# Non-Interactive Keyed-Verification Anonymous Credentials

Geoffroy Couteau, Michael Reichle

# Non-Interactive Keyed-Verification Anonymous Credentials

Geoffroy Couteau* and Michael Reichle**

Karlsruhe Institute of Technology, Karlsruhe, Germany

**Abstract.** Anonymous credential (AC) schemes are protocols which allow for authentication of authorized users without compromising their privacy. Of particular interest are non-interactive anonymous credential (NIAC) schemes, where the authentication process only requires the user to send a single message that still conceals its identity. Unfortunately, all known NIAC schemes in the standard model require pairing based cryptography, which limits them to a restricted set of specific assumptions and requires expensive pairing computations. The notion of keyed-verification anonymous credential (KVAC) was introduced in (Chase et al., CCS'14) as an alternative to standard anonymous credential schemes allowing for more efficient instantiations; yet, making existing KVAC non-interactive either requires pairing-based cryptography, or the Fiat-Shamir heuristic.
In this work, we construct the first non-interactive keyed-verification anonymous credential (NIKVAC) system in the standard model, without pairings. Our scheme is efficient, attribute-based, supports multi-show unlinkability, and anonymity revocation. We achieve this by building upon a combination of algebraic MAC with the recent designated-verifier non-interactive zero-knowledge (DVNIZK) proof of knowledge of (Couteau and Chaidos, Eurocrypt'18). Toward our goal of building NIKVAC, we revisit the security analysis of a MAC scheme introduced in (Chase et al., CCS'14), strengthening its guarantees, and we introduce the notion of *oblivious* non-interactive zero-knowledge proof system, where the prover can generate non-interactive proofs for statements that he *cannot check by himself*, having only a part of the corresponding witness, and where the proof can be checked efficiently given the missing part of the witness. We provide an efficient construction of an oblivious DVNIZK, building upon the specific properties of the DVNIZK proof system of (Couteau and Chaidos, Eurocrypt'18).

**Keywords.** Anonymous credentials, keyed-verification anonymous credentials, non-interactive anonymous credentials, designated-verifier non-interactive zero-knowledge proofs.

## 1 Introduction

### 1.1 Anonymous Credentials

Anonymous credentials, introduced in the seminal work of Chaum [14], allow users to authenticate in an anonymous way to a variety of services. Each user can receive credentials from authorities, and register pseudonyms with authorities and verifiers. These pseudonyms are associated to the identity of the user, but should be *unlinkable* to its exact identity. That is, another entity should not be able to check whether two pseudonyms are associated with the same identity. Authorities can issue credentials to users which can be shown to verifiers, and the presentation of a credential should only leak the information that the user knows the identity associated to the pseudonym, and owns a credential from the authority for this identity. This guarantees the *anonymity* of users. In order for credentials to make sense, they must be *unforgeable*: a user should not be able to present a credential without having received one from the authority first. Due to their wide range of real-world applications, anonymous credentials have received a constant attention from the cryptographic community [1,2,4,6–9,13,17,22,25,26,29,33].

**Non-Interactive Anonymous Credentials.** Non-interactive anonymous credentials (NIAC) are anonymous credentials where the process of showing possession of a valid credential to a verifier requires sending a single message from the user to the verifier. Non-interactivity in anonymous

* geoffroy.couteau@kit.edu
** m.reichle95@outlook.com

credentials is considered to be a highly desirable security property, and was the focus on an important research effort [3,4,26]. However, a downside of existing NIAC scheme is that all known constructions in the standard model require the use of pairing based cryptography, which limits their efficiency (since pairing are a relatively expensive cryptographic operation) and restricts the set of assumptions their security can be based on. While some interactive anonymous credential schemes can be made non-interactive in the random oracle model under the Fiat-Shamir transform, this is known to provide only heuristic security arguments in the standard model [10,20,24].

**Keyed-Verification Anonymous Credential.** Most commonly, anonymous credential schemes allow for a single credential to be shown more than once to multiple verifiers. The notion of *keyed-verification anonymous credentials* (KVAC) was introduced in [13]; it restricts credential to only be valid with respect to one verifier and requires the authority and verifier to share a secret key. The key observation of [13] is that such restricted anonymous credentials can be instantiated very efficiently, using algebraic message authentication codes. Therefore, in numerous applications where the restriction to keyed-verification is not an issue, they can be used to allow for more efficient instantiations. Think for example of a bus company issuing monthly pass, where the pass must be shown each time a user boards a bus; here, it is reasonable to assume that the bus device can share a secret-key with the bus company (since both belong to the same organisation).

A downside of the KVAC scheme of [13], however, is that the process of showing possession of a credential requires an interactive protocol between the user and the verifier. This protocol can be made non-interactive, but this either requires the Fiat-Shamir transform (leading to a protocol secure in the random oracle model only), or the use of pairing-based cryptography, nullifying the efficiency advantages of KVAC with respect to their publicly verifiable counterpart.

### 1.2 Our Contribution

In this work, we construct the first non-interactive keyed-verification anonymous credential scheme (NIKVAC) in the standard model, without relying on pairing-based cryptography. Our NIKVAC is very expressive: it natively supports multi-show unlinkability (*i.e.*, when showing possession of a credential multiple time to a verifier, the latter cannot tell whether these correspond to the same user) or pseudonyms (the verifier knows a pseudonym that he can link a credential to, but that he cannot link to the actual identity of the user), without any additional cost (*i.e.*, we do not require to generate an additional commitment to the identity and prove knowledge of its content to obtain pseudonyms, as in most alternative approaches; rather, such commitments are natively and implicitly defined by our scheme). Our NIKVAC is also attribute-based (it supports vectors of attributes as opposed to identities, and can handle a variety of relations on the attributes), and supports anonymity revocation (there exists a global trapdoor which a trusted authority can use to revoke the anonymity of a misbehaving user, efficiently extracting his identity from any accepting credential).

While our scheme is the first NIKVAC in the standard model without pairings, we observe (this is in fact the starting point of our work) that there is a relatively natural construction of a NIKVAC which is obtained by starting with the (interactive) scheme of [13], and replacing the underlying zero-knowledge proof system by the designated-verifier non-interactive zero-knowledge proof system of [11]. While this observation is interesting in itself, the security analysis of the resulting construction does not present major technical difficulties (although it is not entirely straightforward). In this work, we refine this approach, adopting a different strategy to better exploit the structural properties of the proof system of [11]. Our optimized approach provides strong efficiency improvements (which we detail in Section 1.6) over the previous alternative.

### 1.3 Our Approach

Our starting point is the interactive KVAC scheme of [13]. In this scheme, a credential is an algebraic MAC signature on the identity of the user. Anonymous presentation of a credential is done (informally) by masking the credential, and providing some zero-knowledge proofs of knowledge of the identity together with the masking informations satisfying the appropriate relation, which

allows the verifier (who knows the secret MAC key) to check that the masked credential does indeed verify correctly with respect to the (hidden) identity of the user.

To make this scheme non-interactive, the basic observation is that it suffices to rely on a *designated-verifier* non-interactive zero-knowledge (DVNIZK) proof of knowledge of the appropriate values. Unlike their publicly-verifiable counterpart, there exists efficient constructions of DVNIZK proof systems which does not rely on pairings. However, until recently, all known constructions of DVNIZK proof systems [12,16,18] suffered from two important downside, each of them preventing their use in a NIKVAC scheme:

- they can only deal with *existential* statements, while anonymous credentials crucially rely on proving *knowledge* of the signed identity, and
- they only satisfy a *bounded* notion of security, where the soundness of the proof is only guaranteed to hold if the prover is restricted to query a verification oracle an a priori bounded number of times. In an anonymous credential system, however, the users can interact freely with a verifier and receive feedback on whether proofs of credential possession was accepted or not; hence, for all of these scheme, a malicious user could forge a credential which is accepted by the verifier even though it was not issued by the authority, by interacting a sufficient (polynomial) number of times with the verifier.

This situation recently changed with the introduction in [11] of the first DVNIZK proof system which allows to provide proofs of *knowledge* of a witness, for a wide variety of algebraic statements, where soundness is *unbounded* (it holds even if the prover is given arbitrary access to a verification oracle). Furthermore, the framework of [11] allows for efficient DVNIZK proofs, directly proportional to the size of the algebraic statement to be proven.

A natural approach toward building a NIKVAC scheme is therefore to rely on the KVAC scheme of [13], and to replace the underlying zero-knowledge proofs by appropriate DVNIZK, using the framework of [11]. However, while this approach should lead to a secure NIKVAC, it misses the opportunity to exploit the specific structure of the scheme of [11] to get improved efficiency guarantees. Therefore, we choose instead to tackle the problem directly and construct an optimized NIKVAC system, heavily building upon the specific structure of the DVNIZK of [11].

### 1.4 Our Techniques

To describe our strategy, it is helpful to start from a natural but insecure approach. As in [13], a credential will simply be a signature on the identity of the user using an algebraic MAC. To show possession of a credential, the user can simply send this credential (but not his identity) and prove with a DVNIZK that he knows an identity such that the MAC verification algorithm returns 1 when given as input this identity and the credential. A first observation is that this approach allows for a straightforward optimization: in the most common setting, the verifier must know a pseudonym associated to the user (which cannot be linked to his identity), which will usually take the form of a commitment to the identity of the user. We observe, however, that a DVNIZK proof of knowledge within the framework of [11] does already include an encryption of the witness, and the proof of knowledge property does in particular guarantee that the witness whose knowledge is proven is indeed the one encrypted in the proof. Therefore, it is not necessary to add a commitment to the identity and prove that the committed value is the one for which the user knows a credential; rather, the user can simply compute this encryption ahead of time (this does not require knowing the credential) and send it to the verifier, which will store it as being the user pseudonym. Then, each time the user wants to authenticate, he only have to generate the "missing part" of the proof with respect to this encryption. This strongly reduces the size of the proof (since the proof does not need to include an explicit proof regarding a commitment anymore), and allows to reuse a significant portion of the proof across many authentications.

However, the natural approach of disclosing a credential $\sigma$ and proving knowledge of an identity that verifies correctly with respect to $\sigma$ fails, for two reasons:

- First, the above approach does not guarantee anonymity, because the verifier (who knows the secret MAC key) could find out the identity of the user simply by colluding with the authority, and evaluating the MAC verification algorithms on all identities previously submitted to the authority, to find out which one verifies correctly with respect to this credential.

– Second, and more importantly, the MAC verification requires knowledge of the secret MAC key, which the user does not know; hence, he cannot possibly issue a proof that his credential verifies correctly, since checking this statement does already require knowing the secret MAC key.

We first explain how we address the second concern. Our idea is to build upon the specific malleability property of the DVNIZK proof system of [11] to build an *oblivious* DVNIZK proof system, which allows the prover to issue a proof for a statement *even if he does not know himself whether the statement does hold*. This does not contradict the security of the MAC scheme, since the proof system is not publicly verifiable: hence, even after he builds the proof, the prover cannot check by himself whether this proof verifies correctly. Intuitively, the prover will construct a "partial non-interactive proof" which is malleable in the following sense: given this proof and the secret MAC key, the verifier can reconstruct himself the complete proof that the credential verifies correctly. If the prover does not know the appropriate witness, the reconstructed proof will not verify correctly. The partial proof should not leak any more information about the witness held by the prover than what is leaked by the reconstructed proof; hence, by the zero-knowledge property of the DVNIZK proof system, this proof will only reveal whether the statement (which depends on both the prover witness and the secret key known to the verifier) is true. We believe that the concept of non-interactive oblivious proofs, which allows to prove that a statement is true while knowing only a part of the witness to a verifier knowing the "missing part" of the witness, might be of independent interest (we briefly elaborate on this in Section 1.5); in particular, it formalizes the approach taken (in the interactive setting) in previous works on keyed-verification anonymous credentials [13].

To tackle the first concern, the prover will randomize his credential and mask it with appropriate random values, and issue a partial proof that the *unmasked* credential does verify with respect to the secret key. We formalize both properties at once by introducing a new primitive, oblivious designated-verifier non-interactive zero-knowledge proofs of knowledge, which can be used to prove statements non-interactively even when the prover only knows a part of the witness, and can be simulated by a simulator that does not know neither the witness nor the word for which the proof is constructed, guaranteeing that the verifier will not only be unable to recover the witness, but also that he cannot possibly recover the credential, which would allow him to break anonymity by colluding with the authority.

Next, we provide an optimized construction of an oblivious DVNIZK proof system for the language of valid credentials, building upon the DVNIZK proof system of [11]. Proving security of the resulting proof system, however, runs into a subtle issue: when considering the more general setting of *attribute-based* anonymous credentials, where the user will have a secret *vector of attributes* instead of a secret identity, the unforgeability property of the underlying MAC scheme does not suffice to prove the soundness of the oblivious proof system. We provide two alternatives to overcome this issue:

– When the vector of attributes is of length one (*i.e.*, when we restrict our attention to non-attribute-based anonymous credentials, where the secret of the user is only his identity), we show that the public parameters of the MAC scheme suffice to reduce the security directly to the unforgeability of the MAC scheme. This setting already captures many possible applications.
– In the general setting, where the vector of attributes can be longer than 1, we show that the security can be proven if the MAC scheme satisfies a stronger notion of unforgeability, which we call *extended unforgeability*. Then, we revisit the security analysis of one of the two MAC schemes constructed in [13], which is secure in the generic group model, and prove that this scheme does in fact already satisfy extended unforgeability. While the second MAC scheme constructed in [13] (which is based on the decisional Diffie-Hellman assumption) does plausibly satisfy extended unforgeability, we leave it as an interesting open problem to prove it under a standard assumption, or to construct a MAC scheme with extended unforgeability under the DDH assumption. We note that considering algebraic MACs with stronger unforgeability guarantees is a relatively natural approach in the setting of anonymous credentials (see e.g. [3,4]), but the specific strengthening we require in our construction was not, to our knowledge, considered in previous works.

There is an additional requirement which we must take care of: the MAC schemes of [13] are only proven secure in groups of prime order, while the most natural instantiation of the DVNIZK proof

system of [11] typically requires composite-order groups. While the security of their DDH-based MAC easily extends to the composite order setting by assuming in addition that it is infeasible for any polytime adversary to find a generator of a strict subgroup (which is a standard and well-studied assumption), the proof of their generic-group-model-based (GGM-based) MAC is unconditional, hence it assumes that the adversary is unbounded, in which case there is an explicit attack on the composite-order variant of the scheme where the unbounded adversary constructs an invalid MAC signature in a strict subgroup of the group. We therefore revisit the security proof of the GGM-based MAC, and show that it holds in the generic group model assuming in addition that the adversary is polynomially bounded, and that the computational subgroup assumption holds. Altogether, we show that this gives rise to a highly optimized NIKVAC. In the next section, we discuss in more details the efficiency of our scheme.

### 1.5 Applications of Oblivious DVNIZK

Given the intermediate abstraction of oblivious designated non-interactive zero-knowledge proofs, the construction of NIKVAC follows very naturally. In fact, we could have provided a direct construction of NIKVAC from this approach, without formalizing the intermediate primitive. However, we believe that oblivious DVNIZKs can be interesting in their own right. We elaborate below.

Secure computation protocols allow a group of parties to securely evaluate a public function on their joint private input. We focus in this discussion on the case of two parties for simplicity. A common approach to secure two-party computation is to first design a scheme secure against passive adversaries, which do not deviate from the specifications of the protocol, and then to use zero-knowledge proofs to let all adversaries prove their honest behavior throughout the protocol. This transformation makes the protocol secure against malicious adversaries, which can deviate arbitrarily from the specifications of the protocol. To obtain round-efficient compilation of passively secure computation protocols into maliciously secure protocols, the most natural strategy is to rely on (designated-verifier) non-interactive zero-knowledge proofs (an alternative is to use implicit zero-knowledge proofs [5], but this adds two more rounds to the protocol) to prove honest behavior of each user after each round.

Oblivious DVNIZK allow for an alternative compilation strategy, which starts from a protocol with stronger security guarantees, but is in general more efficient. Let us call (informally) *half-maliciously secure* a secure computation protocol which is passively secure, and such that no malicious adversary can compromise the *privacy* of the inputs (but can possibly compromise the *correctness* of the computation). Let $\Pi$ be a half-maliciously secure protocol, securely computing a function $f$. Let $(x_1, x_2)$ denote the inputs of the parties. To convert $\Pi$ into a fully secure protocol, we first modify $\Pi$ to include commitments $(c_1, c_2)$ to the inputs (if $\Pi$ does not already include them). Then, to guarantee full security, one of the parties, which we call the prover, must send a single oblivious DVNIZK to the other party (the receiver) at the very end of the protocol, which is a proof that $y = f(x_1, x_2)$, where $y$ is the output of the protocol, and $(x_1, x_2)$ is committed in $(c_1, c_2)$. Note that the prover does not have the full witness for this statement (since it depends, in particular, on the private input of the verifier), but the prover and the verifier jointly have the full witness, allowing the verifier to check the proof without further interaction. This is in contrast with DVNIZK-based compilation, which requires proving honest behavior of all users at each round (here, we only prove *correctness* of the computation in the last round). We leave the formal proof of this observation to future work.

### 1.6 Efficiency

There is, to our knowledge, no existing previous construction of NIKVAC in the standard model. However, as we pointed out previously, there is a relatively natural construction which is obtained by starting from the scheme of [13], and replacing the underlying zero-knowledge proofs by DVNIZKs instantiated with [11]. Let us call this construction the CMZ+CC construction. We use CMZ+CC as a basis for comparison with our improved construction. We focus on the communication cost of showing possession of a credential, since the computation is directly proportional to the communication (hence, an improvement factor with respect to communication translates to a comparable improvement factor with respect to computation), and since the cost of issuing a

credential depends on the specific secure computation scheme used to implement it, which is not the focus of our work (we require the same blind issuance of an algebraic MAC as in previous works on KVAC).

For simplicity, we focus on the cost obtained when implementing the MAC with the more efficient GGM-based MAC scheme of [13]; when using the other, DDH-based MAC scheme, all costs must be roughly scaled up by 50% (up to constants), and the improvement factor of our method compared to the naive approach will be essentially identical. Let $\beta$ denote the length of the vector of attributes. In the minimal setting where the verifier knows a pseudonym, implemented as a commitment to the user's vector of attributes, instantiating the zero-knowledge proofs in [13] using the DVNIZK proof system of [11] leads to a proof size of $3\beta + 3$ group elements, and $6\beta + 2$ ciphertexts (in a typical instantiation of the DVNIZK of [11], the group will be a composite order abelian group, and the encryption scheme will be the Paillier encryption scheme).

In comparison, our proof of credential possession requires sending $\beta + 2$ group elements, and $2\beta + 2$ ciphertexts. Furthermore, all the ciphertexts can be sent once for all to the verifier (they form the pseudonym of the prover); each new credential presentation then requires only generating and sending $\beta + 2$ group elements (in comparison, the pseudonym in [13] is a tuple of $\beta$ commitments, hence sending the pseudonym ahead of time saves only $\beta$ group elements). For the important case of $\beta = 1$ attribute, and instantiating the DVNIZK with Paillier and a 2048-bit modulus, this corresponds to a factor of improvement of more than 7 in the proof size compared to [13]. In addition, using an optimization which we describe in Appendix C, the number of ciphertexts can be further reduced, from $2\beta + 2$ to $2\beta$. We summarize the comparison between our scheme and CMZ+CC in the table 1.

