Algebraic (Trapdoor) One-Way Functions and their Applications

Dario Catalano[1], Dario Fiore[2], Rosario Gennaro[3], and Konstantinos Vamvourellis[2]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy
catalano@dmi.unict.it
[2] New York University, USA
fiore@cs.nyu.edu, kv472@nyu.edu
[3] City College of New York, USA
rosario@cs.ccny.cuny.edu

Abstract. In this paper we introduce the notion of *Algebraic (Trapdoor) One Way Functions*, which, roughly speaking, captures and formalizes many of the properties of number-theoretic one-way functions. Informally, a (trapdoor) one way function $F : X \to Y$ is said to be algebraic if X and Y are (finite) abelian cyclic groups, the function is *homomorphic* i.e. $F(x) \cdot F(y) = F(x \cdot y)$, and is *field-homomorphic*, meaning that it is possible to compute linear operations "in the exponent" over some field (which may be different from $\mathbb{Z}_p$ where p is the order of the underlying group X) without knowing the bases. Moreover, algebraic OWFs must be *strongly one-way* in the sense that given $y = F(x)$, it must be infeasible to compute (x', d) such that $F(x') = y^d$ (for $d \neq 0$). Interestingly, algebraic one way functions can be constructed from a variety of *standard* number theoretic assumptions, such as RSA, Factoring and CDH over bilinear groups.

As a second contribution of this paper, we show several applications where algebraic (trapdoor) OWFs turn out to be useful. In particular:

- *Publicly Verifiable Secure Outsourcing of Polynomials*: We present efficient solutions which work for fields of arbitrary size and characteristic. When instantiating our protocol with the RSA/Factoring based algebraic OWFs we obtain the first solution which supports small field size, is efficient and does not require bilinear maps to obtain public verifiability. Moreover, by working over $\mathbb{F}_2$, we can support *public verification of boolean formulas*. This significantly improves on previous work e.g., Parno et al. (TCC 2012), as the latter relies on specific attribute based encryption schemes whose security is based on q-type assumptions.
- *Linearly-Homomorphic Signatures*: We give a direct construction of FDH-like linearly homomorphic signatures from algebraic (trapdoor) one way permutations. Our constructions support messages and homomorphic operations over *arbitrary* finite fields of both binary and prime characteristic. While it was already known how to realize linearly homomorphic signatures over small fields (Boneh-Freeman, Eurocrypt 2011), from lattices in the random oracle model, ours are the first schemes achieving this in a very efficient way from Factoring/RSA.
- *Batch execution of Sigma protocols*: We construct a simple and efficient Sigma protocol for any algebraic OWP and show a "batch" version of it, i.e. a protocol where many statements can be proven at a cost (slightly superior) of the cost of a single execution of the original protocol. Given our RSA/Factoring instantiations of algebraic OWP, this seems to be the first batch verifiable Sigma protocol for groups of unknown order.

1 Introduction

Algebraic One-Way Functions. This paper introduces the notion of *Algebraic One-Way Function*, which aims to capture and formalize many of the properties enjoyed by number-theoretic based one-way functions. Intuitively, an Algebraic One-Way Function (OWF) $F : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa$ is defined over abelian cyclic groups $\mathcal{X}_\kappa, \mathcal{Y}_\kappa$, and it satisfies the following properties:

- *Homomorphic:* the classical property that says that group operations are preserved by the OWF.
- *Field-Homomorphic:* this is a new property that intuitively says that it is possible to efficiently perform linear operations "in the exponent" over some field $\mathbb{K}$. While this property turns out to be equivalent to the homomorphic property for groups of known order p and the field $\mathbb{K} = \mathbb{F}_p$, it might not hold for groups of unknown order. Yet for the case of RSA Moduli we show that this property holds, and more interestingly it holds for *any* finite field.
- *Strongly One-Way:* We strengthen the usual notion of one-wayness by asking that given $y = F(x)$ it should be infeasible to compute (x, d) such that $F(x) = y^d$, for any $d \neq 0$ (rather than $d = 1$ as in the traditional one-wayness definition).

In our work we also consider natural refinements of this notion to the cases when the function is a permutation and when there exists a trapdoor that allows to efficiently invert the function.

Given this new notion, we demonstrate its existence by showing three instantiations of Algebraic OWFs whose security can be deduced from the hardness of the Diffie-Hellman problem in groups with bilinear maps, or alternatively using RSA moduli and the RSA/Factoring assumptions.

Applications. As a second contribution of this paper, we turn our attention to three separate practical problems: outsourcing of polynomial computations, linearly homomorphic signatures and batch executions of identification protocols. In all three separate problems, we show that Algebraic OWFs can be used for building truly efficient schemes that improve in several ways on the "state-of-the-art". In particular, we propose solutions for:

- *Publicly Verifiable Secure Outsourcing of Polynomials* which works over fields of *arbitrary* size and characteristic and does not necessarily use bilinear maps. This allows for simple and efficient solutions for small fields, and, by working over $\mathbb{F}_2$ for publicly verifiable delegation of boolean formulas.
- *Linearly Homomorphic Signature Schemes* also over *arbitrary* fields, and in particular even small fields such as $\mathbb{F}_2$. The only known constructions for the latter case require assumptions over lattices [10] while we can use any of the assumptions above obtaining more efficient algorithms.
- *Batch Executions of Identification Protocols:* we construct a Sigma-protocol based on algebraic one-way functions and then we show that it is possible to construct a "batch" version of it where many statements are proven basically at the cost of a single one. A similar batch version for the Schnorr's Sigma protocol has been proposed in [23] and we generalize it to any of the assumptions above. In particular for the instantiation based on RSA we obtain a batch version of the Guillou-Quisquater protocol [28] which yields, to the best of our knowledge, the first batch verifiable Sigma protocol for groups of unknown order, a problem left open in [23].

Below, we elaborate in detail about the improvements of our solutions.

1.1 Secure Outsourcing of Polynomials

Starting from work by Benabbas *et al.* [7], several papers have been investigating the problem of securely outsourcing the computation of large polynomials. The problem can be described as follows: a computationally weak client stores a large polynomial (say in m variables, of degree d) with a powerful server. Later, the client will request the server to evaluate the polynomial at a certain input x and the server must provide such result together with a "proof" of its correctness. In particular, it is crucial that to be verified such proof must require substantially less resources than computing the polynomial from scratch. Furthermore, the client must store only a "small" amount of secret information, e.g. not the entire polynomial.

Following [7], several other papers (e.g. [38, 39, 19]) have investigated this problem, focusing particularly on the feature of *public verification*, where the correctness of the result provided by the server can be verified by *anyone*, not just by the client who initially stored the polynomial (the original solution in [7] obtained only private verification).

The popularity of this research problem can be explained by its numerous practical applications, including as discussed in [7] *Proofs of Retrievability* (the client stores a large file F with the server and later wants a short proof that the entire file can be retrieved) and *Verifiable Keyword Search* (given a text file $T = \{w_1, \ldots, w_\ell\}$ and a word w, the server tells the client if $w \in T$ or not).

Limitation of Previous Solutions. The solutions for outsourcing of polynomial computations mentioned above suffer from two main drawbacks:

- *Large Field Size.* The schemes presented in [7, 38, 19] work only for polynomials computed over fields of prime characteristic p, which is the same p as the order of the underlying cryptographic group that is used to prove security. As a result, p must be large, and therefore previous schemes cannot handle small field sizes. The solution recently proposed in [39] can support polynomials over $\mathbb{Z}_2$, and thus, by working in a "bit-by-bit" fashion, over any field. However, to work over other fields of any characteristic p, it incurs a $O(\log p)$ computational overhead since $O(\log p)$ parallel instances of the scheme must be run. It would be therefore nice to have a scheme that works for polynomials over arbitrary fields, without a "bit-by-bit" encoding, so that the same scheme would scale well when working over larger field sizes.
- *Public Verifiability via Bilinear Maps.* All previous solutions that achieve public verifiability [38, 39, 19] do so by means of Groups with Bilinear maps as the underlying cryptographic tool. Since pairing computations may be expensive compared to simpler operations such as exponentiations, and given that bilinear maps are the only known algebraic structure under which we can currently build publicly verifiable computation, it is an interesting question to investigate whether we can have solutions that use alternative algebraic tools and cryptographic assumptions (e.g. RSA moduli) to achieve public verifiability.

Our new solution removes these two problems. As discussed above, we can instantiate our protocols over RSA moduli, and prove their security under the DDH/RSA/Factoring Assumptions over such groups, therefore avoiding the use of bilinear maps. Perhaps more interestingly, our protocols can handle finite fields of any size and any characteristic. In particular, by working over $\mathbb{F}_2$ it can also support *public verification of boolean formulas.* This feature allows for much more flexibility and efficiency, as the only known previous solution for arbitrary fields [39] has to work bit by bit in fields larger than 2. Moreover, the schemes in [39] are based on specific Attribute-Based Encryption schemes (e.g. [33]) whose security rely on "q-type" assumptions, whereas our solution can do so based on the well known RSA/Factoring assumptions.

As in the case of [19] our techniques extend for building a protocol for *Matrix Multiplication.* In this problem (also studied in [35]) the client stores a large $(n \times d)$ matrix M with the server and then provides d-dimensional vectors $\boldsymbol{x}$ and obtains $\boldsymbol{y} = M \cdot \boldsymbol{x}$ together with a proof of correctness.

Other comparisons with related work. The subject of verifiable outsourced computation has a large body of prior work, both on the theoretical front (e.g. [4, 27, 32, 34, 26]) and on the more applied arena (e.g. [36, 5, 44, 45]).

Our work follows the "amortized" paradigm introduced in [21] (also adopted in [16, 2]) where a one-time expensive preprocessing phase is allowed. The protocols described in those papers allow a client to outsource the computation of an arbitrary function (encoded as a Boolean circuit) and use fully homomorphic encryption (i.e. [24]) resulting in solutions of limited practical relevance. Instead, we follow [7] by considering a very limited class of computations (polynomial evaluation and matrix multiplication) in order to obtain better efficiency.

As discussed above, we improve on [38] by providing a solution that works for finite fields of arbitrary characteristic (even small fields) and by avoiding the use of bilinear maps. Given that our solution is a generalization of [19] we also inherit all the improvements of that paper. In particular, compared to [38]:

- we get security under constant-size assumptions (i.e. assumptions that do not asymptotically depend on the degree of the polynomial), while their scheme uses a variation of the CDH Assumption that grows with the degree.
- we handle a larger class of polynomial functions: their scheme supports polynomials in m variables and total degree d (which we also support) but we additionally consider also polynomials of degree d in each variable.
- For the case we both support, we enjoy a much faster verification protocol: a constant amount of work (a couple of exponentiations over an RSA modulus) while they require $O(m)$ pairings[4].

1.2 Linearly Homomorphic Signatures

Imagine a user Alice owns some data set $m_1, \ldots, m_n \in M$ that she keeps (signed) in some database stored at a, not necessarily trusted, server. Imagine also that some other user, Bob, is allowed to query the database to perform some basic computation (such as the mean or other statistics) over Alice's data set. The simplest way to do this in a reliable manner (for Bob) is to download the full data set from the server, check all the signatures and compute the desired statistic. This solution, however, has two drawbacks. First, it is inefficient in terms of bandwidth. Second, even though Alice allows Bob to access some statistics over her data, she might not want this data to be explicitly revealed. Homomorphic signatures allow to overcome both these issues in a very elegant fashion [10]. Indeed, using a homomorphic signature scheme, Alice can sign $m_1, \ldots, m_n$, thus producing the signatures $\sigma_1, \ldots, \sigma_n$, which can be verified exactly as ordinary signatures. The homomorphic property provides the extra feature that given $\sigma_1, \ldots, \sigma_n$ and some function $f : M^n \to M$, one can compute a signature σ on the value $f(m_1, \ldots, m_n)$ *without* knowledge of the secret signing key SK. In order to support this property the verification has to take f as well as σ, and the classical security notion for signatures has to be relaxed so as to require that no valid signature for $m \neq f(m_1, \ldots, m_n)$ should be computable without knowing SK.

The notion of homomorphic signature was introduced by Johnson *et al.*[30] and later refined by Boneh *et al.* [9]. Its main motivation was realizing a linear network coding scheme [1, 41] secure against pollution attacks. The construction from [9] uses bilinear groups as the underlying tool

[4] In contrast the delegation phase is basically free in their case, while our delegation step requires $O(md)$ work – note however that in publicly verifiable scheme, the verification algorithm might be run several times and therefore its efficiency is more important.

and authenticates linear functions on vectors defined over large prime fields. Subsequent works considered different settings as well. In particular, the constructions in [22, 14, 15] are based on RSA, while [11, 10] rely on lattices and can support linear functions on vectors over small fields. A general framework for homomorphic signatures in the standard model, was recently provided by Freeman [20].

