Stream ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression

Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, Renaud Sirdey

Stream ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression*

Anne Canteaut[1], Sergiu Carpov[2], Caroline Fontaine[3], Tancrède Lepoint[4], María Naya-Plasencia[1], Pascal Paillier[4], and Renaud Sirdey[2]

[1] Inria, France, {anne.canteaut,maria.naya_plasencia}@inria.fr
[2] CEA LIST, France, {sergiu.carpov,renaud.sirdey}@cea.fr
[3] CNRS/Lab-STICC and Telecom Bretagne and UEB, France, caroline.fontaine@telecom-bretagne.eu
[4] CryptoExperts, France, {tancrede.lepoint,pascal.paillier}@cryptoexperts.com

Abstract In typical applications of homomorphic encryption, the first step consists for Alice to encrypt some plaintext m under Bob's public key pk and to send the ciphertext $c = \mathsf{HE}_{\mathsf{pk}}(m)$ to some third-party evaluator Charlie. This paper specifically considers that first step, *i.e.* the problem of transmitting c as efficiently as possible from Alice to Charlie. As previously noted, a form of compression is achieved using hybrid encryption. Given a symmetric encryption scheme E, Alice picks a random key k and sends a much smaller ciphertext $c' = (\mathsf{HE}_{\mathsf{pk}}(k), \mathsf{E}_k(m))$ that Charlie decompresses homomorphically into the original c using a decryption circuit $\mathcal{C}_{\mathsf{E}^{-1}}$.
In this paper, we revisit that paradigm in light of its concrete implementation constraints; in particular E is chosen to be an additive IV-based stream cipher. We investigate the performances offered in this context by Trivium, which belongs to the eSTREAM portfolio, and we also propose a variant with 128-bit security: Kreyvium. We show that Trivium, whose security has been firmly established for over a decade, and the new variant Kreyvium have an excellent performance.

Keywords. Stream Ciphers, Homomorphic cryptography, Trivium

1 Introduction

Since the breakthrough result of Gentry [30] achieving fully homomorphic encryption (FHE), many works have been published on simpler and more efficient schemes based on homomorphic encryption. Because they allow arbitrary computations on encrypted data, FHE schemes suddenly opened the way to exciting new applications, in particular cloud-based services in several areas (see *e.g.* [46,33,42]).

* This work has received a French governmental support granted to the COMIN Labs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference ANR-10-LABX-07-01, has been supported in part by the European Union's H2020 Programme under grant agreement number ICT-644209 and by the French's FUI project CRYPTOCOMP.

Compressed encryption. In these cloud applications, it is often assumed that some data is sent encrypted under a homomorphic encryption (HE) scheme to the cloud to be processed in a way or another. It is thus typical to consider, in the first step of these applications, that a user (Alice) encrypts some data m under some other user's public key pk (Bob) and sends some homomorphic ciphertext $c = \mathsf{HE}_{\mathsf{pk}}(m)$ to a third-party evaluator in the Cloud (Charlie). The roles of Alice and Bob are clearly distinct, even though they might be played by the same entity in some applications.

However, all HE schemes proposed so far suffer from a very large ciphertext expansion; the transmission of c between Alice and Charlie is therefore a very significant bottleneck in practice. The problem of reducing the size of c as efficiently as possible has first been considered in [46] wherein m is encrypted with a symmetric encryption scheme E under some key k randomly chosen by Alice, who then sends a much smaller ciphertext $c' = (\mathsf{HE}_{\mathsf{pk}}(k), \mathsf{E}_k(m))$ to Charlie. Given c', Charlie then exploits the homomorphic property of HE and recovers

$$c = \mathsf{HE}_{\mathsf{pk}}(m) = \mathcal{C}_{\mathsf{E}^{-1}}\left(\mathsf{HE}_{\mathsf{pk}}(k), \mathsf{E}_k(m)\right)$$

by homomorphically evaluating the decryption circuit $\mathcal{C}_{\mathsf{E}^{-1}}$. This can be assimilated to a *compression method* for homomorphic ciphertexts, c' being the result of applying a *compressed encryption scheme* to the plaintext m and c being recovered from c' using a *ciphertext decompression procedure*. In that approach obviously, the new encryption rate $|c'|/|m|$ becomes asymptotically close to 1 for long messages, which leaves no significant margin for improvement. However, the paradigm of ciphertext compression leaves totally open the question of how to choose E in a way that minimizes the decompression overhead, while preserving the same security level as originally intended.

Prior art. The cost of a homomorphic evaluation of several symmetric primitives has been investigated, including optimized implementations of AES [31,18,23], and of the lightweight block ciphers Simon [43] and Prince [24]. Usually lightweight block ciphers seem natural candidates for efficient evaluations in the encrypted domain. However, they may also lead to *much worse* performances than a homomorphic evaluation of, say, AES. Indeed, contemporary HE schemes use *noisy* ciphertexts, where a fresh ciphertext includes a noise component which grows along with homomorphic operations. Usually a homomorphic multiplication increases the noise by much larger proportions than a homomorphic addition. The maximum allowable level of noise (determined by the system parameters) then depends mostly on the multiplicative depth of the circuit. Many lightweight block ciphers balance out their simplicity by a large number of rounds, *e.g.* KATAN and KTANTAN [12], with the effect of considerably increasing their multiplicative depth. This type of design is therefore prohibitive in a HE context. Still Prince appears to be a much more suitable block cipher for homomorphic evaluation than AES (and than Simon), because it specifically targets applications that require a low latency; it is designed to minimize the cost of an unrolled implementation [10] rather than to optimize *e.g.* silicon area.

At Eurocrypt 2015, Albrecht, Rechberger, Schneider, Tiessen and Zohner observed that the usual criteria that rule the design of lightweight block ciphers are not appropriate when designing a symmetric encryption scheme with a low-cost homomorphic evaluation [3]. Indeed, both the number of rounds and the number of binary multiplications required to evaluate an Sbox have to be taken into account. Minimizing the number of rounds is a crucial issue for low-latency ciphers like PRINCE, while minimizing the number of multiplications is a requirement for efficient masked implementations.

These two criteria have been considered together for the first time by Albrecht *et al.* in the recent design of a family of block ciphers called LOWMC [3] with very small multiplicative size and depth[1]. However, the proposed instances of LOWMC, namely LOWMC-80 and LOWMC-128, have recently had some security issues [21]. They actually present some weaknesses inherent in their low multiplicative complexity. Indeed, the algebraic normal forms (*i.e.*, the multivariate polynomials) describing the encryption and decryption functions are sparse and have a low degree. This type of features is usually exploited in algebraic attacks, cube attacks and their variants, *e.g.* [20,22,5]. While these attacks are rather general, the improved variant used for breaking LOWMC [21], named interpolation attack [38], specifically applies to block ciphers. Indeed it exploits the sparse algebraic normal form of some intermediate bit within the cipher using that this bit can be evaluated both from the plaintext in the forward direction and from the ciphertext in the backward direction. This technique leads to several attacks including a key-recovery attack against LOWMC-128 with time complexity 2^{118} and data complexity 2^{73}, implying that the cipher does not provide the expected 128-bit security level.