Eventually, we sketch a straightforward computational optimization (assuming an instantiation with the Paillier scheme and a 2048-bit modulus for concreteness): the exponents manipulated when constructing and verifying the proof are either attributes, random coins, or masks. If attributes are, say, up to 128-bit long, then under the short-exponent discrete logarithm assumption (which states that it is hard to find $x$ from $g^x$ even if $x$ is random but *short*, e.g. 128-bit long), all exponents can be taken either 128-bit long (for the attributes and the random coins) or 256-bit long (for the masks, since they must statistically mask the attributes over the integers). This makes computing exponentiations and scalar multiplications considerably more efficient than with full-size (i.e., 2048-bit) values.

**Comparison with Plain [13].** We briefly comment on the comparison with the plain scheme of [13] (which is either interactive, or non-interactive in the random oracle model). Our main efficiency bottleneck is the fact that we use the DVNIZK of [11], which requires to use a large order group.[1] Therefore, using natural parameters, we manipulate group elements of size 2048 bits, and ciphertexts of size 4096 bits. In constrast, [13] can work exclusively with group elements and exponents over any DDH-hard group, e.g. of size 256 bits. However, the proof size of [13] (not counting the size of the pseudonym) is $\beta + 2$ group elements and $3\beta + 2$ 256-bit exponents, for a total of 256 Byte. Our proof system achieves a proof size 756 Byte, less than three times larger in spite of our use of an 8-time larger group - and unlike [13], it is secure in the standard model (the ratio remains essentially the same if we instantiate instead the underlying MAC scheme with the DDH-based scheme of [13]).

### 1.7 Organization

In Section 2, we recall necessary preliminaries (further preliminaries are given in Appendix A. In Section 3, we recall the definition of MAC schemes, introduce a general algebraic MAC scheme, and define the stronger notion of extended unforgeability. In Section 4, we formally define non-interactive keyed-verification anonymous credentials and their security properties. In Section 5, we introduce the concept of oblivious DVNIZK and their security properties, provide an explicit instantiation tailored to our application, and formally prove its security. In Section 6, we show how to construct a non-interactive keyed-verification anonymous credential from a MAC scheme and an oblivious

[1] In [11], the size of the group must be equal to the size of the plaintext space of a DVNIZK-friendly encryption scheme, such as Paillier.

**Table 1.** Comparison of our optimized NIKVAC to a direct construction from [13] with the DVNIZK of [11].

| NIKVAC [1] | CMZ+CC | This Work | This Work (optimized) |
|---|---|---|---|
| $\beta$ attributes, group element length $n$, ciphertext size $m$ | | | |
| Pseudonym Size | $\beta n$ | $(2\beta+2)m$ | $2\beta m$ |
| Proof Size | $(2\beta+3)n + (6\beta+2)m$ | $(\beta+2)n$ | $(\beta+2)n$ |
| Prover Computation[2] | $(5\beta+2)A + (3\beta+1)(B+C)$ | $(2\beta+3)A$ | $(2\beta+3)A$ |
| Assumption | GGM+IND-CPA | GGM+IND-CPA | GGM+IND-CPA + short-exp dlog |
| (with Paillier) 1 attribute, group element length 2048, ciphertext size 4096 | | | |
| Pseudonym Size | 256 Byte | 2048 Byte | 1024 Byte |
| Proof Size | 5,38 kB | 756 Byte | 756 Byte |
| Prover Computation | $7A + 4(B+C)$ | $5A$ | $5A$ |
| Assumption | GGM+Paillier | GGM+Paillier | GGM+Paillier + short-exp dlog |

[1] We consider a minimal setting where the prover shows possession of a valid credential with respect to an identity committed in a *pseudonym* known to the verifier. We use the GGM-based scheme of [13] as the underlying algebraic MAC (the efficiency gain of our approach is essentially the same if one uses the DDH-based MAC of [13]).

[2] $A$ denotes the cost of an exponentiation in the group $\mathbb{G}$, $B$ denotes the cost of an encryption, $C$ denotes the cost of an homomorphic scalar multiplication. We note that, under the short-exponent discrete logarithm assumptions, all exponentiations in $\mathbb{G}$ (resp. all homomorphic scalar multiplications) can be performed with exponents (resp. scalars) of length at most 256 bits.

DVNIZK proof system. Eventually, in Appendix B, we prove that the first MAC scheme of [13] satisfies extended unforgeability in the generic group model (with composite order groups), and in Appendix C, we describe further improvements to our NIKVAC construction relying on the short-exponent discrete logarithm assumption.

## 2 Preliminaries

Throughout this paper, $\lambda$ denotes the security parameter. A probabilistic polynomial time algorithm (PPT, also denoted *efficient* algorithm) runs in time polynomial in the (implicit) security parameter $\lambda$. A positive function $f$ is *negligible* if for any polynomial $p$ there exists a bound $B > 0$ such that, for any integer $k \geq B$, $f(k) \leq 1/|p(k)|$. An event depending on $\lambda$ occurs with *overwhelming probability* when its probability is at least $1 - \mathrm{negl}(\lambda)$ for a negligible function negl. Given a finite set $S$, the notation $x \xleftarrow{\$} S$ means a uniformly random assignment of an element of $S$ to the variable $x$. We represent adversaries as interactive probabilistic Turing machines; the notation $\mathscr{A}^{\mathcal{O}}$ indicates that the machine $\mathscr{A}$ is given oracle access to $\mathcal{O}$. Adversaries will sometime output an arbitrary state $\mathsf{st}$ to capture stateful interactions.

**Abelian Groups and Modules.** We use additive notation for groups for convenience, and write $(\mathbb{G}, \boldsymbol{+})$ for an abelian group of order $k$. When it is clear from the context, we denote 0 its neutral element (otherwise, we denote it $0_{\mathbb{G}}$). We denote by $ord(\mathbb{G})$ the order of $\mathbb{G}$. We denote by $\bullet$ the scalar-multiplication algorithm (*i.e.* for any $(x, G) \in \mathbb{Z}_k \times \mathbb{G}$, $x \bullet G = G \boldsymbol{+} G \boldsymbol{+} \ldots \boldsymbol{+} G$, where the sum contains $x$ terms). Observe that we can naturally view $\mathbb{G}$ as a $\mathbb{Z}_k$-module $(\mathbb{G}, \boldsymbol{+}, \bullet)$, for the ring $(\mathbb{Z}_k, +, \cdot)$. For simplicity, we write $\boldsymbol{-}G$ for $(-1) \bullet G$. We use lower case to denote elements of $\mathbb{Z}_k$, upper case to denote elements of $\mathbb{G}$, and bold notations to denote vectors. We extend the notations $(\boldsymbol{+}, \boldsymbol{-})$ to vectors and matrices in the natural way, and write $\boldsymbol{x} \bullet \boldsymbol{G}$ to denote the scalar product $x_1 \bullet G_1 \boldsymbol{+} \ldots \boldsymbol{+} x_t \bullet G_t$ (where $\boldsymbol{x}, \boldsymbol{G}$ are vectors of the same length $t$). For a vector $\boldsymbol{v}$, we denote by $\boldsymbol{v}^{\intercal}$ its transpose. By $\mathsf{GGen}(1^\lambda)$, we denote a probabilistic efficient algorithm that, given the security parameter $\lambda$, generates an abelian group $\mathbb{G}$ in which the CSG and DLSE assumption defined below holds in respect to $\lambda$. Note that this implies that the normal discrete log problem is hard in this group, as well. In the following, we write $(\mathbb{G}, k) \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$. Additionally, we denote by $\mathsf{GGen}(1^\lambda, k)$ a group generation algorithm that allows us to select the order $k$ beforehand.

**RSA Groups.** A *strong prime* is a prime $p = 2p' + 1$ such that $p'$ is also a prime. We call *RSA modulus* a product $n = pq$ of two strong primes. We denote by $\varphi$ Euler's totient function; it holds

that $\varphi(n) = (p-1)(q-1)$. We denote by $\mathbb{J}_n$ the cyclic subgroup of $\mathbb{Z}_n^*$ of elements with Jacobi symbol 1 (the order of this group is $\varphi(n)/2$), and by $\mathsf{QR}_n$ the cyclic subroup of squares of $\mathbb{Z}_n^*$ (which is also a subgroup of $\mathbb{J}_n$ and has order $\varphi(n)/4$). By $\mathsf{Gen}(1^\lambda)$, we denote a probabilistic efficient algorithm that, given the security parameter $\lambda$, generates a strong RSA modulus $n$ and secret parameters $(p, q)$ where $n = pq$, such that the best known algorithm for factoring $n$ takes time $2^\lambda$. In the following, we write $(n, (p, q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and call abelian groups with order $n$ composite order groups, if $n$ is a RSA modulus.

**Generic Group Model.** The generic group model ($\mathsf{GGM}$) was introduced in [35] and is an idealized model of groups. It captures groups with no additional structure apart from being a group. In such generic groups, the only possibility of attacking a cryptographic primitive is utilizing generic algorithms which only make use of group operations.

In proofs, the generic group model is captured by giving an adversary access to the group through random encodings of the group elements as bitstrings and a group operation oracle. Note that if a cryptographic primitive is proven secure in the $\mathsf{GGM}$, it only ensures that it can not be broken with generic algorithms. In order to simulate the oracle in this work, we will require the following lemma, based on [34].

**Lemma 1 (Generalised Schwartz-Zippel).** *Let* $(\mathsf{n}, (p, q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda), \mathbb{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda, \mathsf{n})$ *and* $F \in \mathbb{Z}_\mathsf{n}[\overline{x_1}, \overline{x_2}, .., \overline{x_l}]$ *with* $F \neq 0 \wedge deg(F) = d \geq 0$. *Let* $p' \in \{p, q\}$ *and* $\mathbb{P}$ *a subgroup of* $\mathbb{G}$ *of order* $p'$. *It holds that*

$$\Pr\left[x = (x_1, x_2, .., x_l) \xleftarrow{\$} \mathbb{P}^l : F(x) = 0\right] \leq \frac{d}{p'}$$

### 2.1 Assumptions

**Computational Subgroup Assumption (CSG).** The computational subgroup assumption states that no bounded adversary can output a generator for a non-trivial subgroup. Or more formally, for all PPT adversaries $\mathscr{A}$, it holds that

$$\Pr\begin{bmatrix}(\mathsf{n}, (p, q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda), \\ \mathbb{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda, \mathsf{n}), \\ G \leftarrow \mathscr{A}(\mathbb{G}, \mathsf{n}),\end{bmatrix} : G \neq 0_\mathbb{G} \wedge (p \bullet G = 0_\mathbb{G} \vee q \bullet G = 0_\mathbb{G})\Bigg] \leq \mu(\lambda)$$

where $\mu(\lambda) = \mathrm{negl}(\lambda)$.

**Decisional-Diffie-Hellman (DDH) Assumption.** Let $\mathbb{G}$ be a group with order $\mathsf{n}$. For all PPT adversaries $\mathscr{A}$ it holds that

$$\left|\Pr\begin{bmatrix}a, b, c \xleftarrow{\$} \mathbb{Z}_\mathsf{n} \\ A \leftarrow a \bullet G, B \leftarrow b \bullet G, C \leftarrow ab \bullet G\end{bmatrix} : \mathscr{A}(G, A, B, C) = 1\right] - \Pr\begin{bmatrix}a, b, c \xleftarrow{\$} \mathbb{Z}_\mathsf{n} \\ A \leftarrow a \bullet G, B \leftarrow b \bullet G, C \leftarrow c \bullet G\end{bmatrix} : \mathscr{A}(G, A, B, C) = 1\Bigg]\Bigg| \leq \mu(\lambda)$$

### 2.2 Encryption Schemes

A public-key encryption scheme $S$ is a triple of PPT algorithms $(S.\mathsf{KeyGen}, S.\mathsf{Enc}, S.\mathsf{Dec})$, such that $S.\mathsf{KeyGen}$ generates encryption and decryption keys $(\mathsf{ek}, \mathsf{dk})$, $S.\mathsf{Enc}_\mathsf{ek}$, given a plaintext, outputs a (randomized) ciphertext, and $S.\mathsf{Dec}_\mathsf{dk}$, given a ciphertext, outputs a plaintext. An encryption scheme must be correct ($S.\mathsf{Enc}_\mathsf{dk}(S.\mathsf{Enc}_\mathsf{ek}(m)) = m$ for every message $m$) and $\mathsf{IND\text{-}CPA}$ secure (no adversary can distinguish between the encryptions of two messages of its choice). We defer to Appendix A the formal definition of encryption schemes and their security properties.

In this work, we will focus on additively homomorphic encryption schemes, which are homomorphic for both the message and the random coin. More formally, we require that the message space $\mathcal{M}$ and the random source $\mathcal{R}$ are integer sets $(\mathbb{Z}_M, \mathbb{Z}_R)$ for some integers $(M, R)$, and that there exists an efficient operation $\oplus$ such that for any $(\mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, any

$(m_1, m_2) \in \mathbb{Z}_M^2$ and $(r_1, r_2) \in \mathbb{Z}_R^2$, denoting $(C_i)_{i\leq 2} \leftarrow (S.\mathsf{Enc}_{\mathsf{ek}}(m_i; r_i))_{i\leq 2}$, it holds that $C_1 \oplus C_2 = S.\mathsf{Enc}_{\mathsf{ek}}(m_1 + m_2 \bmod M; r_1 + r_2 \bmod R)$. We say an encryption scheme is *strongly additive* if it satisfies these requirements. Note that the existence of $\oplus$ implies (via a standard square-and-multiply method) the existence of an algorithm that, on input a ciphertext $C = S.\mathsf{Enc}_{\mathsf{ek}}(m; r)$ and an integer $\rho \in \mathbb{Z}$, outputs a ciphertext $C' = S.\mathsf{Enc}_{\mathsf{ek}}(\rho m \bmod M; \rho r \bmod R)$. We denote by $\rho \odot C$ the external multiplication of a ciphertext $C$ by an integer $\rho$, and by $\ominus$ the operation $C \oplus (-1) \odot C'$ for two ciphertexts $(C, C')$. We will sometimes slightly abuse these notations, and write $C \oplus m$ (resp. $C \ominus m$) for a plaintext $m$ to denote $C \oplus S.\mathsf{Enc}_{\mathsf{ek}}(m; 0)$ (resp. $C \ominus S.\mathsf{Enc}_{\mathsf{ek}}(m; 0)$). We extend in a natural way the algorithm $\mathsf{Enc}$ over vectors: for vectors $\boldsymbol{m} = (m_i)_i \in \mathbb{Z}_M^*$ and $\boldsymbol{r} = (r_i)_i \in \mathbb{Z}_R^*$ of the same size, $S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m}; r)$ denotes the vector $(S.\mathsf{Enc}_{\mathsf{ek}}(m_i, r_i))_i$. We extend the algorithm $\mathsf{Dec}$ to vectors of ciphertexts in a similar way.

**The Paillier Encryption Scheme.** The Paillier encryption scheme [30] is a well-known additively homomorphic encryption scheme over $\mathbb{Z}_n$ for an RSA modulus $n$. We describe here a standard variant [19,28], where the random coin is an exponent over $\mathbb{J}_n$ rather than a group element. Note that the exponent space of $\mathbb{J}_n$ is $\mathbb{Z}_{\varphi(n)/2}$, which is a group of unknown order; however, it suffices to draw exponents at random from $\mathbb{Z}_{n/2}$ to get a distribution statistically close from uniform over $\mathbb{Z}_{\varphi(n)/2}$. The $\mathsf{IND\text{-}CPA}$ security of the Paillier encryption scheme reduces to the $\mathsf{DCR}$ assumption, which states that it is computationally infeasible to distinguish random $n$'th powers over $\mathbb{Z}_{n^2}^*$ from random elements of $\mathbb{Z}_{n^2}^*$.

- $\mathsf{KeyGen}(1^\lambda)$: run $(n, (p, q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, pick $g \xleftarrow{\$} \mathbb{J}_n$, set $h \leftarrow g^n \bmod n^2$, and compute $\delta \leftarrow n^{-1} \bmod \varphi(n)$ ($n$ and $\varphi(n)$ are relatively prime). Return $\mathsf{ek} = (n, h)$ and $\mathsf{dk} = \delta$;
- $\mathsf{Enc}(\mathsf{ek}, m; r)$: given $m \in \mathbb{Z}_n$, for a random $r \xleftarrow{\$} \mathbb{Z}_{n/2}$, compute and output $c \leftarrow (1 + n)^m \cdot h^r \bmod n^2$;
- $\mathsf{Dec}(\mathsf{dk}, c)$: compute $x \leftarrow c^{\mathsf{dk}} \bmod n$ and $c_0 \leftarrow [c \cdot x^{-n} \bmod n^2]$. Return $m \leftarrow (c_0 - 1)/n$.

**DVNIZK-Friendly Encryption Scheme.** We say that a strongly additive encryption scheme is $\mathsf{DVNIZK}$-*friendly*, when it satisfies the following additional properties:

- Coprimality Property: we require that the size $M$ of the plaintext space and the size $R$ of the random source are coprime, *i.e.*, $\gcd(M, R) = 1$;
- Decodable: for any $(\mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, the function $f_{\mathsf{ek}} : m \mapsto \mathsf{Enc}_{\mathsf{ek}}(m; 0)$ must be efficiently invertible (*i.e.*, there is a PPT algorithm, which is given $\mathsf{ek}$, computing $f_{\mathsf{ek}}^{-1}$ on any value from the image of $f_{\mathsf{ek}}$).

Note that the Paillier cryptosystem is $\mathsf{DVNIZK}$-friendly: ($\gcd(n, \varphi(n)) = 1$, and any message $m$ can be efficiently recovered from $\mathsf{Enc}_{\mathsf{ek}}(m; 0) = (1 + n)^m \bmod n^2$).

### 2.3 Non-Interactive Zero-Knowledge Proof of Knowledge Systems

A (designated-verifier) non-interactive zero-knowledge ($\mathsf{DVNIZK}$) proof system for a language $\mathscr{L}$ is a quadruple $(\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$, as follows: $\Pi.\mathsf{Setup}$ generates the setup parameters; $\Pi.\mathsf{KeyGen}$ generate the (public) proving key and the verification key (which is private in a designated-verifier scheme, and public in a publicly-verifiable one); $\Pi.\mathsf{Prove}$, given the proving key, a word $x$ an a witness $w$ for $x \in \mathscr{L}$, outputs a proof $\pi$; and $\Pi.\mathsf{Verify}$, given the verification key, $x$, and $\pi$, outputs either accept or reject.

