using a 3-XOR solver, future improvements of the random 3-XOR algorithms will further improve our cryptanalysis. Note that the converse may not hold: it is not trivial to derive an improved computational lower bound for 2EM assuming the 3-XOR algorithms cannot be improved.

We also extend this approach to link the r-round Even-Mansour with the $(r+1)$-XOR problem with a particular structure. However, additional work is required to deduce competitive key-recovery attacks exploiting the particular structure.

Chapter 9
Generic Attack on the Iterated Tweakable FX Construction

Contributions brought forward in this chapter were published in CT-RSA 2020 as a sole author [Sib20]. It is a follow-up on the group discussion initiated in the Asian Symmetric Key Workshop, November 2018.

Introduction

Since they were formalized by Liskov, Rivest and Wagner [LRW11], tweakable block ciphers have received a great deal of attention from the scientific community. However, regular block ciphers such as the DES [DES77] and the AES [AES] benefit from a longer history of research and their security is arguably better understood.

Therefore, we naturally ask ourselves how can we build a tweakable block cipher out of a regular block cipher.

We saw in Section 7.2.2 that simple constructions such as `LRW2` and, by extension, XEX effectively achieve this at a relatively small cost (one block cipher call, one Galois field multiplication and two XOR operations). There is though a cost in security. Indeed, even when the underlying block cipher is assumed to be sprp-secure up to $\mathcal{O}(2^n)$ queries, the resulting `LRW2` / XEX construction is only $\widetilde{\mathsf{sprp}}$-secure up to $\mathcal{O}(2^{n/2})$ queries, that is secure up to the birthday bound.

Therefore, subsequent works aimed at improving this bound and in particular some works started mixing the tweak with the master key to derive the effective key for the block cipher. For instance the $\widetilde{F}[2]$ construction by Mennink [Men15] achieves $\mathcal{O}(2^n)$ $\widetilde{\mathsf{sprp}}$-security assuming an ideal underlying block cipher, at the cost of two block cipher calls and a few XORs as $\widetilde{E}_k^t(m) = E_{2k}(t) \oplus E_{t \oplus k}(m \oplus E_{2k}(t))$. And the `XHX` construction by Jha, List, Minematsu, Mishra and Nandi [Jha+17] which

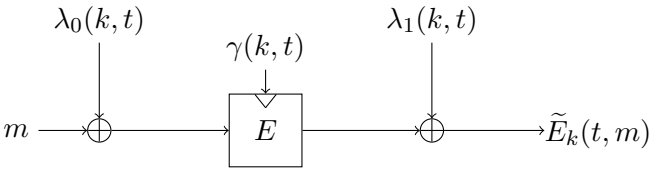


Figure 9.1: GXHX construction with master key k, an AU function $\gamma(\cdot,\cdot)$ and two independent AXU functions $\lambda_0(\cdot,\cdot), \lambda_1(\cdot,\cdot)$. GXHX coincides with the single round Tweakable FX construction.

achieves $\mathcal{O}(2^{(n+\kappa)/2})$ $\widetilde{\mathsf{sprp}}$-security with κ the key size of the block cipher. The XHX proof is assuming an ideal cipher E, a uniform and almost XOR-universal function $\lambda(\cdot,\cdot)$ and a uniform and almost universal function $\gamma(\cdot,\cdot)$ (we recall all definitions in Definition 9.1) and is computed as $\widetilde{E}_k^t(m) = \lambda(k,t) \oplus E_{\gamma(k,t)}(m \oplus \lambda(k,t))$.

Definition 9.1. Let $\delta > 0$ and a function $\lambda : \mathcal{K} \times \mathcal{T} \to \mathcal{Y}$ for non-empty sets $\mathcal{K}, \mathcal{T}, \mathcal{Y}$.

- $\lambda(k,t)$ is said to be δ-almost uniform if for any $t \in \mathcal{T}$ and any $y \in \mathcal{Y}$,

$$\Pr\left(k \leftarrow_{\$} \mathcal{K} : \lambda(k,t) = y\right) \leq \delta \ .$$

- $\lambda(k,t)$ is said to be δ-almost universal (δ-AU) if for any distinct t and $t' \in \mathcal{T}$,

$$\Pr\left(k \leftarrow_{\$} \mathcal{K} : \lambda(k,t) = \lambda(k,t')\right) \leq \delta \ .$$

- $\lambda(k,t)$ is said to be δ-almost XOR-universal (δ-AXU) if for any distinct t and $t' \in \mathcal{T}$ and any $y \in \mathcal{Y}$,

$$\Pr\left(k \leftarrow_{\$} \mathcal{K} : \lambda(k,t) \oplus \lambda(k,t') = y\right) \leq \delta \ .$$

Interestingly, the XHX provable bound shows that we can achieve a security growing in κ and, in particular, achieve a security above 2^n in the ideal cipher setting. Notice that it is impossible prove beyond 2^n security by only assuming an sprp-secure block cipher. There is a matching attack on XHX that looks for distinct tweaks t, t' such that $\widetilde{E}_k^t(m) = \widetilde{E}_k^{t'}(m)$ which happens as soon as both $\lambda(k,t) = \lambda(k,t')$ and $\gamma(k,t) = \gamma(k,t')$.

This is effectively an $(n+\kappa)$-bit collision and thus is expected to require $\mathcal{O}(2^{(n+\kappa)/2})$ queries, matching the proof.

This attack can be adapted to the generalized version of XHX, GXHX (Figure 9.1), where $\lambda(k,t)$ is replaced by two independent AXU functions $\lambda_0(k,t)$ and $\lambda_1(k,t)$ as $\widetilde{E}_k^t(m) = \lambda_0(k,t) \oplus E_{\gamma(k,t)}(m \oplus \lambda_1(k,t))$. Simply look for two distinct tweak t, t' such that $\widetilde{E}_k^t(m) \oplus \widetilde{E}_k^t(m \oplus 1) = \widetilde{E}_k^{t'}(m) \oplus \widetilde{E}_k^{t'}(m \oplus 1)$ which is independent of $\lambda_1(k,\cdot)$ and thus happens as soon as $\lambda_0(k,t) = \lambda_0(k,t')$ and $\gamma(k,t) = \gamma(k,t')$. GXHX has the same provable security bound as XHX. However, using Galois Field multiplication as subkey functions can prevent the attack: it is impossible to find two different tweaks such that $k_0 \cdot t = k_0 \cdot t'$. Nevertheless, the attack on single FX turns out to have the same complexity as our generic attack on iterated FX for a single round. Hence, it is always possible to attack GXHX in $\mathcal{O}(2^{(n+\kappa)/2})$ queries.

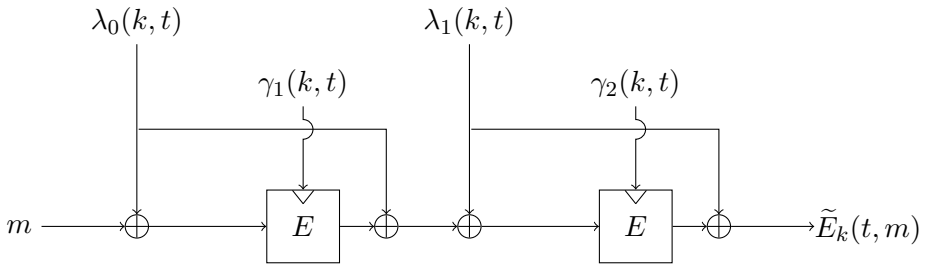


Figure 9.2: The XHX2 construction is the iteration of two independent XHX with AU functions $\gamma 1(\cdot,\cdot), \gamma 2(\cdot,\cdot)$ and AXU functions $\lambda_0(\cdot,\cdot), \lambda_1(\cdot,\cdot)$.

Later, Lee and Lee analyzed XHX2 (Figure 9.2), the iteration of two independent XHX, and showed that it is $\widetilde{\mathsf{sprp}}$ secure up to $\min\{2^{\frac{2}{3}(n+\kappa)}, 2^{n+\kappa/2}\}$ queries [LL18]. The tightness of the bound was left as an open question, and we'll show that our generic attack for two rounds actually matches this bound.

This shows that we can improve the security bound by iterating. Therefore, we ask ourselves what security can be achieved by iterating this strategy even further.

9.1 The Generic Tweakable FX Model

The generic iterated tweakable FX model hopes to capture the most generic strategy to build tweakable block ciphers from block ciphers in order to give a lower-bound on its security advantage.

9.1.1 Notations

Security in the Ideal Cipher Model. First, let us define the $\widetilde{\mathsf{sprp}}$ security game we will use. In Section 7.2.2 we've seen the definition of $\widetilde{\mathsf{sprp}}$, but the fact that we are in the ideal cipher setting also reflects onto the distinguishing game. Concretely, a distinguishing game in the ideal cipher setting captures the fact that the attacker can directly query the underlying block cipher under any key which should behave like a family of independent and random permutations.

Consider a tweakable block cipher $\widetilde{E}_k(t,\cdot)$ defining a family of n to n-bit permutations indexed by a $\widetilde{\kappa}$-bit master key k and a τ-bit tweak t, that is $\widetilde{E} : \{0,1\}^{\widetilde{\kappa}} \times \{0,1\}^{\tau} \times \{0,1\}^{n} \to \{0,1\}^{n}$. Furthermore, consider that $\widetilde{E}_k(t,\cdot)$ is a construction based on a block cipher $E(u,\cdot)$ defining a family of n to n-bit permutations indexed by a κ-bit subkey u, that is $E : \{0,1\}^{\kappa} \times \{0,1\}^{n} \to \{0,1\}^{n}$. Then, we define the $\widetilde{\mathsf{sprp}}$ advantage of $\widetilde{E}$ in the ideal cipher model as:

$$\begin{aligned}\mathbf{Adv}^{\widetilde{\mathsf{sprp}}}(\mathcal{A}) = \mathbf{Pr}(\mathcal{A}^{\widetilde{E}_k(\cdot,\cdot),\widetilde{E}_k^{-1}(\cdot,\cdot),P(\cdot,\cdot),P^{-1}(\cdot,\cdot)} \to 1) \\ - \mathbf{Pr}(\mathcal{A}^{p(\cdot,\cdot),p^{-1}(\cdot,\cdot),P(\cdot,\cdot),P^{-1}(\cdot,\cdot)} \to 1)\end{aligned}$$

where $\widetilde{E}$ is based on the public family of permutations $P(\cdot,\cdot)$ instead of $E(\cdot,\cdot)$, $k \xleftarrow{\$} \{0,1\}^{\widetilde{\kappa}}$, $p(t,\cdot)$ and $P(u,\cdot)$ are independent and random n to n-bit permutations for all $t \in \{0,1\}^{\tau}$ and $u \in \{0,1\}^{\kappa}$.