Our contributions. We emphasize that beyond the task of designing a HE-friendly block cipher, revisiting the whole compressed encryption scheme (in particular its internal mode of operation) is what is really needed in order to take these concrete HE-related implementation constraints into account.

First, we identify that homomorphic decompression is subject to an *offline phase* and an *online phase*. The offline phase is plaintext-independent and therefore can be performed in advance, whereas the online phase completes decompression upon reception of the plaintext-dependent part of the compressed ciphertext. Making the online phase as quick as technically doable leads us to choose *an additive IV-based stream cipher to implement* E. However, we note that the use of a lightweight block cipher as the building-block of that stream cipher usually provides a security level limited to $2^{n/2}$ where n is the block size [48], thus limiting the number of encrypted blocks to (typically) less than 2^{32} (*i.e.* 32GB for 64-bit blocks).

As a result, we propose our own candidate for E: the keystream generator Trivium [14], which belongs to the eSTREAM portfolio of recommended

[1] It is worth noting that in a HE context, reducing the multiplicative size of a symmetric primitive might not be the first concern (while it is critical in a multiparty computation context, which also motivated the work of Albrecht *et al.* [3]), whereas minimizing the multiplicative depth is of prime importance.

stream ciphers, and a new proposal called *Kreyvium*, which shares the same internal structure but allows for bigger keys of 128 bits[2]. The main advantage of Kreyvium over Trivium is that it provides 128-bit security (instead of 80-bit) with the same multiplicative depth, and inherits the same security arguments. It is worth noticing that the design of a variant of Trivium which guarantees a 128-bit security level has been raised as an open problem for the last ten years [1, p. 30]. Beside a higher security level, it also accommodates longer IVs, so that it can encrypt up to $46 \cdot 2^{128}$ plaintext bits under the same key, with multiplicative depth only 12. Moreover, both Trivium and Kreyvium are resistant against the interpolation attacks used for breaking LowMC since these ciphers do not rely on a permutation which would enable the attacker to compute backwards.

We implemented our construction and instantiated it with Trivium, Kreyvium and LowMC in CTR-mode. Our results show that the promising performances attained by the HE-dedicated block cipher LowMC can be achieved with well-known primitives whose security has been firmly established for over a decade.

Organization of the paper. We introduce a general model and a generic construction to compress homomorphic ciphertexts in Section 2. Our construction using Trivium and Kreyvium is described in Section 3. Subsequent experimental results are presented in Section 4.[3]

2 A Generic Design for Efficient Decompression

In this section, we describe our model and generic construction to transmit compressed homomorphic ciphertexts between Alice and Charlie. We use the same notation as in the introduction: Alice wants to send some plaintext m, encrypted under Bob's public key pk (of an homomorphic encryption scheme HE) to a third party evaluator Charlie.

2.1 Offline/Online Phases in Ciphertext Decompression

Most practical scenarios would likely find it important to distinguish between three distinct phases within the homomorphic evaluation of $\mathcal{C}_{\mathsf{E}^{-1}}$:

1. an *offline key-setup* phase which only depends on Bob's public key and can be performed once and for all before Charlie starts receiving compressed ciphertexts encrypted under Bob's key;

[2] Independently from our results, another variant of Trivium named Trivi-A has been proposed [17]. It handles larger keys but uses longer registers. It then needs more rounds for mixing the internal state, which means that it is much less adapted to our setting than Kreyvium.

[3] In [15], we also present a second candidate for E that relies on a completely different technique based on the observation that multiplication in binary fields is $\mathbb{F}_2$-bilinear, making it possible to homomorphically exponentiate field elements with a log-log-depth circuit. We also report a random oracle based proof that compressed ciphertexts are semantically secure under an appropriate complexity assumption. We show, however, that this second approach remains disappointingly impractical.

2. an *offline decompression* phase which can be performed only based on some plaintext-independent material found in the compressed ciphertext;
3. an *online decompression* phase which aggregates the result of the offline phase with the plaintext-dependent part of the compressed ciphertext and (possibly very quickly) recovers the decompressed ciphertext c.

As such, our general-purpose formulation $c' = (\mathsf{HE}_{\mathsf{pk}}(k), \mathsf{E}_k(m))$ does not allow to make a clear distinction between these three phases. In our context, it is much more relevant to reformulate the encryption scheme as an IV-based encryption scheme where the encryption and decryption process are both deterministic but depend on an IV:

$$\mathsf{E}_k(m) \stackrel{\text{def}}{=} \left(IV, \mathsf{E}'_{k,IV}(m)\right) \ .$$

Since the IV has a limited length, it can be either transmitted during an offline preprocessing phase, or may alternately correspond to a state which is maintained by the server. Now, to minimize the latency of homomorphic decompression for Charlie, the online phase should be reduced to a minimum. The most appropriate choice in this respect consists in using an additive IV-based stream cipher Z so that

$$\mathsf{E}'_{k,IV}(m) = Z(k, IV) \oplus m \ .$$

In this reformulation, the decompression process is clearly divided into a offline precomputation stage which only depends on pk, k and IV, and an online phase which is plaintext-dependent. The online phase is thus reduced to a mere XOR between the plaintext-dependent part of the ciphertext $\mathsf{E}'_{k,IV}(m)$ and the HE-encrypted keystream $\mathsf{HE}(Z(k, IV))$, which comes essentially for free in terms of noise growth in HE ciphertexts. All expensive operations (*i.e.* homomorphic multiplications) are performed during the offline decompression phase where $\mathsf{HE}(Z(k, IV))$ is computed from $\mathsf{HE}(k)$ and IV.

2.2 Our Generic Construction

We devise the generic construction depicted on Fig. 1. It is based on a homomorphic encryption scheme HE with plaintext space $\{0,1\}$, an expansion function G mapping ℓ_{IV}-bit strings to strings of arbitrary size, and a fixed-size parametrized function F with input size ℓ_x, parameter size ℓ_k and output size N.

Compressed encryption. Given an ℓ_m-bit plaintext m, Bob's public key pk and $IV \in \{0,1\}^{\ell_{IV}}$, the compressed ciphertext c' is computed as follows:

1. Set $t = \lceil \ell_m / N \rceil$,
2. Set $(x_1, \ldots, x_t) = G(IV; t\ell_x)$,
3. Randomly pick $k \leftarrow \{0,1\}^{\ell_k}$,
4. For $1 \leq i \leq t$, compute $z_i = F_k(x_i)$,
5. Set $\mathsf{keystream}$ to the ℓ_m leftmost bits of $z_1 \,||\, \ldots \,||\, z_t$,
6. Output $c' = (\mathsf{HE}_{\mathsf{pk}}(k), m \oplus \mathsf{keystream})$.