Our Contribution. In this paper we show that algebraic trapdoor one way permutations, *directly* allow for a very simple and elegant extension of Full Domain Hash (FDH) to the case of linearly homomorphic signatures. Similarly to standard FDH signatures our construction is secure in the random oracle model and allows for very efficient instantiations. Our framework allows for great flexibility when choosing a homomorphic signature scheme and the underlying message space. Indeed our constructions support messages and homomorphic operations over *arbitrary* finite fields of both binary and prime characteristic. While it was already known how to realize linearly homomorphic signatures over small fields [11, 10], ours seem to be the first schemes achieving this in a very efficient way and based on simple assumptions such as Factoring and RSA.

1.3 Batch Executions of Sigma Protocols

We show that for any Algebraic One-Way Permutation there exists a simple and efficient Sigma protocol that allows a Prover to convince a Verifier that he "knows" a pre-image of an Algebraic OWP. Our protocol can be seen as an extension of the classical Schnorr and Guillou-Quisquateur protocols [42, 28]. Following [23] we then considered the question of constructing a "batch" version of it where many statements are proven basically at the cost of a single one.

Gennaro *et al.* discuss in [23] many applications of such a protocol. As an example, consider an access control system where users belong to various privilege classes. Access control classes for the data are defined using such privileges, i.e. as the users who own a given subset of privileges. For instance, the access control class for a given piece of data D, can be defined as the users who own privileges P_1, P_2, P_3.

This can be realized by associating a different public key to each privilege [5]. Then a user would prove that she knows the secret keys required for the authorization. Using typical proofs of knowledge, to prove knowledge of k keys the user has to perform k proofs. Although these proofs can be performed in parallel, keeping the round complexity the same, the computational complexity goes up by a factor of k.

The question posed in [23] was to design a proof of knowledge of ℓ secrets at the cost of less than ℓ proofs. They answered this question for the Schnorr's protocol and they left it open for the Guillou-Quisquateur protocol as the same techniques did not seem to work for groups of unknown order.

Following [23] we show a *batch* version of our Sigma protocol where the prover can prove knowledge of ℓ pre-images of the OWP, at a cost slightly superior to the cost of a single execution of the Sigma protocol, thus saving a factor of ℓ in computation and bandwidth over the best previously

[5] Another way to implement such an access control system is to give each user a certified public key. The certificate would indicate the subset of privileges associated with this public key. Then in order to gain access, the user proves knowledge of her secret keys, and if her privileges are a superset of the ones required for the access she is attempting, access is granted. As discussed in [23] this approach violates Alice's privacy, as she is required to reveal *all* her privileges, when, theoretically, in order to gain access she should have had to reveal only a subset of them. Moreover another advantage of associating different keys to different privileges, is that the latter can be easily transferred simply by transferring the corresponding secret key.

known solutions. Given our RSA/Factoring instantiations of Algebraic OWP, this immediately solves the problem left open in [23] thus offering a batch verifiable Sigma protocol even for groups of unknown order.

Related Work. Apart from [23] we are not aware of other work dealing with batch execution of proofs of knowledge. There has been a lot of work on batching the computation of modular exponentiations (e.g. [6]). But the obvious application of such solution to Sigma-protocols would still yield a scheme with higher communication and computation cost by a factor of ℓ (the prover would still have to send and compute the ℓ initial commitments of the Sigma protocol).

2 Background and Definitions

In what follows we will denote with $\lambda \in \mathbb{N}$ a security parameter. We say that a function ϵ is negligible if it vanishes faster than the inverse of any polynomial. If S is a set, we denote with $x \xleftarrow{\$} S$ the process of selecting x uniformly at random in S. Let $\mathcal{A}$ be a probabilistic algorithm. We denote with $x \xleftarrow{\$} \mathcal{A}(\cdot)$ the process of running $\mathcal{A}$ on some appropriate input and assigning its output to x.

2.1 Algebraic Tools and Computational Assumptions

Let $\mathcal{G}(1^\lambda)$ be an algorithm that on input the security parameter 1^λ outputs a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ such that: p is a prime of size at least λ, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate bilinear map.

The co-Computational Diffie-Hellman problem was introduced by Boneh, Lynn and Shacham as a natural generalization of the Computational Diffie-Hellman problem in asymmetric bilinear groups [12]. It is defined as follows.

Definition 1 (co-CDH). *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$, $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ be generators, and $a, b \xleftarrow{\$} \mathbb{Z}_p$ be chosen at random. We define the advantage of an adversary $\mathcal{A}$ in solving the* co-Computational Diffie-Hellman *problem as*

$$\mathbf{Adv}_{\mathcal{A}}^{cdh}(\lambda) = \Pr[\mathcal{A}(p, g_1, g_2, g_1^a, g_2^b) = g_1^{ab}]$$

where the probability is taken over the random choices of $\mathcal{G}, a, b$ and the adversary $\mathcal{A}$. We say that the co-CDH Assumption holds for $\mathcal{G}$ if for every PPT algorithm $\mathcal{A}$ we have that $\mathbf{Adv}_{\mathcal{A}}^{cdh}(\lambda)$ is negligible.

Notice that in symmetric bilinear groups, where $\mathbb{G}_1 = \mathbb{G}_2$, this problem reduces to standard CDH. For asymmetric groups, it is also easy to see that co-CDH reduces to the computational Bilinear Diffie-Hellman problem [8].

We recall below the *decisional* version of the CDH Assumption for groups $\mathbb{G}$ of prime order p.

Definition 2 (DDH). *Let $\mathbb{G}$ be a group of prime order p, $g \in \mathbb{G}$ be a generator and $a, b, c \xleftarrow{\$} \mathbb{Z}_p$ be chosen at random. We define the advantage of an adversary $\mathcal{A}$ in deciding the* Decisional Diffie-Hellman *(DDH) problem as*

$$\mathbf{Adv}_{\mathcal{A}}^{ddh}(\lambda) = \left|\Pr[\mathcal{A}(p, g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(p, g, g^a, g^b, g^c) = 1]\right|$$

We say that the DDH Assumption holds in $\mathbb{G}$ if for every PPT algorithm $\mathcal{A}$: $\mathbf{Adv}_{\mathcal{A}}^{ddh}(\lambda)$ is negligible.

The RSA group Let $\mathbb{Z}_N^*$ be the group of invertible integers modulo N. A group element $g \in \mathbb{Z}_N^*$ can be efficiently sampled by choosing a random value in $\{0, \ldots, N-1\}$ and testing whether $gcd(g, N) = 1$. An element h is called a *quadratic residue* if $h = g^2 \bmod N$ for some $g \in \mathbb{Z}_N^*$. In our work we consider the subgroup $\mathbb{QR}_N \subset \mathbb{Z}_N^*$ of quadratic residues in $\mathbb{Z}_N^*$. Similarly to $\mathbb{Z}_N^*$, $\mathbb{QR}_N$ also allows to efficiently sample a group element: choose $g \xleftarrow{\$} \mathbb{Z}_N^*$ and compute $h = g^2 \bmod N$. For our convenience we consider moduli N which are product of "safe primes" $p \cdot q$. We recall that p is called a *safe prime* if $p = 2p' + 1$ and p' is also a prime number. Moreover, we assume that both p and q are congruent 3 mod 4 so that N is a so-called "Blum integer". In this case a few simple facts hold: $\mathbb{QR}_N$ is a cyclic group of order $p'q'$; almost any element of $\mathbb{QR}_N$ is a generator (unless it is 1 modulo p or q); every element $x \in \mathbb{QR}_N$ has four square roots in $\mathbb{Z}_N^*$, exactly one of which is in $\mathbb{QR}_N$, thus the squaring function $x^2 \bmod N$ is a permutation over $\mathbb{QR}_N$.

Let $\mathsf{RSAGen}(1^\lambda)$ be the following procedure. On input a security parameter λ, choose two random safe primes p and q of size at least λ, compute $N = pq$, and return (N, p, q).

Definition 3 (Factoring Assumption). *We define the advantage of an adversary $\mathcal{A}$ in factoring as:*

$$\mathbf{Adv}_{\mathcal{A}}^{fact}(\lambda) = \Pr[(N, p, q) \xleftarrow{\$} \mathsf{RSAGen}(1^\lambda); (p, q) \leftarrow \mathcal{A}(N)]$$

where the probability is taken over the random choices of RSAGen*, and the adversary. We say that the Factoring assumption holds for* RSAGen *if for every PPT algorithm $\mathcal{A}$: $\mathbf{Adv}_{\mathcal{A}}^{fact}(\lambda)$ is negligible.*

Definition 4 (RSA Assumption). *Let $(N, p, q) \xleftarrow{\$} \mathsf{RSAGen}(1^\lambda)$, τ be a random element in $\mathbb{Z}_N^*$ and $e \geq 3$ be a prime number such that $gcd(e, \phi(N)) = 1$. We define the advantage of an adversary $\mathcal{A}$ in solving the RSA problem as:*

$$\mathbf{Adv}_{\mathcal{A}}^{rsa}(\lambda) = \Pr[x \leftarrow \mathcal{A}(N, e, \tau) : x^e = \tau \bmod N]$$

where the probability is taken over the random choices of RSAGen*, τ and the adversary. We say that the RSA assumption holds for* RSAGen *if for every PPT algorithm $\mathcal{A}$ $\mathbf{Adv}_{\mathcal{A}}^{rsa}(\lambda)$ is negligible.*

According to the distribution from which e is chosen, there are several variants of the RSA assumption. In our work, we consider the case when e is some fixed prime. In this case we say that RSA holds for e.

Below we recall some results that will be useful in our proofs.

Lemma 1 (Shamir [43]). *Given $u, v \in \mathbb{Z}_N^*$ and integers $a, b \in \mathbb{Z}$ such that $u^a = v^b \bmod N$, it is possible to efficiently compute $z \in \mathbb{Z}_N^*$ such that $z^a = v^\gamma$ where $\gamma = gcd(a, b)$.*

Proof. The proof is a straightforward application of the extended Euclidean algorithm. One can indeed use this algorithm to compute integers c, d such that $ac + bd = \gamma = gcd(a, b)$. Finally, setting $z = u^d v^d$ gives the desired result and completes the proof.

Using the above lemma it is possible to show via a simple reduction that the RSA assumption in the subgroup $\mathbb{QR}_N \subset \mathbb{Z}_N^*$ is at least as hard as the RSA assumption in $\mathbb{Z}_N^*$.

We also recall the following result due to Rabin.

Lemma 2 (Rabin [40]). *Let N be an RSA modulus and τ be a random value in $\mathbb{QR}_N$. If there exists an efficient algorithm $\mathcal{A}$ that on input (N, τ) outputs a value $z \in \mathbb{Z}_N^*$ such that $z^2 = \tau \bmod N$ with probability ϵ, then it is possible to build an efficient algorithm $\mathcal{B}$ that on input N uses $\mathcal{A}$ to output its unique prime factorization with probability $\epsilon/2$.*

Finally, we observe that in the subgroup of quadratic residues $\mathbb{QR}_N$ where N is the product of two safe primes, the DDH assumption is assumed to hold (even if the factorization is revealed [31]).

2.2 Closed Form Efficient PRFs

The notion of closed form efficient pseudorandom functions was introduced in [7]. Their definition however seemed geared specifically towards the application of polynomial evaluation and therefore proved insufficient for our matrix multiplication protocol. Here we extend it to include any computations run on a set of pseudo-random values and a set of arbitrary inputs.

A closed form efficient PRF consists of algorithms $(\mathsf{PRF.KG}, \mathsf{PRF.F})$. The key generation $\mathsf{PRF.KG}$ takes as input the security parameter 1^λ, and outputs a secret key K and some public parameters pp that specify domain $\mathcal{X}$ and range $\mathcal{Y}$ of the function. On input $x \in \mathcal{X}$, $\mathsf{PRF.F}_K(x)$ uses the secret key K to compute a value $y \in \mathcal{Y}$. It must of course satisfy the usual pseudorandomness property. Namely, $(\mathsf{PRF.KG}, \mathsf{PRF.F})$ is secure if for every PPT adversary $\mathcal{A}$, the following difference is negligible:

$$\left|\Pr[\mathcal{A}^{\mathsf{PRF.F}_K(\cdot)}(1^\lambda, \mathsf{pp}) = 1] - \Pr[\mathcal{A}^{R(\cdot)}(1^\lambda, \mathsf{pp}) = 1]\right|$$

where $(K, \mathsf{pp}) \xleftarrow{\$} \mathsf{PRF.KG}(1^\lambda)$, and $R(\cdot)$ is a random function from $\mathcal{X}$ to $\mathcal{Y}$.