When multiple block ciphers are used, the number of oracles simply increases accordingly to allow the attacker to query each one of them individually. This is similar to how we handle security games of schemes based on a public permutation, in particular for the Even-Mansour construction studied in Chapter 8.

However, in this chapter we don't directly consider the $\widetilde{\mathsf{sprp}}$ advantage of the iterated tweakable FX construction. Instead, we'll build a key recovery attack on $\widetilde{E}_k(\cdot,\cdot)$ using known plaintexts only. Notice that such a key recovery is an $\widetilde{\mathsf{sprp}}$ distinguisher as well as a $\widetilde{\mathsf{prp}}$ distinguisher (no

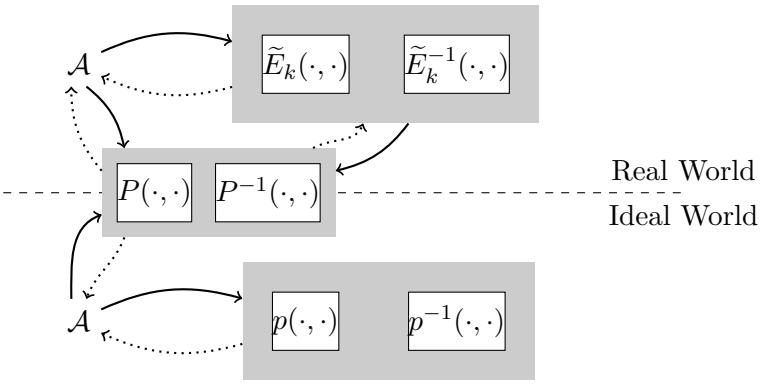


Figure 9.3: Distinguishing game in the ideal cipher setting for the $\widetilde{\mathsf{sprp}}$ security of a tweakable block cipher $\widetilde{E}$ where P and p are independent families of independent random permutations ($P(t,\cdot)$ and $p(t,\cdot)$ are two independent random permutations for all t) and k is a random key value.

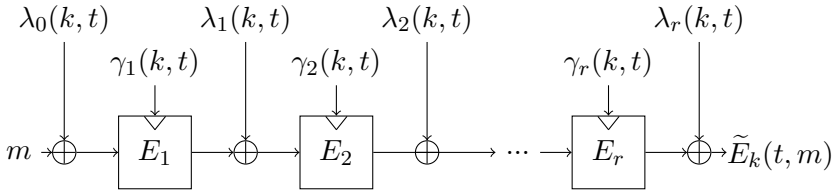


Figure 9.4: r-Round Tweakable FX.

backward queries). Following this, queries to the underlying block cipher $E(\cdot,\cdot)$ are offline queries (they are computations) and make up for the query complexity, while queries to $\widetilde{E}_k(\cdot,\cdot)$ are online queries and make up for the data complexity.

The Generic Construction. We first describe a most generic construction that captures many constructions of tweakable block ciphers. The design strategy of most tweakable block cipher constructions can be seen as a two-step process: first, expand the key space and, then, derive all subkeys from the tweak and the master key. Expanding the key space is usually done by the FX construction that can be iterated r times to allow for alternating $r+1$ n-bit subkeys and r block cipher's κ-bit subkeys (Section 7.1.3). Therefore, we formally define the generic r-round tweakable FX construction shown in Figure 9.4 as follows. Let $E_{1,2,\ldots,r}(u,\cdot)$ be

r block ciphers with κ-bit key u and n-bit input and output. Let k be the $\widetilde{\kappa}$-bit master key of the tweakable block cipher construction and t be a tweak of arbitrary length. Let $\gamma_i(k,t)$ be the subkey for the ith block cipher of length κ-bit for $1 \leq i \leq r$ and $\lambda_i(k,t)$ the n-bit subkey to be XORed in the state for $0 \leq i \leq r$. For every plaintext/tweak input (m,t) the output $\widetilde{E}_k(t,m) = c$ is defined as:

$$\begin{aligned} s_0 &:= m \oplus \lambda_0(k,t) \\ s_i &:= E_i(\gamma_i(k,t), s_{i-1}) \oplus \lambda_i(k,t) \qquad , \text{ for } 1 \leq i \leq r \\ c &:= s_r \ . \end{aligned}$$

For example the $r = 2$-round tweakable FX construction $\widetilde{E}_k(t,m)$ is described as:

$$\widetilde{E}_k(t,m) = E_2\big(\gamma_2(k,t), E_1\big(\gamma_1(k,t), m \oplus \lambda_0(k,t)\big) \oplus \lambda_1(k,t)\big) \oplus \lambda_2(k,t)$$

A description of previous constructions within this generic framework is given in Table 9.1.

9.1.2 Results

In this chapter we ask ourselves what is the best security bound attainable when using the iterated FX framework to build tweakable block ciphers from regular block ciphers. To do this, we improve on the attack by Gaži [Gaž13] (Algorithm 7.5 of Section 7.1.3) on regular iterated FX construction to apply it in the tweakable setting.

Most proposed schemes can be rewritten within this framework and most single round constructions have well understood security with tight bounds and matching attacks. On the other hand, we don't know of any constructions involving more than 2 rounds of tweakable FX. This is why we first focus on $r = 2$ in Section 9.2 and describe an information theoretic key recovery when $\kappa \leq 2n$ with offline and online query complexity of:

$$Q = \mathcal{O}(2^{\frac{2}{3}(n+\kappa)} \cdot \sqrt[3]{\widetilde{\kappa}/n}) \ .$$

Note that $Q = \mathcal{O}(2^{\frac{2}{3}(n+\kappa)})$ under the reasonable assumption that the size of the master secret key is linear with respect to the state size, that is, $\widetilde{\kappa} = \mathcal{O}(n)$.

In particular, the `XHX2` construction by Lee and Lee [LL18] is included in the 2-round tweakable FX framework where $\lambda_1(k,t) = \lambda_0(k,t) \oplus \lambda_2(k,t)$.

Ref	Scheme	r	Subkey functions
[LRW11]	`LRW2`	1	$\lambda_0(k,t) = \lambda_1(k,t)$ a uniform and AXU function. $\gamma_1(k,t) = k$
[Men15]	$\widetilde{F}[1]$	1	$\lambda_0(k,t) = \lambda_1(k,t) = t \cdot k$ $\quad \gamma_1(k,t) = t \oplus k$
[Men15]	$\widetilde{F}[2]$	1	$\lambda_0(k,t) = \lambda_1(k,t) = E_1(2 \cdot k, t)$ $\quad \gamma_1(k,t) = t \oplus k$
[Men16]	`XPX`	1	$\kappa = 0$ so $E_1(\cdot, m) = P(m)$ $\quad t = t_{11} \parallel t_{12} \parallel t_{21} \parallel t_{22}$ $\lambda_0(k,t) = t_{11}k \oplus t_{12}P(k)$ $\quad \lambda_1(k,t) = t_{21}k \oplus t_{22}P(k)$
[Jha+17]	`XHX`	1	$\gamma_1(k,t)$ a uniform and AU function. $\lambda_0(k,t) = \lambda_1(k,t)$ a uniform and AXU function.
[LRW11]	`LRW1`	2	$\lambda_0(k,t) = \lambda_2(k,t) = 0$ $\quad \lambda_1(k,t) = t$ $\gamma_1(k,t) = \gamma_2(k,t) = k$
[LST12]	`CLRW2`	2	$\lambda_0(k,t)$ and $\lambda_2(k,t)$ two uniform and AXU functions. $\lambda_1(k,t) = \lambda_0(k,t) \oplus \lambda_2(k,t)$ $\quad \gamma_1(k,t) = \gamma_2(k,t) = k$
[LL18]	`XHX2`	2	$\gamma_1(k,t)$ and $\gamma_2(k,t)$ two uniform and AU functions. $\lambda_0(k,t)$ and $\lambda_2(k,t)$ two uniform and AXU functions. $\lambda_1(k,t) = \lambda_0(k,t) \oplus \lambda_2(k,t)$

Table 9.1: Some previously proposed schemes and description of how it fits in our iterated tweakable FX generic framework.
Multiplications $(\cdot)$ are over a characteristic 2 finite field.

Their provable security bound is $2^{\frac{2}{3}(n+\kappa)}$ whenever $\kappa \leq 2n$ which matches our attack. Therefore, this attack proves the tightness of their bound and, conversely, their bound proves the optimality of the attack.

Even if we don't know of any construction matching 3 or more rounds of the iterated tweakable FX framework, there seem to be additional security to be gained from iterating further. In Section 9.3 we extend the attack to any number of rounds of the iterated tweakable FX construction. The result is an information theoretic key recovery on r rounds when $\kappa \leq rn$ with offline and online query complexity of:

$$Q = \mathcal{O}(2^{\frac{r}{r+1}(n+\kappa)} \cdot \sqrt[r+1]{\widetilde{\kappa}/n}) \ .$$

Again, note that $Q = \mathcal{O}(2^{\frac{r}{r+1}(n+\kappa)})$ under the assumption that $\widetilde{\kappa} = \mathcal{O}(n)$.

We compare our generic attack with previously known attacks in Table 9.2.

Ref	Scheme	r	Proof	Known Attack	Our Generic Attack
[LRW11]	`LRW2`	1	$2^{n/2}$	$2^{n/2}$	$2^{\frac{1}{2}(n+\kappa)}$
[Men15]	$\widetilde{F}[1]$	1	$2^{\frac{2}{3}n}$	2^n	2^n (as $\kappa = n$)
[Men16]	`XPX`	1	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$ (as $\kappa = 0$)
[Jha+17]	`XHX`	1	$2^{\frac{1}{2}(n+\kappa)}$	$2^{\frac{1}{2}(n+\kappa)}$	$2^{\frac{1}{2}(n+\kappa)}$
[Jha+17]	`GXHX`	1	$2^{\frac{1}{2}(n+\kappa)}$	$2^{\frac{1}{2}(n+\kappa)}$	$2^{\frac{1}{2}(n+\kappa)}$
[Men15]	$\widetilde{F}[2]$	1	2^n	2^n	N.A.
[LRW11]	`LRW1`	2	$2^{n/2}$	$2^{n/2}$	$2^{\frac{2}{3}(n+\kappa)}$
[LST12]	`CLRW2`	2	$2^{2n/3}$	$2^{3n/4}$	$2^{\frac{2}{3}(n+\kappa)}$
[LL18]	`XHX2`	2	$2^{\frac{2}{3}(n+\kappa)}$	$2^{n/2+\kappa}$	$2^{\frac{2}{3}(n+\kappa)}$

Table 9.2: Some previously proposed schemes with their known asymptotic bounds.