Ciphertext decompression. Given c' as above, Bob's public key pk and $IV \in \{0,1\}^{\ell_{IV}}$, the ciphertext decompression is performed as follows:

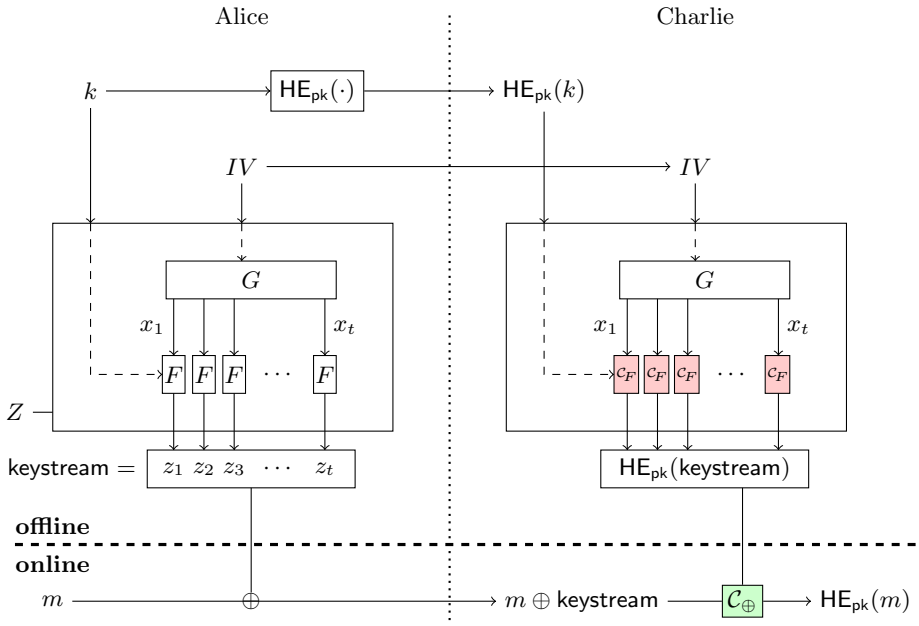


Figure1. Our generic construction. The multiplicative depth of the circuit is equal to the depth of $\mathcal{C}_F$. This will be the bottleneck in our protocol and we want the multiplicative depth of F to be as small as possible. With current HE schemes, the circuit $\mathcal{C}_\oplus$ is usually very fast (addition of ciphertexts) and has a negligible impact on the noise in the ciphertext.

1. Set $t = \lceil \ell_m / N \rceil$,
2. Set $(x_1, \ldots, x_t) = G(IV; t\ell_x)$,
3. For $1 \leq i \leq t$, compute $\mathsf{HE}_{\mathsf{pk}}(z_i) = \mathcal{C}_F\left(\mathsf{HE}_{\mathsf{pk}}(k), x_i\right)$ with some circuit $\mathcal{C}_F$,
4. Deduce $\mathsf{HE}_{\mathsf{pk}}(\mathsf{keystream})$ from $\mathsf{HE}_{\mathsf{pk}}(z_1), \ldots, \mathsf{HE}_{\mathsf{pk}}(z_t)$,
5. Compute $c = \mathsf{HE}_{\mathsf{pk}}(m) = \mathcal{C}_\oplus\left(\mathsf{HE}_{\mathsf{pk}}(\mathsf{keystream}), m \oplus \mathsf{keystream}\right)$.

The circuit $\mathcal{C}_\oplus$ computes $\mathsf{HE}(a \oplus b)$ given $\mathsf{HE}(a)$ and b where a and b are bit-strings of the same size. In our construction, the cost of decompression per plaintext block is *fixed* and roughly equals one single evaluation of the circuit $\mathcal{C}_F$; most importantly, the multiplicative depth of the decompression circuit is also fixed, and set to the depth of $\mathcal{C}_F$.

How secure are compressed ciphertexts? From a high-level perspective, compressed homomorphic encryption is just hybrid encryption and relates to the generic KEM-DEM construct. However it just cannot inherit from the general security results attached to the KEM-DEM framework [2,35] since taking some HE scheme to implement the KEM part does not even fulfill the basic requirements that the KEM be IND-CCA or even IND-CCCA. Thus, the right level of security requirement on the compressed encryption scheme itself seems to be

just IND-CPA for concrete use. However, it is not known what minimal security assumptions to require from a homomorphic KEM and a general-purpose DEM to yield a KEM-DEM scheme that is provably IND-CPA. As a result of that, evidence that CPA security is reached may only be provided on a case-by-case basis given a specific embodiment (see [15] for details).

Instantiating the paradigm. The rest of the paper focuses on how to choose the expansion function G and function F so that the homomorphic evaluation of $\mathcal{C}_F$ is as fast (and its multiplicative depth as low) as possible. In our approach, the value of IV is assumed to be shared between Alice and Charlie and needs not be transmitted along with the compressed ciphertext. For instance, IV is chosen to be an absolute constant such as $IV = 0^\ell$ where $\ell = \ell_{IV} = \ell_x$. Another example is to take for $IV \in \{0,1\}^\ell$ a synchronized state that is updated between transmissions. Also, the expansion function G is chosen to implement a counter in the sense of the NIST description of the CTR mode [47], for instance

$$G(IV; t\ell) = (IV, IV \boxplus 1, \ldots, IV \boxplus (t-1)) \quad \text{where} \quad a \boxplus b = (a+b) \bmod 2^\ell \ .$$

Finally, F is chosen to ensure both an appropriate security level and a low multiplicative depth. We focus in Section 3 on the keystream generator corresponding to Trivium, and on a new variant, called *Kreyvium*.

Interestingly, the output of an iterated PRF used in CTR mode is computationally indistinguishable from random [7, Th. 13]. Hence, under the assumption that Trivium or Kreyvium is a PRF[4], the keystream $z_1 \,||\, \ldots \,||\, z_t$ produced by our construction is also indistinguishable. However, this is insufficient to prove that the compressed encryption scheme is IND-CPA, because the adversary also sees $\mathsf{HE}_{\mathsf{pk}}(k)$ during the IND-CPA game, which cannot be proven not to make the keystream distinguishable. Although the security of this approach is empiric, Section 3 provides a strong rationale for the Kreyvium design and makes it the solution with the smallest homomorphic evaluation latency known so far.

Why not use a block cipher for F? Although not specifically in these terms, the use of lightweight block ciphers like Prince and Simon has been proposed in the context of compressed homomorphic ciphertexts *e.g.* [43,24]. However a complete encryption scheme based on the ciphers has not been defined. This is a major issue since the security provided by all classical modes of operation (including all variants of CBC, CTR, CFB, OFB, OCB...) is inherently limited to $2^{n/2}$ where n is the block size [48] (see also *e.g.* [39, p. 95]). Only a very few modes providing *beyond-birthday* security have been proposed (*e.g.* [37,50] but they induce a higher implementation cost and their security is usually upper-bounded by $2^{2n/3}$.

In other words, the use of a block cipher operating on 64-bit blocks like Prince or Simon-32/64 implies that the number of blocks encrypted under the same key should be significantly less that 2^{32} (*i.e.* 32GB for 64-bit blocks).