In addition, it is required to satisfy the following *closed-form efficiency* property. Consider an arbitrary computation Comp that takes as input ℓ random values $R_1, \ldots, R_\ell \in \mathcal{Y}$ and a vector of m arbitrary values $\boldsymbol{x} = (x_1, \ldots, x_m)$, and assume that the best algorithm to compute $\mathsf{Comp}(R_1, \ldots, R_\ell, x_1, \ldots, x_m)$ takes time T. Let $z = (z_1, \ldots, z_\ell)$ a ℓ-tuple of arbitrary values in the domain $\mathcal{X}$ of $\mathsf{PRF.F}$. We say that a PRF $(\mathsf{PRF.KG}, \mathsf{PRF.F})$ is *closed-form efficient* for (Comp, z) if there exists an algorithm $\mathsf{PRF.CFEval}_{\mathsf{Comp},z}$ such that

$$\mathsf{PRF.CFEval}_{\mathsf{Comp},z}(K, x) = \mathsf{Comp}(F_K(z_1), \ldots, F_K(z_\ell), x_1, \ldots, x_m)$$

and its running time is $o(T)$. For $z = (1, \ldots, \ell)$ we usually omit the subscript z.

Note that depending on the structure of Comp, this property may enforce some constraints on the range $\mathcal{Y}$ of $\mathsf{PRF.F}$. In particular in our case, $\mathcal{Y}$ will be an abelian group. We also remark that due to the pseudorandomness property the output distribution of $\mathsf{PRF.CFEval}_{\mathsf{Comp},z}(K, x)$ (over the random choice of K) is indistinguishable from the output distribution of $\mathsf{Comp}(R_1, \ldots, R_\ell, x_1, \ldots, x_m)$ (over the random choices of the R_i).

In this paper we do not introduce new PRFs with closed form efficiency but we use previous proposals (in one case with a small modification). For the CDH-based solution we use the PRFs based on the Decision Linear Assumption described in [19].

For the RSA/Factoring based solutions we use the PRF constructions described in [7] that are based on the Naor-Reingold PRF [37]. The only difference is that in our case we have to instantiate the PRFs in the group $\mathbb{QR}_N$, and thus claim their security under the hardness of DDH in the group $\mathbb{QR}_N$.

2.3 Verifiable Computation

A verifiable computation scheme is a tuple of distributed algorithms that enable a client to outsource the computation of a function f to an untrusted worker, in such a way that the client can verify the correctness of the result returned by the worker. In order for the outsourcing to make sense, it is crucial that the cost of verification at the client must be cheaper than computing the function locally.

In our work we are interested in computation schemes that are *publicly verifiable* as defined by Parno *et al.* [39]: any third party (possibly different from the delegator) can verify the correctness of the results returned by the worker.

Let $\mathcal{F}$ be a family of functions. A Verifiable Computation scheme $\mathcal{VC}$ for $\mathcal{F}$ is defined by the following algorithms:

$\mathsf{KeyGen}(1^\lambda, f) \rightarrow (\mathsf{SK}_f, \mathsf{PK}_f, \mathsf{EK}_f)$**:** on input a function $f \in \mathcal{F}$, it produces a secret key SK_f that will be used for input delegation, a public verification key PK_f, used to verify the correctness of the delegated computation, and a public evaluation key EK_f which will be handed to the server to delegate the computation of f.

$\mathsf{ProbGen}(\mathsf{PK}_f, \mathsf{SK}_f, x) \rightarrow (\sigma_x, \mathsf{VK}_x)$**:** given a value $x \in \mathsf{Dom}(f)$, the problem generation algorithm is run by the delegator to produce an encoding σ_x of x, together with a public verification key VK_x.

$\mathsf{Compute}(\mathsf{EK}_f, \sigma_x) \rightarrow \sigma_y$**:** given the evaluation key EK_f and the encoding σ_x of an input x, this algorithm is run by the worker to compute an encoded version of $y = f(x)$.

$\mathsf{Verify}(\mathsf{PK}_f, \mathsf{VK}_x, \sigma_y) \rightarrow y \cup \perp$**:** on input the public key PK_f, the verification key VK_x, and an encoded output σ_y, this algorithm returns a value y or an error $\perp$.

Correctness. Informally, a verifiable computation scheme $\mathcal{VC}$ is *correct* if the values generated by the problem generation algorithm allows a honest worker to output values that will verify correctly. More formally, for any $f \in \mathcal{F}$, any $(\mathsf{SK}_f, \mathsf{PK}_f, \mathsf{EK}_f) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda, f)$, any $x \in \mathsf{Dom}(f)$, if $(\sigma_x, \mathsf{VK}_x) \overset{\$}{\leftarrow} \mathsf{ProbGen}(\mathsf{PK}_f, \mathsf{SK}_f, x)$ and $\sigma_y \leftarrow \mathsf{Compute}(\mathsf{EK}_f, \sigma_x)$, then $f(x) \leftarrow \mathsf{Verify}(\mathsf{PK}_f, \mathsf{VK}_x, \sigma_y)$ holds with all but negligible probability.

Security. For any verifiable computation scheme $\mathcal{VC}$, let us define the following experiment:

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{PubVer}}[\mathcal{VC}, f, \lambda]$

$(\mathsf{SK}_f, \mathsf{PK}_f, \mathsf{EK}_f) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda, f)$
For $i = 1$ to q:
$x_i \leftarrow \mathcal{A}(\mathsf{PK}_f, \mathsf{EK}_f, \sigma_{x,1}, \mathsf{VK}_{x,1}, \ldots, \sigma_{x,i-1}, \mathsf{VK}_{x,i-1})$
$(\sigma_{x,i}, \mathsf{VK}_{x,i}) \overset{\$}{\leftarrow} \mathsf{ProbGen}(\mathsf{SK}_f, x_i)$
$x^* \leftarrow \mathcal{A}(\mathsf{PK}_f, \mathsf{EK}_f, \sigma_{x,1}, \mathsf{VK}_{x,1}, \ldots, \sigma_{x,q}, \mathsf{VK}_{x,q})$
$(\sigma_{x^*}, \mathsf{VK}_{x^*}) \overset{\$}{\leftarrow} \mathsf{ProbGen}(\mathsf{SK}_f, x^*)$
$\hat{\sigma}_y \leftarrow \mathcal{A}(\mathsf{PK}_f, \mathsf{EK}_f, \sigma_{x,1}, \mathsf{VK}_{x,1}, \ldots, \sigma_{x,q}, \mathsf{VK}_{x,q}, \mathsf{VK}_{x^*})$
$\hat{y} \leftarrow \mathsf{Verify}(\mathsf{PK}_f, vk_{x^*}, \hat{\sigma}_y)$
If $\hat{y} \neq \perp$ and $\hat{y} \neq f(x^*)$, output 1, else output 0.

For any $\lambda \in \mathbb{N}$, any function $f \in \mathcal{F}$, we define the advantage of an adversary $\mathcal{A}$ making at most $q = poly(\lambda)$ queries in the above experiment against $\mathcal{VC}$ as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{PubVer}}(\mathcal{VC}, f, q, \lambda) = \Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{PubVer}}[\mathcal{VC}, f, \lambda] = 1].$$

Definition 5. *A verifiable computation scheme $\mathcal{VC}$ is secure for $\mathcal{F}$ if for any $f \in \mathcal{F}$, and any PPT $\mathcal{A}$ it holds that $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{PubVer}}(\mathcal{VC}, f, q, \lambda)$ is negligible.*

Note that our definition captures full adaptive security, where the adversary decides "on the fly" on which input x^* it will try to cheat. The weaker selective security notion requires the adversary to commit to x^* at the beginning of the game.

2.4 Linearly-Homomorphic Signatures

Digital signature schemes allow a user to create a signature σ on a message m (in some appropriate set $\mathcal{M}$), such that any other user knowing only a public verification key PK can verify the validity of σ on m. Boneh and Freeman [10] recently introduced the notion of homomorphic signatures which extends regular signatures as follows: given a set of signatures $(\sigma_1, \sigma_2, \ldots, \sigma_m)$, corresponding set of messages $(M_1, M_2, \ldots, M_m) \in \mathcal{M}^m$, and a function f (in an appropriate set $\mathcal{F} = \{f | f : \mathcal{M}^m \to \mathcal{M}\}$) any user can produce a valid signature on the message $f(M_1, M_2, \ldots, M_m)$. Furthermore, any message M can be verified against a signature σ as well as a function f. A *linearly homomorphic signature* scheme is a homomorphic signature scheme where the only admissible functions f are linear, i.e. $\mathcal{F} = \{f : \mathcal{M}^m \to \mathcal{M} | f \text{ is linear}\}$.

We recall below the formal notion of linearly-homomorphic signatures, as defined by Freeman in [20].

Definition 6 (Linearly-Homomorphic Signatures). *A linearly-homomorphic signature scheme is a tuple of probabilistic, polynomial-time algorithms* $(\mathsf{Hom.KG}, \mathsf{Hom.Sign}, \mathsf{Hom.Ver}, \mathsf{Hom.Eval})$ *with the following properties:*

$\mathsf{Hom.KG}(1^\lambda, m)$ *takes a security parameter λ, a maximum data set size m, and outputs a public key PK and a secret key SK. The public key PK defines implicitly a message space $\mathcal{M}$, a signature space Σ, and a set $\mathcal{F}$ of admissible linear functions, that in our case is $\mathcal{F} = \{f : \mathcal{M}^n \to \mathcal{M} | f \text{ is linear}\}$.*

$\mathsf{Hom.Sign}(\mathsf{SK}, \tau, M, i)$ *takes a secret key SK, a tag τ, a message $M \in \mathcal{M}$ and an index $i \in \{1, 2, \ldots, m\}$. It outputs a signature $\sigma \in \Sigma$.*

$\mathsf{Hom.Ver}(\mathsf{VK}, \tau, M, \sigma, f)$ *takes a public key PK, a tag τ, a message $M \in \mathcal{M}$, a signature $\sigma \in \Sigma$, and a function $f \in \mathcal{F}$. It outputs either 0 (reject) or 1 (accept).*

$\mathsf{Hom.Eval}(\mathsf{VK}, \tau, f, \boldsymbol{\sigma})$ *takes a public key PK, a tag τ, a function $f \in \mathcal{F}$, and a tuple of signatures $\{\sigma_i\}_{i=1}^m$. It outputs a new signature $\sigma' \in \Sigma$.*

In order to define the correctness we first fix some notation. We denote by π_i the projection function $\pi_i : X^m \to X$, where $X \in \{\mathcal{M}, \Sigma, \mathcal{F}\}$, as follows: $\pi_i(x_1, x_2 \ldots, x_m) = x_i$.

Informally speaking, a linearly-homomorphic signature scheme is **correct** if: (i) the signature on any initial message with index i as output by $\mathsf{Hom.Sign}$ must verify correctly against the corresponding projection function π_i; (ii) if any vector of signatures $\boldsymbol{\sigma}$ verifies correctly on respective messages $\boldsymbol{M}$, then the output of $\mathsf{Hom.Eval}(\mathsf{VK}, \tau, f, \boldsymbol{\sigma})$ should verify correctly for $f(M_1, M_2, \ldots, M_m)$.