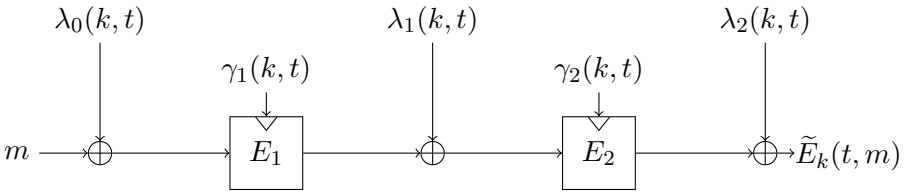


Figure 9.5: 2-Round Tweakable FX.

9.2 Cryptanalysis of 2-Round Tweakable FX

In this section we give an algorithm to extract the master key of a 2-round tweakable FX construction (Algorithm 9.1). Then, we analyze its probability of success by deriving the required total query complexity.

9.2.1 Algorithm

The cryptanalysis shown in Algorithm 9.1 is a key recovery attack following the idea of the original cryptanalysis by Gaži [Gaž13]: we want just enough data to construct contradictory paths for each wrong key. First, we do a large amount of offline computations under all possible κ-bit key for the block ciphers. Input values are the sets S_1 and S_2 which can be chosen randomly and the input/output pairs under the key j are stored in $\mathcal{L}_{j,1}$ and $\mathcal{L}_{j,2}$ for $E_1(j,\cdot)$ and $E_2(j,\cdot)$ respectively. In Step 12 we store all known tweak/plaintext/ciphertext triples in $\mathcal{L}_0$. We don't need to choose the set S_0 of inputs to the tweakable block cipher as the attack works in the known plaintext setting. Finally, we can test all the κ-bit values that are potential master keys k only using the stored values by reconstructing the paths round by round.

Indeed, sets $\mathcal{A}$ and $\mathcal{B}$ reconstruct the paths under the current key guess. For completeness, we provide Algorithm 9.2 to show how to construct the sets $\mathcal{A}$ and $\mathcal{B}$. To construct $\mathcal{A}$ is to apply Algorithm 9.2 with $\mathcal{A} = \textsc{MergeSet}(S_0, \mathcal{L}_{\gamma_1(k,t),1}, \lambda_0(k,t))$. For every guess of k, the goal is to check every known tweak/message pairs, compute the input to the first block cipher $m \oplus \lambda_0(k,t)$ and see in set $\mathcal{L}_{\gamma_1(k,t),1}$ whether we already know its output. If we know it, we record the guessed internal state. Then, starting from the many guessed states (under the guessed key), we do the same with $\mathcal{B}$ and record, if possible, the internal state after the second block cipher call.

Algorithm 9.1 Cryptanalysis of 2-round tweakable FX construction.

input: $\widetilde{E}$ is tweakable FX with given block ciphers and subkeys, $\kappa \leq 2n$.
output: k : the master key of $\widetilde{E}$.
procedure 2KeyRecovery($\widetilde{\kappa}, n, \kappa, \widetilde{E}, E_1, E_2, \gamma_1, \gamma_2, \lambda_0, \lambda_1, \lambda_2$)
    $\nu \leftarrow \widetilde{\kappa}/n$
    $Q \leftarrow 2^{\frac{2}{3}(n+\kappa)} \cdot \sqrt[3]{\nu}$ ▷ Constants derived in Section 9.2.2
    Randomly sample $S_1 \subset \{0,1\}^n$ with $|S_1| = Q/2^\kappa = 2^{\frac{2n-\kappa}{3}} \sqrt[3]{\nu}$.
    Randomly sample $S_2 \subset \{0,1\}^n$ with $|S_2| = Q/2^\kappa = 2^{\frac{2n-\kappa}{3}} \sqrt[3]{\nu}$.
    for all $j \in \{0,1\}^\kappa$ **do**
        $\mathcal{L}_{j,1} \leftarrow \{(m, E_1(j,m)) : m \in S_1\}$
        $\mathcal{L}_{j,2} \leftarrow \{(m, E_2(j,m)) : m \in S_2\}$ ▷ Offline Queries Sets
    Let $S_0 \subset \{0,1\}^* \times \{0,1\}^n$ with $|S_0| = Q$ be a known tweak/message set.
    $\mathcal{L}_0 \leftarrow \{(t, m, \widetilde{E}(t,m)) : (t,m) \in S_0\}$ ▷ Online Queries Set
    for all $k \in \{0,1\}^{\widetilde{\kappa}}$ **do**
        $\mathcal{A} \leftarrow \{(t,m,a) : (t,m) \in S_0, (m \oplus \lambda_0(k,t), a) \in \mathcal{L}_{\gamma_1(k,t),1}\}$
        $\mathcal{B} \leftarrow \{(t,m,b) : (t,m,a) \in \mathcal{A}, (a \oplus \lambda_1(k,t), b) \in \mathcal{L}_{\gamma_2(k,t),2}\}$ ▷ by Algorithm 9.2
        if $|\mathcal{B}| \geq \nu$ **and** $\forall (t,m,b) \in \mathcal{B} : (t, m, b \oplus \lambda_2(k,t)) \in \mathcal{L}_0$ **then**
            return k
    return $\perp$ ▷ No proper key in the set

Algorithm 9.2 Set construction.

input: $S_1 \subset \mathcal{X} \times \{0,1\}^n, S_2 \subset \{0,1\}^n \times \{0,1\}^n, \ell \in \{0,1\}^n$.
output: $\{(e, s_3) : (e, s_1) \in \mathcal{S}_1, (s_1 \oplus \ell, s_3) \in \mathcal{S}_2\}$.
procedure MergeSet($\mathcal{S}_1, \mathcal{S}_2, \ell$)
    $\mathcal{S}_3 \leftarrow \emptyset$
    for all $(e, s_1) \in \mathcal{S}_1$ **do**
        if $\exists s_3 : (s_1 \oplus \ell, s_3) \in \mathcal{S}_2$ **then**
            $\mathcal{S}_3 \leftarrow \mathcal{S}_3 \cup \{(e, s_3)\}$
    return $\mathcal{S}_3$

At last, the condition $\forall(t, m, b) \in \mathcal{B} : (t, m, b \oplus \gamma_5(k, t)) \in \mathcal{L}_0$ is checking whether the path is consistent with the known tweak/plaintext/ciphertext triples. The additional condition $|\mathcal{B}| \geq \nu$ is simply here to ensure a good reduction.

The constants ν and Q are derived in Section 9.2.2, and the algorithm already ensures that the total query complexity is of magnitude Q. Indeed, once we construct the sets $\mathcal{L}_{j,i}$ and $\mathcal{L}_0$ we will have all the necessary queries to perform the attack. Since $|\mathcal{L}_{j,i}| = |S_i| = Q/2^\kappa$ and there are 2^κ different possible subkeys, the total number of queries to $E_1(\cdot, \cdot)$ and $E_2(\cdot, \cdot)$ is Q. The set $\mathcal{L}_0$ records the online queries so that $|\mathcal{L}_0| = |S_0| = Q$.

9.2.2 Analysis

The Query Complexity. To derive the constant Q used in Algorithm 9.1 we focus on what happens when we guess the correct master key k. In particular, we look at the test of Step 16 and wish to avoid false negative that would reject it even though it is the correct key. Concretely, we need to ensure that $|\mathcal{B}| \geq \nu$ happens with good probability as the second constraint is satisfied by construction when the guess is correct.

First, let's look at the construction of $\mathcal{A}$ in Step 14:

$$\mathcal{A} \leftarrow \left\{(t, m, a) : (t, m) \in S_0, (m \oplus \lambda_0(k, t), a) \in \mathcal{L}_{\gamma_1(k,t),1}\right\}$$

Remember that there are Q values $(t, m) \in S_0$, and, as S_1 is chosen randomly and independently, there is a $|S_1|/2^n$ probability that $(m \oplus \lambda_0(k, t)) \in S_1$ for each (t, m) observed meaning that there exists an a such that $(m \oplus \lambda_0(k, t), a) \in \mathcal{L}_{\gamma_1(k,t),1}$. Therefore, in expectation, we have $|\mathcal{A}| = Q^2/2^{n+\kappa}$.

We do the same reasoning for $\mathcal{B}$ in Step 15:

$$\mathcal{B} \leftarrow \left\{(t, m, b) : (t, m, a) \in \mathcal{A}, (a \oplus \lambda_1(k, t), b) \in \mathcal{L}_{\gamma_2(k,t),2}\right\}$$

to find that in expectation $|\mathcal{B}| = Q^3/2^{2n+2\kappa}$.

With some regularity assumptions, if $|\mathcal{B}| = \nu$ in expectation then $|\mathcal{B}| \geq \nu$ with constant probability. Therefore, we can derive the constant Q in terms of ν as:

$$Q^3/2^{2n+2\kappa} = \nu \implies Q = 2^{\frac{2}{3}(n+\kappa)} \cdot \sqrt[3]{\nu}$$

The Number of Paths. Let us now derive the constant ν so that Step 16 doesn't result in a false positive. In other words, the test must fail for all the wrong guesses of k with good probability.

First, notice that $Q = 2^{\frac{2}{3}(n+\kappa)} \cdot \sqrt[3]{\nu}$ implies that $|\mathcal{B}| = \nu$ in expectation for all guesses of k, good or wrong. If $|\mathcal{B}| < \nu$, the test fails as it should. If $|\mathcal{B}| \geq \nu$, we need to look at the second condition, that is: $\forall (t, m, b) \in \mathcal{B} : (t, m, b \oplus \lambda_3(k, t)) \in \mathcal{L}_0$. When the key guess k is wrong, for a given $(t, m, b) \in \mathcal{B}$ we have $(b \oplus \lambda_3(k, t)) = \widetilde{E}(t, m)$ with a 2^{-n} probability. Since $|\mathcal{B}| \geq \nu$, the second condition is satisfied with probability at most $(2^{-n})^{\nu} = 2^{-\nu \cdot n}$. We need the test to fail for all the wrong guesses and there are $2^{\widetilde{\kappa}} - 1$ such wrong guesses. Therefore, all the tests fail with constant probability when:

$$2^{\widetilde{\kappa}} \cdot 2^{-\nu \cdot n} \leq 1 \implies \widetilde{\kappa} - \nu \cdot n \leq 0 \implies \nu \geq \widetilde{\kappa}/n \ ,$$

thus we take $\nu = \widetilde{\kappa}/n$.