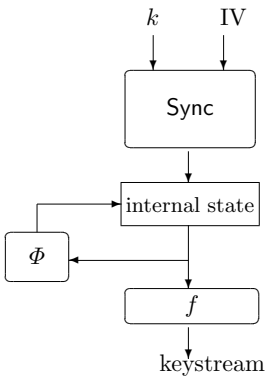
[4] Note that this equivalent to say that Kreyvium instantiated with a random key and mapping the IV's to the keystream is secure [8, Sec. 3.2].

Therefore, only block ciphers with a large enough block size, like the LowMC instantiation with a 256-bit block proposed in [3], are suitable in applications which may require the encryption of more than 2^{32} bits under the same key.

3 Trivium and Kreyvium, Two Low-Depth Stream Ciphers

Since an additive stream cipher is the optimal choice, we now focus on keystream generation, and on its homomorphic evaluation. An IV-based keystream generator is decomposed into:

- a resynchronization function, Sync, which takes as input the IV and the key (possibly expanded by some precomputation phase), and outputs some n-bit initial state;
- a transition function Φ which computes the next state of the generator;
- a filtering function f which computes a keystream segment from the internal state.



Since generating N keystream bits may require a circuit of depth up to

$$(\mathrm{depth}(\mathsf{Sync}) + N\,\mathrm{depth}(\Phi) + \mathrm{depth}(f))\,,$$

the best design strategy for minimizing this value consists in choosing a transition function with a small depth. The extreme option is to choose for Φ a linear function as in the CTR mode where the counter is implemented by an LFSR. An alternative strategy consists in choosing a nonlinear transition whose depth does not increase too fast when it is iterated. The influence of Sync on the multiplicative depth of the circuit is further investigated in [15].

Size of the internal state. A major specificity of our context is that a large internal state can be easily handled. Indeed, in most classical stream ciphers, the internal-state size usually appears as a bottleneck because the overall size of the quantities to be stored highly influences the number of gates in the implementation. This is not the case in our context. It might seem, a priori, that increasing the size of the internal state automatically increases the number of nonlinear operations (because the number of inputs of Φ increases). But, this is not the case if a part of this larger internal state is used, for instance, for storing the secret key. This strategy can be used for increasing the security at no implementation cost. Indeed, the complexity of all generic attacks aiming at recovering the internal state of the generator is $\mathcal{O}(2^{n/2})$ where n is the size of the secret part of the internal state even if some part is not updated during the keystream generation. For instance, the time-memory-data-tradeoff attacks in [6,32,9] aim at inverting the function which maps the internal state of the generator to the first keystream

bits. But precomputing some values of this function must be feasible by the attacker, which is not the case if the filtering or transition function depends on some secret material. On the other hand, the size n' of the non-constant secret part of the internal state determines the data complexity for finding a collision on the internal state: the length of the keystream produced from the same key is limited to $2^{n'/2}$. But, if the transition function or the filtering function depends on the IV, this limitation corresponds to the maximal keystream length produced from the same key/IV pair. It is worth noticing that many attacks require a very long keystream generated from the same key/IV pair and do not apply in our context since the keystream length is strictly limited by the multiplicative depth of the circuit.

3.1 Trivium in the HE setting

Trivium [14] is one of the 7 stream ciphers recommended by the eSTREAM project [25]. Due to the small number of nonlinear operations in its transition function, it appears as a natural candidate in our context.

Description. Trivium is a synchronous stream cipher with a key and an IV of 80 bits each. Its internal state is composed of 3 registers of sizes 93, 84 and 111 bits, corresponding to a size of 288 bits in total. We use the notation introduced by the designers: the leftmost bit of the 93-bit register is s_1, and its rightmost one is s_{93}; the leftmost bit of the register of size 84 is s_{94} and the rightmost s_{177}; the leftmost bit of register of size 111 is s_{178} and the rightmost s_{288}. The initialization and the generation of an N-bit keystream are described below.

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_0, \ldots, K_{79}, 0, \ldots, 0)$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_0, \ldots, IV_{79}, 0, \ldots, 0)$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
for $i = 1$ to $1152 + N$ **do**
  $t_1 \leftarrow s_{66} + s_{93}$
  $t_2 \leftarrow s_{162} + s_{177}$
  $t_3 \leftarrow s_{243} + s_{288}$
    if $i > 1152$ **do**
      output $z_{i-1152} \leftarrow t_1 + t_2 + t_3$
    end if
  $t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
  $t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
  $t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

No attack better than an exhaustive key search is known so far on the full Trivium. It can then be considered as secure. The family of attacks that seems

to provide the best result on round-reduced versions is the cube attack and its variants [22,5,28]. They recover some key bits (resp. provide a distinguisher on the keystream) if the number of initialization rounds is reduced to 799 (resp. 885) rounds out of 1152. The highest number of initialization rounds that can be attacked is 961: in this case, a distinguisher exists for a class of weak keys [41].

Multiplicative depth. It is easy to see that the multiplicative depth grows quite slowly with the number of iterations. An important observation is that, in the internal state, only the first 80 bits in Register 1 (the keybits) are initially encrypted under the HE and that, as a consequence, performing hybrid clear and encrypted data calculations is possible (this is done by means of the following simple rules: $0 \cdot [x] = 0$, $1 \cdot [x] = [x]$, $0 + [x] = [x]$ and $1 + [x] = [1] + [x]$, where the square brackets denote encrypted bits and where in all but the latter case, a homomorphic operation is avoided which is specially desirable for multiplications). This optimization allows for instance to increase the number of bits which can be generated (after the 1152 blank rounds) at depth 12 from 42 to 57 (*i.e.*, a 35% increase). Then, the relevant quantity in our context is the multiplicative depth of the circuit which computes N keystream bits from the 80-bit key. The proof of the following proposition is given in [15].

Proposition 1. *In Trivium, the keystream length $N(d)$ which can be produced from the 80-bit key with a circuit of multiplicative depth d, $d \geq 4$, is given by*

$$N(d) = 282 \times \left\lfloor \frac{d}{3} \right\rfloor + \begin{cases} 81 & \text{if } d \equiv 0 \bmod 3 \\ 160 & \text{if } d \equiv 1 \bmod 3 \\ 269 & \text{if } d \equiv 2 \bmod 3 \end{cases}.$$

3.2 Kreyvium

Our first aim is to offer a variant of Trivium with 128-bit key and IV, without increasing the multiplicative depth of the corresponding circuit. Besides a higher security level, another advantage of this variant is that the number of possible IVs, and then the maximal length of data which can be encrypted under the same key, increases from $2^{80} N_{\mathsf{trivium}}(d)$ to $2^{128} N_{\mathsf{kreyvium}}(d)$. Increasing the key and IV-size in Trivium is a challenging task, mentioned as an open problem in [1, p. 30] for instance. In particular, Maximov and Biryukov [45] pointed out that increasing the key-size in Trivium without any additional modification cannot be secure due to some attack with complexity less than 2^{128}. A first attempt in this direction has been made in [45] but the resulting cipher accommodates 80-bit IV only, and its multiplicative complexity is higher than in Trivium since the number of AND gates is multiplied by 2.