A $\mathsf{DVNIZK}$ proof system must be complete (if $x \in \mathscr{L}$, the verifier accept), sound (if $x \notin \mathscr{L}$, no malicious prover can cause the verifier to accept; we usually want a stronger security notion, *unbounded extractability*, which states that a polytime extractor can extract a valid witness from any accepting proof, even if the proof was adversarially generated with arbitrary access to a verification oracle), and *zero-knowledge* (the proof can be simulated without knowledge of the witness). We defer to Appendix A the formal definition of $\mathsf{DVNIZKs}$ and their security properties.

**The DVNIZK of Chaidos and Couteau.** This DVNIZK proof of knowledge system was introduced in [11] and satisfies composable zero-knowledge, and statistical adaptive unbounded knowledge-extractability. The proofs are generated for statements defined by a linear map over $\mathbb{G}$:
Let $k$ be an integer, $(\mathbb{G}, \boldsymbol{+})$ be an abelian group of order $k$, and $(\alpha, \beta, \gamma)$ be three integers. Let $\boldsymbol{G} \in \mathbb{G}^\alpha$ denote a vector of *public parameters*, and let $\boldsymbol{C} \in \mathbb{G}^\beta$ denote a public *word*. This system considers statements $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ defined by a linear map $\Gamma : (\mathbb{G}^\alpha, \mathbb{G}^\beta) \mapsto \mathbb{G}^{\gamma \times \beta}$ such that $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C}$ coresponds to the statement "I know $\boldsymbol{x} \in \mathbb{Z}_k^\gamma$ such that $\boldsymbol{x} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = \boldsymbol{C}$". Let $S = (S.\mathsf{KeyGen}, S.\mathsf{Enc}, S.\mathsf{Dec})$ denote a DVNIZK-friendly encryption scheme with plaintext space $\mathbb{Z}_k$. The algorithms $(\Pi_\mathsf{K}.\mathsf{Setup}, \Pi_\mathsf{K}.\mathsf{KeyGen}, \Pi_\mathsf{K}.\mathsf{Prove}, \Pi_\mathsf{K}.\mathsf{Verify})$ form a DVNIZK of knowledge $\Pi_\mathsf{K}$ for a statement $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$ over a word $\boldsymbol{C} \in \mathbb{G}^\beta$, with public parameters $\boldsymbol{G} \in \mathbb{G}^\alpha$, defined by a linear map $\Gamma : (\mathbb{G}^\alpha, \mathbb{G}^\beta) \mapsto \mathbb{G}^{\gamma \times \beta}$:

- $\Pi_\mathsf{K}.\mathsf{Setup}(1^\lambda)$ : compute $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Note that $\mathsf{ek}$ defines a plaintext space $\mathbb{Z}_k$ and a random source $\mathbb{Z}_R$. As the IND-CPA and strong additive properties of $S$ require $R$ to be unknown, we assume that a bound $B$ on $R$ is publicly available. We denote $\ell \leftarrow 2^\lambda k B$.
- $\Pi_\mathsf{K}.\mathsf{KeyGen}(1^\lambda)$: pick $e \leftarrow \mathbb{Z}_\ell$, set $\mathsf{pk} \leftarrow S.\mathsf{Enc}_\mathsf{ek}(0; e)$ and $\mathsf{vk} \leftarrow e$.
- $\Pi_\mathsf{K}.\mathsf{Prove}(\mathsf{pk}, \boldsymbol{C}, \boldsymbol{x})$: on a word $\boldsymbol{C} \in \mathbb{Z}_k^\beta$, with witness $\boldsymbol{x}$ for the statement $\mathsf{St}_\Gamma(\boldsymbol{G}, \boldsymbol{C})$, pick $\boldsymbol{x'} \xleftarrow{\$} \mathbb{Z}_k^\gamma$, $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^\gamma$, compute

$$\begin{aligned} \boldsymbol{X} &\leftarrow S.\mathsf{Enc}_\mathsf{ek}(\boldsymbol{x}; \boldsymbol{r}), \\ \boldsymbol{X'} &\leftarrow S.\mathsf{Enc}_\mathsf{ek}(\boldsymbol{x'}; 0) \ominus (\boldsymbol{r} \odot \mathsf{pk}) = S.\mathsf{Enc}_\mathsf{ek}(\boldsymbol{x'}; -e \cdot \boldsymbol{r}), \\ \boldsymbol{C'} &\leftarrow \boldsymbol{x'} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}), \end{aligned}$$

  and output $\boldsymbol{\pi} \leftarrow (\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$.
- $\Pi_\mathsf{K}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \boldsymbol{C}, \boldsymbol{\pi})$: parse $\boldsymbol{\pi}$ as $(\boldsymbol{X}, \boldsymbol{X'}, \boldsymbol{C'})$. Check that $e \odot \boldsymbol{X} \oplus \boldsymbol{X'}$ is decodable, and decode it to a vector $\boldsymbol{d} \in \mathbb{Z}_k^\gamma$. Check that

$$\boldsymbol{d} \bullet \Gamma(\boldsymbol{G}, \boldsymbol{C}) = e \bullet \boldsymbol{C} \boldsymbol{+} \boldsymbol{C'}.$$

  If all checks succeeded, accept. Otherwise, reject.

## 3 Message Authentication Codes

In this section, we recall the definition of message authentication codes (MAC), and outline a general MAC scheme (which we call "abstract MAC"), which unifies several existing MAC scheme with a natural algebraic structure. Then, we introduce a stronger unforgeability notion for this abstract MAC scheme. In Appendix B, we prove that one of the MAC schemes of [13] does satisfy this security notion in the generic group model.

### 3.1 Definition

**Definition 1 (Message Authentication Code).** We recall the definition of a message authentication code. A message authentication code (MAC) $M$ is a quadruple of PPT algorithms $(M.\mathsf{Setup}, M.\mathsf{KeyGen}, M.\mathsf{Sign}, M.\mathsf{Verify})$, such that

- $M.\mathsf{Setup}(1^\lambda)$ generates the public parameters $\mathsf{pp}$ of the MAC. We assume that $\mathsf{pp}$ specifies the tag space $\mathcal{S}$ and the message space $\mathcal{M}$;
- $M.\mathsf{KeyGen}(\mathsf{pp})$ generates and outputs a key $\mathsf{sk}$ and optionally public issuer parameters $\mathsf{ipp}$.
- $M.\mathsf{Sign}_\mathsf{sk}(m)$ given the message $m \in \mathcal{M}$, outputs a tag $\sigma$;
- $M.\mathsf{Verify}_\mathsf{sk}(m, \sigma)$ given the message $m \in \mathcal{M}$ and a tag $\sigma \in \mathcal{S}$, outputs a bit $b$ whose value depends on the validity of the tag $\sigma$ with respect to $m$.

We assume for simplicity that once generated, the public parameters $\mathsf{pp}$ are implicitly passed as an argument to the algorithms $(M.\mathsf{KeyGen}, M.\mathsf{Sign}, M.\mathsf{Verify})$.

**Definition 2 (Correctness of a MAC).** A Message Authentication Code M is *correct* if for any $\mathsf{pp} \xleftarrow{\$} M.\mathsf{Setup}(1^\lambda)$, any $\mathsf{sk} \xleftarrow{\$} M.\mathsf{KeyGen}(\mathsf{pp})$, any message $m \in \mathcal{M}$ and for $\sigma \xleftarrow{\$} M.\mathsf{Sign}_\mathsf{sk}(m)$, it holds that $M.\mathsf{Verify}_\mathsf{sk}(m, \sigma) = 1$.

**Definition 3 (UF-CMVA Security of a MAC).** A MAC $M$ is UF-CMVA *secure* if for any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\left[\begin{array}{l} Q \leftarrow \emptyset, \mathsf{pp} \xleftarrow{\$} M.\mathsf{Setup}(1^\lambda), \\ \mathsf{sk} \xleftarrow{\$} M.\mathsf{KeyGen}(\mathsf{pp}), \\ (m,\sigma) \xleftarrow{\$} \mathscr{A}^{\mathcal{O}_{\mathsf{sk}}[Q]}(\mathsf{pp}) \end{array} : M.\mathsf{Verify}_{\mathsf{k}}(m,\sigma) = 1 \wedge m \notin Q\right] \leq \frac{1}{2} + \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$. $\mathscr{A}$ has access to an oracle $\mathcal{O}_{\mathsf{sk}}[Q]$ which answers to verification and signing queries:

- $\mathcal{O}.\mathsf{Sign}(m)$ sets $Q \leftarrow Q \cup \{m\}$ and outputs $M.\mathsf{Sign}_{\mathsf{sk}}(m)$;
- $\mathcal{O}.\mathsf{Verify}(m,\sigma)$ outputs $M.\mathsf{Verify}_{\mathsf{sk}}(m,\sigma)$.

In this paper we will need algebraic MACs which means that the signing and verification algorithms require only group operations to be performed.

### 3.2 An Abstract MAC Scheme

Let $\mathbb{G}$ be an abelian group of order $\mathsf{n}$. Given a vector $\boldsymbol{x} = (x_0, \cdots, x_\beta)$ for some integer $\beta$, we denote by $H_{\boldsymbol{x}} : \mathbb{Z}_{\mathsf{n}}^{\beta} \mapsto \mathbb{Z}_{\mathsf{n}}$ the affine function which, on input $(m_1, \cdots, m_\beta)$, outputs $x_0 + \sum_{i=1}^{\beta} x_i \cdot m_i$. Consider now the following generic MAC scheme, parametrized with integers $(\alpha, \beta)$:

- $M.\mathsf{Setup}(1^\lambda)$ : pick a generator $G$ of $\mathbb{G}$ and output $\mathsf{pp} \leftarrow (\mathbb{G}, \mathsf{n}, G, \alpha, \beta)$;
- $M.\mathsf{KeyGen}(\mathsf{pp})$ : pick $\alpha$ vectors $(\mathbf{k}_i)_{i\leq\alpha} \in (\mathbb{Z}_{\mathsf{n}}^{\beta+1})^\alpha$ (which can be either random or fixed) of length $\beta+1$, and $\alpha$ random group elements $(G_i)_{i\leq\alpha} \xleftarrow{\$} \mathbb{G}^\alpha$. Set $H_{i,j} \leftarrow \mathsf{k}_{i,j}^{-1} \bullet G$ for $i \in [1..\alpha], j \in [1..\beta]$, $G'_i \leftarrow \mathsf{k}_{i,0} \bullet G_i$ for $i \in [1..\alpha]$, and $\mathsf{ipp} \leftarrow ((H_{i,j})_{1\leq j\leq\beta})_{i\leq\alpha}, (G_i, G'_i)_{i\leq\alpha})$. Output $\mathsf{sk} = (\mathbf{k}_i)_{i\leq\alpha}$ and $\mathsf{ipp}$.
- $M.\mathsf{Sign}_{\mathsf{sk}}(\boldsymbol{m})$ : given a message $\boldsymbol{m} = (m_1, .., m_\beta) \in \mathbb{Z}_{\mathsf{n}}^{\beta}$, pick a random group element $U \xleftarrow{\$} \mathbb{G}$ and output
$$\sigma \leftarrow (U, (H_{\mathbf{k}_i}(\boldsymbol{m}) \bullet U)_{i\leq\alpha}).$$
- $M.\mathsf{Verify}_{\mathsf{sk}}(\boldsymbol{m}, \sigma)$ : parse $\sigma$ as $(U, (V_i)_{i\leq\alpha})$ and check that for $i = 1$ to $\alpha$, $V_i = H_{\mathbf{k}_i}(\boldsymbol{m}) \bullet U$.

**Example 1.** The scheme $\mathsf{MAC}_{\mathsf{GGM}}$ from [13] is obtained by setting $\alpha = 1$, and sampling the key $\mathbf{k}$ uniformly at random. This scheme is UF-CMVA-secure in the generic group model. Similarly, we recover the scheme $\mathsf{MAC}_{\mathsf{DDH}}$ from [13] by setting $\alpha = 3$, sampling $\mathbf{k}_1, \mathbf{k}_2$ at random, and setting $\mathbf{k}_3 \leftarrow (\mathsf{k}_{3,0}, 0, \cdots, 0)$ for a uniformly random $\mathsf{k}_{3,0}$. This scheme is UF-CMVA-secure under the decisional Diffie-Hellman assumption.

Note that for our construction of an anonymous credential scheme, we will require the security of the underlying MAC scheme to hold in a group of composite order. In Appendix B, we slightly modify $\mathsf{MAC}_{\mathsf{GGM}}$ and prove that the modified version is secure in non-prime order groups in the generic group model.

### 3.3 Extended Unforgeability

The UF-CMVA security property states that no PPT adversary should be able to forge a MAC on a message, even given access to signing and verification oracles, as long as this message was never queried to the signing oracle. One can consider stronger notions of unforgeability, where the adversary is given access to an additional oracle. In particular, it will be useful in our setting to consider the following *extended unforgeability* property for the abstract MAC scheme defined above:

**Definition 4 (Extended Unforgeability).** An abstract MAC $M$ is XUF-CMVA *secure* if for any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\left[\begin{array}{l} Q \leftarrow \emptyset, \mathsf{pp} \xleftarrow{\$} M.\mathsf{Setup}(1^\lambda), \\ \mathsf{sk} \xleftarrow{\$} M.\mathsf{KeyGen}(\mathsf{pp}), \\ (m,\sigma) \xleftarrow{\$} \mathscr{A}^{\mathcal{O}_{\mathsf{sk}}[Q]}(\mathsf{pp}) \end{array} : M.\mathsf{Verify}_{\mathsf{sk}}(m,\sigma) = 1 \wedge m \notin Q\right] \leq \frac{1}{2} + \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$. $\mathscr{A}$ has access to an oracle $\mathcal{O}_{\mathsf{sk}}[Q]$ which answers to verification and signing queries, as well as another specific type of query:

- $\mathcal{O}.\mathsf{Sign}(m)$ sets $Q \leftarrow Q \cup \{m\}$ and outputs $M.\mathsf{Sign}_{\mathsf{sk}}(m)$;
- $\mathcal{O}.\mathsf{Verify}(m, \sigma)$ outputs $M.\mathsf{Verify}_{\mathsf{sk}}(m, \sigma)$;
- $\mathcal{O}.\mathsf{Check}((A_{i,j})_{i\leq\alpha,j\leq\beta}, (B_{i,j})_{i\leq\alpha,j\leq\beta})$ checks $\sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet A_{i,j} = \sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet B_{i,j}$ for all $i \leq \alpha$, and outputs 1 iff all checks succeed (note: the $\mathcal{O}.\mathsf{Check}$ oracle could equivalently check whether $\sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet A_{i,j} = 0$).

In Appendix B, we will prove that the scheme $\mathsf{MAC}_{\mathsf{GGM}}$ from [13], which was proven UF-CMVA-secure over prime order groups in the generic group model in [13], is in fact XUF-CMVA-*secure* in the generic group model over *composite order groups* (the use of groups of composite order is required for compatibility of the MAC scheme with the DVNIZK scheme of [11]), under the computational subgroup assumption. Note that, while it is uncommon to prove security in the GGM under an additional assumption, it is necessary here: there exists an explicit attack against the security of the MAC if the adversary is able to compute a generator of a strict subgroup of $\mathbb{G}$. However, in the usual formulation of the GGM, the adversary is unbounded and receives as input the order of the group, hence he can trivially factor this order and efficiently compute generators of strict subgroups of $\mathbb{G}$, showing that $\mathsf{MAC}_{\mathsf{GGM}}$ is in fact *not* unconditionally secure in the GGM over composite order groups.

## 4 Non-Interactive Keyed-Verification Anonymous Credentials

In this section, we formally introduce non-interactive keyed-verification anonymous credentials and their security properties. Our definition mostly follows the blueprint of [13].

**Definition 5 (Non-Interactive Keyed-Verification Anonymous Credentials).** An non-interactive keyed-verification anonymous credentials (NIKVAC) scheme $\Theta$ is a set of algorithms $(\Theta.\mathsf{Setup}, \Theta.\mathsf{CredKeyGen}, \Theta.\mathsf{BlindIssue}, \Theta.\mathsf{BlindObtain}, \Theta.\mathsf{Show}, \Theta.\mathsf{ShowVerify})$ such that

- $\Theta.\mathsf{Setup}(1^\lambda)$, outputs the public parameters $\mathsf{pp}$ of the AC and a trapdoor $\mathsf{td}$, the public parameters fix the set of supported statements $\mathbf{\Phi}$, the universe of attributes $\mathcal{U}$ and are passed to the following algorithms implicitly, the trapdoor can be used to revoke anonymity;
- $\Theta.\mathsf{CredKeyGen}(\mathsf{pp})$, generates a secret key $\mathsf{sk}$ and public issuer parameters $\mathsf{ipp}$ for an issuing organization;
- $\Theta.\mathsf{BlindIssue}(\mathsf{sk}, S) \leftrightarrow \Theta.\mathsf{BlindObtain}(\mathsf{ipp}, (m_1, ..., m_l))$, interactively generates a credential $\mathsf{cred}$ for the attributes $(m_1, .., m_l) \in \mathcal{U}$, where $S \subset \{m_1, .., m_l\}$ (here, $S$ refers to the subset of attributes that the user wants to keep private; it allows to flexibly choose which attributes should be revealed, and which should not);
- $\Theta.\mathsf{Show}(\mathsf{ipp}, \mathsf{cred}, (m_1, ..., m_l), \Phi)$, outputs a proof of possession $\pi$ of a credential $\mathsf{cred}$ for organization with issuer parameters $\mathsf{ipp}$ in respect to the attributes $(m_1, .., m_l) \in \mathcal{U}$ with associated statements $\Phi \in \mathbf{\Phi}$;
- $\Theta.\mathsf{ShowVerify}(\mathsf{sk}, \pi, \Phi)$, checks the proof $\pi$ with $\mathsf{sk}$ with respect to the statements $\Phi \in \mathbf{\Phi}$;

which satisfies the *correctness*, *anonymity*, *unforgeability*, *blind issuance* and *key-parameter consistency* properties defined below.

We define two extra algorithms to simplify the security definitions:

- $\mathsf{Issue}(\mathsf{sk}, (m_1, .., m_l))$: generates a credential for the attributes $(m_1, .., m_l)$ using $\mathsf{sk}$;
- $\mathsf{CredVerify}(\mathsf{sk}, (m_1, .., m_l), \mathsf{cred})$: verifies the credential $\mathsf{cred}$ using $\mathsf{sk}$.

Here we define correctness, which guarantees that $\mathsf{Issue}$ always outputs proper credentials and that a proof of possession for a valid credential verifies correctly.