Constraints. For this attack to be coherent we need to make sure that all quantities are well-defined. In particular, we require:

$$\begin{aligned} & 1 \leq |S_i| \\ \iff & 1 \leq 2^{\frac{2}{3}n - \frac{1}{3}\kappa} \cdot \sqrt[3]{\nu} \\ \iff & \kappa \leq 2n + \log(\nu) \end{aligned}$$

which limits the block cipher key size κ to a multiple of the state size n. In practice, block ciphers rarely admit a key larger than $2n$ so this is not a strong limitation.

We also require that all master key / tweak combinations (k, t) induce a different sequence of subkeys. We didn't put any requirement on the functions $\lambda_i(k, t)$ and $\gamma_i(k, t)$ and they may even not depend on k or t, but we nevertheless require that changing one of them induces another permutation for our attack to work. Concretely:

$$\begin{aligned} & \forall k \in \{0,1\}^{\widetilde{\kappa}} \, \forall (t, m) \in S_0 \, \forall (t', m') \in S_0 \ : \\ & t \neq t' \implies \exists i : \gamma_i(k, t) \neq \gamma_i(k, t') \text{ OR } \lambda_i(k, t) \neq \lambda_i(k, t') \ . \end{aligned}$$

This condition mostly ensures that the construction behaves like a tweakable block cipher. Indeed, if this condition is not fulfilled, there is an

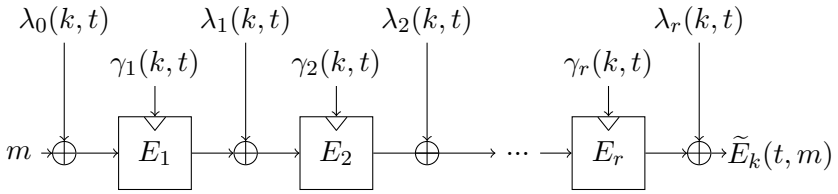


Figure 9.6: r-Round Tweakable FX.

even easier distinguisher: if two tweaks induce the same subkeys, then the permutation is also the same which is a near zero probability event for a family of random permutation, and hence it is an $\widetilde{\mathsf{prp}}$ distinguisher.

9.3 Cryptanalysis of Iterated Tweakable FX

We now generalize the attack of Section 9.2 to attack an arbitrary number of rounds $r \geq 1$ of the iterated tweakable FX construction (Figure 9.6).

9.3.1 Generic Algorithm

The attack is described in Algorithm 9.3. The strategy is the same, we start by doing all the necessary offline queries (Step 10) and online queries (Step 12) before reconstructing paths round by round for all guesses of k (Step 16) to finally check whether the obtained ciphertext is consistent with the known values (Step 17).

9.3.2 Analysis

The Query Complexity. We first derive the constant Q used in Algorithm 9.3 in the same way as we did for the 2-round version. We focus on what happens when we guess the correct master key k. In Step 17 we want to avoid getting a false negative meaning that we need to ensure that $|\mathcal{B}| \geq \nu$ happens with good probability. Again, the second condition is always fulfilled when the guess is correct.

First, the set $\mathcal{A}_1$ is built in Step 14 as:

$$\mathcal{A}_1 \leftarrow \{(t,m,a) : (t,m) \in S_0, (m \oplus \lambda_0(k,t), a) \in \mathcal{L}_{\gamma_1(k,t),1}\} .$$

There are Q values $(t,m) \in S_0$ and S_1 is chosen randomly and independently, there is a $|S_1|/2^n$ probability that $\exists a : (m \oplus \lambda_0(k,t), a) \in \mathcal{L}_{\gamma_1(k,t),1}$

Algorithm 9.3 Cryptanalysis of r-round tweakable FX construction.

input: $\widetilde{E}$ is tweakable FX with given block ciphers and subkeys, $\kappa \leq rn$.
output: k, the master key of $\widetilde{E}$.
procedure rKeyRecovery$(\widetilde{\kappa}, n, \kappa, \widetilde{E}, E_1, ..., E_r, \gamma_1, ..., \gamma_r, \lambda_0, ..., \lambda_r)$
    $\nu \leftarrow \widetilde{\kappa}/n$
    $Q \leftarrow 2^{\frac{r}{r+1}(n+\kappa)} \cdot \sqrt[r+1]{\nu}$

    for all $i \in \{1, ..., r\}$ **do**
        Randomly sample $S_i \subset \{0,1\}^n$ with $|S_i| = Q/2^\kappa = 2^{\frac{rn-\kappa}{r+1}} \sqrt[r+1]{\nu}$.
    for all $j \in \{0,1\}^\kappa$ **do**
        for all $i \in \{1, ..., r\}$ **do**
            $\mathcal{L}_{j,i} \leftarrow \{(m, E_i(j, m)) : m \in S_i\}$ ▷ Offline Queries Sets

    Let $S_0 \subset \{0,1\}^* \times \{0,1\}^n$ with $|S_0| = Q$ be an observable tweak/message set.
    $\mathcal{L}_0 \leftarrow \{(t, m, \widetilde{E}(t, m)) : (t, m) \in S_0\}$ ▷ Online Queries Set

    for all $k \in \{0,1\}^{\widetilde{\kappa}}$ **do**
        $\mathcal{A}_1 \leftarrow \{(t, m, a) : (t, m) \in S_0, (m \oplus \lambda_0(k, t), a) \in \mathcal{L}_{\gamma_1(k,t),1}\}$
        for all $i \in \{2, ..., r\}$ **do**
            $\mathcal{A}_i \leftarrow \{(t, m, a) : (t, m, \bar{a}) \in \mathcal{A}_{i-1}, (\bar{a} \oplus \lambda_{i-1}(k, t), a) \in \mathcal{L}_{\gamma_i(k,t),i}\}$
            ▷ by Algorithm 9.2
        if $|\mathcal{A}_r| \geq \nu$ **and** $\forall (t, m, a) \in \mathcal{A}_r : (t, m, a \oplus \lambda_r(k, t)) \in \mathcal{L}_0$ **then**
            return k
    return $\perp$ ▷ No proper key in the set

for all observed tweak/message pairs (t, m). Therefore, in expectation, we have $|\mathcal{A}_1| = Q^2/2^{n+\kappa}$.

Moreover, it is easy to show by induction that $|\mathcal{A}_i| = Q^{i+1}/2^{i(n+\kappa)}$ in expectation as it is true for $|\mathcal{A}_1|$ and, in expectation and following Step 16, $|\mathcal{A}_{i+1}| = |\mathcal{A}_i| \cdot |S_{i+1}|/2^n$. Thus, we get $|\mathcal{A}_r| = Q^{r+1}/2^{r(n+\kappa)}$.

Under some regularity assumptions, if in expectation we set $|\mathcal{A}_r| = \nu$ then $|\mathcal{A}_r| \geq \nu$ with constant probability. We deduce the value of Q depending on ν as:

$$Q^{r+1}/2^{r(n+\kappa)} = \nu \implies Q = 2^{\frac{r}{r+1}(n+\kappa)} \cdot \sqrt[r+1]{\nu}$$

The Number of Paths. Again, let us derive the constant ν to avoid all false positives in Step 17 of Algorithm 9.3.

If $|\mathcal{A}_r| < \nu$, the test fails as it should. If $|\mathcal{A}_r| \geq \nu$, the second condition is satisfied with probability $(2^{-n})^\nu = 2^{-\nu \cdot n}$. There are $2^{\widetilde{\kappa}} - 1$ wrong guesses so all the tests should fail at least with constant probability when:

$$2^{\widetilde{\kappa}} \cdot 2^{-\nu \cdot n} \leq 1 \implies \widetilde{\kappa} - \nu \cdot n \leq 0 \implies \nu \geq \widetilde{\kappa}/n$$

thus we take $\nu = \widetilde{\kappa}/n$.

Constraints. At last, we check the coherence of our values so that:

$$\begin{aligned} & 1 \leq |S_i| \\ \iff & \kappa \leq rn + \log(\nu) \end{aligned}$$

which limits κ to a multiple of the state size n.

And the condition on the induced subkeys is:

$$\begin{aligned} & \forall k \in \{0,1\}^{\widetilde{\kappa}} \; \forall (t,m) \in S_0 \; \forall (t',m') \in S_0 \; : \\ & t \neq t' \implies \exists i : \gamma_i(k,t) \neq \gamma_i(k,t') \text{ OR } \lambda_i(k,t) \neq \lambda_i(k,t') \end{aligned}$$

Note that this condition prevents the known matching attack on `XHX`. Indeed, as for `XHX` $r = 1$ and $\lambda_0 = \lambda_1$, a collision on the full subkeys is expected after trying $\mathcal{O}(2^{(n+\kappa)/2})$ different tweaks. However, our attack works with the same complexity even when we can't observe a lot of tweaks. It also works on the generalized setting `GXHX` that doesn't enforce $\lambda_0 = \lambda_1$.

In fact, for $r = 1$ and $\kappa \leq n$ our attack is equivalent to the generic attack on the FX construction (Algorithm 7.4). Indeed, when $k \leq n$ the online query complexity is less than 2^n, so we don't take full advantage of the tweakable setting.

9.4 Remarks and Conclusion

Comparison with Attacks on FX. Our attack on iterated tweakable FX is closely related to the attack on regular iterated FX by Gaži [Gaž13] with two main differences. First, in the tweakable case, the number of online queries is unbounded (it is bounded by 2^n in [Gaž13]), so we better balance out the queries to obtain a lower query complexity. Moreover, our attack shows that the tweak space does not matter: we don't even need to choose the tweak for this to work. Indeed, to generically apply the regular FX attack we would need to fix a tweak and perform all of our query, but our analysis shows that this is not required.

The query complexity of the regular FX attack is $\mathcal{O}(2^{\frac{r-1}{r}n+\kappa})$ offline and 2^n online queries. When $\kappa \leq \frac{n}{r}$ though, the offline query complexity fall below the online query complexity so it is actually easy to rebalance the query complexity to $\mathcal{O}(2^{\frac{r}{r+1}(n+\kappa)})$ in the same way as our attack. So the total query complexity of the original attack by Gaži [Gaž13] is $\mathcal{O}(2^{\frac{r-1}{r}n+\kappa})$ or $\mathcal{O}(2^{\frac{r}{r+1}(n+\kappa)})$ when $\kappa \leq \frac{n}{r}$. Notice that for a single round this matches the complexity of our attack, that is $\mathcal{O}(2^{\frac{r}{r+1}(n+\kappa)})$ when $\kappa \leq n$.