Description. Our proposal, Kreyvium, accommodates a key and an IV of 128 bits each. The only difference with the original Trivium is that we have added to the 288-bit internal state a 256-bit part corresponding to the secret key and the IV. This part of the state aims at making both the filtering and transition functions key- and IV-dependent. More precisely, these two functions f and Φ

depend on the key bits and IV bits, through the successive outputs of two shift-registers K^* and IV^* initialized by the key and by the IV respectively. The internal state is then composed of five registers of sizes 93, 84, 111, 128 and 128 bits, having an internal state size of 544 bits in total, among which 416 become unknown to the attacker after initialization.

We will use the same notation as the description of Trivium, and for the additional registers we use the usual shift-register notation: the leftmost bit is denoted by K^*_{127} (or IV^*_{127}), and the rightmost bit (*i.e.*, the output) is denoted by K^*_0 (or IV^*_0). Each one of these two registers are rotated independently from the rest of the cipher. The generator is described below, and depicted on Fig. 2.

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_0, \ldots, K_{92})$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_0, \ldots, IV_{83})$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (IV_{84}, \ldots, IV_{127}, 1, \ldots, 1, 0)$
$(K^*_{127}, K^*_{126}, \ldots, K^*_0) \leftarrow (K_0, \ldots, K_{127})$
$(IV^*_{127}, IV^*_{126}, \ldots, IV^*_0) \leftarrow (IV_0, \ldots, IV_{127})$
for $i = 1$ to $1152 + N$ **do**
  $t_1 \leftarrow s_{66} + s_{93}$
  $t_2 \leftarrow s_{162} + s_{177}$
  $t_3 \leftarrow s_{243} + s_{288} + \boldsymbol{K^*_0}$
    if $i > 1152$ **do**
      output $z_{i-1152} \leftarrow t_1 + t_2 + t_3$
    end if
  $t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171} + \boldsymbol{IV^*_0}$
  $t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
  $t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
  $t_4 \leftarrow K^*_0$
  $t_5 \leftarrow IV^*_0$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
  $(K^*_{127}, K^*_{126}, \ldots, K^*_0) \leftarrow (t_4, K^*_{127}, \ldots, K^*_1)$
  $(IV^*_{127}, IV^*_{126}, \ldots, IV^*_0) \leftarrow (t_5, IV^*_{127}, \ldots, IV^*_1)$
end for

Related ciphers. KATAN [12] is a lightweight block cipher with a lot in common with Trivium. It is composed of two registers, whose feedback functions are very sparse, and have a single nonlinear term. The key, instead of being used for initializing the state, is introduced by XORing two key information-bits per round to each feedback bit. The recently proposed stream cipher Sprout [4], inspired by Grain but with much smaller registers, also inserts the key in a similar way: instead of using the key for initializing the state, one key information-bit is XORed at each clock to the feedback function. We can see the parallelism between these two ciphers and our newly proposed variant. In particular, the

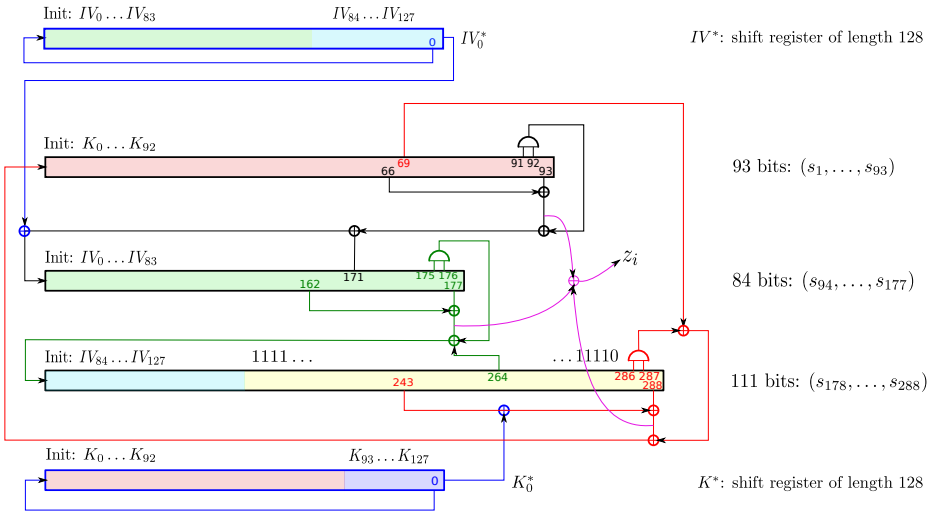


Figure2. Kreyvium. The three registers in the middle correspond to Trivium. The modifications defining Kreyvium correspond to the two registers in blue.

previous security analysis on KATAN shows that this type of design does not introduce any clear weakness. Indeed, the best attacks on round-reduced versions of KATAN so far [29] are meet-in-the-middle attacks, that exploit the knowledge of the values of the first and the last internal states (due to the block-cipher setting). As this is not the case here, such attacks, as well as the recent interpolation attacks against LowMC [21], do not apply. The best attacks against KATAN, when excluding MitM techniques, are conditional differential attacks [40,41].

Design rationale. We have decided to XOR the keybit K_0^* to the feedback function of the register that interacts with the content of $(s_1, \dots, s_{63})$ the later, since $(s_1, \dots, s_{63})$ is initialized with some key bits. The same goes for the IV^* register. Moreover, as the keybits that start entering the state are the ones that were not in the initial state, all the keybits affect the state at the earliest.

We also decided to initialize the state with some keybits and with all the IV bits, and not with a constant value, as this way the mixing will be performed quicker. Then we can expect that the internal-state bits after initialization are expressed as more complex and less sparse functions in the key and IV bits.

Our change of constant is motivated by the conditional differential attacks from [41]: the conditions needed for a successful attack are that 106 bits from the IV or the key are equal to '0' and a single one needs to be '1'. This suggests that values set to zero "encourage" non-random behaviors, leading to our new constant. In other words, in Trivium, an all-zero internal state is always updated in an all-zero state, while an all-one state will change through time. The 0 at the end of the constant is added for preventing slide attacks.

Multiplicative depth. Exactly as for Trivium, we can compute the number of keystream bits which can be generated from the key at a given depth (see [15]).

Proposition 2. *In Kreyvium, the keystream length $N(d)$ which can be produced from the* 128*-bit key with a circuit of multiplicative depth d, $d \geq 4$, is given by*

$$N(d) = 282 \times \left\lfloor \frac{d}{3} \right\rfloor + \begin{cases} 70 & \text{if } d \equiv 0 \bmod 3 \\ 149 & \text{if } d \equiv 1 \bmod 3 \\ 258 & \text{if } d \equiv 2 \bmod 3 \end{cases}.$$

Security analysis. We investigate how all the known attacks on Trivium can apply to Kreyvium. A more detailled analysis is provided in [15].

TMDTO. TMDTO attacks aiming at recovering the initial state of the cipher do not apply since the size of the secret part of the internal state (416 bits) is much larger than twice the key-size: the size of the whole secret internal state has to be taken into account, even if the additional 128-bit part corresponding to K^* is independent from the rest of the state. On the other hand, TMDTO aiming at recovering the key have complexity larger than exhaustive key search since the key and the IV have the same size [36,13].