**Definition 6 (Correctness).** A NIKVAC scheme $\Theta$ is *correct* if it holds that

$$\Pr\left[\begin{array}{ll}(\mathsf{pp}, \mathsf{td}) \xleftarrow{\$} \Theta.\mathsf{Setup}(1^\lambda), (m_1, .., m_l) \xleftarrow{\$} \mathcal{U}, & \\ (\mathsf{sk}, \mathsf{ipp}) \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(1^\lambda), & \\ \mathsf{cred} \xleftarrow{\$} \mathsf{Issue}(\mathsf{sk}, (m_1, .., m_l)), & : b = 1 \\ b \xleftarrow{\$} \mathsf{CredVerify}(\mathsf{sk}, (m_1, .., m_l), \mathsf{cred}) & \end{array}\right] = 1$$

and

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\mathsf{td}) \xleftarrow{\$} \Theta.\mathsf{Setup}(1^\lambda),\\ \Phi \in \boldsymbol{\Phi}, (m_1,..,m_l) \xleftarrow{\$} \mathcal{U} \text{ with } \Phi(m_1,..,m_l)=1,\\ (\mathsf{sk},\mathsf{ipp}) \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(1^\lambda),\\ \mathsf{cred} \xleftarrow{\$} \mathsf{Issue}(\mathsf{sk},(m_1,..,m_l)),\\ \pi \xleftarrow{\$} \Theta.\mathsf{Show}(\mathsf{ipp},\mathsf{cred},(m_1,...,m_l),\Phi)\\ b \xleftarrow{\$} \Theta.\mathsf{ShowVerify}(\mathsf{sk},\pi,\Phi),\end{array} : b=1\right] = 1$$

Unforgeability ensures that users cannot successfully show credentials without having received one from the authority.

**Definition 7 (Unforgeability).** A NIKVAC scheme $\Theta$ is *unforgeable* if for any PPT adversary $\mathscr{A}$ it holds that

$$\Pr\left[\begin{array}{l}\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda), Q \leftarrow \emptyset,\\ (\mathsf{sk},\mathsf{ipp}) \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(1^\lambda),\\ (\Phi,\pi) \xleftarrow{\$} \mathscr{A}^{\mathcal{O}_{\mathsf{sk}}[Q]}(\mathsf{pp},\mathsf{ipp}),\\ b \xleftarrow{\$} \Theta.\mathsf{ShowVerify}(\mathsf{sk},\pi,\Phi),\end{array} : \begin{array}{l} b=1 \wedge \forall (m_1,..,m_l) \in Q :\\ \Phi(m_1,..,m_l)=0\end{array}\right] \leq \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$. $\mathscr{A}$ has access to an oracle $\mathcal{O}_{\mathsf{sk}}[Q]$ which answers to issuing and verification queries:

- $\mathcal{O}.\mathsf{Issue}(m_1,..,m_l)$ sets $Q \leftarrow Q \cup \{m_1,..,m_l\}$ and outputs $\mathsf{Issue}(\mathsf{sk},\mathsf{ipp},(m_1,..,m_l))$;
- $\mathcal{O}.\mathsf{Verify}(\Phi,\pi)$ outputs $\Theta.\mathsf{ShowVerify}(\mathsf{sk},\pi,\Phi)$.

Anonymity ensures that a user that shows a credential stays anonymous. Note that, as observed in [13], this simulation-style notion of anonymity implies in particular the standard notion of multi-show unlinkability, which states that anonymity is preserved throughout multiple presentations of the credential (a property which is not satisfied by e.g. U-Prove [31]).

**Definition 8 (Anonymity).** A NIKVAC scheme $\Theta$ is *anonymous* if for any PPT adversary $\mathscr{A}$, there exists a PPT simulator $\mathsf{Sim}$ such that it holds that

$$\left|\Pr\left[\begin{array}{l}(\mathsf{pp},\mathsf{td}) \xleftarrow{\$} \Theta.\mathsf{Setup}(1^\lambda),\\ (\mathsf{sk},\mathsf{ipp}) \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(1^\lambda),\\ (\Phi,\mathsf{cred},(m_1,..,m_l),\mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{pp},\mathsf{ipp},\mathsf{sk}),\\ \pi \xleftarrow{\$} \Theta.\mathsf{Show}(\mathsf{ipp},\mathsf{cred},(m_1,...,m_l),\Phi)\end{array} : \begin{array}{l}\mathsf{CredVerify}(\mathsf{sk},(m_1,..,m_l),\mathsf{cred})\\ =1 \wedge \Phi(m_1,..,m_l)=1 \wedge \mathscr{A}(\mathsf{st},\pi)=1\end{array}\right] -$$
$$\Pr\left[\begin{array}{l}(\mathsf{pp},\mathsf{td}) \xleftarrow{\$} \Theta.\mathsf{Setup}(1^\lambda),\\ (\mathsf{sk},\mathsf{ipp}) \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(1^\lambda),\\ (\Phi,\mathsf{cred},(m_1,..,m_l),\mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{pp},\mathsf{ipp},\mathsf{sk}),\\ \pi \xleftarrow{\$} \mathsf{Sim}(\mathsf{ipp},\mathsf{sk},\Phi)\end{array} : \begin{array}{l}\mathsf{CredVerify}(\mathsf{sk},(m_1,..,m_l),\mathsf{cred})\\ =1 \wedge \Phi(m_1,..,m_l)=1 \wedge \mathscr{A}(\mathsf{st},\pi)=1\end{array}\right]\Bigg| \leq \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

**Blind Issuance.** The protocol $\mathsf{BlindIssue} \leftrightarrow \mathsf{BlindObtain}$ defines a secure two-party protocol for the function $f((S,\mathsf{pp},\mathsf{ipp}),(\mathsf{sk},r),(m_1,..,m_l))$ for shared input $(S,\mathsf{pp},\mathsf{ipp})$, issuer input $(\mathsf{sk},r)$ and user input $(m_1,..,m_l)$ which returns $\mathsf{cred} \leftarrow \mathsf{Issue}(\mathsf{sk},(m_1,..,m_l);r)$ to the user, if the input is correct. Since we will not cover this property explicitly in this paper, refer to [13] for more details.

**Definition 9 (Key-Parameter Consistency).** A NIKVAC scheme $\Theta$ fulfills the *key-parameter consistency* property if for any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\mathsf{td}) \xleftarrow{\$} \Theta.\mathsf{Setup}(1^\lambda), \quad \text{for } i \in \{0,1\},\\ (\mathsf{ipp},\mathsf{sk}_0,\mathsf{sk}_1) \xleftarrow{\$} \mathscr{A}(\mathsf{pp}) \end{array} : (\mathsf{ipp},\mathsf{sk}_i) \in \{x \mid x \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(1^\lambda)\}\right] \leq \mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

### 4.1 Additional Properties

**Anonymity Revocation.** The following property would allow a trusted third party to revoke anonymity if desired.

**Definition 10 (Extractability).** A NIKVAC scheme $\Theta$ is *extractable* if there exists an efficient extractor $\mathsf{Ext}$ such that

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\mathsf{td}) \xleftarrow{\$} \Theta.\mathsf{Setup}(1^\lambda),\\ \Phi \in \mathbf{\Phi}, (m_1,..,m_l) \xleftarrow{\$} \mathcal{U},\\ (\mathsf{sk},\mathsf{ipp}) \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(1^\lambda), \\ \mathsf{cred} \xleftarrow{\$} \mathsf{Issue}(\mathsf{sk},(m_1,..,m_l)),\\ \pi \xleftarrow{\$} \Theta.\mathsf{Show}(\mathsf{ipp},\mathsf{cred},(m_1,...,m_l),\Phi)\end{array} : (m_1,..,m_l) \leftarrow \mathsf{Ext}(\mathsf{td},\pi)\right] = 1$$

## 5 Oblivious Designated-Verifier Non-Interactive Zero-Knowledge

In this section, we introduce oblivious (designated-verifier, non-interactive) zero-knowledge proof system. Intuitively, an oblivious DVNIZK enhances the security and the functionality of a DVNIZK with two properties:

- First, the oblivious DVNIZK on a word $x$ can be used to show knowledge of a witness $w$ such that $R_{\mathsf{sk}}(x,w)=1$, where $R_{\mathsf{sk}}$ is a *secret* witness relation, which depends on a secret information which is not known to the prover. The knowledge of $\mathsf{sk}$ is not required to generate a proof – but it is, obviously, necessary to verify the proof.
- Second, we consider words $x$ which can be divided in subwords $(x_0,x_1)$, such that $x_0$ is a *public subword*, while $x_1$ is a *private subword*. The privacy of $x_1$ is ensured by requiring, for the zero-knowledge property, the existence of a simulator which can simulate a proof without knowing the witness $w$ and/or $x_1$. Note that this formalism is mainly chosen for notational convenience: the word $x_1$ could always be thought of as being part of the witness. However, defining it as a part of the word allows us to set the secret relation $R_{\mathsf{sk}}$ to be exactly the MAC verification, where the word is the signature and the witness is the message, in our concrete instantiation.

### 5.1 Definition

**Definition 11 (Oblivious DVNIZK).** An oblivious designated-verifier non-interactive zero-knowledge proof of knowledge $\Pi$ for a family of *secret* witness relations $\{R_{\mathsf{crs}}(\cdot,\cdot,\cdot)\}_{\mathsf{crs}}$ (which take as input triples $(\mathsf{sk},x,w)$ where $\mathsf{sk}$ is a *secret relation key*, $x$ is a word, and $w$ is a witness for the relation $R_{\mathsf{crs}}(\mathsf{sk},\cdot,\cdot)$) is a five-tuple $(\Pi.\mathsf{Setup},\Pi.\mathsf{RelSetup},\Pi.\mathsf{KeyGen},\Pi.\mathsf{Prove},\Pi.\mathsf{Verify})$ of efficient algorithms such that

- $\Pi.\mathsf{Setup}(1^\lambda)$, on input the security parameter in unary, outputs a pair $(\mathsf{crs},\mathsf{td})$ where $\mathsf{crs}$ is a *common reference string* and $\mathsf{td}$ is a *trapdoor*;
- $\Pi.\mathsf{RelSetup}(\mathsf{crs})$, on input $\mathsf{crs}$, outputs a pair $(\mathsf{sk},\mathsf{ipp})$, where $\mathsf{sk}$ is a secret *relation key*, and $\mathsf{ipp}$ are public issuer parameters;
- $\Pi.\mathsf{KeyGen}(\mathsf{crs})$, on input $\mathsf{crs}$, outputs a pair $(\mathsf{pk},\mathsf{vk})$ where $\mathsf{pk}$ is a public *proving key*, and $\mathsf{vk}$ is a secret *verification key*;
- $\Pi.\mathsf{Prove}(\mathsf{crs},\mathsf{pk},\mathsf{ipp},(x_0,x_1),w)$, on input $\mathsf{crs}$, the public key $\mathsf{pk}$, the issuer parameters $\mathsf{ipp}$, a word $(x_0,x_1)$, where $x_0$ is a public subword and $x_1$ is a secret subword, and a witness $w$ such that $R_{\mathsf{crs}}(\mathsf{sk},(x_0,x_1),w)=1$, outputs a proof $\pi$;
- $\Pi.\mathsf{Verify}(\mathsf{crs},\mathsf{pk},\mathsf{ipp},x_0,\mathsf{vk},\mathsf{sk},\pi)$, on input $\mathsf{crs}$, the public key $\mathsf{pk}$, the issuer parameters $\mathsf{ipp}$, the public subword $x_0$, the verification key $\mathsf{vk}$, the secret relation key $\mathsf{sk}$, and a proof $\pi$, outputs a bit $b\in\{0,1\}$;

which satisfies the completeness, oblivious zero-knowledge, and oblivious knowledge-extractability properties defined below.

**Definition 12 (Completeness).** An oblivious $\mathsf{DVNIZK}$ proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{RelSetup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of secret witness relations $\{R_{\mathsf{crs}}\}_{\mathsf{crs}}$ satisfies *completeness* if for every $(\mathsf{crs}, \mathsf{td})$ in the image of $\Pi.\mathsf{Setup}(1^\lambda)$, every $(\mathsf{sk}, \mathsf{ipp})$ in the image of $\Pi.\mathsf{RelSetup}(\mathsf{crs})$, every $(\mathsf{pk}, \mathsf{vk}, \mathsf{sk})$ in the image of $\Pi.\mathsf{KeyGen}(\mathsf{crs})$, every $((x_0, x_1), w)$ such that $R_{\mathsf{crs}}(\mathsf{sk}, (x_0, x_1), w) = 1$, and every $\pi$ in the image of $\Pi.\mathsf{Prove}(\mathsf{pk}, \mathsf{ipp}, (x_0, x_1), w)$, it holds that $\Pi.\mathsf{Verify}(\mathsf{pk}, \mathsf{ipp}, x_0, \mathsf{vk}, \mathsf{sk}, \pi) = 1$.

**Definition 13 (Oblivious Zero-Knowledge).** An oblivious $\mathsf{DVNIZK}$ proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{RelSetup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of witness relations $\{R_{\mathsf{crs}}\}_{\mathsf{crs}}$ satisfies *oblivious zero-knowledge* if for any stateful PPT $\mathsf{Adv}$, there exists a probabilistic polynomial-time simulator $\mathsf{Sim}$ such that

$$\left| \Pr\left[\begin{array}{ll} (\mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda), & \\ (\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(\mathsf{crs}), & : \ (R_{\mathsf{crs}}(\mathsf{sk}, (x_0, x_1), w) = 1) \\ ((x_0, x_1), w, \mathsf{ipp}, \mathsf{sk}) \leftarrow \mathscr{A}(\mathsf{crs}, \mathsf{pk}, \mathsf{vk}), & \wedge (\mathscr{A}(\pi) = 1) \\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, (x_0, x_1), w), & \end{array}\right] - \right.$$

$$\left. \Pr\left[\begin{array}{ll} (\mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda), & \\ (\mathsf{pk}, \mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(\mathsf{crs}), & : \ (R_{\mathsf{crs}}(\mathsf{sk}, (x_0, x_1), w) = 1) \\ ((x_0, x_1), w, \mathsf{sk}, \mathsf{ipp}) \leftarrow \mathscr{A}(\mathsf{crs}, \mathsf{pk}, \mathsf{vk}), & \wedge (\mathscr{A}(\pi) = 1) \\ \pi \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, x_0, \mathsf{vk}, \mathsf{sk}), & \end{array}\right] \right| \leq \mu(\lambda)$$

where $\mu(\lambda) = \mathrm{negl}(\lambda)$.

**Definition 14 ($(\mathcal{O}_0, \mathcal{O}_1)$-Knowledge-Extractability).** An oblivious $\mathsf{DVNIZK}$ proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{RelSetup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of secret witness relations $\{R_{\mathsf{crs}}\}_{\mathsf{crs}}$ satisfies $(\mathcal{O}_0, \mathcal{O}_1)$-*knowledge-extractability* if the following two conditions hold:

- for every PPT adversary $\mathscr{A}$, there is an efficient extractor $\mathsf{Ext}$ such that

$$\Pr\left[\begin{array}{ll} (\mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda), & \\ (\mathsf{sk}, \mathsf{ipp}) \xleftarrow{\$} \mathsf{RelSetup}(\mathsf{crs}), & R_{\mathsf{crs}}(\mathsf{sk}, (x_0, x_1), w) = 0 \wedge \\ (\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(\mathsf{crs}), & : \Pi.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, x_0, \mathsf{vk}, \\ (\pi, x_0) \leftarrow \mathscr{A}^{\mathcal{V}, \mathcal{O}_0[\mathsf{sk}]}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}), & \mathsf{sk}, \pi) = 1 \\ (x_1, w) \leftarrow \mathsf{Ext}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, x_0, \mathsf{td}, \pi), & \end{array}\right] \approx 0,$$

  where $\mathcal{V}$ denotes $\Pi.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, \cdot, \mathsf{vk}, \mathsf{sk}, \cdot)$;
- there exists an efficient simulator that simulates the answers of $\Pi.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, \cdot, \mathsf{vk}, \mathsf{sk}, \cdot)$, which is not given $\mathsf{sk}$ but is instead given oracle access to $\mathcal{O}_1[\mathsf{sk}]$.

### 5.2 Instantiation

We now provide an instantiation of an oblivious $\mathsf{DVNIZK}$ suitable for our construction. At a high level, the secret witness relation we consider will be the one that checks, for triples $(\mathsf{sk}, x, w)$, that the message $w$ is the one signed in the credential $x$ (with respect to the secret key $\mathsf{sk}$ of the abstract MAC scheme defined in Section 3.1). Our construction heavily builds upon the $\mathsf{DVNIZK}$ proof system of [11]. Let $S = (S.\mathsf{KeyGen}, S.\mathsf{Enc}, S.\mathsf{Dec})$ denote a $\mathsf{DVNIZK}$-friendly encryption scheme with plaintext space $\mathbb{Z}_\mathsf{n}$ and $M = (M.\mathsf{Setup}, M.\mathsf{KeyGen}, M.\mathsf{Sign}, M.\mathsf{Verify})$ be a MAC scheme, which we assume to have the abstract structure given in Section 3.1, over a group $\mathbb{G}$ of order $\mathsf{n}$ with generator $G$. We will consider the following witness relation: $R_{\mathsf{crs}}(\mathsf{sk}, x, w)$, given as input a vector $x = (U, (V_i)_{i \leq \alpha}) \in \mathbb{G}^{\alpha+1}$ of group elements, a witness $w = (m_1, \cdots, m_\beta)$, and given $\mathsf{sk}$, checks that $M.\mathsf{Ver}_{\mathsf{sk}}(m_1, \cdots, m_\beta, x) = 1$, where $\mathsf{sk} = (\mathsf{k}_i)_{i \leq \alpha}$ is the MAC key. Since the purpose of the public word $x_0$ is mainly to allow more expressivity when considering a more complex relation, and we focus here on the most basic relation (the scheme can be enhanced to work with more complex relations), we simply consider that $x = x_1$ is entirely a secret word. The scheme works as follows:

- $\Pi.\mathsf{Setup}(1^\lambda)$ : compute $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$ and $\mathsf{pp} \xleftarrow{\$} M.\mathsf{Setup}(1^\lambda)$. Output $\mathsf{crs} \leftarrow (\mathsf{ek}, \mathsf{pp})$. Note that $\mathsf{ek}$ defines a plaintext space $\mathbb{Z}_\mathsf{n}$ and a random source $\mathbb{Z}_R$. As the $\mathsf{IND\text{-}CPA}$ and strong additive properties of $S$ require $R$ to be unknown, we assume that a bound $B$ on $R$ is publicly available. We denote $\ell \leftarrow 2^\lambda \mathsf{n} B$.