Using Tweakable Block Ciphers. If, instead of regular block ciphers, we use tweakable block ciphers, then it is not trivial to adapt this attack. Indeed, the cryptanalysis exploits the fact that the master key and the tweak must be blended before computation and not separately plugged in a tweakable block cipher. Such a construction of a tweakable block cipher based on another tweakable block cipher could be used to increase the security and/or the size of the tweak, in the same way that the original FX construction builds a stronger block cipher from another block cipher. On the cryptanalytic side, it is always possible to fix a single tweak and perform an attack on regular iterated FX.

Weaker Constructions. The attack described is generic given any reasonable key schedule represented by the λ and γ functions. However, they are particular cases where better attacks are possible. In particular, the cascaded `LRW2` construction is a 2-round tweakable FX construction where the key in the block cipher does not vary with the tweaks (γ_1 and γ_2 don't depend on t). This construction permits an attack in $\mathcal{O}(2^{\frac{3n}{4}})$ as shown by Mennink [Men18] using only two different tweaks which beats our generic attack as soon as $\kappa > \frac{n}{8}$.

Tweak-rekeying. In fact, our generic attack being a key recovery attack, it necessarily requires at least 2^κ calls to the underlying block cipher. As soon as $\kappa \geq n$, this implies a complexity above 2^n. Mennink [Men17] showed that provable 2^n security is unattainable in the standard block cipher model where one only assumes the block cipher to be a good pseudo-random permutation. Therefore, our generic attack can only hope to be tight for schemes that are proven secure with an ideal cipher and use tweak-rekeying.

Towards Simplicity. The attack on generic 2-round tweakable FX is also tight since Lee and Lee could prove with `XHX2` [LL18] that we can reach this level of security even when $\lambda_1(k,t) = \lambda_0(k,t) \oplus \lambda_2(k,t)$ with some conditions on those functions. Moreover, the previously known matching attack on `XHX` [Jha+17] exploited the fact that $\lambda_0(k,t) = \lambda_1(k,t)$ but our generic attack shows that it cannot be made more secure without this simplification. Another way to say it is that enforcing $\lambda_0(k,t) = \lambda_1(k,t)$ does not affect the provable security bound.

Using the iterated tweakable FX framework, one can therefore wonder how much can we simplify the subkey functions while maintaining an optimal provable security with respect to the generic security upper bound shown in this work.

General Conclusion

This manuscript introduced many provably secure cryptographic constructions along with the contributions made during my thesis. In this general conclusion I wish to insist on a few points that link most of my works as well as give orientations for future research topics.

Proofs and Cryptanalysis. Even though the techniques used can be quite different, proofs and cryptanalysis fundamentally complete each others. The intuition of cryptanalysis often gives insight on its dual proof.

While this manuscript clearly focuses on cryptanalysis, techniques for proofs are introduced in Chapter 6 to show the robustness guarantee of the authenticated encryption mode `MONDAE` which is derived from `SUNDAE`. However, the intuition for the design of `MONDAE` clearly comes from the simple RUP attack shown in Section 6.1.2. Indeed, `MONDAE` was first thought as a quick fix to avoid this simple attack and, as we couldn't come up with anything else, we derived a proof to formally show its AERUP security. Hence, even simple attacks can give the necessary insights to build stronger provably secure designs.

Another good example is shown in Chapter 5 about Double-block Hash-then-Sum MACs. The attacks described are using $\mathcal{O}(2^{3n/4})$ tag queries for small messages and a single verification query [LNS18]. At the time of publication, this result was not known to be information theoretically optimal, but about a year and a half later Kim, Lee and Lee [KLL20] improved the best proof guaranteeing security up to $\Omega(2^{3n/4})$ short queries, matching our result for many constructions. We finally have a tight bound on the security of `SUM-ECBC`, about 10 years after it's been proposed by Yasuda [Yas10]. The timing of the proof quickly following after the cryptanalysis is not a coincidence and, in a private communication, one author of [KLL20] wrote that our attack inspired them to improve the proof to this tight bound. Hence, it is clear that the

intuition behind one cryptanalysis actually helps to improve designs and build better proofs. Combining proof and analysis can reduce the gap and lead to tight proofs.

From Information Theoretic to Computational Security. There are many examples of substantial gaps between the best information theoretic proof and the best time complexity cryptanalysis even when the proof is tight in the IT model. In particular, this happened with the cryptanalysis of `Poly1305` and `GMAC` in Chapter 4 ($\Omega(2^{n/2})$ vs $\mathcal{O}(2^{2n/3})$), with the cryptanalysis on 2-round Even-Mansour scheme in Chapter 8 ($\Omega(2^{2n/3})$ vs $\mathcal{O}(2^n/n)$), and with the generic attack on Double-block Hash-then-Sum in Chapter 5 ($\Omega(2^{3n/4})$ vs $\mathcal{O}(2^n)$).

In fact, the best attack on those cited cases solves some algorithmic problem that is a particular instance of the missing difference, the 3-XOR and the 4-XOR, respectively. While it might be possible to speed up those attacks, it may well be impossible to reach a lower time complexity. To prove such an impossibility would provide a computational security bound that potentially goes beyond the information theoretic one to match the best known time complexity.

Having computational security in cryptography is not a new thing; this is what is basic `prp` assumption for block cipher is about. This is also the basis of all public key schemes that relies on security reduction to a conjectured hard problem such as the discrete logarithm or learning with error. Moreover, the relatively recent fine-grained complexity theory aims at categorizing and relating algorithmic problems with polynomial time complexity solvers. Typically, it is now fairly common to assume that random 3-SUM, and also random 3-XOR, requires a quadratic (relative to the lists size and ignoring log factors) amount of computations to solve.

How can we link the security of the mentioned symmetric key constructions to a group of problem in the fine-grained complexity theory remains an open question, but it is a promising way to accurately characterize their practical security.

Ideal Primitives against Practice. We've seen many schemes proved with ideal primitives, be it ideal ciphers with the tweakable FX construction in Chapter 9 or ideal permutations with the iterated Even-Mansour in Chapter 8. The proof for those schemes ignores the actual primitive and randomly draws one at the start of the security game. In practice, those

primitives need to be publicly described and efficiently computable which creates a gap between what is proven and what is used. Even worse, Black proposed a hash construction that is provably secure in the ideal cipher model but insecure for any instantiation [Bla06].

Black's construction is a rather unnatural proof of concept as it exploits the compact representation of any instantiation, but it nevertheless means that there is necessarily a gap between the practical primitive and its ideal counterpart. However, there has been no natural scheme that can be proven in the ideal setting but hard to instantiate. Quite the contrary, some schemes seem to be secure even when using relatively weak primitives. For instance, consider the combination of Merkle-Damgård with Davies-Meyer we've seen in Section 7.2.1: Winternitz [Win84] showed that some amount of weak keys and a complementation property (typical of the DES block cipher) posed no lethal threat for the security of the hash scheme. This is also especially true for authenticated encryption schemes using a sponge-like construction as most properties found on various permutations are not exploitable to build an attack on the final mode.

Overall, proving security under an ideal primitive is a good indication that the construction is sound and secure, but it hardly tells what is required from the primitive's instantiation. Since a gap necessarily exists between the ideal version and the practical version, designers may tend to increase it in order to gain efficiency. This makes the proof of security less and less relevant. In fact, proofs in the ideal setting fail to reduce the security to a well-defined but strictly easier cryptanalytic problem on the primitive in the same way as it is done with the classical PRP notion for block ciphers.

Bibliography

[ADL17] Tomer Ashur, Orr Dunkelman, and Atul Luykx. “Boosting Authenticated Encryption Robustness with Minimal Modifications”. In: *CRYPTO 2017, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. LNCS. Springer, Heidelberg, Aug. 2017, pp. 3–33 (cit. on p. 172).

[AES] *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce. Nov. 2001 (cit. on pp. 38, 201, 239).

[Al-ry] Al-Kindi. *A Manuscript on Deciphering Cryptographic Messages*. 9th century (cit. on p. 31).

[Alb+14] Martin R. Albrecht, Benedikt Driessen, Elif Bilge Kavun, Gregor Leander, Christof Paar, and Tolga Yalçin. “Block Ciphers - Focus on the Linear Layer (feat. PRIDE)”. In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 57–76 (cit. on p. 204).

[Alb+16] Martin R. Albrecht, Jean Paul Degabriele, Torben Brandt Hansen, and Kenneth G. Paterson. “A Surfeit of SSH Cipher Suites”. In: *ACM CCS 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM Press, Oct. 2016, pp. 1480–1491 (cit. on p. 123).

[AlF+13] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. “On the Security of RC4 in TLS”. In: *USENIX Security 2013*. Ed. by Samuel T. King. USENIX Association, 2013, pp. 305–320. ISBN: 978-1-931971-03-4 (cit. on p. 102).

[Alk+02] Ammar Alkassar, Alexander Geraldy, Birgit Pfitzmann, and Ahmad-Reza Sadeghi. "Optimized Self-Synchronizing Mode of Operation". In: *FSE 2001*. Ed. by Mitsuru Matsui. Vol. 2355. LNCS. Springer, Heidelberg, Apr. 2002, pp. 78–91 (cit. on p. 51).

[And+14] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. "How to Securely Release Unverified Plaintext in Authenticated Encryption". In: *ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 105–125 (cit. on pp. 15, 82, 162).

[Ban+18] Subhadeep Banik, Andrey Bogdanov, Atul Luykx, and Elmar Tischhauser. "SUNDAE: Small Universal Deterministic Authenticated Encryption for the Internet of Things". In: *IACR Trans. Symm. Cryptol.* 2018.3 (2018), pp. 1–35. ISSN: 2519-173X (cit. on pp. 15, 82, 161, 163, 164).

[Ban+19] Subhadeep Banik, Andrey Bogdanov, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, Elmar Tischhauser, and Yosuke Todo. *SUNDAE-GIFT v1.0*. Submission to NIST Lightweight Cryptography Standardization Process. Mar. 2019 (cit. on p. 164).

[Bao+19a] Zhenzhen Bao, Jian Guo, Tetsu Iwata, and Ling Song. *SIV-Rijndael256 Authenticated Encryption and Hash Family*. Submission to NIST Lightweight Cryptography Standardization Process. Feb. 2019 (cit. on p. 164).

[Bao+19b] Zhenzhen Bao, Jian Guo, Tetsu Iwata, and Ling Song. *SIV-TEM-PHOTON Authenticated Encryption and Hash Family*. Submission to NIST Lightweight Cryptography Standardization Process. Feb. 2019 (cit. on p. 164).

[BCC] *The estimated number of terahashes per second the bitcoin network is performing in the last 24 hours*. Blockchain Company. https://www.blockchain.com/en/charts/hash-rate (cit. on p. 38).