More formally, for correctness we require that:

1. For all public keys $(\mathsf{PK}, \mathsf{SK}) \xleftarrow{\$} \mathsf{Hom.KG}(1^\lambda, m)$, any tag τ, any message $M \in \mathcal{M}$, any index $i \in \{1, 2, \ldots, m\}$ and any signature $\sigma \xleftarrow{\$} \mathsf{Hom.Sign}(\mathsf{SK}, \tau, M, i)$, $\mathsf{Hom.Ver}(\mathsf{VK}, \tau, M, \sigma, f) = 1$ holds with overwhelming probability.
2. For all public keys $(\mathsf{PK}, \mathsf{SK}) \xleftarrow{\$} \mathsf{Hom.KG}(1^\lambda, m)$, any tag τ the following holds with overwhelming probability as well. Suppose a message-vector $\boldsymbol{\mu} \in \mathcal{M}^m$, a function-vector $\boldsymbol{f} \in \mathcal{F}^m$ and a signature-vector $\boldsymbol{\sigma}$ are such that for all $i = 1, \ldots, m$

$$\mathsf{Hom.Ver}(\mathsf{VK}, \tau, M_i = f_i(\mu_1, \mu_2, \ldots, \mu_m), \sigma_i, f_i) = 1.$$

Then, the following must hold with overwhelming probability:

$$\mathsf{Hom.Ver}(\mathsf{VK}, \tau, g(M_1, M_2, \ldots, M_m), \mathsf{Eval}(\mathsf{VK}, \tau, g, \boldsymbol{M}, \boldsymbol{\sigma}), g \circ f) = 1,$$

where $g \circ f : \mathcal{M}^m \to \mathcal{M}$ is defined as $[g \circ \boldsymbol{f}(\mu_1, \mu_2, \ldots, \mu_m)]_i = \pi_i(g(f_1(\boldsymbol{\mu}), f_2(\boldsymbol{\mu}), \ldots, f_m(\boldsymbol{\mu})))$, so that

$$g \circ \boldsymbol{f}(\mu_1, \mu_2, \ldots, \mu_m) = g(M_1, M_2, \ldots, M_m).$$

Security of linearly-homomorphic signatures Recall that in a linearly homomorphic signature scheme, given valid signatures on a set of messages $M_1, M_2, \ldots, M_m$, anyone (only with the knowledge of the public key) can produce valid signatures on any message $M = f(M_1, M_2, \ldots, M_m)$, for some linear function f. In particular, in order for the homomorphic property to work, these messages must be in the same "data set", which is identified by a tag τ. Freeman recently proposed in [20] a security notion for linearly-homomorphic signatures, which is stronger than the ones proposed by earlier works, such as [9, 22, 10, 15]. In our work we adopt this definition. Informally, the goal of the adversary is to produce a signature on a message M that cannot be obtained by applying functions on previously observed data sets. This means, that the forgery is either a signature for a new data set (Type 1 forgery), or it is a signature on a previously observed data set $(M_1, \ldots, M_m)$, but on an incorrect value, i.e., a value which is not obtained by applying $f(M_1, \ldots, M_m)$.

More formally, we define the following security game:

Key generation The challenger runs $(\mathsf{PK}, \mathsf{SK}) \xleftarrow{\$} \mathsf{Hom.KG}(1^\lambda, m)$ and gives PK to the adversary.

Queries The adversary submits queries of the form (F, i, M), where F is a filename (i.e., an identifier for the data set), $i \in \{1, \ldots, m\}$, and $M \in \mathcal{M}$. For each queried file name F, the challenger generates a tag τ_F and keeps a state so that he returns the same τ_F next time the same F is queried. The challenger computes $\sigma \xleftarrow{\$} \mathsf{Hom.Sign}(\mathsf{SK}, \tau_F, M, i)$ and returns the tag τ_F together with the signature σ. The challenger also keeps a state of the indices i queried for each file F so that it rejects queries of the form (F, i, M) if (F, i, M') has been queried before for some message $M' \neq M$, and it returns the same signature as before if $M = M'$.

This stage is repeated a polynomial number of times. At the end of the querying stage the challenger (and the adversary) have a list of states with file names F_j and corresponding tags τ_j; and for each file name F_k there is also a list of indices i with corresponding messages M_i for $0 \leq i \leq m$.

Forgery The adversary outputs a tuple $(\tau^*, M^*, \sigma^*, f^*)$

In order to define all possible forgeries we need to fix some notation. We denote by i_F the number of messages asked for the data set with filename F. A function f is said to be *well-defined on* F if either $i_F = m$, or $i_F < m$ and

$$f(M_1, \ldots, M_{i_F}, M_{i_F+1}, \ldots, M_m)$$

takes the same value for all possible choices of $(M_{i_F+1}, \ldots, M_m) \in \mathcal{M}^{m-i_F}$.

The adversary wins the game if $\mathsf{Hom.Ver}(\mathsf{VK}, \tau^*, M^*, \sigma^*, f^*) = 1$ and any of the following holds:

1. $\tau^* \neq \tau_j$ for all τ_j chosen by the challenger
2. $\tau^* = \tau_j$ for some τ_j chosen by the challenger, corresponding to file name F_j and set of $(M_1, M_2, \ldots, M_m)$ queried with that file in total. Then for the adversary to win it must be that $M^* \neq f^*(M_1, M_2, \ldots, M_m)$.

3. $\tau^* = \tau_j$ for some τ_j chosen by the challenger, corresponding to file name F_j and set of $(M_1, M_2, \ldots, M_k)$ queried with that file in total. Then the adversary to win it must be that f^* is not well-defined on F_j.

It has been shown in [20] that for linearly-homomorphic schemes Type 3 forgeries reduce to Type 2. Therefore, in our work we will focus only on Type 1 and Type 2 forgeries.

We define the advantage $\mathbf{Adv}_{\mathcal{A}}^{LHS}(\lambda)$ of an adversary against a linearly-homomorphic signature scheme as the probability of $\mathcal{A}$ winning the above game.

Definition 7 (Unforgeability of Linearly Homomorphic Signatures [20]). *A linearly-homomorphic signature scheme is* unforgeable *if for all m the advantage $\mathbf{Adv}_{\mathcal{A}}^{LHS}(\lambda)$ of a any PPT algorithm $\mathcal{A}$ is negligible.*

2.5 *Σ*-protocols

Let L be an NP language with associated relation $\mathcal{R}$. Informally, a Σ-protocol for $\mathcal{R}$ is a two party (interactive) protocol, consisting of 3 rounds of communications and involving two parties: an (honest) prover P and an (honest) verifier V. Both P and V start with some common input statement of the form $x \in L$, where L is an NP language. The private input for P is a *witness* $w \in \{0,1\}^{p(|x|)}$ (where $p(\cdot)$ is some polynomial), certifying the fact that $x \in L$ (i.e., such that $(x, w) \in \mathcal{R}$). At the end of the protocol V should be able to efficiently decide whether the produced transcript is accepting with respect to the statement or not.

More formally, a Σ-protocol for a relation $\mathcal{R}$ consists of algorithms $(\Sigma.\mathsf{Setup}, \Sigma.\mathsf{Com}, \Sigma.\mathsf{Resp}, \Sigma.\mathsf{Ver})$ such that:

- $\Sigma.\mathsf{Setup}(1^\lambda, \mathcal{R}) \to (x, w)$ is a PPT algorithm that on input the security parameter and a relation $\mathcal{R}$ outputs a statement x and a witness w such that $(x, w) \in \mathcal{R}$.
- $\Sigma.\mathsf{Com}(x; r) \to R$ is a PPT algorithm run by the prover that on input the public value x and random coins r in some appropriate randomness space RndSp, outputs the first message R of the protocol.
- $\Sigma.\mathsf{Resp}(x, w, r, c) \to s$ is a PPT algorithm that is run by the prover to compute the third message s of the Σ-protocol. The algorithm takes as input the pair (x, w) generated by $\Sigma.\mathsf{Setup}$, random coins $r \in \mathsf{RndSp}$, and the second message of the verifier $c \in \mathsf{ChSp}$. Here ChSp denotes the challenge space.
- $\Sigma.\mathsf{Ver}(x, R, c, s) \to 0/1$ is the verification algorithm that on input the message R, a challenge $c \in \mathsf{ChSp}$ and a response s, outputs 1 (accept) or 0 (reject).

Here we will focus on Σ-protocols having the following properties

Completeness. $\forall (x, w) \xleftarrow{\$} \Sigma.\mathsf{Setup}(1^\lambda, \mathcal{R})$, any $R \xleftarrow{\$} \Sigma.\mathsf{Com}(x, r)$ for $r \xleftarrow{\$} \mathsf{RndSp}$, any $c \in \mathsf{ChSp}$, and $s \xleftarrow{\$} \Sigma.\mathsf{Resp}(x, w, r, c)$,

$$\Sigma.\mathsf{Ver}(x, R, c, s) = 1$$

holds with overwhelming probability.

Special Soundness. There exists an extractor algorithm $\Sigma.\mathsf{Ext}$ such that $\forall x \in L$, $\forall R, c, s, c', s'$ such that $\Sigma.\mathsf{Ver}(x, R, c, s) = 1$ and $\Sigma.\mathsf{Ver}(x, R, c', s') = 1$, $\Sigma.\mathsf{Ext}(x, R, c, s, c', s') = w'$ such that $(x, w') \in \mathcal{R}$.

Special HVZK. There exists a simulator Sim such that $\forall c \in \mathsf{ChSp}$, $\mathsf{Sim}(x, c)$ generates a pair (R, s) such that $\Sigma.\mathsf{Ver}(x, R, c, s) = 1$ and the probability distribution of (R, c, s) is identical to that obtained by running the real algorithms.

3 Algebraic (Trapdoor) One-Way Functions

A family of one-way functions consists of two efficient algorithms (Gen, F) that work as follows. $\mathsf{Gen}(1^\lambda)$ takes as input a security parameter 1^λ and outputs a key κ. Such key κ determines a member $F_\kappa(\cdot)$ of the family, and in particular it specifies two sets $\mathcal{X}_\kappa$ and $\mathcal{Y}_\kappa$ such that $F_\kappa : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa$. Given κ, on any input $x \in \mathcal{X}_\kappa$, the function $F_\kappa(x) = y$ is efficiently computable and $y \in \mathcal{Y}_\kappa$. In addition, we also assume that κ specifies a finite field $\mathbb{K}$ that will be used as described below.

(Gen, F) is a family of *algebraic one-way functions* if it satisfies the following properties:

Algebraic: $\forall \lambda \in \mathbb{N}$, and every $\kappa \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, the sets $\mathcal{X}_\kappa$, $\mathcal{Y}_\kappa$ are abelian cyclic groups. In our work we denote the group operation by multiplication, and we assume that given κ, sampling a (random) generator as well as computing the group operation can be done efficiently (in PPT).

Homomorphic: $\forall \lambda \in \mathbb{N}$, every $\kappa \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, for any inputs $x_1, x_2 \in \mathcal{X}_\kappa$, it holds:

$$F_\kappa(x_1) \cdot F_\kappa(x_2) = F_\kappa(x_1 \cdot x_2)$$

Field-homomorphic: intuitively, this property states that it is possible to evaluate inner product operations in the exponent given some "blinded" bases. More precisely, let $\kappa \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, $h_1, \ldots, h_m \in \mathcal{X}_\kappa$ be generators (for $m \geq 1$), and let $W_1, \ldots, W_\ell \in \mathcal{X}_\kappa$ be group elements, each of the form $W_i = h_1^{\omega_i^{(1)}} \cdots h_m^{\omega_i^{(m)}} \cdot R_i$, for some $R_i \in \mathcal{X}_\kappa$ and some integers $\omega_i^{(j)} \in \mathbb{Z}$ (note that this decomposition may not be unique).

We say that (Gen, F) is field-homomorphic (for the field $\mathbb{K}$ specified by κ) if there exists an efficient algorithm Eval such that for any $\kappa \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, any set of generators $h_1, \ldots, h_m \in \mathcal{X}_\kappa$, any vector of elements $\boldsymbol{W} \in \mathcal{X}_\kappa^\ell$ of the above form, and any vector of integers $\boldsymbol{\alpha} \in \mathbb{Z}^\ell$, it holds

$$\mathsf{Eval}(\kappa, \boldsymbol{A}, \boldsymbol{W}, \boldsymbol{\Omega}, \boldsymbol{\alpha}) = h_1^{\langle \boldsymbol{\omega}^{(1)}, \boldsymbol{\alpha} \rangle} \cdots h_m^{\langle \boldsymbol{\omega}^{(m)}, \boldsymbol{\alpha} \rangle} \prod_{i=1}^{\ell} R_i^{\alpha_i}$$

where $\boldsymbol{A} = (A_1, \ldots, A_m) \in \mathcal{Y}_\kappa^m$ is such that $A_i = F_\kappa(h_i)$, $\boldsymbol{\Omega} = (\omega_i^{(j)})_{i,j} \in \mathbb{Z}^{\ell \times m}$, and each product $\langle \boldsymbol{\omega}^{(j)}, \boldsymbol{\alpha} \rangle$ in the exponent is computed over the field $\mathbb{K}$. We notice that over all the paper we often abuse notation by threating elements of the field $\mathbb{K}$ as integers and viceversa. For this we assume a canonical interpretation of $d \in \mathbb{K}$ as an integer $[d] \in \mathbb{Z}$ between 0 and $|\mathbb{K}| - 1$, and that both d and $[d]$ are efficiently computable from one another.