- $\Pi.\mathsf{RelSetup}(\mathsf{crs})$ : same as $M.\mathsf{KeyGen}$, namely: pick $\alpha$ vectors $(\mathsf{k}_i)_{i\leq\alpha} \in (\mathbb{Z}_{\mathsf{n}}^{\beta})^{\alpha}$ (which can be either random or fixed) of length $\beta$, and $\alpha$ random group elements $(G_i)_{i\leq\alpha} \xleftarrow{\$} \mathbb{G}^{\alpha}$. Set $H_{i,j} \leftarrow \mathsf{k}_{i,j}^{-1} \bullet G$ for $i \in [1..\alpha], j \in [1..\beta]$, $G'_i \leftarrow \mathsf{k}_{i,0} \bullet G_i$ for $i \in [1..\alpha]$, and $\mathsf{ipp} \leftarrow ((H_{i,j})_{1\leq j\leq\beta})_{i\leq\alpha}, (G_i, G'_i)_{i\leq\alpha})$. Output $\mathsf{sk} = (\mathsf{k}_i)_{i\leq\alpha}$ and $\mathsf{ipp}$.
- $\Pi.\mathsf{KeyGen}(\mathsf{crs})$ : pick $e \leftarrow \mathbb{Z}_{\ell}$, set $\mathsf{pk} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(0; e)$ and $\mathsf{vk} \leftarrow e$. Output $(\mathsf{pk}, \mathsf{vk})$.
- $\Pi.\mathsf{Prove}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, x, w)$ : given $x = (U, (V_i)_{i\leq\alpha})$ and a witness $w = \boldsymbol{m}$, pick $(\boldsymbol{m'}, \boldsymbol{t}, \boldsymbol{t'}) \xleftarrow{\$} \mathbb{Z}_{\mathsf{n}}^{\beta} \times (\mathbb{Z}_{\mathsf{n}}^{\alpha})^2$, $(\boldsymbol{r_m}, \boldsymbol{r_t}) \xleftarrow{\$} \mathbb{Z}_{2^{\lambda}B}^{\beta} \times \mathbb{Z}_{2^{\lambda}B}^{\alpha}$, $z \xleftarrow{\$} \mathbb{Z}_{\mathsf{n}}$. Let $(\boldsymbol{t'_j})_{j\leq\beta} = (t'_{1,j}, \cdots, t'_{\alpha,j})_{j\leq\beta}$ denote uniformly random additive shares of $\boldsymbol{t'}$ over $\mathbb{Z}_{\mathsf{n}}^{\alpha}$. Compute

$$
\begin{aligned}
&(U', (V'_i)_{i\leq\alpha}) \leftarrow (z \bullet U, ((z \bullet V_i) \mathbf{+} (t_i \bullet G))_{i\leq\alpha}) \\
&W_{i,j} \leftarrow m'_j \bullet U' \mathbf{+} (t'_{i,j} \bullet H_{i,j}) \text{ for } i \in [1..\alpha], j \in [1..\beta] \\
&(\boldsymbol{X_m}, \boldsymbol{X_t}) \leftarrow (S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m}; \boldsymbol{r_m}), S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{t}; \boldsymbol{r_t})), \\
&\boldsymbol{X'_m} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m'}; 0) \ominus (\boldsymbol{r_m} \odot \mathsf{pk}) = S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m'}; -e \cdot \boldsymbol{r_m}), \\
&\boldsymbol{X'_t} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{t'}; 0) \ominus (\boldsymbol{r_t} \odot \mathsf{pk}) = S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{t'}; -e \cdot \boldsymbol{r_t}),
\end{aligned}
$$

  and output $\pi \leftarrow (U', (V'_i)_{i\leq\alpha}, (W_{i,j})_{i\leq\alpha, j\leq\beta}, \boldsymbol{X_m}, \boldsymbol{X_t}, \boldsymbol{X'_m}, \boldsymbol{X'_t})$.
- $\Pi.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, \mathsf{vk}, \mathsf{sk}, \pi)$ : parse $\pi$ as

$$(U', (V'_i)_{i\leq\alpha}, (W_{i,j})_{i\leq\alpha, j\leq\beta}, \boldsymbol{X_m}, \boldsymbol{X_t}, \boldsymbol{X'_m}, \boldsymbol{X'_t}).$$

  Check that $e \odot \boldsymbol{X_m} \oplus \boldsymbol{X'_m}$ and $e \odot \boldsymbol{X_t} \oplus \boldsymbol{X'_t}$ are decodable, and decode them to vectors $\boldsymbol{d_m}, \boldsymbol{d_t}$. Reconstruct

$$(W'_i)_{i\leq\alpha} \leftarrow \left(\sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet W_{i,j}\right)_{i\leq\alpha}$$

  and check that

$$(e \bullet (V'_i \mathbf{-} (\mathsf{k}_{i,0} \bullet U')) \mathbf{+} W'_i)_{i\leq\alpha} = (\boldsymbol{d_m} \bullet (\mathsf{k}_{i,j} \bullet U')_{1\leq j\leq\beta})_{i\leq\alpha} \mathbf{+} \boldsymbol{d_t} \bullet G.$$

  Output 1 if and only if all checks succeeded.

**Theorem 2.** *The scheme $\Pi$ is an oblivious designated-verifier zero-knowledge proof of knowledge for the family of secret witness relations $\{R_{\mathsf{crs}}\}_{\mathsf{crs}}$, whose oblivious zero-knowledge property reduces to the semantic security of the* DVNIZK*-friendly encryption scheme $S$, and which satisfies statistical $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge extractability for the oracle $\mathcal{O}_0[\mathsf{sk}] \equiv M.\mathsf{Sign}_{\mathsf{sk}}$, and an oracle $\mathcal{O}_1[\mathsf{sk}]$ which is either*

- *$M.\mathsf{Verify}_{\mathsf{sk}}(\cdot,\cdot)$ if $\beta = 1$, or*
- *$M.\mathsf{Verify}_{\mathsf{sk}}(\cdot,\cdot)$ together with $M.\mathsf{Check}_{\mathsf{sk}}(\cdot,\cdot)$ otherwise.*

### 5.3 Extensions and Optimizations

In itself, the above oblivious DVNIZK does not seem to provide a strong unforgeability guarantee. Indeed, recall that the unforgeability of keyed-verification anonymous credential states (informally) that it should be infeasible to come up with a pair $(\boldsymbol{m}, \sigma)$ such that $M.\mathsf{Ver}_{\mathsf{sk}}(\boldsymbol{m}, \sigma) = 1$ and $\Phi(\boldsymbol{m}) = 1$, if all previous queries to the signing authority where on vectors $\boldsymbol{m'}$ such that $\Phi(\boldsymbol{m'}) = 0$. The exact choice of $\Phi$ depends on the particular application; typically, $\Phi(\boldsymbol{m})$ could correspond to the statement that $\boldsymbol{m}$ is the value committed in some *pseudonym* known to the verifier; that way, the condition "all previous queries to the signing authority where on vectors $\boldsymbol{m'}$ such that $\Phi(\boldsymbol{m'}) = 0$" boils down to the standard guarantee of anonymous credentials: it should be infeasible to come up with an accepting credential on a vector that was never signed before by the authority. But $\Phi$ can also check a more complex statement on the vector of attributes (e.g. it could check that the attribute "age" is above 18).

In the construction given above, we directly focus on enforcing $M.\mathsf{Ver}_{\mathsf{sk}}(\boldsymbol{m}, \sigma) = 1$; there is no additional $\Phi$ to, for example, bind $\boldsymbol{m}$ to a commitment. However, we observe that this typical choice of $\Phi$ is *for free* in our construction above. Indeed, a proof $\pi$ does contain, by construction,

a perfectly binding commitment (in fact, an encryption with $S$) of the vector $\boldsymbol{m}$, which is $\boldsymbol{X_m}$. Furthermore, it will immediately follow from the security analysis that the proof does not only guarantee the knowledge of a witness $w = \boldsymbol{m}$ (recovered by the extractor): it further guarantees that this witness is exactly the one encrypted in $\boldsymbol{X_m}$. Therefore, to bind the user to a pseudonym known to the verifier, it is unecessary to add a commitment to $\boldsymbol{m}$. Instead, the user can simply compute $(\boldsymbol{X_m}, \boldsymbol{X_t}, \boldsymbol{X'_m}, \boldsymbol{X'_t})$ in advance (observe that this does not require the knowledge of a credential) and send it to the verifier, who will simply define it to be the pseudonym of the user. Then, each time he wants to show possession of a credential $(U, (V_i)_{i\leq\alpha})$, the user only needs to compute the *missing part* of the proof, $(U', (V'_i)_{i\leq\alpha}, (W_{i,j})_{i\leq\alpha, j\leq\beta})$. This significantly reduces the size of a proof of possession, and in scenario where $\Phi$ is only intended to check that the vector matches with a pseudonym, the basic construction suffices as is. Of course, it can be extended to more complex statements $\Phi$, as long as they fit in the framework of statements handled by [11].

### 5.4 Security Analysis

Completeness follows from a straightforward (although tedious) inspection. We now establish oblivious zero-knowledge and $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge extractability.

**Oblivious Zero-Knowledge.** We exhibit a simulator $\mathsf{Sim}$ which simulates a proof $\pi$ given $(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, x_0, \mathsf{vk} = e, \mathsf{sk} = (\mathbf{k}_i)_{i\leq\alpha})$. Note that $\mathsf{Sim}$ is not given the witness $w$ nor $x_1$. The simulator $\mathsf{Sim}$ proceeds as follows:

Pick $(\tilde{\boldsymbol{m}}, \boldsymbol{m'}, \boldsymbol{d_m}, \boldsymbol{t}, \boldsymbol{t'}, \boldsymbol{d_t}) \xleftarrow{\$} (\mathbb{Z}_{\mathsf{n}}^{\beta})^3 \times (\mathbb{Z}_{\mathsf{n}}^{\alpha})^3$, $(\boldsymbol{r_m}, \boldsymbol{r_t}) \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^{\beta} \times \mathbb{Z}_{2^\lambda B}^{\alpha}$. Let $(\boldsymbol{t'_j})_{j\leq\beta} = (t'_{1,j}, \cdots, t'_{\alpha,j})_{j\leq\beta}$ denote uniformly random additive shares of $\boldsymbol{t'}$ over $\mathbb{Z}_{\mathsf{n}}^{\alpha}$. Compute

$$
\begin{aligned}
&(U', (V'_i)_{i\leq\alpha}) \xleftarrow{\$} \mathbb{G}^{\alpha+1}\\
&(W'_i)_{i\leq\alpha} \leftarrow (\boldsymbol{d_m} \bullet (\mathsf{k}_{i,j} \bullet U')_{1\leq j\leq\beta})_{i\leq\alpha} \boldsymbol{+} \boldsymbol{d_t} \bullet G \boldsymbol{-} e \bullet (V'_i \boldsymbol{-} (\mathsf{k}_{i,0} \bullet U'))_{i\leq\alpha}\\
&(\boldsymbol{X_m}, \boldsymbol{X_t}) \leftarrow (S.\mathsf{Enc}_{\mathsf{ek}}(\tilde{\boldsymbol{m}}; \boldsymbol{r_m}), S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{t}; \boldsymbol{r_t})),\\
&(\boldsymbol{X'_m}, \boldsymbol{X'_t}) \leftarrow (S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{d_m} - e \cdot \tilde{\boldsymbol{m}}; -e \cdot \boldsymbol{r_m}), S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{d_t} - e \cdot \boldsymbol{t}; -e \cdot \boldsymbol{r_t})).
\end{aligned}
$$

Then, for $i \in [1..\alpha], j \in [1..\beta]$, pick random $W_{i,j}$ conditioned on

$$W'_i = \sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet W_{i,j},$$

and output $\pi \leftarrow (U', (V'_i)_{i\leq\alpha}, (W_{i,j})_{i\leq\alpha,j\leq\beta}, \boldsymbol{X_m}, \boldsymbol{X_t}, \boldsymbol{X'_m}, \boldsymbol{X'_t})$.

We now show how to use an adversary $\mathsf{Adv}$ which outputs $((x_0, x_1), w, \mathsf{ipp}, \mathsf{sk})$ and distinguishes $\pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, (x_0, x_1), w)$ from $\pi \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, , x_0, \mathsf{vk}, \mathsf{sk})$ conditioned on $R_{\mathsf{crs}}(\mathsf{sk}, (x_0, x_1), w) = 1$ to break the semantic security of $S$. The reduction obtains $\boldsymbol{m}$ from $\mathsf{Adv}$, samples a random $\tilde{\boldsymbol{m}}$, and sends $(\boldsymbol{m}, \tilde{\boldsymbol{m}})$ to a challenger for the $\mathsf{IND\text{-}CPA}$ game of $S$. It receives a ciphertext $\boldsymbol{X_m}$. It samples $(\boldsymbol{m'}, \boldsymbol{d_m}, \boldsymbol{t}, \boldsymbol{t'}, \boldsymbol{d_t}) \xleftarrow{\$} (\mathbb{Z}_{\mathsf{n}}^{\beta})^2 \times (\mathbb{Z}_{\mathsf{n}}^{\alpha})^3$, $\boldsymbol{r_t} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^{\alpha}$ as before, and sets $\boldsymbol{X'_m} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{d_m}; 0) \ominus \boldsymbol{X_m} \odot e$. Finally, it computes $(U', (V'_i)_{i\leq\alpha})$, $(W_{i,j})_{i\leq\alpha,j\leq\beta}$, and $(\boldsymbol{X_t}, \boldsymbol{X'_t})$ as before. Observe that $(U', (V'_i)_{i\leq\alpha})$ are distributed identically in the real game and the simulated game; direct calculations show that when $\boldsymbol{X_m}$ encrypts $\boldsymbol{m}$, the proof $\pi$ is distributed exactly as in the real game, while when $\boldsymbol{X_m}$ encrypts $\tilde{\boldsymbol{m}}$, the proof $\pi$ is distributed exactly as in the simulated game.

**$(\mathcal{O}_0, \mathcal{O}_1)$-Knowedge-Extractability.** We now turn our attention to the $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge extractability property. The extractor $\mathsf{Ext}$ proceeds as follows: given a proof $\pi = (U', (V'_i)_{i\leq\alpha}, (W_{i,j})_{i\leq\alpha,j\leq\beta}, \boldsymbol{X_m}, \boldsymbol{X_t}, \boldsymbol{X'_m}, \boldsymbol{X'_t})$, it computes $\boldsymbol{m} \leftarrow S.\mathsf{Dec}_{\mathsf{td}}(\boldsymbol{X_m})$, $\boldsymbol{t} \leftarrow S.\mathsf{Dec}_{\mathsf{td}}(\boldsymbol{X_t})$, and outputs $x \leftarrow (U', (V'_i \boldsymbol{-} t_i \bullet G)_{i\leq\alpha})$, and $w \leftarrow \boldsymbol{m}$. We now analyze the probability that $R_{\mathsf{crs}}(\mathsf{sk}, (x_0, x_1), w) = 0 \wedge \Pi.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, x_0, \mathsf{vk}, \mathsf{sk}, \pi) = 1$. To do so, we proceed in two steps:

**Game 1.** In this game, we modify the behavior of the oracle $\Pi.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}, \mathsf{vk}, \cdot, \mathsf{sk}, \cdot)$ that $\mathsf{Adv}$ is given access to. Namely, the oracle is not given $\mathsf{vk}$ anymore. Rather, we generate $\mathsf{vk}$ as before,

and set $e_R \leftarrow \mathsf{vk} \bmod R$. Each time $\mathsf{Adv}$ sends a query $\pi$ to the oracle, we proceed as follows: we parse $\pi$ as

$$\pi = (U', (V'_i)_{i\leq\alpha}, (W_{i,j})_{i\leq\alpha, j\leq\beta}, \boldsymbol{X_m}, \boldsymbol{X_t}, \boldsymbol{X'_m}, \boldsymbol{X'_t}),$$

and use $\mathsf{td}$ to decrypt $(\boldsymbol{X_m}, \boldsymbol{X_t}, \boldsymbol{X'_m}, \boldsymbol{X'_t})$, obtaining vectors $(\boldsymbol{m}, \boldsymbol{t}, \boldsymbol{m'}, \boldsymbol{t'})$. Then, we perform the following checks:

1. we check that $-e_R \odot (\boldsymbol{X_m} \ominus \boldsymbol{m}) = \boldsymbol{X'_m} \ominus \boldsymbol{m'}$;
2. we check that $-e_R \odot (\boldsymbol{X_t} \ominus \boldsymbol{t}) = \boldsymbol{X'_t} \ominus \boldsymbol{t'}$;
3. we check $V'_i \mathbin{-} t_i \bullet G = H_{\mathsf{k}_i}(\boldsymbol{m}) \bullet U'$ for every $i \leq \alpha$ (that is, we run $M.\mathsf{Ver}_{\mathsf{sk}}(\boldsymbol{m}, \sigma)$ on the $\mathsf{MAC}$ $\sigma = (U', (V'_i \mathbin{-} t_i \bullet G)_{i\leq\alpha})$);
4. we reconstruct $(W'_i)_{i\leq\alpha} \leftarrow \left(\sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet W_{i,j}\right)_{i\leq\alpha}$ and check $W'_i \mathbin{-} t'_i \bullet G = \sum_{j=1}^{\beta} (\mathsf{k}_{i,j} \cdot m'_j) \bullet U'$ for every $i \leq \alpha$.

Note that this follows exactly the proof strategy of [11, Section 3.3]. It follows by the exact same argument that it is statistically infeasible to distinguish the simulated oracle in Game 1 from the real oracle, and the distinguishing advantage is at most $Q(\alpha+1)p$, where $p$ is the smallest prime factor of $\mathsf{n}$ and $Q$ is the number of queries of $\mathsf{Adv}$ to the oracle. Intuitively, the argument stems from the fact that if $\mathsf{Adv}$ ever submits a proof that would be accepted by the oracle, but not by the simulated oracle (or the converse), then this proof information-theoretically determines $\mathsf{vk}$. However, even given $e_R = \mathsf{vk} \bmod R$, it follows from the chinese remainder theorem that the value $\mathsf{vk} \bmod \mathsf{n}$ remains *statistically hidden*, since $\mathsf{vk}$ was initially picked at random in $\mathbb{Z}_\ell$ and $\ell$ satisfies $\ell > 2^\lambda \mathsf{n} R$. Observe that game already suffices to establish that the probability of $R_{\mathsf{crs}}(\mathsf{sk}, (x_0, x_1), w) = 0 \wedge \Pi.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \mathsf{sk}, \pi) = 1$ must be negligible, since in this game the simulation of $\Pi.\mathsf{Verify}$ does in particular check that $R_{\mathsf{crs}}(\mathsf{sk}, (x_0, x_1), w) = 1$. However, the simulation of $\Pi.\mathsf{Verify}$ still uses $\mathsf{sk}$; to establish the second property of the $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge-extractability, we proceed with a second game.

**Game 2.** In this game, we further modify the simulated oracle, so that it does not use $\mathsf{sk}$ anymore. Instead, the simulation will itself rely on the $\mathsf{MAC}$ verification oracle. More precisely, the key $\mathsf{sk}$ is only used in the checks 3 and 4 of Game 1. The third check is straightforward given oracle access to $M.\mathsf{Verify}_{\mathsf{sk}}(\cdot,\cdot)$: just call $M.\mathsf{Verify}_{\mathsf{sk}}(\boldsymbol{m}, \sigma)$ with $\sigma = (U', (V'_i \mathbin{-} t_i \bullet G)_{i\leq\alpha})$ (this is perfectly identical to the third check in the previous game).