[BDF18] Charles Bouillaguet, Claire Delaplace, and Pierre-Alain Fouque. "Revisiting and Improving Algorithms for the 3XOR Problem". In: *IACR Trans. Symm. Cryptol.* 2018.1 (2018),

pp. 254–276. ISSN: 2519-173X (cit. on pp. 18, 98, 214, 218–220, 226, 229, 230).

[BDP08] Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. "Subquadratic Algorithms for 3SUM". In: *Algorithmica* 50.4 (Apr. 2008), pp. 584–596. ISSN: 1432-0541 (cit. on pp. 18, 19, 98, 214, 218, 219, 226, 230).

[Bea+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. *SIMON and SPECK: Block Ciphers for the Internet of Things*. Cryptology ePrint Archive, Report 2015/585. http://eprint.iacr.org/2015/585. 2015 (cit. on p. 116).

[Bel+97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. "A Concrete Security Treatment of Symmetric Encryption". In: *38th FOCS*. IEEE Computer Society Press, Oct. 1997, pp. 394–403 (cit. on p. 50).

[Ber+12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications". In: *SAC 2011*. Ed. by Ali Miri and Serge Vaudenay. Vol. 7118. LNCS. Springer, Heidelberg, Aug. 2012, pp. 320–337 (cit. on p. 74).

[Ber05a] Daniel J. Bernstein. "Stronger Security Bounds for Wegman-Carter-Shoup Authenticators". In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, Heidelberg, May 2005, pp. 164–180 (cit. on pp. 62, 63, 125).

[Ber05b] Daniel J. Bernstein. "The Poly1305-AES Message Authentication Code". In: *FSE 2005*. Ed. by Henri Gilbert and Helena Handschuh. Vol. 3557. LNCS. Springer, Heidelberg, Feb. 2005, pp. 32–49 (cit. on p. 126).

[Ber07] Daniel J Bernstein. "Better price-performance ratios for generalized birthday attacks". In: *Workshop Record of SHARCS*. Vol. 7. 2007, p. 160 (cit. on p. 228).

[BGM04] Mihir Bellare, Oded Goldreich, and Anton Mityagin. *The Power of Verification Queries in Message Authentication and Authenticated Encryption*. Cryptology ePrint Archive, Report 2004/309. http://eprint.iacr.org/2004/309. 2004 (cit. on p. 58).

[BKN06] M. Bellare, T. Kohno, and C. Namprempre. *The Secure Shell (SSH) Transport Layer Encryption Modes*. IETF RFC 4344. 2006 (cit. on p. 123).

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. "The Security of the Cipher Block Chaining Message Authentication Code". In: *Journal of Computer and System Sciences* 61.3 (2000), pp. 362–399 (cit. on p. 133).

[BL16] Karthikeyan Bhargavan and Gaëtan Leurent. "On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN". In: *ACM CCS 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM Press, Oct. 2016, pp. 456–467 (cit. on pp. 79, 102, 124).

[Bla06] John Black. "The Ideal-Cipher Model, Revisited: An Uninstantiable Blockcipher-Based Hash Function". In: *FSE 2006*. Ed. by Matthew J. B. Robshaw. Vol. 4047. LNCS. Springer, Heidelberg, Mar. 2006, pp. 328–340 (cit. on p. 259).

[BN00] Mihir Bellare and Chanathip Namprempre. "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm". In: *ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Vol. 1976. LNCS. Springer, Heidelberg, Dec. 2000, pp. 531–545 (cit. on p. 67).

[Bog+12] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert, John P. Steinberger, and Elmar Tischhauser. "Key-Alternating Ciphers in a Provable Setting: Encryption Using a Small Number of Public Permutations - (Extended Abstract)". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 45–62 (cit. on pp. 195, 201, 202, 235).

[Bor+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. "PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract". In: *ASIACRYPT 2012*. Ed. by

Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 208–225 (cit. on p. 204).

[BR02] John Black and Phillip Rogaway. "A Block-Cipher Mode of Operation for Parallelizable Message Authentication". In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, Apr. 2002, pp. 384–397 (cit. on p. 145).

[BR06] Mihir Bellare and Phillip Rogaway. "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 409–426 (cit. on p. 38).

[Cha+18] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. "Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers". In: *IACR TCHES* 2018.2 (2018). https://tches.iacr.org/index.php/TCHES/article/view/881, pp. 218–241. ISSN: 2569-2925 (cit. on p. 75).

[Cha+19a] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas Lopez, Mridul Nandi, and Yu Sasaki. *ESTATE*. Submission to NIST Lightweight Cryptography Standardization Process. Mar. 2019 (cit. on p. 164).

[Cha+19b] Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi, Somitra Sanadhya, and Ferdinand Sibleyras. "Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE". In: *IACR Trans. Symm. Cryptol.* 2019.4 (2019), pp. 119–146. ISSN: 2519-173X (cit. on pp. 8, 17, 28, 161).

[Che+14] Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John P. Steinberger. "Minimizing the Two-Round Even-Mansour Cipher". In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 39–56 (cit. on pp. 221, 222, 229, 233).

[CJM02] Philippe Chose, Antoine Joux, and Michel Mitton. "Fast Correlation Attacks: An Algorithmic Point of View". In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332.

LNCS. Springer, Heidelberg, Apr. 2002, pp. 209–221 (cit. on pp. 95, 96, 137).

[CL14] Anne Canteaut and Gaëtan Leurent. "Distinguishing and Key-recovery Attacks against Wheesht". working paper or preprint. Mar. 2014 (cit. on p. 42).

[CS14] Shan Chen and John P. Steinberger. "Tight Security Bounds for Key-Alternating Ciphers". In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 327–350 (cit. on pp. 181, 201, 203).

[CY99] Scott Contini and Yiqun Lisa Yin. *On Differential Properties of Data-Dependent Rotations and Their Use in MARch and RC6*. 1999 (cit. on p. 179).

[Dae93] Joan Daemen. "Limitations of the Even-Mansour Construction (Rump Session)". In: *ASIACRYPT'91*. Ed. by Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto. Vol. 739. LNCS. Springer, Heidelberg, Nov. 1993, pp. 495–498 (cit. on pp. 199, 200).

[Dam90] Ivan Damgård. "A Design Principle for Hash Functions". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 416–427 (cit. on p. 206).

[Dat+15] Nilanjan Datta, Avijit Dutta, Mridul Nandi, Goutam Paul, and Liting Zhang. *Building Single-Key Beyond Birthday Bound Message Authentication Code*. Cryptology ePrint Archive, Report 2015/958. http://eprint.iacr.org/2015/958. 2015 (cit. on pp. 12, 131, 132, 134, 157).

[Dat+17] Nilanjan Datta, Avijit Dutta, Mridul Nandi, Goutam Paul, and Liting Zhang. "Single Key Variant of PMAC_Plus". In: *IACR Trans. Symm. Cryptol.* 2017.4 (2017), pp. 268–305. ISSN: 2519-173X (cit. on pp. 12, 132, 134).

[Dat+18] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Goutam Paul. "Double-block Hash-then-Sum: A Paradigm for Constructing BBB Secure PRF". In: *IACR Trans. Symm. Cryptol.* 2018.3 (2018), pp. 36–92. ISSN: 2519-173X (cit. on pp. 134, 160).

[Dat+19] Nilanjan Datta, Ashrujit Ghoshal, Debdeep Mukhopadhyay, Sikhar Patranabis, Stjepan Picek, and Rajat Sadhukhan. *TRIFLE*. Submission to NIST Lightweight Cryptography Standardization Process. Mar. 2019 (cit. on p. 164).

[DES77] *Data Encryption Standard*. National Bureau of Standards, NBS FIPS PUB 46, U.S. Department of Commerce. Jan. 1977 (cit. on pp. 196, 203, 239).

[DH76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654 (cit. on p. 34).

[DH79] Whitfield Diffie and Martin E Hellman. "Privacy and authentication: An introduction to cryptography". In: *Proceedings of the IEEE* 67.3 (1979), pp. 397–427 (cit. on pp. 53, 102).

[Din+13] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. "Key Recovery Attacks on 3-round Even-Mansour, 8-step LED-128, and Full AES2". In: *ASIACRYPT 2013, Part I*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. LNCS. Springer, Heidelberg, Dec. 2013, pp. 337–356 (cit. on pp. 18, 215, 218, 219).

[Din+16] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. "Key Recovery Attacks on Iterated Even-Mansour Encryption Schemes". In: *Journal of Cryptology* 29.4 (Oct. 2016), pp. 697–728 (cit. on pp. 18, 216, 218–221, 233, 236).

[DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir. "Minimalism in Cryptography: The Even-Mansour Scheme Revisited". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 336–354 (cit. on p. 199).

[DKS15] Orr Dunkelman, Nathan Keller, and Adi Shamir. "Slidex Attacks on the Even-Mansour Encryption Scheme". In: *Journal of Cryptology* 28.1 (Jan. 2015), pp. 1–28 (cit. on pp. 199–201).

[DR11] Thai Duong and Juliano Rizzo. "Here Come The $\oplus$ Ninjas". In: (2011) (cit. on pp. 78, 102).

[Dwo01] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation*. Tech. rep. NIST Special Publication 800-38A, Dec. 2001 (cit. on p. 53).

[Dwo04] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation*. Tech. rep. NIST Special Publication 800-38C, May 2004 (cit. on p. 69).

[Dwo10] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Condidentiality on Storage Devices*. Tech. rep. NIST Special Publication 800-38E, Jan. 2010 (cit. on p. 211).

[Dwo15] Morris Dworkin. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Tech. rep. NIST Federal Inf. Process. Stds. (NIST FIPS) - 202, Aug. 2015 (cit. on pp. 73, 209).

[EM93] Shimon Even and Yishay Mansour. "A Construction of a Cipher From a Single Pseudorandom Permutation". In: *ASIACRYPT'91*. Ed. by Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto. Vol. 739. LNCS. Springer, Heidelberg, Nov. 1993, pp. 210–224 (cit. on pp. 198, 199).

[Fer05a] Niels Ferguson. *Authentication weaknesses in GCM*. Comment to NIST. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf. 2005 (cit. on p. 68).

[Fer05b] Niels Ferguson. *Authentication weaknesses in GCM*. Comment to NIST. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf. 2005 (cit. on p. 126).

[FIPS113] *Computer Data Authentication*. National Bureau of Standards, NIST FIPS PUB 113, U.S. Department of Commerce. 1985 (cit. on p. 133).

[FIPS81] *DES Modes of Operation*. National Institute of Standards and Technology (NIST), FIPS PUB 81, U.S. Department of Commerce. Dec. 1980 (cit. on pp. 48, 50).