We note that in the case when the field $\mathbb{K}$ is $\mathbb{Z}_p$, where p is the order of the group $\mathcal{X}_\kappa$, then this property is trivially realized: *every* OWF where $\mathcal{X}_\kappa$ is a group of order p, is field-homomorphic for $\mathbb{Z}_p$. To see this, observe that the following efficient algorithm trivially follows from the simple fact that $\mathcal{X}_\kappa$ is a finite group:

$$\overline{\mathsf{Eval}}(\kappa, \boldsymbol{A}, \boldsymbol{W}, \boldsymbol{\Omega}, \boldsymbol{\alpha}) = \prod_{i=1}^{\ell} W_i^{\alpha_i}$$

What makes the property non-trivial for some instantiations (in particular the RSA and Factoring based ones shown in the next section) is that the algorithm Eval must compute the inner products $\langle \boldsymbol{\omega}^{(j)}, \boldsymbol{\alpha} \rangle$ over the field $\mathbb{K}$, which might be different from $\mathbb{Z}_p$, where p is the order of the group $\mathcal{X}_\kappa$ over which the function is defined.

Strongly One-way: finally, we require a family (Gen, F) to be non-invertible in a strong sense. Formally, we say that (Gen, F) is *strongly one-way* if for any PPT adversary $\mathcal{A}$ it holds:

$$\Pr[\mathcal{A}(1^\lambda, \kappa, y) = (x', d) : d \neq 0 \wedge d \in \mathbb{K} \wedge F_\kappa(x') = y^d]$$

is negligible, where $\kappa \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, $x \xleftarrow{\$} \mathcal{X}_\kappa$ is chosen uniformly at random and $y = F_\kappa(x)$.
Our definition asks for $d \neq 0$ as we additionally require that in the case when $d = 0$ (over the field $\mathbb{K}$) the function must be efficiently invertible. More precisely, given a value $y = F_\kappa(x) \in \mathcal{Y}_\kappa$ (for any $x \in \mathcal{X}_\kappa$) and an integer d such that $d = 0$ over the field $\mathbb{K}$ (d may though be different from zero over the integers), there is an efficient algorithm that computes $x' \in \mathcal{X}_\kappa$ such that $F_\kappa(x') = y^d$.
This notion of one-wayness is stronger than the standard one (in which d is basically 1). Even though this property may look non-standard, in the next section we demonstrate that our candidates satisfy it under very simple and standard assumptions.

Algebraic Trapdoor One-Way Functions. Our notion of algebraic one-way functions can be easily extended to the trapdoor case, in which there exists a trapdoor key that allows to efficiently invert the function. More formally, we define a family of *trapdoor one-way functions* as a set of efficient algorithms $(\mathsf{Gen}, F, \mathsf{Inv})$ that work as follows. $\mathsf{Gen}(1^\lambda)$ takes as input a security parameter 1^λ and outputs a pair (κ, td). Given κ, F_κ is the same as before. On input the trapdoor td and a value $y \in \mathcal{Y}_\kappa$, the inversion algorithm Inv computes $x \in \mathcal{X}_\kappa$ such that $F_\kappa(x) = y$. Often we will write $\mathsf{Inv}_{\mathsf{td}}(\cdot)$ as $F_\kappa^{-1}(\cdot)$. Then we say that $(\mathsf{Gen}, F, \mathsf{Inv})$ is a family of *algebraic trapdoor one-way functions* if it is algebraic, homomorphic and field-homomorphic, in the same way as defined above.

Finally, when the input space $\mathcal{X}_\kappa$ and the output space $\mathcal{Y}_\kappa$ are the same (i.e., $\mathcal{X}_\kappa = \mathcal{Y}_\kappa$) and the function $F_\kappa : \mathcal{X}_\kappa \to \mathcal{X}_\kappa$ is a permutation, then we call $(\mathsf{Gen}, F, \mathsf{Inv})$ a family of *algebraic trapdoor permutations.*

3.1 Instantiations

We give three simple constructions of algebraic (trapdoor) one-way functions from a variety of number theoretic assumptions: CDH in bilinear groups, RSA and factoring.

CDH in Bilinear Groups

$\mathsf{Gen}(1^\lambda)$: use $\mathcal{G}(1^\lambda)$ to generate groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of the same prime order p, together with an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Sample two random generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and output $\kappa = (p, e, g_1, g_2)$. The finite field $\mathbb{K}$ is $\mathbb{F}_p$.
$F_\kappa(x)$: the function $F_\kappa : \mathbb{G}_1 \to \mathbb{G}_T$ is defined by:

$$F_\kappa(x) = e(x, g_2)$$

The algebraic and homomorphic properties are easy to check. Moreover, the function is trivially field-homomorphic for $\mathbb{F}_p$ as p is the order of $\mathbb{G}_1$.

Its security can be shown via the following Theorem:

Theorem 1. *If the co-CDH assumption holds for $\mathcal{G}(\cdot)$, then the above function is strongly one-way.*

The proof can be obtained via a straightforward reduction. Given a co-CDH instance $(p, g_1, g_2, g_1^a, g_2^b)$ compute $y = e(g_1^a, g_2^b)$ and run $\mathcal{A}$ on input (p, g_1, g_2, y). If $\mathcal{A}$ returns $(x, d) \in \mathbb{G}_1 \times \mathbb{Z}_p$ such that $e(x, g_2) = y^d$, then compute $g_1^{ab} = x^{1/d}$.

Since $\mathbb{K} = \mathbb{F}_p$, for $d = 0 \bmod p$ computing a pre-image of y^d is trivial, i.e., $1_{\mathbb{G}_1}$.

RSA (over $\mathbb{QR}_N$) This construction is an algebraic trapdoor permutation, and it allows to explicitly choose the field $\mathbb{K}$, which can be any finite field $\mathbb{F}_e$ of prime characteristic $e \geq 3$.

$\mathsf{Gen}(1^\lambda, e)$**:** let $e \geq 3$ be a prime number. Run $(N, p, q) \xleftarrow{\$} \mathsf{RSAGen}(1^\lambda)$ to generate a Blum integer N, product of two safe primes p and q. If $gcd(e, \phi(N)) \neq 1$, then reject the tuple (N, p, q) and try again. Output $\kappa = (N, e)$ and $\mathsf{td} = (p, q)$.

$F_\kappa(x)$**:** the function $F_\kappa : \mathbb{QR}_N \to \mathbb{QR}_N$ is defined by:

$$F_\kappa(x) = x^e \bmod N$$

$\mathsf{Inv}_{\mathsf{td}}(y)$**:** the inversion algorithm computes $c = e^{-1} \bmod \phi(N)$, and then outputs:

$$\mathsf{Inv}_{\mathsf{td}}(y) = x^c \bmod N$$

$\mathsf{Eval}(\kappa, \boldsymbol{A}, \boldsymbol{W}, \boldsymbol{\Omega}, \boldsymbol{\alpha})$**:** for $j = 1$ to m, compute $\omega^{(j)} = \langle \boldsymbol{\omega}^{(j)}, \boldsymbol{\alpha} \rangle$ over the integers and write it as $\omega^{(j)} = \omega^{(j)\prime} + e\omega^{(j)\prime\prime}$, for some $\omega^{(j)\prime}, \omega^{(j)\prime\prime} \in \mathbb{Z}$. Finally, output

$$V = \frac{\prod_{i=1}^{\ell} W_i^{\alpha_i}}{\prod_{j=1}^{m} A_j^{\omega^{(j)\prime\prime}}} \bmod N$$

The algebraic and homomorphic properties are easy to check. To see that the function is field-homomorphic for $\mathbb{K} = \mathbb{F}_e$, we show the correctness of the Eval algorithm as follows:

$$\begin{aligned}
V &= \frac{\prod_{i=1}^{\ell} W_i^{\alpha_i}}{\prod_{j=1}^{m} A_j^{\omega^{(j)\prime\prime}}} \bmod N = \frac{\prod_{i=1}^{l} (\prod_{j=1}^{m} h_j^{\omega_i^{(j)}} \cdot R_i)^{\alpha_i}}{\prod_{j=1}^{m} h_j^{(e\omega^{(j)\prime\prime} \bmod \phi(N))}} \bmod N \\
&= \frac{\prod_{j=1}^{m} h_j^{(\langle \boldsymbol{\omega}^{(j)}, \boldsymbol{\alpha} \rangle \bmod \phi(N))} \prod_{i=1}^{l} R_i^{\alpha_i}}{\prod_{j=1}^{m} h_j^{(e\omega^{(j)\prime\prime} \bmod \phi(N))}} \bmod N \\
&= \frac{\prod_{j=1}^{m} h_j^{(\omega^{(j)\prime} + e\omega^{(j)\prime\prime} \bmod \phi(N))} \prod_{i=1}^{l} R_i^{\alpha_i}}{\prod_{j=1}^{m} h_j^{(e\omega^{(j)\prime\prime} \bmod \phi(N))}} \bmod N \\
&= h_1^{\omega^{(1)\prime}} \cdots h_m^{\omega^{(m)\prime}} \prod_{i=1}^{l} R_i^{\alpha_i} \bmod N.
\end{aligned}$$

The security of the function is shown via the following Theorem:

Theorem 2. *If the RSA assumption holds for* RSAGen*, the above function is strongly one-way.*

To prove the theorem, we simply observe that since $d \neq 0$ and $d \in \mathbb{F}_e$, it holds $gcd(e, d) = 1$. Therefore, it is possible to apply the result of Lemma 1 to transform any adversary against the security of our OWF to an adversary which solves the RSA problem for the fixed e.

On the other hand, given $y \in \mathcal{Y}_\kappa$, in the special case when $d = 0 \bmod e$, finding a pre-image of y^d can be done efficiently by computing $y^{d'}$ where d' is the integer such that $d = e \cdot d'$.

Factoring Also this construction allows to explicitly choose the field $\mathbb{K}$, which can be any binary field $\mathbb{F}_{2^t}$ for any integer $t \geq 1$.

$\mathsf{Gen}(1^\lambda, t)$**:** run $(N, p, q) \xleftarrow{\$} \mathsf{RSAGen}(1^\lambda)$ to generate a Blum integer N product of two safe primes p and q. Output $\kappa = (N, t)$ and $\mathsf{td} = (p, q)$.

$F_\kappa(x)$**:** The function $F_\kappa : \mathbb{QR}_N \to \mathbb{QR}_N$ is defined by:

$$F_\kappa(x) = x^{2^t} \bmod N$$

$\mathsf{Inv}_{\mathsf{td}}(y)$**:** given $\mathsf{td} = (p, q)$ and on input $y \in \mathbb{QR}_N$, the inversion algorithm proceeds as follows. First, it uses the factorization of N to compute the four square roots $x, -x, x', -x' \in \mathbb{Z}_N^*$ of y, and then it outputs the only one which is in $\mathbb{QR}_N$ (recall that since N is a Blum integer exactly one of the roots of y is a quadratic residue).

$\mathsf{Eval}(\kappa, \boldsymbol{A}, \boldsymbol{W}, \boldsymbol{\omega}, \boldsymbol{\alpha})$**:** for $j = 1$ to m, compute $\omega^{(j)} = \langle \boldsymbol{\omega}^{(j)}, \boldsymbol{\alpha} \rangle$ over the integers and write it as $\omega^{(j)} = \omega^{(j)\prime} + 2^t \omega^{(j)\prime\prime}$. Finally, output

$$V = \frac{\prod_{i=1}^{\ell} W_i^{\alpha_i}}{\prod_{j=1}^{m} A_j^{\omega^{(j)\prime\prime}}} \bmod N$$

The algebraic and homomorphic properties are easy to check. To see that the function is field-homomorphic for $\mathbb{F}_{2^t}$, observe that its correctness can be checked similarly to the RSA case. We notice that this construction is an algebraic trapdoor permutation.

The security of the function can be shown via the following Theorem:

Theorem 3. *If Factoring holds for* RSAGen*, then the above function is strongly one-way.*

Proof. To prove the theorem, we first show that any adversary $\mathcal{A}$ who is able to break the one-wayness of this construction with probability ϵ can be used to build an adversary $\mathcal{B}$ that computes square roots with the same probability. Then, by applying Lemma 2, we finally obtain an adversary who can factor N with probability $\epsilon/2$.