Internal-state collision. A distinguisher may be built if the attacker is able to find two colliding internal states, since the two keystreams produced from colliding states are identical. Finding such a collision requires around 2^{144} keystream bits generated from the same key/IV pair, which is much longer than the maximal keystream length allowed by the multiplicative depth of the circuit. We also show in [15] that, for a given key, finding two internal states colliding on all bits except on IV^* does not provide any valid distinguisher. The birthday-bound of 2^{144} bits then provides a limit on the number of bits produced from the same key/IV pair, not on the bits produced from the same key.

Cube attacks [22,28] and cube testers [5]. They provide the best attacks for round-reduced Trivium. In our case, as we keep the same main function, but we have two additional XORs per round, thus a better mixing of the variables, we can expect the relations to get more involved and hamper the application of previously defined round-reduced distinguishers. One might wonder if the fact that more variables are involved could ease the attacker's task, but we point out here that the limitation in the previous attacks was not the IV size, but the size of the cubes themselves. Therefore, having more variables available is of no help with respect to this point. We can conclude that the resistance of Kreyvium to these types of attacks is at least the resistance of Trivium, and even better.

Conditional differential cryptanalysis. Because of its applicability to Trivium and KATAN, the attack from [41] is definitely of interest in our case. In particular, the highest number of blank rounds is reached if some conditions on two registers are satisfied at the same time (and not only conditions on the register controlled by the IV bits in the original Trivium). In our case, as we have

IV bits in two registers, it is important to elucidate whether an attacker can take advantage of introducing differences in two registers simultaneously. First, let us recall that we have changed the constant to one containing mostly 1. We previously saw that the conditions that favor the attacks are values set to zero in the initial state. In Trivium, we have $(108 + 4 + 13) = 125$ bits already fixed to zero in the initial state, 3 are fixed to one and the others can be controlled by the attacker in the weak-key setting (and the attacker will force them to be zero most of the time). Now, instead, we have 64 bits forced to be 1, 1 equal to zero, and $(128 + 93) = 221$ bits of the initial state controlled by the attacker in the weak-key setting, plus potentially 21 additional bits from the key still not used, that will be inserted during the first rounds. We can conclude that, while in Trivium it is possible in the weak-key setting, to introduce zeros in the whole initial state but in 3 bits, in Kreyvium, we will never be able to set to zero 64 bits, implying that applying the techniques from [41] becomes much harder.

Algebraic attacks. Several algebraic attacks have been proposed against Trivium, aiming at recovering the 288-bit internal state at the beginning of the keystream generation (i.e., at time $t = 1153$) from the knowledge of the keystream bits. The most efficient attack of this type is due to Maximov and Biryukov [45]. It exploits the fact that the 22 keystream bits at time $3t'$, $0 \leq t' < 22$, are determined by all bits of the initial state at indexes divisible by 3 (starting from the leftmost bit in each register). Moreover, once all bits at positions $3i$ are known, then guessing that the outputs of the three AND gates at time $3t'$ are zero provides 3 linear relations between the bits of the internal state and the keystream bits. The attack then consists of an exhaustive search for some bits at indexes divisible by 3. The other bits in such positions are then deduced by solving the linear system derived from the keystream bits at positions $3t'$. Once all these bits have been determined, the other 192 bits of the initial state are deduced from the other keystream equations. This process must be iterated until the guess for the outputs of the AND gates is correct. In the case of Trivium, the outputs of at least 125 AND gates must be guessed in order to get 192 linear relations involving the 192 bits at indexes $3i + 1$ and $3i + 2$. This implies that the attack has to be repeated $(4/3)^{125} = 2^{52}$ times. From these guesses, we get many linear relations involving the bits at positions $3i$ only, implying that only an exhaustive search with complexity 2^{32} for the other bits at positions $3i$ is needed. Therefore, the overall complexity of the attack is around $2^{32} \times 2^{52} = 2^{84}$. A similar algorithm can be applied to Kreyvium, but the main difference is that every linear equation corresponding to a keystream bit also involves one key bit. Moreover, the key bits involved in the generation of any 128 consecutive output bits are independent. It follows that each of the first 128 linear equations introduces a new unknown in the system to solve. For this reason, it is not possible to determine all bits at positions $3i$ by an exhaustive search on less than 96 bits like for Trivium. Moreover, the outputs of more than 135 AND gates must be guessed for obtaining enough equations on the remaining bits of the initial state. Therefore the overall complexity of the attack exceeds $2^{96} \times 2^{52} = 2^{148}$ and is much higher that the cost of the exhaustive key search. It is worth noticing

that the attack would have been more efficient if only the feedback bits, and not the keystream bits, would have been dependent on the key. In this case, 22 linear relations independent from the key would have been available to the attacker.

4 Experimental Results

We now discuss and compare the practicality of our generic construction when instantiated with Trivium, Kreyvium and LowMC. The expansion function G implements a mere counter, and the aforementioned algorithms are used to instantiate the function F that produces N bits of keystream per iteration as defined by Prop. 1 and 2.[5]

HE framework. In our experiments, we considered two HE schemes: the BGV scheme [11] and the FV scheme [26] (a scale-invariant version of BGV). The BGV scheme is implemented in the library HElib [34] and has become *de facto* a standard benchmarking library for HE applications. Similarly, the FV scheme was previously used in several HE benchmarkings [27,43,16], is conceptually simpler than the BGV scheme, and is one of the most efficient HE schemes.[6] Additionally, batching was used [49], *i.e.* the HE schemes were set up to encrypt vectors in an SIMD fashion (componentwise operations, and rotations via the Frobenius endomorphism). The number of elements that can be encrypted depends on the number of terms in the factorization modulo 2 of the cyclotomic polynomial used in the implementation. This batching allowed us to perform several Trivium/Kreyvium/LowMC in parallel in order to increase the throughput.

Parameter selection for subsequent homomorphic processing. In all the previous works on the homomorphic evaluation of symmetric encryption schemes, the parameters of the underlying HE scheme were selected for the exact multiplicative depth required and not beyond [31,19,43,24,3]. This means that once the ciphertext is decompressed, no further homomorphic computation can actually be performed by Charlie – this makes the claimed timings considerably less meaningful in a real-world context.

We benchmarked both parameters for the exact multiplicative depth and parameters able to handle circuits of the minimal multiplicative depth plus 7 to allow further homomorphic processing by Charlie (which is obviously what is expected in applications of homomorphic encryption). We chose 7 because, in practice, numerous applications use algorithms of multiplicative depth smaller than 7 (see *e.g.* [33,42]). In what follows we compare the results we obtain using Trivium, Kreyvium and also the LowMC cipher. For LowMC, we benchmarked not only our own implementation but also the LowMC implementation

[5] Note that these propositions only hold when hybrid clear and encrypted data calculations are possible between IV and HE ciphertexts. This explains the slight differences in the number of keystream bits per iteration (column "N") between Tab. 1 and 2.