[FO90] Philippe Flajolet and Andrew M. Odlyzko. "Random Mapping Statistics". In: *EUROCRYPT'89*. Ed. by Jean-Jacques Quisquater and Joos Vandewalle. Vol. 434. LNCS. Springer, Heidelberg, Apr. 1990, pp. 329–354 (cit. on p. 88).

[FSK11] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography engineering: design principles and practical applications*. John Wiley & Sons, 2011 (cit. on p. 101).

[Gaž13] Peter Gaži. "Plain versus Randomized Cascading-Based Key-Length Extension for Block Ciphers". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 551–570 (cit. on pp. 21, 205, 233, 244, 247, 254).

[GL15] Shay Gueron and Yehuda Lindell. "GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte". In: *ACM CCS 2015*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. ACM Press, Oct. 2015, pp. 109–119 (cit. on p. 82).

[GLL17] Shay Gueron, Adam Langley, and Yehuda Lindell. *AES-GCM-SIV: Specification and Analysis*. Cryptology ePrint Archive, Report 2017/168. http://eprint.iacr.org/2017/168. 2017 (cit. on p. 82).

[Gre13] Matthew Green. *Why I hate CBC-MAC*. Johns Hopkins University, https://blog.cryptographyengineering.com/2013/02/15/why-i-hate-cbc-mac/. Feb. 2013 (cit. on p. 80).

[Gui+11] Bertoni Guido, Daemen Joan, P Michaël, and VA Gilles. *Cryptographic sponge functions*. 2011 (cit. on p. 209).

[HKR15] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. "Robust Authenticated-Encryption AEZ and the Problem That It Solves". In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 15–44 (cit. on p. 171).

[Hoa+15] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. "Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 493–517 (cit. on pp. 79, 119).

[HP08] Helena Handschuh and Bart Preneel. "Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms". In: *CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. Springer, Heidelberg, Aug. 2008, pp. 144–161 (cit. on p. 126).

[IM16] Tetsu Iwata and Kazuhiko Minematsu. "Stronger Security Variants of GCM-SIV". In: *IACR Trans. Symm. Cryptol.* 2016.1 (2016). http://tosc.iacr.org/index.php/ToSC/article/view/539, pp. 134–157. ISSN: 2519-173X (cit. on pp. 12, 132, 134, 143).

[IM18] Akiko Inoue and Kazuhiko Minematsu. *Cryptanalysis of OCB2*. Cryptology ePrint Archive, Report 2018/1040. https://eprint.iacr.org/2018/1040. 2018 (cit. on p. 73).

[IMV16] Tetsu Iwata, Bart Mennink, and Damian Vizár. *CENC is Optimally Secure*. Cryptology ePrint Archive, Report 2016/1087. http://eprint.iacr.org/2016/1087. 2016 (cit. on pp. 76, 129).

[Ino+19] Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, and Bertram Poettering. "Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality". In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. LNCS. Springer, Heidelberg, Aug. 2019, pp. 3–31 (cit. on p. 73).

[IOM12] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. "Breaking and Repairing GCM Security Proofs". In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 31–49 (cit. on p. 68).

[IS17] Takanori Isobe and Kyoji Shibutani. "New Key Recovery Attacks on Minimal Two-Round Even-Mansour Ciphers". In: *ASIACRYPT 2017, Part I*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, Heidelberg, Dec. 2017, pp. 244–263 (cit. on pp. 18, 216–219, 221, 230).

[Iwa06] Tetsu Iwata. "New Blockcipher Modes of Operation with Beyond the Birthday Bound Security". In: *FSE 2006*. Ed. by Matthew J. B. Robshaw. Vol. 4047. LNCS. Springer, Heidelberg, Mar. 2006, pp. 310–327 (cit. on pp. 76, 129).

[Jha+17] Ashwin Jha, Eik List, Kazuhiko Minematsu, Sweta Mishra, and Mridul Nandi. "XHX - A Framework for Optimally Secure Tweakable Block Ciphers from Classical Block Ciphers and Universal Hashing". In: *LATINCRYPT 2017*. Ed. by Tanja Lange and Orr Dunkelman. Vol. 11368. LNCS. Springer, Heidelberg, Sept. 2017, pp. 207–227 (cit. on pp. 22, 211, 239, 245, 246, 255).

[JJV02] Éliane Jaulmes, Antoine Joux, and Frédéric Valette. "On the Security of Randomized CBC-MAC Beyond the Birthday Paradox Limit: A New Construction". In: *FSE 2002*. Ed. by Joan Daemen and Vincent Rijmen. Vol. 2365. LNCS. Springer, Heidelberg, Feb. 2002, pp. 237–251 (cit. on p. 133).

[JL09] Antoine Joux and Stefan Lucks. "Improved Generic Algorithms for 3-Collisions". In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 347–363 (cit. on p. 216).

[Jon03] Jakob Jonsson. "On the Security of CTR + CBC-MAC". In: *SAC 2002*. Ed. by Kaisa Nyberg and Howard M. Heys. Vol. 2595. LNCS. Springer, Heidelberg, Aug. 2003, pp. 76–93 (cit. on p. 70).

[Jou04] Antoine Joux. "Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions". In: *CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. LNCS. Springer, Heidelberg, Aug. 2004, pp. 306–316 (cit. on pp. 154, 206, 207).

[Jou06] Antoine Joux. *Authentication failures in NIST version of GCM*. Comment to NIST. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/GCM/Joux_comments.pdf. 2006 (cit. on p. 126).

[Jou09] Antoine Joux. *Algorithmic Cryptanalysis*. 1st. Chapman & Hall/CRC, 2009. ISBN: 1420070029, 9781420070026 (cit. on pp. 18, 97, 218, 225–227).

[KLL20] Seongkwang Kim, ByeongHak Lee, and Jooyoung Lee. "Tight Security Bounds for Double-Block Hash-then-Sum MACs". In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Heidelberg, May 2020, pp. 435–465 (cit. on pp. 11, 12, 76, 131, 132, 134, 149, 160, 257).

[KM03] Lars R Knudsen and Chris J Mitchell. "Analysis of 3gpp-MAC and two-key 3gpp-MAC". In: *Discrete Applied Mathematics* 128.1 (2003). International Workshop on Coding and Cryptography (WCC2001)., pp. 181–191. ISSN: 0166-218X (cit. on p. 149).

[KR14] T. Krovetz and P. Rogaway. *The OCB Authenticated-Encryption Algorithm*. RFC 7253, DOI 10.17487/RFC7253, https://www.rfc-editor.org/info/rfc7253. May 2014 (cit. on pp. 72, 210).

[KR96] Joe Kilian and Phillip Rogaway. "How to Protect DES Against Exhaustive Key Search". In: *CRYPTO'96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, Heidelberg, Aug. 1996, pp. 252–267 (cit. on pp. 195, 203, 204).

[KVW04] Tadayoshi Kohno, John Viega, and Doug Whiting. "CWC: A High-Performance Conventional Authenticated Encryption Mode". In: *FSE 2004*. Ed. by Bimal K. Roy and Willi Meier. Vol. 3017. LNCS. Springer, Heidelberg, Feb. 2004, pp. 408–426 (cit. on p. 127).

[KZ19] Patrick Kresmer and Alexander Zeh. *CCM-SIV: Single-PRF Nonce-Misuse-Resistant Authenticated Encryption*. Cryptology ePrint Archive, Report 2019/892. https://eprint.iacr.org/2019/892. 2019 (cit. on p. 82).

[Leu15] Gaëtan Leurent. *Generic Attacks against MAC algorithms*. SAC 2015, Invited Talk. Aug. 2015 (cit. on p. 56).

[LL18] ByeongHak Lee and Jooyoung Lee. "Tweakable Block Ciphers Secure Beyond the Birthday Bound in the Ideal Cipher Model". In: *ASIACRYPT 2018, Part I*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11272. LNCS. Springer, Heidelberg, Dec. 2018, pp. 305–335 (cit. on pp. 21, 22, 211, 241, 244–246, 255).

[LNS18] Gaëtan Leurent, Mridul Nandi, and Ferdinand Sibleyras. "Generic Attacks Against Beyond-Birthday-Bound MACs". In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Heidelberg, Aug. 2018, pp. 306–336 (cit. on pp. 8, 14, 28, 76, 131, 257).

[LP16] Atul Luykx and Kenneth G. Paterson. *Limits on Authenticated Encryption Use in TLS*. http://www.isg.rhul.ac.uk/~kp/TLS-AEbounds.pdf. Mar. 2016 (cit. on pp. 86, 123, 129).

[LP18] Atul Luykx and Bart Preneel. "Optimal Forgeries Against Polynomial-Based MACs and GCM". In: *EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. LNCS. Springer, Heidelberg, Apr. 2018, pp. 445–467 (cit. on pp. 63, 125).

[LPS12] Rodolphe Lampe, Jacques Patarin, and Yannick Seurin. "An Asymptotically Tight Security Analysis of the Iterated Even-Mansour Cipher". In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 278–295 (cit. on pp. 201, 203).

[LR88] Michael Luby and Charles Rackoff. "How to construct pseudorandom permutations from pseudorandom functions". In: *SIAM Journal on Computing* 17.2 (1988) (cit. on pp. 195–197).

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. "Tweakable Block Ciphers". In: *CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. LNCS. Springer, Heidelberg, Aug. 2002, pp. 31–46 (cit. on p. 73).

[LRW11] Moses Liskov, Ronald L. Rivest, and David Wagner. "Tweakable Block Ciphers". In: *Journal of Cryptology* 24.3 (July 2011), pp. 588–613 (cit. on pp. 20, 22, 195, 209, 210, 239, 245, 246).

[LS18] Gaëtan Leurent and Ferdinand Sibleyras. "The Missing Difference Problem, and Its Applications to Counter Mode Encryption". In: *EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. LNCS. Springer, Heidelberg, Apr. 2018, pp. 745–770 (cit. on pp. 8, 11, 28, 63, 101).

[LS19] Gaëtan Leurent and Ferdinand Sibleyras. "Low-Memory Attacks Against Two-Round Even-Mansour Using the 3-XOR Problem". In: *CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. LNCS.

Springer, Heidelberg, Aug. 2019, pp. 210–235 (cit. on pp. 8, 19, 28, 98, 213).

[LST12] Will Landecker, Thomas Shrimpton, and R. Seth Terashima. "Tweakable Blockciphers with Beyond Birthday-Bound Security". In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 14–30 (cit. on pp. 22, 245, 246).