Let (N, τ) be $\mathcal{B}$'s input such that $\tau \in \mathbb{QR}_N$. $\mathcal{B}$ sets $\kappa = N$ and runs the adversary $\mathcal{A}(N, \tau)$. Let us suppose that $\mathcal{A}$ returns a pair $(x, d) \in \mathbb{QR}_N \times \mathbb{F}_{2^t}$ such that $x^{2^t} = \tau^d$. Since $d \in \mathbb{F}_{2^t}$, we can write $d = 2^\ell \cdot u$, for some $0 \leq \ell < t$ and an odd integer u. By applying Lemma 1 we can then compute a value v such that $v^{2^t} = \tau^{gcd(2^t, 2^\ell \cdot u)} = \tau^{2^\ell} \bmod N$. Since $\ell < t$ and the squaring function is a permutation over $\mathbb{QR}_N$ (as N is a Blum integer), it holds $v^{2^{t-\ell}} = \tau \bmod N$. Therefore, $\mathcal{B}$ computes $z = v^{2^{t-\ell-1}} \bmod N$ and returns z. It is easy to see that under the assumption that $\mathcal{A}$'s output is correct, it holds $z^2 = \tau \bmod N$.

Finally, similarly to the RSA case, given $y \in \mathcal{Y}_\kappa$, in the special case when $d = 0 \bmod 2^t$, finding a pre-image of y^d can be done efficiently by computing $y^{d'}$ where d' is the integer such that $d = 2^t d'$.

4 Our Verifiable Computation Schemes

In this section we propose the construction of verifiable computation schemes for the delegation of multivariate polynomials and matrix multiplication. Our constructions make a generic use of our new notion of algebraic one-way functions.

An overview of our solutions. Our starting point is the protocol of [7]: assume the client has a polynomial $F(\cdot)$ of large degree d, and it wants to compute the value $F(x)$ for arbitrary inputs

x. In [7] the client stores the polynomial in the clear with the server as a vector of coefficients c_i in $\mathbb{Z}_p$. The client also stores with the server a vector of group elements t_i of the form $g^{ac_i+r_i}$ where g generates a cyclic group $\mathbb{G}$ of order p, $a \in_R \mathbb{Z}_p$, and r_i is the i^{th}-coefficient of a polynomial $R(\cdot)$ of the same degree as $F(\cdot)$. When queried on input x, the server returns $y = F(x)$ and $t = g^{aF(x)+R(x)}$, and the client accepts y iff $t = g^{ay+R(x)}$.

If $R(\cdot)$ was a random polynomial, then this is a secure way to authenticate y, however checking that $t = g^{ay+R(x)}$ would require the client to compute $R(x)$ – the exact work that we set out to avoid! The crucial point, therefore, is how to perform this verification fast, i.e., in $o(d)$ time. The fundamental tool in [7] is the introduction of pseudo-random functions (PRFs) with a special property called *closed-form efficiency*: if we define the coefficients r_i of $R(\cdot)$ as $PRF_K(i)$ (which preserves the security of the scheme), then for any input x the value $g^{R(x)}$ can be computed very efficiently (sub-linearly in d) by a party who knows the secret key K for the PRF.

Our first observation was to point out that one of the PRFs proposed in [7] was basically a variant of the Naor-Reingold PRF [37] which can be easily istantiated over RSA moduli assuming the DDH holds over such groups (in particular over the subgroup of quadratic residues).

Note, however, that this approach implies a private verification algorithm by the same client who outsourced the polynomial in the first place, since it requires knowledge of the secret key K. To make verification public, Fiore and Gennaro proposed the use of Bilinear Maps [19].

Our second observation was to note that the scheme in [7] is really an information-theoretic authentication of the polynomial "in the exponent". Instead of using exponentiation, we observed that any "one-way function" with the appropriate "homomorphic properties" would do. We teased out the relevant properties and defined the notion of an *Algebraic One-Way Function* and showed that it is possible to instantiate it using the RSA/Rabin functions.

If we use our algebraic one-way functions based on RSA and factoring described in Section 3.1, then we obtain new verifiable computation schemes whose security rely on these assumptions and that support polynomials over a large variety of finite fields: $\mathbb{F}_e$ for any prime $e \geq 3$, $\mathbb{F}_{2^t}$ for any integer $t \geq 1$. Previously known solutions [38, 19] could support only polynomials over $\mathbb{F}_p$ where p must be a *large* prime whose size strictly depends on the security parameter 1^λ (basically, p must be such that the discrete logarithm problem is hard in a group of order p). In contrast, our factoring and RSA solutions allow for much more flexibility.

Precisely, using the RSA function allow us to compute polynomials over $\mathbb{F}_e$ for any prime $e \geq 3$, where e is the prime used by the RSA function. Moreover, using the Rabin function allows us to handle polynomials over $\mathbb{F}_{2^t}$ for any integer $t \geq 1$. Notice also that in the specific case of $\mathbb{F}_2$, our solution can handle public verifiable delegation of boolean formulas.

A solution for Polynomials of Degree d in each variable. In this section we propose the construction of a scheme for delegating the computation of m-variate polynomials of degree at most d in each variable. These polynomials have up to $l = (d+1)^m$ terms which we index by $(i_1, \ldots, i_m)$, for $0 \leq i_j \leq d$. Similarly to [7, 19], we define the function $h : \mathbb{Z}_p^m \to \mathbb{Z}_p^l$ which expands the input $\boldsymbol{x}$ to the vector $(h_1(\boldsymbol{x}), \ldots, h_l(\boldsymbol{x}))$ of all monomials as follows: for all $1 \leq j \leq l$, write $j = (i_1, \ldots, i_m)$ with $0 \leq i_k \leq d$, then $h_j(\boldsymbol{x}) = (x_1^{i_1} \cdots x_m^{i_m})$. So, using this notation we can write the polynomial as $f(\boldsymbol{x}) = \langle \boldsymbol{f}, h(\boldsymbol{x}) \rangle = \sum_{j=1}^{l} f_j \cdot h_j(\boldsymbol{x})$ where the f_j's are its coefficients.

Our scheme uses two main building blocks: an algebraic one-way function (see definition in Section 3) (Gen, F) and a pseudorandom function $(\mathsf{PRF.KG}, \mathsf{PRF.F}, \mathsf{PRF.CFEval})$ with closed form efficiency (see definition in Section 2.2). Our construction works generically for any family of functions $\mathcal{F}$ that is the set of m-variate polynomials of degree d over a finite field $\mathbb{K}$ such that: (1) the algebraic one-way function $F_\kappa : \mathcal{X}_\kappa \rightarrow \mathcal{Y}_\kappa$ is field-homomorphic for $\mathbb{K}$, and (2) there exists a PRF that has closed form efficiency relative to the computation of polynomials, and whose range is $\mathcal{X}_\kappa$.

If we instantiate these primitives with the CDH-based algebraic OWF of Section 3.1 and the PRFs based on Decision Linear described in [19], then our generic construction captures the verifiable computation scheme of Fiore and Gennaro [19].

If we use our algebraic one-way functions based on RSA and Factoring described in Section 3.1, then we obtain new verifiable computation schemes that support polynomials over a *large* variety of finite fields: $\mathbb{F}_e$ for any prime $e \geq 3$, $\mathbb{F}_{2^t}$ for any integer $t \geq 1$.

Previously known solutions [38, 19] could support only polynomials over $\mathbb{F}_p$ where p must be a *large* prime whose size strictly depends on the security parameter 1^λ (basically, p must be such that the discrete logarithm problem is hard in a group of order p). In contrast, our factoring and RSA solutions allow for much more flexibility. In particular, it is interesting to notice that our scheme for $\mathbb{F}_2$ allows for verifiable computation of boolean formulas, and its security is based on a very standard assumption, whereas the only previous scheme for boolean formulas [39] rely on non-constant size assumptions in bilinear groups.

Both our algebraic OWFs based on RSA and factoring have input and output space $\mathcal{X}_\kappa = \mathcal{Y}_\kappa = \mathbb{QR}_N$, the subgroup of quadratic residues in $\mathbb{Z}_N^*$. To complete the instantiation of the scheme $\mathcal{VC}_{\mathsf{Poly}}$, we need a PRF with closed form efficiency whose range is $\mathbb{QR}_N$. For this purpose we can use the PRF constructions described in [7] that are based on the Naor-Reingold PRF. The only difference is that in our case we have to instantiate the PRFs in the group $\mathbb{QR}_N$, and thus claim their security under the hardness of DDH in the group $\mathbb{QR}_N$.

The description of our generic construction $\mathcal{VC}_{\mathsf{Poly}}$ follows.

$\mathsf{KeyGen}(1^\lambda, f)$**.** Run $\kappa \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ to obtain a one-way function $F_\kappa : \mathcal{X}_\kappa \rightarrow \mathcal{Y}_\kappa$ that is field-homomorphic for $\mathbb{K}$. Let f be encoded as the set of its coefficients $(f_1, \ldots, f_l) \in \mathbb{K}^l$.
Generate the seed of a PRF, $K \xleftarrow{\$} \mathsf{PRF.KG}(1^\lambda, \lceil \log d \rceil, m)$, whose output space is $\mathcal{X}_\kappa$, the input of the one-way function. Choose a random generator $h \xleftarrow{\$} \mathcal{X}_\kappa$, and compute $A = F_\kappa(h)$.
For $i = 1$ to l, compute $W_i = h^{f_i} \cdot \mathsf{PRF.F}_K(i)$. Let $W = (W_1, \ldots, W_l) \in (\mathcal{X}_\kappa)^l$.
Output $\mathsf{EK}_f = (f, W, A)$, $\mathsf{PK}_f = A$, $\mathsf{SK}_f = K$.

$\mathsf{ProbGen}(\mathsf{PK}_f, \mathsf{SK}_f, \boldsymbol{x})$**.** Output $\sigma_x = \boldsymbol{x}$ and $\mathsf{VK}_x = F_\kappa(\mathsf{PRF.CFEval}_{\mathsf{Poly}}(K, h(\boldsymbol{x})))$.

$\mathsf{Compute}(\mathsf{EK}_f, \sigma_x)$**.** Let $\mathsf{EK}_f = (f, W, A)$ and $\sigma_x = \boldsymbol{x}$. Compute $y = f(\boldsymbol{x}) = \sum_{i=1}^{l} f_i \cdot h_i(\boldsymbol{x})$ (over $\mathbb{K}$) and

$$V = \mathsf{Eval}(\kappa, A, W, f, h(\boldsymbol{x}))$$

and return $\sigma_y = (y, V)$.

$\mathsf{Verify}(\mathsf{PK}_f, \mathsf{VK}_x, \sigma_y)$**.** Parse σ_y as (y, V). If $y \in \mathbb{K}$ and $F_\kappa(V) = A^y \cdot \mathsf{VK}_x$, then output y, otherwise output $\perp$.

The correctness of the scheme follows from the properties of the algebraic one-way function and the correctness of $\mathsf{PRF.CFEval}$.

Theorem 4. *If* (Gen, F) *is a family of algebraic one-way functions and* $\mathsf{PRF.F}$ *is a family of pseudo-random functions then any PPT adversary* $\mathcal{A}$ *making at most* $q = poly(\lambda)$ *queries has negligible advantage* $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{PubVer}}(\mathcal{VC}_{\mathsf{Poly}}, \mathcal{F}, q, \lambda)$.

To prove the theorem, we define the following games, where $G_i(\mathcal{A})$ denotes the output of Game i run with adversary $\mathcal{A}$:

Game 0: it is $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{PubVer}}(\mathcal{VC}_{\mathsf{Poly}}, \mathcal{F}, q, \lambda)$.

Game 1: this is the same as Game 0 except that the challenger performs a different evaluation of the algorithm $\mathsf{ProbGen}$. Let $\boldsymbol{x}$ be the input asked by the adversary. The challenger computes $\mathsf{VK}_x = \prod_{i=1}^{l} \mathsf{PRF.F}_K(i)^{h_i(\boldsymbol{x})}$.

Game 2: this game proceeds as Game 1, except that the function $\mathsf{PRF.F}_k(i)$ is replaced by a truly random function that on every i lazily samples a value $R_i \xleftarrow{\$} \mathcal{X}_\kappa$ uniformly at random.

The proof of the theorem is obtained by the proofs of the following claims.

Claim 1 $\Pr[G_0(\mathcal{A}) = 1] = \Pr[G_1(\mathcal{A}) = 1]$.

Proof. By correctness of $\mathsf{PRF.CFEval}$, these two games produce the same distribution. In particular, the distribution of the values VK_x does not change. Therefore, the probability of the adversary winning in Game 1, i.e., $\Pr[G_1(\mathcal{A}) = 1]$, remains the same.