The fourth check, however, is more problematic, since it's not clear how to reconstruct the $(W'_i)_{i\leq\alpha}$ without knowing $\mathsf{sk}$. Rewriting a bit the fourth check, we need to check is

$$\sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet W_{i,j} \mathbin{-} t'_i \bullet G = \sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet (m'_j \bullet U')$$

for every $i \leq \alpha$. Letting $(t'_{i,1}, \cdots, t'_{i,\beta})$ denote an arbitrary additive sharing of $t'_i$ for every $i \leq \alpha$, this equation can be rewritten as

$$\sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet (W_{i,j} \mathbin{-} t'_{i,j} \bullet H_{i,j}) = \sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet (m'_j \bullet U')$$

Now, we distinguish two cases:

- **Case 1.** If it holds that $\beta = 1$, corresponding to the case where the vector of attributes has length 1 (or, equivalently, we consider a simplified scenario without attributes, and credentials computed directly on the identity of the user), then the equation becomes

  $$\mathsf{k}_{i,1} \bullet (m'_1 \bullet U' \mathbin{-} W_{i,1}) = t'_i \bullet G.$$

  Observe that this check can be performed efficiently: since we are given $H_{i,1} = \mathsf{k}_{i,1}^{-1} \bullet G$, this is perfectly equivalent to checking

  $$m'_1 \bullet U' \mathbin{-} W_{i,1} = t'_i \bullet H_{i,1}$$

  for every $i \leq \alpha$, which does not require the knowledge of $\mathsf{sk}$.

– **Case 2.** In the general case, where $\beta$ can be larger than 1, there is no immediate shortcut. In this case, we have to rely on a MAC with a stronger unforgeability property, the XUF-CMVA security property defined in Section 3, and we simulate the verification using the following two oracles:
  - $M.\mathsf{Verify}(m, \sigma)$ outputs $M.\mathsf{Verify}_{\mathsf{sk}}(m, \sigma)$;
  - $M.\mathsf{Check}((A_{i,j})_{i\leq\alpha,j\leq\beta}, (B_{i,j})_{i\leq\alpha,j\leq\beta})$ checks $\sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet A_{i,j} = \sum_{j=1}^{\beta} \mathsf{k}_{i,j} \bullet B_{i,j}$ for all $i \leq \alpha$, and outputs 1 iff all checks succeed,

  where the first oracle allows to check the third equation, and the second oracle allows to check the last equation.

In both cases, it is immediate to see that the answers of the simulated oracle are distributed exactly as in Game 1. Furthermore, the simulation only requires access to an oracle $\mathcal{O}_1[\mathsf{sk}]$, which is $M.\mathsf{Verify}$ in case 1, and the pair of oracles $M.\mathsf{Verify}, M.\mathsf{Check}$ in case 2.

# 6 A Construction of NIKVAC from Algebraic MAC and Oblivious DVNIZK

In this section, we will use the system introduced in section 5 to construct a NIKVAC scheme $\Theta$.

## 6.1 Construction

Let $M$ be a MAC and $\Phi$ a set of statements for attributes $m_1, .., m_l$. Let $\Pi_\Phi$ be an oblivious DVNIZK system which runs on a common Setup algorithm with $M$ for the relation $R_{\mathsf{crs}}$, for $\mathsf{crs} \xleftarrow{\$} M.\mathsf{Setup}(1^\lambda)$, defined as

$$R_{\mathsf{crs}}((x_0, x_1), (m_1, .., m_l), \mathsf{k}) = 1 \text{ iff } M.\mathsf{Verify}_{\mathsf{k}}((m_1, .., m_l), x_1) = 1 \wedge \Phi(m_1, .., m_l),$$

where $x_0$ is a public word needed to prove the statements $\Phi$ and $\Pi_\Phi.\mathsf{RelSetup} = M.\mathsf{KeyGen}$. We assume $\Pi_\Phi$ satisfies $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge-extractability, where $\mathcal{O}_0[\mathsf{k}](\cdot) = M.\mathsf{Sign}_{\mathsf{k}}(\cdot)$ and $\mathcal{O}_1[\mathsf{k}]$ is either the MAC verification oracle, if $M$ is UF-CMVA secure (and the attribute vectors are of length 1), or the MAC verification and additional check oracle, if $M$ is XUF-CMVA secure. Since $x_0$ depends on the choice of $\Phi$, we omit it entirely in the following and simply set $x = x_1$. Note that $\Pi_\Phi.\mathsf{Setup}$, $\Pi_\Phi.\mathsf{CredKeyGen}$ do not rely on the choice of $\Phi$, so we simply write $\Pi.\mathsf{Setup}, \Pi.\mathsf{CredKeyGen}$. We now construct a NIKVAC scheme using $\{\Pi_\Phi\}_\Phi$.

– $\Theta.\mathsf{Setup}(1^\lambda)$, outputs $(\mathsf{pp}, \mathsf{td}) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, we assume that $\mathsf{pp}$ fixes the supported statements $\boldsymbol{\Phi}$, the universe of attributes $\mathcal{U}$ is the message space of $M$;
– $\Theta.\mathsf{CredKeyGen}(\mathsf{pp})$, runs $(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(\mathsf{pp})$, $(\mathsf{k}, \mathsf{ipp}_M) \xleftarrow{\$} \Pi.\mathsf{RelSetup}(\mathsf{pp})$, outputs secret key $\mathsf{sk} \leftarrow (\mathsf{vk}, \mathsf{k})$ and issuer parameters $\mathsf{ipp} \leftarrow (\mathsf{pk}, \mathsf{ipp}_M)$, we assume that CredKeyGen satisfies key-parameter consistency;
– $\Theta.\mathsf{BlindIssue}(\mathsf{sk}, S) \leftrightarrow \Theta.\mathsf{BlindObtain}(\mathsf{ipp}, (m_1, ..., m_l))$, performs a secure two-party computation that issues a tag of $M$ to the user on valid input, we assume that this protocol satisfies *blind issuance* property[2];
– $\Theta.\mathsf{Show}(\mathsf{ipp}, \mathsf{cred}, (m_1, ..., m_l), \Phi)$, parses $\mathsf{ipp}$ as $(\mathsf{pk}, \mathsf{ipp}_M)$ outputs $\pi \xleftarrow{\$}$ $\Pi_\Phi.\mathsf{Prove}(\mathsf{pk}, \mathsf{ipp}_M, \mathsf{cred}, (m_1, ..., m_l))$;
– $\Theta.\mathsf{ShowVerify}(\mathsf{sk}, \pi, \Phi)$, parses $\mathsf{sk}$ as $(\mathsf{vk}, \mathsf{k})$ and $\mathsf{ipp}$ as $(\mathsf{pk}, \mathsf{ipp}_M)$, checks $\Pi_\Phi.\mathsf{Verify}(\mathsf{pk}, \mathsf{ipp}_M, \mathsf{vk}, \mathsf{k}, \pi)$.

## 6.2 Security Analysis

For $\Theta$, the functions Issue and CredVerify are defined as follows:

– $\mathsf{Issue}(\mathsf{sk}, (m_1, .., m_l))$: for $\mathsf{sk} = (\mathsf{vk}, \mathsf{k})$ outputs $M.\mathsf{Sign}_{\mathsf{k}}(m_1, .., m_l)$;
– $\mathsf{CredVerify}(\mathsf{sk}, (m_1, .., m_l), \mathsf{cred})$: for $\mathsf{sk} = (\mathsf{vk}, \mathsf{k})$ outputs $M.\mathsf{Verify}_{\mathsf{k}}((m_1, .., m_l), \mathsf{cred})$.

**Theorem 3 (Correctness).** *The NIKVAC scheme $\Theta$ satisfies correctness if $M$ is correct and $\Pi_\Phi$ is complete.*

[2] The protocol depends highly on the chosen MAC scheme. Thus, we omit details in abstract instantiation.

*Proof.* Let $(\mathsf{pp}, \mathsf{td}) \leftarrow \Theta.\mathsf{Setup}(1^\lambda)$, $(m_1, .., m_l) \xleftarrow{\$} \mathcal{U}$, $(\mathsf{sk}, \mathsf{ipp}) \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(\mathsf{pp})$ and $\mathsf{credIssue}(\mathsf{sk}, (m_1, .., m_l))$. It follows that $\mathsf{CredVerify}(\mathsf{sk}, (m_1, .., m_l), \mathsf{cred}) = 1$ from the correctness of the MAC scheme $M$.

Now, let $(\mathsf{pp}, \mathsf{td}) \leftarrow \Theta.\mathsf{Setup}(1^\lambda)$, $\Phi \xleftarrow{\$} \mathbf{\Phi}, (m_1, .., m_l) \xleftarrow{\$} \mathcal{U}$ with $\Phi(m_1, .., m_l) = 1$, $(\mathsf{sk} = (\mathsf{vk}, \mathsf{k}), \mathsf{ipp}) \xleftarrow{\$} \Theta.\mathsf{CredKeyGen}(\mathsf{pp})$ and $\mathsf{cred} \xleftarrow{\$} \mathsf{Issue}(\mathsf{sk}, (m_1, .., m_l))$. Let $\pi \xleftarrow{\$} \Theta.\mathsf{Show}(\mathsf{ipp}, \mathsf{cred}, (m_1, .., m_l), \Phi)$. Note that $R_\mathsf{k}(\mathsf{cred}, (m_1, .., m_l)) = 1$ and thus $\Theta.\mathsf{ShowVerify}(\mathsf{sk}, \pi, \Phi) = 1$ by the completeness of $\Pi_\Phi$.

**Theorem 4 (Unforgeability).** *The* NIKVAC *scheme* $\Theta$ *is unforgeable if* $M$ *is unforgeable and* $\Pi_\Phi$ *is* $(\mathcal{O}_0, \mathcal{O}_1)$*-knowledge-extractable for all* $\Phi \in \mathbf{\Phi}$*.*

*Proof.* Let $\mathscr{A}$ be a PPT adversary on the unforgeability of $\Theta$. We build an adversary $\mathscr{B}$ which either breaks the unforgeability of $M$ (so either the UF-CMVA or XUF-CMVA security) or the $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge-extractability of $\Pi_\Phi$ for some $\Phi \in \mathbf{\Phi}$.

$\mathscr{B}$ receives $(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}_M)$[3] and access to a proof verification oracle $\mathcal{V}_\Phi$ and an MAC issuing oracle $\mathcal{O}_0$ (defined in section 6.1) from the $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge-extractability[4] game with $\Pi_\Phi$ for $\Phi \in \mathbf{\Phi}$. $\mathscr{B}$ sends $\mathsf{pp} \leftarrow \mathsf{crs}, \mathsf{ipp} \leftarrow (\mathsf{pk}, \mathsf{ipp}_M)$ to $\mathscr{A}$ and gives access to the following issuing and verification oracle $\mathcal{O}$:

- $\mathcal{O}.\mathsf{Issue}(m_1, .., m_l)$ sets $Q \leftarrow Q \cup \{m_1, .., m_l\}$ and outputs $\mathcal{O}_0(m_1, .., m_l)$;
- $\mathcal{O}.\mathsf{Verify}(\Phi, \pi)$ outputs $\mathcal{V}_\Phi(\pi)$.

By the second property of definition 14, $\mathcal{V}_\Phi(\cdot)$ can be simulated only using $\mathcal{O}_1$ without access to the secret key. Now, all answers to queries which require the secret MAC key can be computed using solely access to the MAC oracles. Note that $\mathscr{B}$ simulates the unforgeability game of $\Theta$ with overwhelming probability. At some point, if $\mathscr{A}$ is successful, he will output $\pi, \Phi$ such that the pair $(\pi, \Phi)$ verifies correclty and for all queried $(m_1, .., m_l) \in Q : \Phi(m_1, .., m_l) = 0$. Subsequently, $\mathscr{B}$ forwards $\pi$ to the $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge-extractability game for $\Pi_\Phi$, which in turn forwards the extracted values $(x_1, w)$ to the MAC unforgeability game.

We now analyze the success probability of $\mathscr{B}$ assuming $\mathscr{A}$ is successful. If $\mathscr{B}$ won the $(\mathcal{O}_0, \mathcal{O}_1)$-knowledge-extractability game, we are finished. In the other case, the MAC unforgeability game receives $(x_1 = \sigma, w = (m_1, .., m_l))$. Because $\mathscr{A}$ is successful, $\pi$ verifies correctly with regards to $\Theta$ and thus also verifies correctly with regards to $\Pi_\Phi$. Because $\mathscr{B}$ failed the first game, it necessarily holds that $R(\sigma, (m_1, .., m_l)) = 1$. Since $\forall (m'_1, .., m'_l) \in Q : \Phi(m'_1, .., m'_l) = 0$ and $\Phi(m_1, .., m_l) = 1$, it holds that $(m_1, .., m_l) \notin Q$ and $\sigma$ verifies correctly. Thus, $\mathscr{B}$ breaks the unforgeability of $M$. □

**Theorem 5 (Anonymity).** *The* NIKVAC *scheme* $\Theta$ *is anonymous if* $\Pi_\Phi$ *satisfies oblivious zero-knowledge.*

*Proof.* Let $\mathscr{A}$ be an adversary on the anonymity of $\Theta$. We construct an adversary $\mathscr{B}$ that breaks the oblivious zero-knowledge property of $\Pi_\Phi$ for some $\Phi \in \mathbf{\Phi}$ with overwhelming probability if $\mathscr{A}$ is successful.

$\mathscr{B}$ receives $\mathsf{crs}, \mathsf{pk}, \mathsf{vk}$ from the zero-knowledge game with $\Pi_\Phi$ for some arbitrary $\Phi \in \mathbf{\Phi}$. Note that these values are independent of the particular choice of $\Phi$. $\mathscr{B}$ then runs $(\mathsf{k}, \mathsf{ipp}_M) \xleftarrow{\$} M.\mathsf{KeyGen}(\mathsf{crs})$ and sends $\mathsf{pp} \leftarrow \mathsf{crs}, \mathsf{ipp} \leftarrow (\mathsf{pk}, \mathsf{ipp}_M), \mathsf{sk} \leftarrow (\mathsf{vk}, \mathsf{k})$ to $\mathscr{A}$. In turn, $\mathscr{B}$ receives $(\Phi, \mathsf{cred}, (m_1, .., m_l))$ from $\mathscr{A}$. Next, $\mathscr{B}$ outputs $(\mathsf{cred}, (m_1, .., m_l), \mathsf{k}, \mathsf{ipp}_M)$ to the oblivious zero-knowledge game for the now fixed $\Phi$ and receives $\pi$ in return which he forwards to $\mathscr{A}$. Note that $\mathscr{B}$ simulates the anonymity game with overwhelming probability. Also, $R_\mathsf{crs}(\mathsf{sk}, \mathsf{cred}, (m_1, .., m_l)) = 1 \iff \mathsf{CredVerify}(\mathsf{sk}, (m_1, .., m_l), \mathsf{cred}) = 1 \wedge \Phi(m_1, .., m_l)$. The simulation of $\pi$ in the zero-knowledge game only uses $\mathsf{ipp}, \mathsf{sk}, \Phi$ and will thus be a simulation for the anonymity game. Otherwise, $\pi$ is built honestly in both games and thus, if $\mathscr{A}$ is successful, $\mathscr{B}$ is successful with overwhelming probability. □

[3] The parameters $(\mathsf{crs}, \mathsf{pk}, \mathsf{ipp}_M)$ are fixed for all $\Phi \in \mathbf{\Phi}$, since they do not depend on the particular choice of $\Phi$.

[4] In this proof, this refers to the first property of definition 14.

**Missing Properties.** The missing properties are *blind issuance* and *key-parameter consistency*. In practice, *key-parameter consistency* can easily be fulfilled by adding additional commitments to the components of the secret key and the two-party computation for *blind issuance* depends highly on the structure of the MAC scheme and can be implemented with any standard two party computation protocol; we briefly outline a possible candidate for an optimized version of our scheme in Appendix C.

## References


1. Acar, T., Nguyen, L.: Revocation for delegatable anonymous credentials. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 423–440. Springer, Heidelberg (Mar 2011)
2. Barki, A., Brunet, S., Desmoulins, N., Traoré, J.: Improved algebraic MACs and practical keyed-verification anonymous credentials. In: Avanzi, R., Heys, H.M. (eds.) SAC 2016. LNCS, vol. 10532, pp. 360–380. Springer, Heidelberg (Aug 2016)
3. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (Aug 2009)
4. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (Mar 2008)
5. Benhamouda, F., Couteau, G., Pointcheval, D., Wee, H.: Implicit zero-knowledge arguments and applications to the malicious setting. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 107–129. Springer, Heidelberg (Aug 2015)
6. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (Mar 2009)
7. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. Cryptology ePrint Archive, Report 2001/019 (2001), `http://eprint.iacr.org/2001/019`
8. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (Aug 2002)
9. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (Aug 2004)
10. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC. pp. 209–218. ACM Press (May 1998)
11. Chaidos, P., Couteau, G.: Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 193–221. Springer, Heidelberg (Apr / May 2018)
12. Chaidos, P., Groth, J.: Making sigma-protocols non-interactive without random oracles. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 650–670. Springer, Heidelberg (Mar / Apr 2015)
13. Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic MACs and keyed-verification anonymous credentials. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 14. pp. 1205–1216. ACM Press (Nov 2014)
14. Chaum, D.: Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In: Pichler, F. (ed.) EUROCRYPT'85. LNCS, vol. 219, pp. 241–244. Springer, Heidelberg (Apr 1986)
15. Corrigan-Gibbs, H., Kogan, D.: The discrete-logarithm problem with preprocessing. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 415–447. Springer, Heidelberg (Apr / May 2018)
16. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., shelat, a., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (Dec 2007)
17. Damgård, I.: Payment systems and credential mechanisms with provable security against abuse by individuals. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 328–335. Springer, Heidelberg (Aug 1990)
18. Damgård, I., Fazio, N., Nicolosi, A.: Non-interactive zero-knowledge from homomorphic encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 41–59. Springer, Heidelberg (Mar 2006)
19. Damgård, I., Jurik, M., Nielsen, J.B.: A generalization of paillier's public-key system with applications to electronic voting. International Journal of Information Security 9(6), 371–385 (2010)

20. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. In: 40th FOCS. pp. 523–534. IEEE Computer Society Press (Oct 1999)
21. Galbraith, S.D., Ruprai, R.S.: Using equivalence classes to accelerate solving the discrete logarithm problem in a short interval. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 368–383. Springer, Heidelberg (May 2010)
22. Garman, C., Green, M., Miers, I.: Decentralized anonymous credentials. In: NDSS 2014. The Internet Society (Feb 2014)
23. Gennaro, R.: An improved pseudo-random generator based on discrete log. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 469–481. Springer, Heidelberg (Aug 2000)
24. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th FOCS. pp. 102–115. IEEE Computer Society Press (Oct 2003)
25. Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 491–511. Springer, Heidelberg (Dec 2014)
26. Izabachène, M., Libert, B., Vergnaud, D.: Block-wise P-signatures and non-interactive anonymous credentials with efficient attributes. In: Chen, L. (ed.) 13th IMA International Conference on Cryptography and Coding. LNCS, vol. 7089, pp. 431–450. Springer, Heidelberg (Dec 2011)
27. Koshiba, T., Kurosawa, K.: Short exponent diffie-hellman problems. In: International Workshop on Public Key Cryptography. pp. 173–186. Springer (2004)
28. Lipmaa, H.: Optimally sound sigma protocols under DCRA. Cryptology ePrint Archive, Report 2017/703 (2017), `http://eprint.iacr.org/2017/703`
29. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (Aug 1999)
30. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (May 1999)
31. Paquin, C., Zaverucha, G.: U-Prove cryptographic specification V1.1 (revision 2), `www.microsoft.com/uprove` (2013)
32. Patel, S., Sundaram, G.S.: An efficient discrete log pseudo random generator. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 304–317. Springer, Heidelberg (Aug 1998)
33. Sadiah, S., Nakanishi, T., Funabiki, N.: Anonymous credential system with efficient proofs for monotone formulas on attributes. In: Tanaka, K., Suga, Y. (eds.) IWSEC 15. LNCS, vol. 9241, pp. 262–278. Springer, Heidelberg (Aug 2015)
34. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM (JACM) 27(4), 701–717 (1980)
35. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 256–266. Springer (1997)
36. van Oorschot, P.C., Wiener, M.J.: On Diffie-Hellman key agreement with short exponents. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 332–343. Springer, Heidelberg (May 1996)

# A Classical Preliminaries

## A.1 Preliminaries on Encryption Schemes

**Definition 15.** (Public-Key Encryption Scheme) A public-key encryption scheme $S$ is a triple of PPT algorithms $(S.\mathsf{KeyGen}, S.\mathsf{Enc}, S.\mathsf{Dec})$, such that

- $S.\mathsf{KeyGen}(1^\lambda)$, generates a pair $(\mathsf{ek}, \mathsf{dk})$, $\mathsf{ek}$ is the public encryption key and $\mathsf{ek}$ is the secret decryption key. We assume that $\mathsf{ek}$ specifies the ciphertext space $\mathcal{C}$, the message space $\mathcal{M}$, and the random source $\mathcal{R}$;
- $S.\mathsf{Enc}_{\mathsf{ek}}(m; r)$, given the message $m \in \mathcal{M}$ and some random coins $r \in \mathcal{R}$, outputs a ciphertext $c$;
- $S.\mathsf{Dec}_{\mathsf{dk}}(c)$, output a message $m \in \mathcal{M}$;

which satisfies the correctness and IND-CPA security properties defined below.