[6] We used the Armadillo compiler implementation of FV [16]. This source-to-source compiler turns a C++ algorithm into a Boolean circuit, optimizes it, and generates an OpenMP parallel code which can then be combined with a HE scheme.

Table1. Latency and throughput using HElib on a single core of a mid-end 48-core server (4 x AMD Opteron 6172 processors with 64GB of RAM).

Algorithm	security level κ	N	used × depth	#slots	latency sec.	throughput bits/min
Trivium-12	80	45	12	600	1417.4	1143.0
			19	720	4420.3	439.8
Trivium-13	80	136	13	600	3650.3	1341.3
			20	720	11379.7	516.3
Kreyvium-12	128	42	12	504	1715.0	740.5
			19	756	4956.0	384.4
Kreyvium-13	128	124	13	682	3987.2	1272.6
			20	480	12450.8	286.8
LowMC-128	$? \leq 118$	256	13	682	3608.4	2903.1
			20	480	10619.6	694.3
LowMC-128 [3]	$? \leq 118$	256	13	682	3368.8	3109.6
			20	480	9977.1	739.0

of [3] available at `https://bitbucket.org/malb/lowmc-helib`. Minor changes to this implementation were made in order to obtain an equivalent parametrization of HElib. The main difference is that the implementation from [3] uses an optimized method for multiplying a Boolean vector and a Boolean matrix, namely the "Method of Four Russians". This explains why our implementation is approximately 6% slower, as it performs 2–3 times more ciphertext additions.

Experimental results using HElib. For sake of comparison with [3], we ran our implementations and their implementation of LowMC on a single core using HElib. The results are provided in Tab. 1. We recall that the latency refers to the time required to perform the entire homomorphic evaluation whereas the throughput is the number of blocks processed per time unit.

Experimental results using FV. On Tab. 2, we present the benchmarks when using the FV scheme. The experiments were performed using either a single core (in order to compare with BGV) or on all the cores of the machine the tests were performed on. The execution time acceleration factor between 48-core parallel and sequential executions is given in the column "Speed gain". While good accelerations (at least 25 times) were obtained for Trivium and Kreyvium algorithms, the acceleration when using LowMC is significantly smaller (~ 10 times). This is due to the huge number of operations in LowMC that created memory contention and huge slowdown in memory allocation.

Table2. Latency of our construction when using the FV scheme on a mid-end 48-core server (4 x AMD Opteron 6172 processors with 64GB of RAM).

Algorithm	security level κ	N	used $\times$ depth	latency (sec.) 1 core	latency (sec.) 48 cores	Speed gain
Trivium-12	80	57	12	681.5	26.8	$\times$ 25.4
			19	2097.1	67.6	$\times$ 31.0
Trivium-13	80	136	13	888.2	33.9	$\times$ 26.2
			20	2395.0	77.2	$\times$ 31.0
Kreyvium-12	128	46	12	904.4	35.3	$\times$ 25.6
			19	2806.3	82.4	$\times$ 34.1
Kreyvium-13	128	125	13	1318.6	49.7	$\times$ 26.5
			20	3331.4	97.9	$\times$ 34.0
LowMC-128	$? \leq 118$	256	14	1531.1	171.0	$\times$ 9.0
			21	3347.8	329.0	$\times$ 10.2

Interpretation. First, we would like to recall that LowMC-128 must be considered in a different category because of the existence of a key-recovery attack with time complexity 2^{118} and data complexity 2^{73} [21]. However, it has been included in the table in order to show that the performances achieved by Trivium and Kreyvium are of the same order of magnitude. An increase in the number of rounds of LowMC-128 (typically by 4 rounds) is needed to achieve 128-bit security, but this would have a non-negligible impact on its homomorphic evaluation performance, as it would require to increase the depth of the cryptosystem supporting the execution. For instance, a back-of-the-envelope estimation for four additional rounds leads to a degradation of its homomorphic execution performances by a factor of about 2 to 3 (more computations with larger parameters). It is also worth noticing that the minimal multiplicative depth for which valid LowMC output ciphertexts were obtained was 14 for the FV scheme and 13 for the BGV scheme. The theoretical multiplicative depth is 12 but the high number of additions explains this difference[7].

Our results show that Trivium and Kreyvium have a smaller latency than LowMC, but have a slightly smaller throughput. As already emphasized in [43], real-world applications of homomorphic encryption (which are often cloud-based applications) should be implemented in a transparent and user-friendly way. In the context of our approach, the latency of the offline phase is still an important parameter aiming at an acceptable experience for the end-user even when a

[7] The multiplicative depth is only an *approximation* of the homomorphic depth required to absorb the noise generated by the execution of an algorithm [44]. It neglects the noise induced by additions and thus does not hold for too addition-intensive algorithms such as those in the LowMC family.

sufficient amount of homomorphic keystream could not be precomputed early enough because of overall system dimensioning issues.

Also Trivium and Kreyvium are more parallelizable than LowMC is. Therefore, our work shows that the promising performances obtained by the recently proposed *HE-dedicated cipher* LowMC can also be achieved with Trivium, a well-analyzed stream cipher, and a variant aiming at achieving 128 bits of security. Last but not least, we recall that our construction was aiming at compressing the size of transmissions between Alice and Charlie. We support an encryption rate $|c'|/|m|$ that becomes asymptotically close to 1 for long messages, *e.g.* for $\ell_m = 1$GB message length, our construction instantiated with Trivium (resp. Kreyvium), yields an expansion rate of 1.08 (resp. 1.16).

5 Conclusion

Our work shows that the promising performances obtained by the recent HE-dedicated cipher LowMC can also be achieved with Trivium, a well-known primitive whose security has been thoroughly analyzed, *e.g.* [45,22,5,28,41]. The 10-year analysis effort from the community, initiated by the eSTREAM competition, enables us to gain confidence in its security. cipher is essentially the same.

From a more fundamental perspective, one may wonder how many multiplicative levels are *strictly necessary* to achieve a secure compressed encryption scheme, irrespective of any performance metric such as the number of homomorphic bit multiplications to perform in the decompression circuit. We already know that a multiplicative depth of $\lceil \log \kappa \rceil + 1$ is achievable for κ-bit security (*cf.* [15]). Can one do better or prove that this is a lower bound?

The provable security of a KEM-DEM construct where the KEM is homomorphic also remains an open question. In particular, assuming the KEM part is just IND-CPA, what would be the minimum security requirements expected from the DEM part to yield an IND-CPA construction?