Claim 2 $|\Pr[G_1(\mathcal{A}) = 1] - \Pr[G_2(\mathcal{A}) = 1]|$ *is negligible*

Proof. The difference between Game 2 and Game 1 is that the output of the pseudorandom function $\mathsf{PRF.F}_K$ is replaced by values chosen at random in $\mathcal{X}_\kappa$. If there exists an adversary $\mathcal{A}$ such its success probability in Game 2 decreases by more than a non-negligible quantity, then $\mathcal{A}$ can be used to build an efficient distinguisher that breaks the security of the PRF with such non-negligible probability.

Claim 3 $\Pr[G_2(\mathcal{A}) = 1]$ *is negligible.*

Proof. Assume by contradiction there exists a PPT adversary $\mathcal{A}$ such that $\Pr[G_2(\mathcal{A}) = 1]$ is a non-negligible ϵ.

We show that from such $\mathcal{A}$ it is possible to construct an efficient algorithm $\mathcal{B}$ that breaks the strong one-wayness of the algebraic one-way function with the same probability ϵ.

$\mathcal{B}$ receives the pair (κ, A) as its input, where $A \in \mathcal{Y}_\kappa$. It proceeds as follows. It chooses l random values $W_1, \ldots, W_l \xleftarrow{\$} \mathcal{X}_\kappa$, and it sets $\mathsf{EK}_f = (f, W, A)$ and $\mathsf{PK}_f = A$. Notice that the public and evaluation keys are perfectly distributed as in Game 2.

Next, for $i = 1$ to l, $\mathcal{B}$ computes $Z_i = F_\kappa(W_i) \cdot A^{-f_i}$. $\mathcal{B}$ runs $\mathcal{A}(\mathsf{PK}_f, \mathsf{EK}_f)$ and answers its queries as follows. Let $\boldsymbol{x}$ be the queried value. $\mathcal{B}$ computes $\mathsf{VK}_x = \prod_{i=1}^{l} Z_i^{h_i(\boldsymbol{x})}$, and returns VK_x to $\mathcal{A}$. By the homomorphic property of F_κ this computation of VK_x is equivalent to the one made by the challenger in Game 2.

Finally, let $\hat{\sigma}_y = (\hat{y}, \hat{V})$ be the output of $\mathcal{A}$ at the end of the game, such that for some input value $\boldsymbol{x}^*$ chosen by $\mathcal{A}$ it holds: $\mathsf{Verify}(\mathsf{PK}_f, \mathsf{VK}_{x^*}, \hat{\sigma}_y) = \hat{y}$, $\hat{y} \neq \perp$ and $\hat{y} \neq f(\boldsymbol{x}^*)$. By verification, this means that

$$F_\kappa(\hat{V}) = A^{\hat{y}} \cdot \mathsf{VK}_{x^*}$$

Let $y = f(\boldsymbol{x}^*) \in \mathbb{K}$ be the correct output of the computation, and let $V = \mathsf{Eval}(\kappa, A, W, f, h(\boldsymbol{x}))$ be the proof as obtained by honestly running $\mathsf{Compute}$. By correctness of the scheme it holds:

$$F_\kappa(V) = A^{y} \cdot \mathsf{VK}_{x^*}$$

Hence, we can divide the two verification equations and by the homomorphic property of F_κ, we obtain $F_\kappa(\hat{V}/V) = A^{\delta}$ where $\delta = \hat{y} - y \neq 0$. $\mathcal{B}$ outputs $U = \hat{V}/V$ and δ as a solution for the one-wayness of $F_\kappa(A)$. As one can see, if $\mathcal{A}$ wins in Game 2 with probability ϵ, then $\mathcal{B}$ breaks the one-wayness of F_κ with the same probability.

4.1 m-Variate Polynomials of Total Degree d

We observe that it is possible to change the protocol $\mathcal{VC}_{\mathsf{Poly}}$ described in the previous section in order to support the class of polynomials in m variables and maximum degree d in each monomial. As hinted in [19], this can be done as follows: (i) adjust the number of monomials to $l = (m+1)^d$; (ii) use a PRF with closed-form efficiency for polynomials of this form (such as the DDH-based one given in [7]).

4.2 Matrix Multiplication

We show that the same techniques used to construct a verifiable computation scheme for the delegation of multivariate polynomials can be adapted for the case of matrix multiplications. Again, the building blocks are an algebraic one-way function and a PRF with closed form efficiency for this type of computations.

By using our constructions of algebraic OWFs based on RSA and factoring we obtain schemes that can support delegation of matrix computations over arbitrary finite fields of both prime and binary characteristic. As for the algebraic PRF, we can use the DDH-based construction (instantiated over $\mathbb{QR}_N$) proposed in [19] that is closed-form efficient for matrix multiplication.

$\mathsf{KeyGen}(1^\lambda, M)$**.** Run $\kappa \xleftarrow{\$} \mathsf{Gen}(1^{\mathrm{sec}})$ to obtain a one-way function $F_\kappa : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa$ that is field-homomorphic for $\mathbb{K}$. Let $M \in \mathbb{K}^{n \times d}$ be a matrix.
Generate a seed K for an algebraic PRF with domain $[1..n] \times [1..d]$ and range $\mathcal{X}_\kappa$. Sample a random generator $h \xleftarrow{\$} \mathcal{X}_\kappa$, and compute $A = F_\kappa(h)$.
For $1 \le i \le d$, $1 \le j \le n$, compute $W_{i,j} = h^{M_{i,j}} \cdot \mathsf{PRF.F}_K(i,j)$, and let $W = (W_{i,j}) \in \mathcal{X}_\kappa^{n \times d}$.
Output $\mathsf{SK}_M = K$, $\mathsf{EK}_M = (M, W, A)$, and $\mathsf{VK}_M = A$.

$\mathsf{ProbGen}(\mathsf{SK}_M, \boldsymbol{x})$**.** Let $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{K}^d$ be the input. Let R be the matrix defined by $R = [\mathsf{PRF.F}_K(i,j)]$. Compute $\boldsymbol{\rho}_x = \mathsf{PRF.CFEval}_{\mathsf{Matrix}}(K, x)$ in $O(n+d)$ using the closed form efficiency. Recall that $\rho_{x,i} = \prod_{j=1}^{d} \mathsf{PRF.F}_K(i,j)^{x_j}$, and define $\tau_{x,i} = F_\kappa(\rho_{x,i})$. Finally, output the encoding $\sigma_{\boldsymbol{x}} = \boldsymbol{x}$, and the verification key $\mathsf{VK}_{\boldsymbol{x}} = (\tau_{x,1}, \ldots, \tau_{x,n})$.

$\mathsf{Compute}(\mathsf{EK}_M, \sigma_{\boldsymbol{x}})$**.** Let $\mathsf{EK}_M = (M, W, A)$ and $\sigma_{\boldsymbol{x}} = \boldsymbol{x}$. Compute $\boldsymbol{y} = M \cdot \boldsymbol{x}$ over the field $\mathbb{K}$, and the vector $\boldsymbol{V} = (V_1, \ldots, V_n)$ as $V_j = \mathsf{Eval}(\kappa, A, (W_{i,j})_i, (M_{i,j})_i, \boldsymbol{x})$, $\forall j = 1$ to n.
Output $\sigma_y = (\boldsymbol{y}, \boldsymbol{V})$.

$\mathsf{Verify}(\mathsf{VK}_M, \mathsf{VK}_{\boldsymbol{x}}, \sigma_y)$**.** Parse σ_y as $(\boldsymbol{y}, \boldsymbol{V})$. If $\boldsymbol{y} \in \mathbb{K}^n$ and $F_\kappa(V_i) = A^{y_i} \cdot \tau_{x,i}$, $\forall i = 1, \ldots, n$, then output $\boldsymbol{y}$, otherwise output $\perp$.

The security of the scheme is proven via the following theorem.

Theorem 5. *If* (Gen, F) *is a secure family of algebraic one-way functions and* $\mathsf{PRF.F}$ *is a secure PRF family, then any PPT adversary* $\mathcal{A}$ *making at most* $q = poly(\lambda)$ *queries has negligible advantage* $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{PubVer}}(\mathcal{VC}_{\mathsf{Matrix}}, \mathcal{F}, q, \lambda)$.

The proof proceeds in a way very similar to that of Theorem 4. Consider the following games, where $G_i(\mathcal{A})$ denotes the output of Game i with adversary $\mathcal{A}$:

Game 0: it is $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{PubVer}}(\mathcal{VC}_{\mathsf{Matrix}}, \mathcal{F}, q, \lambda)$.

Game 1: this is the same as Game 0, except that the challenger performs a different computation of the algorithm $\mathsf{ProbGen}$. Let $\boldsymbol{x}$ be the input asked by the adversary. The challenger computes $\mathsf{VK}_x = \boldsymbol{\rho}_x$ as $\rho_{x,i} = \prod_{j=1}^{d} \mathsf{PRF.F}_K(i,j)^{x_j}$.

Game 2: this game proceeds as Game 1, except that the matrix W is computed as $W_{i,j} = h^{M_{i,j}} \cdot R_{i,j}$ where for all i, j $R_{i,j} \xleftarrow{\$} \mathcal{X}_\kappa$ is chosen uniformly at random, instead of being the output of $\mathsf{PRF.F}_K(i, j)$.

By the same ideas used in the proof of Theorem 4, it is not hard to see that the following two claims hold.

Claim 4 $\Pr[G_0(\mathcal{A}) = 1] = \Pr[G_1(\mathcal{A}) = 1]$.

Claim 5 $|\Pr[G_1(\mathcal{A}) = 1] - \Pr[G_2(\mathcal{A}) = 1]|$ *is negligible*

The proof of the following claim is a simple extension of the proof of Claim 3. We describe it below for completeness.

Claim 6 $\Pr[G_2(\mathcal{A}) = 1]$ *is negligible*

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that the probability of $\mathcal{A}$ winning in Game 2 is a non-negligible function ϵ, then we show that we can build an efficient algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to break the security of the algebraic one-way function with probability ϵ. $\mathcal{B}$ takes as input a pair (κ, A) where $A \in \mathcal{Y}_\kappa$ and proceeds as follows. For $i = 1, \dots, d$ and $j = 1, \dots, n$, $\mathcal{B}$ chooses $W_{i,j} \xleftarrow{\$} \mathcal{X}_\kappa$, sets $\mathsf{EK}_M = (M, W, A)$, and $\mathsf{PK}_M = A$. It is easy to check that the public and evaluation keys are perfectly distributed as in Game 2. Next, for $i = 1, \dots, d$ and $j = 1, \dots, n$, it computes $Z_{i,j} = F_\kappa(W_{i,j}) \cdot A^{-M_{i,j}}$. Then $\mathcal{B}$ runs $\mathcal{A}(\mathsf{PK}_M, \mathsf{EK}_M)$ and answers its queries as follows. Let $\boldsymbol{x}$ be the queried vector. $\mathcal{B}$ computes $\tau_{x,j} = \prod_{i=1}^{d} Z_{i,j}^{x_i}$ for $j = 1$ to n, and returns $\mathsf{VK}_x = (\tau_{x,1}, \dots, \tau_{x,n})$ to $\mathcal{A}$. By the homomorphic property of F_κ this computation of VK_x is equivalent to the one done in Game 2.

Finally, let $\hat{\sigma}_y = (\hat{\boldsymbol{y}}, \hat{\boldsymbol{V}})$ be the output of $\mathcal{A}$ at the end of the game, such that for some $\boldsymbol{x}^*$ chosen by $\mathcal{A}$ it holds $\mathsf{Verify}(\mathsf{PK}_f, \mathsf{VK}_{\boldsymbol{x}^*}, \hat{\sigma}_y) = \hat{\boldsymbol{y}}$, $\hat{\boldsymbol{y}} \neq \perp$ and $\hat{\boldsymbol{y}} \neq M \cdot \boldsymbol{x}^*$. Let $\boldsymbol{y} = M \cdot \boldsymbol{x}^*$ be the correct output of the multiplication. Since $\hat{\boldsymbol{y}} \neq \boldsymbol{y}$ there must exist an index $j \in \{1, \dots, n\}$ such that $\hat{y}_j \neq y_j$. However, if we let $V_j = \mathsf{Eval}(\kappa, A, (W_{i,j})_i, (M_{i,j})_i, \boldsymbol{x})$ be the honest computation for the j-th vector entry, then by correctness we have:

$$F_\kappa(V_j) = A^{y_j} \cdot \tau_{\boldsymbol{x}^*,j}$$

Hence, if we divide the two verification equations, we obtain $F_\kappa(\hat{V}_j/V_j) = A^\delta$ where $\delta = \hat{y}_j - y_j \neq 0$. Therefore, $\mathcal{B}$ can output $U = \hat{V}_j/V_j$ and δ. It is easy to see that if $\mathcal{A}$ wins in Game 2 with probability ϵ, then $\mathcal{B}$ breaks the one-wayness of F_κ with the same probability.