We extend in a natural way the algorithm $\mathsf{Enc}$ over vectors: for vectors $\boldsymbol{m} = (m_i)_i \in \mathbb{Z}_M^*$ and $\boldsymbol{r} = (r_i)_i \in \mathbb{Z}_R^*$ of the same size, $S.\mathsf{Enc}_{\mathsf{ek}}(\boldsymbol{m}; r)$ denotes the vector $(S.\mathsf{Enc}_{\mathsf{ek}}(m_i, r_i))_i$. We extend the algorithm $\mathsf{Dec}$ to vectors of ciphertexts in a similar way.

**Definition 16 (Correctness of an Encryption Scheme).** A public-key encryption scheme $S$ is *correct* if for any pair $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$, any message $m \in \mathcal{M}$, and any random coin $r \in \mathcal{R}$, decryption is the reverse operation of encryption: $S.\mathsf{Dec}_{\mathsf{dk}}(S.\mathsf{Enc}_{\mathsf{ek}}(m; r)) = m$.

**Definition 17 (IND-CPA Security Property of a Public-Key Encryption Scheme).** A public-key encryption scheme $S$ is IND-CPA *secure* if for any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\left[\begin{array}{l}(\mathsf{ek},\mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda),\\ (m_0,m_1,\mathsf{st}) \xleftarrow{\$} \mathscr{A}(\mathsf{ek}), b \xleftarrow{\$} \{0,1\} \ : \ \mathscr{A}(\mathsf{st},c)=b\\ r \xleftarrow{\$} \mathcal{R}, c \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(m_b;r)\end{array}\right] \leq \frac{1}{2}+\mu(\lambda)$$

for some function $\mu(\lambda) = \mathrm{negl}(\lambda)$.

### A.2 Preliminaries on Non-Interactive Zero-Knowledge Proofs

We recall the definition of *designated verifier non-interactive zero-knowledge proof of knowledge system* (DVNIZK) from [11]. In the definitions below, we focus on proof systems for NP-languages that admit an efficient (polynomial-time) prover. For an NP-language $\mathscr{L}$, we denote $R_{\mathscr{L}}$ its associated relation, *i.e.*, a polynomial-time algorithm which satisfies $\mathscr{L} = \{x \mid \exists w, |w| = \mathsf{poly}(|x|) \wedge R_{\mathscr{L}}(x,w) = 1\}$. We only recall DVNIZKs in the common reference string model. For conciseness, the common reference string is always implictily given as input to all algorithms.

**Definition 18 (DVNIZK).** A designated verifier non-interactive zero-knowledge proof of knowledge system $\Pi$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ is a quadruple of probabilistic polynomial-time algorithms $(\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ such that

- $\Pi.\mathsf{Setup}(1^\lambda)$, outputs a common reference string $\mathsf{crs}$ (which specifies the language $\mathscr{L}_{\mathsf{crs}}$),
- $\Pi.\mathsf{KeyGen}(1^\lambda)$, outputs a public key $\mathsf{pk}$ and a verification key $\mathsf{vk}$,
- $\Pi.\mathsf{Prove}(\mathsf{pk}, x, w)$, on input the public key $\mathsf{pk}$, a word $x \in \mathscr{L}_{\mathsf{crs}}$, and a witness $w$, outputs a proof $\pi$,
- $\Pi.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, x, \pi)$, on input the public key $\mathsf{pk}$, the verification key $\mathsf{vk}$, a word $x$, and a proof $\pi$, outputs $b \in \{0,1\}$,

which satisfies the completeness, zero-knowledge, and knowledge-extractability properties defined below.

We assume for simplicity that once it is generated, the common reference string $\mathsf{crs}$ is implicitly passed as an argument to the algorithms $(\Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$. Instead of standard soundness properties, we require the stronger notion of knowledge-extractability in this work.

**Definition 19 (Completeness).** A DVNIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations $R_{\mathsf{crs}}$ satisfies the (perfect,statistical) completeness property if for $\mathsf{crs} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, for every $x \in \mathscr{L}_{\mathsf{crs}}$ and every witness $w$ such that $R_{\mathsf{crs}}(x,w) = 1$,

$$\Pr\left[\begin{array}{ll}(\mathsf{pk},\mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk},x,w) & : \Pi.\mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi)=1\end{array}\right] = 1-\mu(\lambda)$$

where $\mu(\lambda) = 0$ for perfect completeness, and $\mu(\lambda) = \mathrm{negl}(\lambda)$ for statistical completeness.

**Definition 20 (Composable Zero-Knowledge).** A DVNIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations $R_{\mathsf{crs}}$ satisfies the (perfect, statistical) composable zero-knowledge property if for any $\mathsf{crs} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, there exists a probabilistic polynomial-time simulator $\mathsf{Sim}$ such that for any stateful adversary $\mathscr{A}$,

$$\left|\Pr\left[\begin{array}{ll}(\mathsf{pk},\mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ (x,w) \leftarrow \mathscr{A}(\mathsf{pk},\mathsf{vk}), & : (R_{\mathsf{crs}}(x,w)=1) \wedge (\mathscr{A}(\pi)=1)\\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk},x,w) & \end{array}\right] - \right.$$
$$\left.\Pr\left[\begin{array}{ll}(\mathsf{pk},\mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ (x,w) \leftarrow \mathscr{A}(\mathsf{pk},\mathsf{vk}), & : (R_{\mathsf{crs}}(x,w)=1) \wedge (\mathscr{A}(\pi)=1)\\ \pi \leftarrow \mathsf{Sim}(\mathsf{pk},\mathsf{vk},x) & \end{array}\right]\right| \leq \mu(\lambda)$$

where $\mu(\lambda) = 0$ for perfect composable zero-knowledge, and $\mu(\lambda) = \mathrm{negl}(\lambda)$ for statistical composable zero-knowledge. If the composable zero-knowledge property holds against efficient (PPT) verifiers, the proof system satisfies computational composable zero-knowledge.

**Definition 21 ($Q$-bounded Knowledge-Extractability).** A DVNIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations $R_{\mathsf{crs}}$ satisfies the $Q$-bounded knowledge-extractability property if for $(\mathsf{crs}, \tau) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, and every adversary $\mathscr{A}$ making at most $Q$ queries to $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$, there is an efficient extractor $\mathsf{Ext}$ such that

$$\Pr\left[\begin{array}{ll}(\mathsf{pk},\mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ (\pi, x) \leftarrow \mathscr{A}^{\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]}(\mathsf{pk}), & : R_{\mathsf{crs}}(x,w) = 0 \wedge \Pi.\mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi) = 1 \\ w \leftarrow \mathsf{Ext}(\pi, x, \tau), & \end{array}\right] \approx 0.$$

## B Security of $\mathsf{MAC}_{\mathsf{GGM}}$ in Composite Order Groups

The DVNIZK scheme from [11] operates in composite order groups if instantiated with the Paillier encryption scheme. Since $\mathsf{MAC}_{\mathsf{GGM}}$ is only proven to be secure in prime order groups, we prove the following theorem.

**Definition 22.** ($\mathsf{MAC}_{\mathsf{GGM}}$ in Composite Order Groups.) The modified version of $\mathsf{MAC}_{\mathsf{GGM}}$ with message length $l$ functions as follows:

- $\mathsf{Setup}(1^\lambda)$: Run $\mathbb{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda, \mathsf{n})$ for $(\mathsf{n}, (p,q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$. The message space $\mathcal{M}$ is $\mathbb{Z}_\mathsf{n}$ and the tag space $\mathcal{S}$ is $\mathbb{G}^2$. Return $\mathsf{pp} \leftarrow (\mathbb{G}, \mathsf{n}, G, H)$ for generators $G, H \xleftarrow{\$} \mathbb{G}$ of $\mathbb{G}$.
- $\mathsf{KeyGen}(\mathsf{pp})$: Pick $\mathsf{k}_0, .., \mathsf{k}_l \xleftarrow{\$} \mathbb{Z}_\mathsf{n}$, set $G_0 \leftarrow \mathsf{k}_0 \bullet G, H_i \leftarrow \mathsf{k}_i \bullet H$ for $i \in \{1, .., l\}$ and output key $\mathsf{sk} \leftarrow (\mathsf{k}_0, .., \mathsf{k}_l)$ and issuer parameters $\mathsf{ipp} \xleftarrow{\$} (G_0, H_1, .., H_l)$.
- $\mathsf{Sign}_{\mathsf{sk}}(m_1, .., m_l)$: Pick $U \xleftarrow{\$} \mathbb{G} \setminus \{0_\mathbb{G}\}$ and output $\sigma \leftarrow (U, (\mathsf{k}_0 + \sum_{i=1}^{l} \mathsf{k}_i m_i) \bullet U)$.
- $\mathsf{Verify}_{\mathsf{sk}}(m_1, .., m_l, \sigma)$: Parse $\sigma$ as $(U, U') \in \mathbb{G}^2$. Check that $U \neq 0_\mathbb{G}$ and $U' = (\mathsf{k}_0 + \sum_{i=1}^{l} \mathsf{k}_i m_i) \bullet U$.

**Theorem 6 (XUF-CMVA Security of $\mathsf{MAC}_{\mathsf{GGM}}$ in Composite Order Groups).** *Let $M$ be $\mathsf{MAC}_{\mathsf{GGM}}$ over a group $\mathbb{G}$ of composite order. Then $M$ is XUF-CMVA-secure in the generic group model under the computational subgroup assumption.*

*Proof.* We follow the proof strategy of [13] and only proof the theorem for $l = 1$ as well. Let $\mathscr{A}$ be a PPT adversary on the UF-CMVA security of $\mathsf{MAC}_{\mathsf{GGM}}$ in the generic group model and $\mathscr{C}$ a challenger for the CSG assumption. We show that if $\mathscr{A}$ outputs a forgery with non-negligible probability, we can win the game against $\mathscr{C}$ with non-negligible probability.

The challenger $\mathscr{C}$ computes $\mathsf{pp} = (\mathbb{G}, \mathsf{n}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ for $(\mathsf{n}, (p,q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ and outputs $(\mathbb{G}, \mathsf{n})$ to us. Now, if we were to build the public $\mathsf{MAC}_{\mathsf{GGM}}$ parameters, we would perform the following steps: Let $P \xleftarrow{\$} \mathbb{G} \setminus \{0_\mathbb{G}\}$ which is a generator for $\mathbb{G}$ with overwhelming probability, we choose $\mathsf{k}_0, \mathsf{k}_1 \xleftarrow{\$} \mathbb{Z}_\mathsf{n}$ and set $G \leftarrow g \bullet P, G \leftarrow h \bullet P$ for $g, h \xleftarrow{\$} Z_\mathsf{n}$, $G_0 \leftarrow \mathsf{k}_0 \bullet G, H_1 \leftarrow \mathsf{k}_1 \bullet H$. Again, the probability of $G, H$ being generators is overwhelming. In our simulation, we actually fix the exponents $\mathsf{k}_0, \mathsf{k}_1, g, h$ after the interaction with the adversary $\mathscr{A}$.

We count the queries to the group operation oracle with variable $q_G$, the queries to the tag oracle with $q_T$, the queries to the verification oracle with $q_V$ and the oracle to the special check oracle with $q_C$. Note that a tag oracle query counts as two group operation queries. Since the adversary is polynomially bounded, it holds that $\frac{c(q_G + q_V + q_T + q_C)^2}{max(p,q)}$ is negligible for any constant $c$. Subsequently, we already count the two group oracle queries of the tag oracle in $q_G$.

Let $S \subset \{0,1\}^*$ with $|S| \geq \mathsf{n}$. We identify group elements $U \in \mathbb{G}$ as $(\log_P(U) \bmod \mathsf{n})$ and represent them with random elements $\zeta \xleftarrow{\$} S$. The adversary only has access to this representation. Throughout the simulation, we maintain a list of polynomials $F \in \mathbb{Z}_\mathsf{n}[\overline{\mathsf{k}_0}, \overline{\mathsf{k}_1}, \overline{h}, \overline{g}, \overline{e_1}, .., \overline{e_{q_T}}]$ representing the group elements, encoded as $\zeta$. The indeterminates $\overline{e_i}$ represent the random choices from the tag oracle. Also, we remember messages for which a tag was issued in the list $Q_{tag}$ and messages that were queried in the verification oracle in $Q_{ver}$. Also, we safe the queried group elements of the check oracle in $Q_{check}$.

In the beginning, we give $P, G, H, G_0, H_1$ to $\mathscr{A}$, encoded as distinct strings $\zeta_p, \zeta_g, \zeta_h, \zeta_{\mathsf{k}_0}, \zeta_{\mathsf{k}_1} \xleftarrow{\$} S$, internally represented in $L$ as $1, \overline{g}, \overline{h}, \overline{g\mathsf{k}_0}, \overline{h\mathsf{k}_1} \in \mathbb{Z}_\mathsf{n}[\overline{\mathsf{k}_0}, \overline{\mathsf{k}_1}, \overline{h}, \overline{g}, \overline{e_1}, .., \overline{e_{q_T}}]$[5].

We simulate the oracles $\mathcal{O}$ for $\mathscr{A}$ as follows:

[5] Note that we do not need to fix the represented elements yet.

- $\mathcal{O}.\mathsf{Operation}(\zeta_i, \zeta_j, \pm)$: for $i, j < q_G$, sets $F_{q_G} = F_i \pm F_j \in \mathbb{Z}_{\mathsf{n}}[\overline{\mathsf{k}_0}, \overline{\mathsf{k}_1}, \overline{h}, \overline{g}, \overline{e_1}, .., \overline{e_{q_T}}]$, if $F_{q_G} = F_l$ for $l < q_G$, then sets $\zeta_{q_G} \leftarrow \zeta_l$, otherwise picks $\zeta_{q_G} \xleftarrow{\$} S$ with $\zeta_{q_G} \neq \zeta_{prev}$ for $0 \leq prev < q_G$, adds tuple $(F_{q_G}, \zeta_{q_G})$ to $L$, outputs $\zeta_{q_G}$ and increments $q_G$;
- $\mathcal{O}.\mathsf{Sign}(m)$: sets $F_{q_G} \leftarrow \overline{e_{q_T}}$, picks $\zeta_{q_G} \xleftarrow{\$} S$ with $\zeta_{q_G} \neq \zeta_{prev}$ for $0 \leq prev < q_G$. Then computes $F_{q_G+1} \leftarrow \overline{e_{q_T}}(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) \in \mathbb{Z}_{\mathsf{n}}[\overline{\mathsf{k}_0}, \overline{\mathsf{k}_1}, \overline{h}, \overline{g}, \overline{e_1}, .., \overline{e_{q_T}}]$. If $F_{q_G+1} = F_l$ for $l \leq q_G$, then sets $\zeta_{q_G+1} \leftarrow \zeta_l$, otherwise picks $\zeta_{q_G+1} \xleftarrow{\$} S$ with $\zeta_{q_G+1} \neq \zeta_{prev}$ for $0 \leq prev \leq q_G$, adds tuples $(F_{q_G}, \zeta_{q_G}), (F_{q_G+1}, \zeta_{q_G+1})$ to $L$ and $m$ to $Q_{tag}$, outputs $\zeta_{q_G}, \zeta_{q_G+1}$ and finally, increments $q_T$ and $q_G$ twice;
- $\mathcal{O}.\mathsf{Verify}(m, \zeta, \zeta')$: checks $(F, \zeta) \in L \wedge (F', \zeta') \in L$ and $F(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) = F'$, finally increments $q_V$ and adds $m$ to $Q_{ver}$.
- $\mathcal{O}.\mathsf{Check}(\zeta, \zeta')$: checks $(F, \zeta) \in L \wedge (F', \zeta') \in L$ and $F(\overline{\mathsf{k}_1}) = F'(\overline{\mathsf{k}_1})$, finllay incremets $q_C$ and adds $(F, F')$ to $Q_{check}$.

At some point, $\mathscr{A}$ outputs $(m, \zeta, \zeta')$ after $q_G$ group operation queries, $q_T$ tag queries, $q_V$ verification queries and $q_C$ check oracle queries with $(F, \zeta), (F', \zeta') \in L$[6]. Note that due to the limited amount of query choices of the adversary $\mathscr{A}$, the polynomial F will have the structure

$$F = a\overline{g} + b\overline{h} + s\overline{g}\overline{\mathsf{k}_0} + t\overline{h}\overline{\mathsf{k}_1} + \sum_{i=1}^{q_T} x_i\overline{e_i} + \sum_{i=1}^{q_T} y_i\overline{e_i}(m_i\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) + c$$

for $a, b, c, s, t \in \mathbb{Z}_q$ and where $m_i \in Q_{tag}$ is the $i^{th}$ queried message via $\mathcal{O}.\mathsf{Sign}$. Lastly, we pick a random $v$[7] $:= (\mathsf{k}_0, \mathsf{k}_1, g, h, e_1, .., e_{q_T}) \xleftarrow{\$} \mathbb{Z}_{\mathsf{n}}^{3+q_T}$, choose $i \in \{1..q_T\} : x_i \neq 0$ and output $x_i \bullet P$ to the challenger $\mathscr{C}$. If no such $i$ exists, output $P$[8].