References

1. Algorithms, key size and parameters report 2014. Tech. rep., ENISA (2014)
2. Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM. In: EUROCRYPT. LNCS, vol. 3494, pp. 128–146. Springer (2005)
3. Albrecht, M., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: EUROCRYPT. LNCS, vol. 9056, pp. 430–454. Springer (2015)
4. Armknecht, F., Mikhalev, V.: On Lightweight Stream Ciphers with Shorter Internal States. In: FSE. LNCS, vol. 9054, pp. 451–470. Springer (2015)
5. Aumasson, J., Dinur, I., Meier, W., Shamir, A.: Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In: FSE. LNCS, vol. 5665, pp. 1–22. Springer (2009)
6. Babbage, S.: A space/time trade-off in exhaustive search attacks on stream ciphers. In: European Convention on Security and Detection. No. 408, IEEE (1995)

7. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: FOCS. pp. 394–403. IEEE Computer Society (1997)
8. Berbain, C., Gilbert, H.: On the Security of IV Dependent Stream Ciphers. In: FSE. LNCS, vol. 4593, pp. 254–273. Springer (2007)
9. Biryukov, A., Shamir, A.: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In: ASIACRYPT. LNCS, vol. 1976, pp. 1–13. Springer (2000)
10. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications. In: ASIACRYPT. LNCS, vol. 7658, pp. 208–225. Springer (2012)
11. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) Fully Homomorphic Encryption without Bootstrapping. TOCT 6(3), 13 (2014)
12. Cannière, C.D., Dunkelman, O., Knezevic, M.: KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: CHES. LNCS, vol. 5747, pp. 272–288. Springer (2009)
13. Cannière, C.D., Lano, J., Preneel, B.: Comments on the rediscovery of time memory data tradeoffs. Tech. rep., eSTREAM - ECRYPT Stream Cipher Project (2005)
14. Cannière, C.D., Preneel, B.: Trivium. In: New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 244–266. Springer (2008)
15. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: How to Compress Homomorphic Ciphertexts. IACR Cryptology ePrint Archive 2015, 113 (2015), https://eprint.iacr.org/2015/113
16. Carpov, S., Dubrulle, P., Sirdey, R.: Armadillo: a compilation chain for privacy preserving applications. In: ACM CCSW (2015)
17. Chakraborti, A., Chattopadhyay, A., Hassan, M., Nandi, M.: TriviA: A fast and secure authenticated encryption scheme. In: CHES. Lecture Notes in Computer Science, vol. 9293, pp. 330–353. Springer (2015)
18. Cheon, J.H., Coron, J., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch Fully Homomorphic Encryption over the Integers. In: EUROCRYPT. LNCS, vol. 7881, pp. 315–335. Springer (2013)
19. Coron, J., Lepoint, T., Tibouchi, M.: Scale-Invariant Fully Homomorphic Encryption over the Integers. In: PKC. LNCS, vol. 8383, pp. 311–328. Springer (2014)
20. Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: EUROCRYPT. LNCS, vol. 2656, pp. 345–359. Springer (2003)
21. Dinur, I., Liu, Y., Meier, W., Wang, Q.: Optimized Interpolation Attacks on LowMC. IACR Cryptology ePrint Archive 2015, 418 (2015)
22. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: EUROCRYPT. LNCS, vol. 5479, pp. 278–299. Springer (2009)
23. Doröz, Y., Hu, Y., Sunar, B.: Homomorphic AES Evaluation using NTRU. IACR Cryptology ePrint Archive 2014, 39 (2014)
24. Doröz, Y., Shahverdi, A., Eisenbarth, T., Sunar, B.: Toward Practical Homomorphic Evaluation of Block Ciphers Using Prince. In: WAHC. LNCS, vol. 8438, pp. 208–220. Springer (2014)
25. ECRYPT - European Network of Excellence in Cryptology: The eSTREAM Stream Cipher Project. http://www.ecrypt.eu.org/stream/ (2005)
26. Fan, J., Vercauteren, F.: Somewhat Practical Fully Homomorphic Encryption. IACR Cryptology ePrint Archive 2012, 144 (2012)
27. Fau, S., Sirdey, R., Fontaine, C., Aguilar, C., Gogniat, G.: Towards practical program execution over fully homomorphic encryption schemes. In: IEEE International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. pp. 284–290 (2013)

28. Fouque, P., Vannet, T.: Improving Key Recovery to 784 and 799 Rounds of Trivium Using Optimized Cube Attacks. In: FSE. LNCS, vol. 8424, pp. 502–517. Springer (2013)
29. Fuhr, T., Minaud, B.: Match Box Meet-in-the-Middle Attack against KATAN. In: FSE. LNCS, vol. 8540, pp. 61–81. Springer (2014)
30. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC. pp. 169–178. ACM (2009)
31. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic Evaluation of the AES Circuit. In: CRYPTO. LNCS, vol. 7417, pp. 850–867. Springer (2012)
32. Golic, J.D.: Cryptanalysis of alleged A5 stream cipher. In: EUROCRYPT'97. LNCS, vol. 1233, pp. 239–255. Springer-Verlag (1997)
33. Graepel, T., Lauter, K.E., Naehrig, M.: ML Confidential: Machine Learning on Encrypted Data. In: ICISC. LNCS, vol. 7839, pp. 1–21. Springer (2012)
34. Halevi, S., Shoup, V.: Algorithms in HElib. In: CRYPTO, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 554–571 (2014)
35. Hofheinz, D., Kiltz, E.: Secure Hybrid Encryption from Weakened Key Encapsulation. In: CRYPTO. LNCS, vol. 4622, pp. 553–571. Springer (2007)
36. Hong, J., Sarkar, P.: New Applications of Time Memory Data Tradeoffs. In: ASIACRYPT. LNCS, vol. 3788, pp. 353–372. Springer (2005)
37. Iwata, T.: New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In: FSE. LNCS, vol. 4047, pp. 310–327. Springer (2006)
38. Jakobsen, T., Knudsen, L.R.: The interpolation attack on block ciphers. In: FSE. LNCS, vol. 1267, pp. 28–40. Springer (1997)
39. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Second Edition. Chapman and Hall/CRC Press (2014)
40. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems. In: ASIACRYPT. LNCS, vol. 6477, pp. 130–145. Springer (2010)
41. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential Cryptanalysis of Trivium and KATAN. In: SAC. LNCS, vol. 7118, pp. 200–212. Springer (2011)
42. Lauter, K., López-Alt, A., Naehrig, M.: Private Computation on Encrypted Genomic Data. In: LATINCRYPT. LNCS (2014)
43. Lepoint, T., Naehrig, M.: A Comparison of the Homomorphic Encryption Schemes FV and YASHE. In: AFRICACRYPT. LNCS, vol. 8469, pp. 318–335. Springer (2014)
44. Lepoint, T., Paillier, P.: On the Minimal Number of Bootstrappings in Homomorphic Circuits. In: WAHC. LNCS, vol. 7862, pp. 189–200. Springer (2013)
45. Maximov, A., Biryukov, A.: Two Trivial Attacks on Trivium. In: SAC. vol. 4876, pp. 36–55. Springer (2007)
46. Naehrig, M., Lauter, K.E., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: ACM CCSW. pp. 113–124. ACM (2011)
47. National Institute of Standards and Technology: Recommendation for Block Cipher Modes of Operation. NIST Special Publication 800-38A (2001)
48. Rogaway, P.: Evaluation of some blockcipher modes of operation. Cryptrec (2011)
49. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptography 71(1), 57–81 (2014)
50. Yasuda, K.: A New Variant of PMAC: Beyond the Birthday Bound. In: CRYPTO. LNCS, vol. 6841, pp. 596–609. Springer (2011)