[McG12] David McGrew. *Impossible plaintext cryptanalysis and probable-plaintext collision attacks of 64-bit block cipher modes*. Cryptology ePrint Archive, Report 2012/623. http://eprint.iacr.org/2012/623. 2012 (cit. on pp. 102, 104, 105).

[Men15] Bart Mennink. "Optimally Secure Tweakable Blockciphers". In: *FSE 2015*. Ed. by Gregor Leander. Vol. 9054. LNCS. Springer, Heidelberg, Mar. 2015, pp. 428–448 (cit. on pp. 22, 211, 239, 245, 246).

[Men16] Bart Mennink. "XPX: Generalized Tweakable Even-Mansour with Improved Security Guarantees". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Heidelberg, Aug. 2016, pp. 64–94 (cit. on pp. 22, 211, 245, 246).

[Men17] Bart Mennink. "Insuperability of the Standard Versus Ideal Model Gap for Tweakable Blockcipher Security". In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017, pp. 708–732 (cit. on p. 255).

[Men18] Bart Mennink. "Towards Tight Security of Cascaded LRW2". In: *TCC 2018, Part II*. Ed. by Amos Beimel and Stefan Dziembowski. Vol. 11240. LNCS. Springer, Heidelberg, Nov. 2018, pp. 192–222 (cit. on p. 255).

[Mer79] Ralph Merkle. *Secrecy, Authentication, and public key system*. Tech. rep. Stanford University, June 1979 (cit. on p. 206).

[Min10] Kazuhiko Minematsu. "How to Thwart Birthday Attacks against MACs via Small Randomness". In: *FSE 2010*. Ed. by Seokhie Hong and Tetsu Iwata. Vol. 6147. LNCS. Springer, Heidelberg, Feb. 2010, pp. 230–249 (cit. on p. 133).

[MS15] Brice Minaud and Yannick Seurin. "The Iterated Random Permutation Problem with Applications to Cascade Encryption". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 351–367 (cit. on p. 41).

[MS20] Bill Marczak and John Scott-Railton. *Move Fast and Roll Your Own Crypto*. Tech. rep. Citizen Lab, Munk School of Global Affairs & Public Policy, Apr. 2020 (cit. on p. 49).

[MV04] David A. McGrew and John Viega. "The Security and Performance of the Galois/Counter Mode (GCM) of Operation". In: *INDOCRYPT 2004*. Ed. by Anne Canteaut and Kapalee Viswanathan. Vol. 3348. LNCS. Springer, Heidelberg, Dec. 2004, pp. 343–355 (cit. on pp. 67, 123).

[Nai17] Yusuke Naito. "Blockcipher-Based MACs: Beyond the Birthday Bound Without Message Length". In: *ASIACRYPT 2017, Part III*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10626. LNCS. Springer, Heidelberg, Dec. 2017, pp. 446–470 (cit. on pp. 12, 132, 134, 148).

[Nai18] Yusuke Naito. "Improved Security Bound of LightMAC_Plus and Its Single-Key Variant". In: *CT-RSA 2018*. Ed. by Nigel P. Smart. Vol. 10808. LNCS. Springer, Heidelberg, Apr. 2018, pp. 300–318 (cit. on pp. 77, 131, 134, 148, 159).

[Nan09] Mridul Nandi. "Fast and Secure CBC-Type MAC Algorithms". In: *FSE 2009*. Ed. by Orr Dunkelman. Vol. 5665. LNCS. Springer, Heidelberg, Feb. 2009, pp. 375–393 (cit. on p. 164).

[NS15] Ivica Nikolic and Yu Sasaki. "Refinements of the k-tree Algorithm for the Generalized Birthday Problem". In: *ASIACRYPT 2015, Part II*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, Heidelberg, Nov. 2015, pp. 683–703 (cit. on pp. 96, 97, 137, 215, 225, 226, 228).

[NWW14] Ivica Nikolic, Lei Wang, and Shuang Wu. "Cryptanalysis of Round-Reduced LED". In: *FSE 2013*. Ed. by Shiho Moriai. Vol. 8424. LNCS. Springer, Heidelberg, Mar. 2014, pp. 112–129 (cit. on pp. 18, 215, 218, 219, 226–228).

[Pat09] Jacques Patarin. "The "Coefficients H" Technique (Invited Talk)". In: *SAC 2008*. Ed. by Roberto Maria Avanzi, Liam Keliher, and Francesco Sica. Vol. 5381. LNCS. Springer, Heidelberg, Aug. 2009, pp. 328–345 (cit. on p. 181).

[PC15] Gordon Procter and Carlos Cid. "On Weak Keys and Forgery Attacks Against Polynomial-Based MAC Schemes". In: *Journal of Cryptology* 28.4 (Oct. 2015), pp. 769–795 (cit. on p. 126).

[PGV94] Bart Preneel, René Govaerts, and Joos Vandewalle. "Hash Functions Based on Block Ciphers: A Synthetic Approach". In: *CRYPTO'93*. Ed. by Douglas R. Stinson. Vol. 773. LNCS. Springer, Heidelberg, Aug. 1994, pp. 368–378 (cit. on pp. 207, 208).

[Pv95] Bart Preneel and Paul C. van Oorschot. "MDx-MAC and Building Fast MACs from Hash Functions". In: *CRYPTO'95*. Ed. by Don Coppersmith. Vol. 963. LNCS. Springer, Heidelberg, Aug. 1995, pp. 1–14 (cit. on pp. 61, 92, 133, 136).

[Que86] Mary Queen of Scots. *Page of ciphers used by Mary Queen of Scots, c.1586 (SP 53/22 f.1)*. The National Archives (United Kingdom). 1586 (cit. on p. 32).

[Rog+01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption". In: *ACM CCS 2001*. Ed. by Michael K. Reiter and Pierangela Samarati. ACM Press, Nov. 2001, pp. 196–205 (cit. on p. 72).

[Rog04] Phillip Rogaway. "Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC". In: *ASIACRYPT 2004*. Ed. by Pil Joong Lee. Vol. 3329. LNCS. Springer, Heidelberg, Dec. 2004, pp. 16–31 (cit. on pp. 145, 210).

[Rog11] Phillip Rogaway. *Evaluation of Some Blockcipher Modes of Operation*. Tech. rep. University of California, Davis, Feb. 2011 (cit. on p. 50).

[RP95] Motwani Rajeev and Raghavan Prabhakar. *Randomized Algorithms*. Cambridge University Press, 1995 (cit. on p. 109).

[RS06] Phillip Rogaway and Thomas Shrimpton. "A Provable-Security Treatment of the Key-Wrap Problem". In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 373–390 (cit. on p. 80).

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the Association for Computing Machinery* 21.2 (1978), pp. 120–126 (cit. on p. 34).

[Saa12] Markku-Juhani Olavi Saarinen. "Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes". In: *FSE 2012*. Ed. by Anne Canteaut. Vol. 7549. LNCS. Springer, Heidelberg, Mar. 2012, pp. 216–225 (cit. on p. 126).

[Sha49] Claude E. Shannon. "Communication theory of secrecy systems". In: *Bell Systems Technical Journal* 28.4 (1949), pp. 656–715 (cit. on p. 33).

[Sho96] Victor Shoup. "On Fast and Provably Secure Message Authentication Based on Universal Hashing". In: *CRYPTO'96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, Heidelberg, Aug. 1996, pp. 313–328 (cit. on p. 62).

[Sib20] Ferdinand Sibleyras. "Generic Attack on Iterated Tweakable FX Constructions". In: *CT-RSA 2020*. Ed. by Stanislaw Jarecki. Vol. 12006. LNCS. Springer, Heidelberg, Feb. 2020, pp. 1–14 (cit. on pp. 8, 22, 28, 239).

[Sin00] Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor Books, 2000. ISBN: 9780385495325 (cit. on pp. 32, 33).

[Sys17] Cisco Systems. *Cisco Visual Networking Index Predicts Global Annual IP Traffic to Exceed Three Zettabytes by 2021*. Tech. rep. June 2017 (cit. on p. 86).

[VN17] Baptiste Vinh Mau and Koji Nuida. "Correction of a Secure Comparison Protocol for Encrypted Integers in IEEE WIFS 2012 (Short Paper)". In: *IWSEC 17*. Ed. by Satoshi Obana and Koji Chida. Vol. 10418. LNCS. Springer, Heidelberg, Aug. 2017, pp. 181–191 (cit. on p. ii).

[vW99] Paul C. van Oorschot and Michael J. Wiener. “Parallel Collision Search with Cryptanalytic Applications”. In: *Journal of Cryptology* 12.1 (Jan. 1999), pp. 1–28 (cit. on p. 89).

[Wag02] David Wagner. “A Generalized Birthday Problem”. In: *CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. LNCS. Springer, Heidelberg, Aug. 2002, pp. 288–303 (cit. on pp. 93–96, 137, 236).

[WC81] Mark N. Wegman and Larry Carter. “New Hash Functions and Their Use in Authentication and Set Equality”. In: *Journal of Computer and System Sciences* 22 (1981), pp. 265–279 (cit. on pp. 61, 133).

[Win83] Robert S. Winternitz. “Producing a One-Way Hash Function from DES”. In: *CRYPTO’83*. Ed. by David Chaum. Plenum Press, New York, USA, 1983, pp. 203–207 (cit. on p. 207).

[Win84] Robert S. Winternitz. “A Secure One-Way Hash Function Built from DES”. In: *1984 IEEE Symposium on Security and Privacy* (1984), pp. 88–88 (cit. on pp. 207, 259).

[WN95] David J. Wheeler and Roger M. Needham. “TEA, a Tiny Encryption Algorithm”. In: *FSE’94*. Ed. by Bart Preneel. Vol. 1008. LNCS. Springer, Heidelberg, Dec. 1995, pp. 363–366 (cit. on p. 109).

[Yas10] Kan Yasuda. “The Sum of CBC MACs Is a Secure PRF”. In: *CT-RSA 2010*. Ed. by Josef Pieprzyk. Vol. 5985. LNCS. Springer, Heidelberg, Mar. 2010, pp. 366–381 (cit. on pp. 12, 76, 132, 133, 138, 257).

[Yas11] Kan Yasuda. “A New Variant of PMAC: Beyond the Birthday Bound”. In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 596–609 (cit. on pp. 12, 76, 132, 134, 145).

[Zha+12] Liting Zhang, Wenling Wu, Han Sui, and Peng Wang. “3kf9: Enhancing 3GPP-MAC beyond the Birthday Bound”. In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 296–312 (cit. on pp. 12, 132, 134, 149).

[Zoo20] Inc. Zoom Video Communications. *White Paper: Zoom Encryption*. Tech. rep. Apr. 2020 (cit. on p. 49).