We now analyze the success probability of the constructed adversary. Without loss of generality, we assume $p > q$. First, we show that our simulation is perfect with overwhelming probability. All polynomials in $L$ have at most degree two based on the operations available. Let $(F_i, \zeta_i), (F_j, \zeta_j) \in L$. If $F_i \neq F_j$, but both polynomials are evaluated to the same value in $\mathbb{Z}_{\mathsf{n}}$, the simulation would be invalid because we represented the same value with different representations in $S$. This is formalized by the following event:

$$\exists i, j \leq q_T : F_i(v) = F_j(v) \bmod \mathsf{n} \wedge F_i \neq F_j. \tag{1}$$

This is bounded by $2/p$ using lemma 1 for each fixed $i, j$[9]. Also, it could be the case that our verification oracle failed. This event is described as:

$$(F_i(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}))(v) = F_j(v) \bmod \mathsf{n} \tag{2}$$

for $m \in Q_{ver}$. This is bounded by $3/p$ using lemma 1 for fixed $i, j, m$. Note that $F_i$ can be of degree two and thus $F_i(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) - F_j$ of degree three. Lastly, the check oracle simulation could fail which is described as:

$$(F(\overline{\mathsf{k}_1}))(v) = (F'(\overline{\mathsf{k}_1}))(v) \bmod \mathsf{n} \tag{3}$$

for $(F, F') \in Q_{check}$. Again, the probability of this event is bounded by $3/p$ using lemma 1 for fixed $F, F'$.

Thus, the probability of event 1 occurring for all pairs $(i, j) \in \mathbb{Z}_{q_G}^2$, event 2 for all $m \in Q_{ver}$[10] or event 3 for all $(F, F') \in Q_{check}$ is at most

$$\binom{q_G}{2} \cdot \frac{2}{p} + (q_V + q_C) \cdot \frac{3}{p} \leq \frac{2q_G^2}{p} + \frac{3(q_V + q_C)}{p} \leq \frac{6(q_G + q_V + q_C)^2}{p}$$

[6] If $\mathscr{A}$ outputs non-queried group elements, the success probability negligible.
[7] This finally fixes our group elements represented as polynomials to real group elements.
[8] Since $P$ is a generator of $\mathbb{G}$ with overwhelming probability, $P$ will most likely not be a non-trivial subgroup generator, but we show that if $\mathscr{A}$ is successful, such $i$ must exist in the following.
[9] We set $v_p = v \mod p$ and check if $(F_i - F_j)(v_p) = 0$. Since $v$ and thus $v_p$ is chosen uniformly random and $F_i - F_j \neq 0$, the probability of this happening is bounded by $deg(F_i - F_j)/p$.
[10] For each $m \in Q_{ver}$ the queried polynomials are fixed.

In conclusion, the probability of our simulation failing is negligible for the polynomially bounded adversary $\mathscr{A}$.

If $\mathscr{A}$ succeeds, it holds that[11]

$$F(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) = F' \tag{4}$$

Due to the restriction of available operations for $\mathscr{A}$, we have

$$F = a\overline{g} + b\overline{h} + s\overline{g}\overline{\mathsf{k}_0} + t\overline{h}\overline{\mathsf{k}_1} + \sum_{i=1}^{q_T} x_i\overline{e_i} + \sum_{i=1}^{q_T} y_i\overline{e_i}(m_i\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) + c$$

$$F' = a'\overline{g} + b'\overline{h} + s'\overline{g}\overline{\mathsf{k}_0} + t'\overline{h}\overline{\mathsf{k}_1} + \sum_{i=1}^{q_T} x'_i\overline{e_i} + \sum_{i=1}^{q_T} y'_i\overline{e_i}(m_i\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) + c'$$

for $a, a', b, b', c, c', s, s', t, t' \in \mathbb{Z}_q$ and where $m_i \in Q_{tag}$ is the $i^{th}$ queried message via $\mathcal{O}.\mathsf{Sign}$.

First, we look at $F = w$ for some $w \in \mathbb{Z}_{\mathsf{n}}$. Then $F' = w(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0})$. Because $F'$ has no subterm $\overline{\mathsf{k}_0}$, the equality in 4 is only possible for $w = 0$. This does not constitute a valid forgery, though.

Now, we look at $F$ with $deg(F) > 0$. Since the degree of $F(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0})$ will be at least two and the degree of $F'$ is at most two, we have $deg(F) = 1, deg(F') = 2$. Thus $F = a\overline{g} + b\overline{h} + \sum_{i=1}^{q_T} x_i\overline{e_i} + c$. Since $F'$ has no subterm of the form $\overline{g}\overline{\mathsf{k}_1}, \overline{\mathsf{k}_0}, \overline{\mathsf{k}_1}$ or $\overline{h}\overline{\mathsf{k}_0}$, $F$ has to be of the form $\sum_{i=1}^{q_T} x_i\overline{e_i}$. Thus, equation 4 implies that $F' = \sum_{i=1}^{q_T} y'_i\overline{e_i}(m_i\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0})$. In conclusion, we have

$$\begin{aligned}
& F' - F(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) = 0 \\
\implies & \sum_{i=1}^{q_T} y'_i\overline{e_i}(m_i\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) - \sum_{i=1}^{q_T} x_i\overline{e_i}(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) = 0 \\
\implies & \sum_{i=1}^{q_T} \overline{e_i}(y'_i(m_i\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) - x_i(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0})) = 0 \\
\implies & \forall i \in \{1..q_T\} : y'_i \cdot m_i\overline{\mathsf{k}_1} + y'_i\overline{\mathsf{k}_0} - x_i \cdot m\overline{\mathsf{k}_1} - x_i\overline{\mathsf{k}_0} = 0 \\
\implies & \forall i \in \{1..q_T\} : y'_i \cdot m_i\overline{\mathsf{k}_1} = x_i \cdot m\overline{\mathsf{k}_1} \wedge y'_i\overline{\mathsf{k}_0} = x_i\overline{\mathsf{k}_0} \\
\overset{x_i=y'_i}{\implies} & \forall i \in \{1..q_T\} : x_i \cdot m_i\overline{\mathsf{k}_1} = x_i \cdot m\overline{\mathsf{k}_1} \\
\implies & \forall i \in \{1..q_T\} : x_i \cdot m_i = x_i \cdot m \mod \mathsf{n} \\
\implies & \begin{cases} m = m_j \in Q, & \exists j \in \{1..q_T\} : x_j \in Z^*_{\mathsf{n}} \\ \forall i \in \{1..q_T\} : x_i = 0 \bmod p_i, & \text{for } p_i \in \{p, q\} \text{ otherwise} \end{cases}
\end{aligned}$$

In the first case, it holds that $m = m_j$ for $j \in \{1..q_T\}$. This is not a valid forgery. In the second case, if $\forall i \in \{1..q_t\} : x_i = 0 \mod n$ implies that $F = 0 \mod n$. This would not constitute a valid forgery. Thus, $\exists j \in \{1..q_t\} : x_j \neq 0 \mod n$, but $x_j = 0 \mod p_j$. Finally, this implies that if $\mathscr{A}$ is successful, there exists some $j \in \{1..q_t\}$ such that $x_j \bullet P$ is a non-trivial subgroup generator.

## C Further Improvements from the Short-Exponent Discrete Logarithm Assumption

In this section, we discuss an approach to further improve the efficiency of our NIKVAC scheme, under the additional assumption that it is infeasible to solve discrete logarithms over $\mathbb{G}$, even when the exponent is not uniformly random, but random conditioned on being shorter than some threshold. Considering discrete logarithm with short exponent has been done many times in the past (e.g. [15,23,27,32,36]), since it is a natural approach to improve the computational efficiency of a scheme when working over a group with a large order. The best known attack on the discrete logarithm with short exponent (DLSE) assumption solves it in time $O(\sqrt{T})$, where $T$ is the length of the interval from which the exponent is drawn [21].

[11] The probability of $F(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}) \neq F'$, but $(F(m\overline{\mathsf{k}_1} + \overline{\mathsf{k}_0}))(v) = F'(v)$ is negligible with the same argument as before.

### C.1 Definition

***T*-Bounded Short Exponent Discrete Logarithm (DLSE) Assumption.** Let $\mathbb{G}$ be a group with order $\mathsf{n}$, $G \in \mathbb{G}$ a generator and $T \in \mathbb{Z}_{\mathsf{n}}, T < \mathsf{n}$. For all PPT adversaries $\mathscr{A}$ it holds that

$$\Pr\begin{bmatrix} z \xleftarrow{\$} \{0..T-1\}, \\ z' \leftarrow \mathscr{A}(z \bullet G) \end{bmatrix} : z = z' \Big] \leq \mu(\lambda)$$

where $\mu(\lambda) = \mathrm{negl}(\lambda)$.

**Indistinguishability of Short Exponents.** As shown in [27], the following statement is equivalent to the DLSE assumption in prime order groups. This is also true in composite order groups, which was shown in [11].
Let $\mathbb{G}$ be a group with order $\mathsf{n}$, $G \in \mathbb{G}$ a generator and $T \in \mathbb{Z}_{\mathsf{n}}, T < \mathsf{n}$. For all PPT adversaries $\mathscr{A}$ it holds that

$$\Big|\Pr\Big[z \xleftarrow{\$} \{0..T-1\} \ : \ \mathscr{A}(z \bullet G) = 1\Big] - \Pr\Big[z \xleftarrow{\$} \mathbb{Z}_{\mathsf{n}} \ : \ \mathscr{A}(z \bullet G) = 1\Big]\Big| \leq \mu(\lambda)$$

### C.2 Our Approach

Unlike previous works, we seek to use the DLSE assumption to improve not only the computational efficiency of our scheme, but also reduce its communication. The main observation is that in a group $\mathbb{G}$ of composite order, where exponents are typically of size at least 2048 bits, there is a clear "waste of space" for attributes: typical attributes will in general be much shorter (say, 128 or 256 bits).

**Recalling the Previous Approach.** For simplicity, we focus in our explanation on the case of the scheme $\mathsf{MAC}_{\mathsf{GGM}}$, with a single attributes – all the results will trivially extend to larger vectors of attributes, and to the more general abstract MAC. A proof with our scheme of possession of a valid credential $(U, V = (\mathsf{k}_0 + \mathsf{k}_1 \cdot m) \bullet U)$ on a message $m$ is computed as follows:

- $(U', V') \leftarrow (z \bullet U, z \bullet V \mathbin{+} t \bullet G)$,
- $W \leftarrow m' \bullet U + t' \bullet H$,
- $(X_m, X_t) \leftarrow (S.\mathsf{Enc}_{\mathsf{ek}}(m; r_m), S.\mathsf{Enc}_{\mathsf{ek}}(t; r_t))$
- $X'_m \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(m'; 0) \ominus (r_m \odot \mathsf{pk}) = S.\mathsf{Enc}_{\mathsf{ek}}(m'; -e \cdot r_m)$, and
- $X'_t \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(t'; 0) \ominus (r_t \odot \mathsf{pk}) = S.\mathsf{Enc}_{\mathsf{ek}}(t'; -e \cdot r_t)$,

for random $(z, t, m', t', r_m, r_t)$ in the appropriate space, and where $H = \mathsf{k}_1^{-1} \bullet G$. The purpose of the component $t \bullet G$ in the computation of $V' = z \bullet V \mathbin{+} t \bullet G$ is to perfectly mask $V$, preventing the verifier to break the anonymity by verifying the credential $(U', V')$ with respect to several candidate messages $m$. However, this mask $t$ is also the reason why the proof of possession must include two additional ciphertexts $(X_t, X'_t)$, since the user must now also prove knowledge of this masking value without revealing it.

**Alternative Strategy.** Under the DLSE assumption, there is a more efficient approach: the idea will be to directly "randomize the message $m$" to maintain anonymity, by padding it with a random value. Let $\ell$ be the bit-length of the attribute $m$, which we assume to be much shorter than $\log \mathsf{n}$, where $\mathsf{n}$ is the order of $\mathbb{G}$: we require $\ell \leq \log \mathsf{n} - 2\lambda$, where $\lambda$ is the security parameter. We modify the blind issuance protocol such that it does not issue a credential $(U, V)$ on $m$ anymore: instead, it should output a credential $(U, V)$ on the value $x = m + 2^{\ell} r \bmod \mathsf{n}$, where $r$ is a uniformly random *padding* picked by the user (and hidden from the signing authority) of bit-length at least $2\lambda$, and at most $\log \mathsf{n} - \ell$ (to ensure that no wraparound modulo $\mathsf{n}$ occurs, which guarantees that $x$ uniquely defines $m$). Consider now the following simplifies proof of credential possession:

- $(U', V') \leftarrow (z \bullet U, z \bullet V)$,
- $W \leftarrow m' \bullet U$,
- $X_m \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(m; r_m)$, and

– $X'_m \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(m';0) \ominus (r_m \odot \mathsf{pk}) = S.\mathsf{Enc}_{\mathsf{ek}}(m';-e \cdot r_m)$.

We explain why the above simplified proof guarantees anonymity. In the anonymity game, we must exhibit a simulator which can generate a valid-looking proof of possession, without knowing the signed message $m$. The simulator will rely on the security of the blind issuance two-party protocol to simulate his interaction with the signing authority without knowing $m$. Then, to simulate the proof of credential possession, he simply pick $(U', V')$ at random, compute $W$ honestly, and simulate the $(X_m, X'_m)$ using the simulation strategy of [11]. Now, we argue that this is indistinguishable from an honest proof, using a sequence of games:

**Game 1.** In the first game, the simulator still obtains $(U, V)$ by playing the blind issuance protocol honestly with $m$, and computes $(U', V')$ honestly from $(U, V)$. He uses the simulation strategy of [11] to simulate the $(X_m, X'_m)$ using the verification key $\mathsf{vk}$. By the same argument as in [11], this game is statistically indistinguishable from the real game.

**Game 2.** In this game, the simulator will now simulate the blind issuance protocol, without knowing $m$ (hence he does not obtain $(U, V)$), and generates $(U', V')$ uniformly at random. Under the security of the two-party computation protocol for blind issuance, his interaction with the signing authority is indistinguishable from an interaction with an honest user. It remains to show that the simulated $(U', V')$ is indistinguishable from the real one. Note that in the previous game, we had

$$V' = (\mathsf{k}_0 + x \cdot \mathsf{k}_1) \bullet U'$$

where $U' = z \bullet U$ is random, and $x = m + 2^\ell r$ for an $\ell$-bit message $m$ and a padding $r$ of size at most $\log \mathsf{n} - \ell$ bits. We can rewrite this as

$$V' = (\mathsf{k}_0 \bullet U' \boldsymbol{+} (m \cdot \mathsf{k}_1) \bullet U') \boldsymbol{+} r \bullet ((2^\ell \cdot \mathsf{k}_1) \bullet U').$$

under the short exponent discrete logarithm assumption, the above is indistinguishable to

$$V' = (\mathsf{k}_0 \bullet U' \boldsymbol{+} (m \cdot \mathsf{k}_1) \bullet U') \boldsymbol{+} r' \bullet ((2^\ell \cdot \mathsf{k}_1) \bullet U'),$$

where $r'$ is now picked uniformly at random in $\mathbb{Z}_{\mathsf{n}}$. But then, since $r'$ is uniformly random, the above $V'$ is distributed perfectly uniformly, which concludes the proof.

### C.3 Blind Issuance

Generalizing to vectors of $\beta$ attributes and an abstract $\mathsf{MAC}$ with $\alpha$ components, the above strategy reduces the size of a proof of credential possession from $\alpha(1+\beta)+1$ group elements and $2(\beta+\alpha)$ ciphertexts to $\alpha(1+\beta)+1$ group elements and $2\beta$ ciphertexts. We now exhibit a very simple and efficient blind issuance protocol for the above padding-based strategy. The basic idea is that, because of the padding, the user can simply send $G' = x \cdot G$ to the authority, with $x = m + 2^\ell r$: under the $\mathsf{DLSE}$ assumption, this computationally hides the message $m$. Furthermore, it is straightforward to sign a message "in the exponent" with the abstract $\mathsf{MAC}$: focusing for simplicity on $\mathsf{MAC}_{\mathsf{GGM}}$, the authority would just need to compute $(U, V) \leftarrow (y \bullet G, y \bullet (\mathsf{k}_0 \bullet G \boldsymbol{+} \mathsf{k}_1 \bullet G'))$, for a random exponent $y$. It is immediate to check that $(U, V)$ is a valid $\mathsf{MAC}$ signature on $x$; this issuance protocol is therefore extremely efficient, requiring only three group elements in total.

This, however, raises an apparent issue: in many applications, the user will need to prove to the authority that the signed message is indeed $m$ (if, for example, the authority only issues credentials for users with an identity known to him), but still cannot reveal $x = m + 2^\ell r$ to the authority (this would break anonymity when the user authenticate to some verifier). However, this message is only uniquely defined if the value $r$ in $x = m = 2^\ell r$ is guaranteed to be smaller than $\mathsf{n}/2^\ell$; proving that this is the case seems to require a range proof on $r$, which is typically very expensive. We observe that there is a very simple way around this issue: it only suffices here to guarantee that $r$ is smaller than $\mathsf{n}/2^\ell$, but it is also fine to ask an honest user to choose a padding $r$ which is smaller than that, as long as it is at least $2\lambda$ bits long. Therefore, we get the following simple solution:

– The user picks $r \xleftarrow{\$} [1..\mathsf{n}/2^{\ell+2\lambda}]$ and sets $x \leftarrow m + 2^\ell r$. He sends $G' \leftarrow x \bullet G$ to the authority. From $G'$, the authority computes $G_r \leftarrow (2^\ell)^{-1} \bullet (G' \boldsymbol{-} m \bullet G) = r \bullet G$.

– The authority and the user executes a standard three-move proof of knowledge of $r$ with *statistical security* by picking the mask and the challenge in an appropriate set, where the user plays the role of the prover, and the authority of the verifier:
  - The user picks $\rho \xleftarrow{\$} [1..\mathsf{n}/2^\ell]$ and sends $G'' \leftarrow \rho \bullet G$ to the authority.
  - The authority sends a random $\lambda$-bit challenge $c$.
  - The users computes and sends $d \leftarrow r \cdot c + \rho$ (over the integers).

  The authority accepts the interaction if and only if $d \bullet G = c \bullet G_r \mathbf{+} G''$, and furthermore $d \leq \mathsf{n}/2^\ell$.

The above protocol guarantees that the user knows the $r$ in $x = m + 2^\ell r$, and furthermore that this padding is *short* (below $\mathsf{n}/2^\ell$); to maintain correctness, however, the user must initially choose the padding smaller than $\mathsf{n}/2^{\ell+2\lambda}$. This is fine as long as $\mathsf{n}/2^{\ell+2\lambda} > 2^{2\lambda}$. In a typical scenario, we can have $\ell = \lambda = 128$ and $\mathsf{n} = 2048$, in which case the above condition is easily satisfied.