5 Linearly-Homomorphic FDH Signatures

In this section we show a direct application of Algebraic Trapdoor One Way Permutations (TDP) to build linearly-homomorphic signatures.

An intuitive overview of our solution. Our construction can be seen as a linearly-homomorphic version of Full-Domain-Hash (FDH) signatures. Recall that a FDH signature on a message m is $F^{-1}(H(m))$ where F is any TDP and H is a hash function modeled as a random oracle. Starting from this basic scheme, we build our linearly homomorphic signatures by defining a signature on a message m, tag τ and index i as $\sigma = F^{-1}(H(\tau, i) \cdot G(m))$ where F is now an algebraic TDP,

H is a classical hash function that will be modeled as a random oracle and G is a homomorphic hash function (i.e, such that $G(x) \cdot G(y) = G(x+y)$). Then, we will show that by using the special properties of algebraic TDPs (in particular, field-homomorphicity and strong one-wayness) both the security and the homomorphic property of the signature scheme follow immediately.

Precisely, if the algebraic TDP used in the construction is field-homomorphic for a field $\mathbb{K}$, then our signature scheme supports the message space $\mathbb{K}^n$ (for some integer $n \geq 1$) and all linear functions over this field. Interestingly, by instantiating our generic construction with our two algebraic TDPs based on Factoring and RSA (see Section 3.1), we obtain schemes that are linearly-homomorphic for *arbitrary* finite fields, i.e., $\mathbb{F}_{2^t}$ or $\mathbb{F}_e$, for any $t \geq 1$ and a prime e. As we will detail at the end of this section, previous solutions (e.g., [9, 22, 3, 11, 10, 14, 15, 20]) could support only large fields whose size strictly depends on the security parameter. The only exception are the lattice-based schemes of Boneh and Freeman [11, 10] that work for small fields, but are less efficient than our solution. In this sense, one of our main contributions is to propose a solution that offers a great flexibility as it can support arbitrary finite fields, both small and large, whose characteristic can be basically chosen ad-hoc (e.g., according to the desired application) at the moment of instantiating the scheme.

Our Scheme. The scheme is defined by the following algorithms.

$\mathsf{Hom.KG}(1^\lambda, m, n)$ On input the security parameter λ, the maximum data set size m, and an integer $n \geq 1$ used to determine the message space $\mathcal{M}$ as we specify below, the key generation algorithm proceeds as follows.

Run $(\kappa, \mathsf{td}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$ to obtain an algebraic TDP, $F_\kappa : \mathcal{X}_\kappa \to \mathcal{X}_\kappa$ that is field-homomorphic for the field $\mathbb{K}$. Next, sample $n+1$ group elements $u, g_1, \ldots, g_n \xleftarrow{\$} \mathcal{X}_\kappa$ and choose a hash function $H : \{0,1\}^* \to \mathcal{X}_\kappa$.

The public key is set as $\mathsf{VK} = (\kappa, u, g_1, \ldots, g_n, H)$, while the secret key is the trapdoor $\mathsf{SK} = \mathsf{td}$. The message space $\mathcal{M} = (\mathbb{K})^n$ is the set of n-dimensional vectors whose components are elements of $\mathbb{K}$, while the set of admissible functions $\mathcal{F}$ is all degree-1 polynomials over $\mathbb{K}$ with m variables and constant-term zero.

$\mathsf{Hom.Sign}(\mathsf{SK}, \tau, M, i)$ The signing algorithm takes as input the secret key SK, a tag $\tau \in \{0,1\}^\lambda$, a message $M = (M_1, \ldots, M_n) \in \mathbb{K}^n$ and an index $i \in \{1, \ldots, m\}$. To sign, choose $s \xleftarrow{\$} \mathbb{K}$ uniformly at random and use the trapdoor td to compute

$$x = F_\kappa^{-1}(H(\tau, i) \cdot u^s \cdot \prod_{j=1}^{n} g_j^{M_j})$$

and output $\sigma = (x, s)$.

$\mathsf{Hom.Ver}(\mathsf{VK}, \tau, M, \sigma, f)$ To verify a signature $\sigma = (x, s)$ on a message $M \in \mathcal{M}$, w.r.t. tag τ and the function f, the verification algorithm proceeds as follows. Let f be encoded as its set of coefficients $(f_1, f_2, \ldots, f_m)$. Check that all values f_i and M_j are in $\mathbb{K}$ and then check that the following equation holds

$$F_\kappa(x) = \prod_{i=1}^{m} H(\tau, i)^{f_i} \cdot u^s \cdot \prod_{j=1}^{n} g_j^{M_j}$$

If both checks are satisfied, then output 1 (accept), otherwise output 0 (reject).

$\mathsf{Hom.Eval}(\mathsf{VK}, \tau, f, \boldsymbol{\sigma}, \boldsymbol{M}, \boldsymbol{f})$ The public evaluation algorithm takes as input the public key VK, a tag τ, a function $f \in \mathcal{F}$ encoded as $(f_1, \ldots, f_m) \in \mathbb{K}^m$, a vector of signatures $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)$ where $\sigma_i = (x_i, s_i)$, a vector of messages $\boldsymbol{M} = (M^{(1)}, \ldots, M^{(m)})$ and a vector of functions $\boldsymbol{f} = (f^{(1)}, \ldots, f^{(m)})$. If each signature σ_i is valid for the tag τ, the message $M^{(i)}$ and the function $f^{(i)}$, then the signature σ output by $\mathsf{Hom.Eval}$ is valid for the message $M = f(M^{(1)}, \ldots, M^{(m)})$. In order to do this, our algorithm first computes $s = f(s_1, \ldots, s_m) = \sum_{i=1}^{m} f_i \cdot s_i$ (over $\mathbb{K}$). Next, it defines:

$$\boldsymbol{A} = (H(\tau, 1), \ldots, H(\tau, m), u, g_1, \ldots, g_n) \in \mathcal{X}_\kappa^{m+n+1},$$

$$\Omega = \begin{bmatrix} f_1^{(1)} & \cdots & f_m^{(1)} & s_1 & M_1^{(1)} & \cdots & M_n^{(1)} \\ \vdots & & \vdots & & \vdots & & \vdots \\ f_1^{(m)} & \cdots & f_m^{(m)} & s_m & M_1^{(m)} & \cdots & M_n^{(m)} \end{bmatrix} \in \mathbb{Z}^{m \times m+n+1}$$

and uses the Eval algorithm of the algebraic TDP to compute $x = \mathsf{Eval}(\kappa, \boldsymbol{A}, \boldsymbol{x}, \Omega, f)$. Finally, it outputs $\sigma = (x, s)$.

We remark that our construction requires the $\mathsf{Hom.Eval}$ algorithm to know the messages $M^{(i)}$ for which the signatures σ_i are supposed to verify correctly.

Since our scheme follows the FDH paradigm, its security holds in the random oracle model, and according to recent impossibility results [17] it is unlikely to hope for a standard model proof. Though, following similar results for FDH signatures, we propose in Appendix A a variant of our scheme that can be proven secure in the standard model in the weaker security model of Q-time security, in which the adversary is restricted to query signatures on at most Q different datasets, and Q is a pre-fixed bound.

The security of our scheme from algebraic TDP follows from the following theorem.

Theorem 6. *If $(\mathsf{Gen}, F, \mathsf{Inv})$ is a family of algebraic trapdoor permutations and H is modeled as a random oracle, then the linearly-homomorphic signature scheme described above is secure.*

Proof. As usual, the proof proceeds by contradiction. Assume there exists an efficient adversary $\mathcal{A}$ that has non-negligible probability ϵ of winning the unforgeability game. Let $(\tau^*, M^*, \sigma^*, f^*)$ be the valid forgery returned by the adversary, i.e., such that $\mathsf{Verify}(\mathsf{VK}, \tau^*, M^*, \sigma^*, f^*) = 1$. According to whether the forgery is of Type 1 or Type 2, we distinguish two types of adversaries. For every such adversary $\mathcal{A}$ we describe a simulation in which we reduce $\mathcal{A}$ to an algorithm $\mathcal{B}$ that breaks the strong one-wayness of the algebraic TDP with non-negligible probability.

Type 1. $\mathcal{B}$ takes as input (κ, ρ) where κ is the description of an algebraic TDP $F_\kappa : \mathcal{X}_\kappa \to \mathcal{X}_\kappa$ and $\rho \in \mathcal{X}_\kappa$. The goal of $\mathcal{B}$ is to find values $(x, d) \in \mathcal{X}_\kappa \times \mathbb{K}$ such that $F_\kappa(x) = \rho^d$ and $d \neq 0$. Our simulator $\mathcal{B}$ proceeds as follows.

Key Generation. Let $Q = poly(\lambda)$ be an upper bound on the number of data sets for which the adversary asks signatures. $\mathcal{B}$ chooses in advance all tags $\tau_1, \ldots, \tau_Q \xleftarrow{\$} \{0,1\}^\lambda$ that it will use in the signing queries. Let T be the set of all such tags. $\mathcal{B}$ chooses an index $\mu \xleftarrow{\$} \{1, \ldots, m\}$ and group elements $y_0, y_1, \ldots, y_n \xleftarrow{\$} \mathcal{X}_\kappa$ uniformly at random. For $j = 1$ to n, it sets $g_j = F_\kappa(y_j)$, and $u = F_\kappa(y_0)$. It gives the public key $\mathsf{VK} = (\kappa, u, g_1, \ldots, g_n, H)$ to the adversary where H is a random oracle whose queries are answered as described below. We notice that since F_κ is a permutation over $\mathcal{X}_\kappa$, the public key VK is distributed as in the real case.

Hash queries. The simulator maintains a table $\bar{H}$ whose entries, indexed by pairs (τ, i), are tuples of the form (δ, h). If an entry $\bar{H}[\tau, i]$ is empty we denote it by $\bar{H}[\tau, i] = \perp$. When the adversary makes an oracle query $H(\tau, i)$ the simulator looks in the table the entry $\bar{H}[\tau, i]$. If $\bar{H}[\tau, i] = (\delta, h)$, then $\mathcal{B}$ returns h. Otherwise, if $\bar{H}[\tau, i] = \perp$, $\mathcal{B}$ picks a random $\delta_{\tau,i} \xleftarrow{\$} \mathcal{X}_\kappa$, and it proceeds as follows. If $\tau \notin T \wedge i = \mu$, then it sets $h_{\tau,i} = F_\kappa(\delta_{\tau,i}) \cdot \rho$. Otherwise $\mathcal{B}$ sets $h_{\tau,i} = F_\kappa(\delta_{\tau,i})$. Finally, it returns $h_{\tau,i}$ to $\mathcal{A}$ and stores $H[\tau, i] = (\delta_{\tau,i}, h_{\tau,i})$. Notice that regardless of whether $\tau \in T$ and $i = \mu$, all answers have the same distribution, uniform over $\mathcal{X}_\kappa$ (as the group is cyclic).

Signing queries. Let (F, i, M) be a signing query. If this is the first query with filename F, then $\mathcal{B}$ takes the next unused tag τ from T. Otherwise, let τ be the tag already chosen for F. Let $\bar{H}[\tau, i] = (\delta_{\tau,i}, h_{\tau,i})$ (if $\bar{H}[\tau, i] = \perp$, then $\mathcal{B}$ proceeds as above to generate it). Since $\tau \in T$ we have $h_{\tau,i} = F_\kappa(\delta_{\tau,i})$. Thus, $\mathcal{B}$ simulates a signature by choosing $s \xleftarrow{\$} \mathbb{K}$ at random, computing $x = \delta_{\tau,i} y_0^s \prod_{j=1}^n y_j^{M_j}$, and returning $\sigma = (x, s)$ to the adversary. It is easy to see that σ is correctly distributed.

Forgery. Let $(\tau^*, M^*, \sigma^*, f^*)$ be the forgery returned by $\mathcal{A}$, and let $T' = \{\tau_1, \ldots, \tau_{Q'}\}$ be the set of all tags used in the signing queries. Notice that $T' \subseteq T$, $|T \setminus T'| \leq Q$ and that all unrevealed tags are completely unpredictable. By our assumption in this case of the proof, this is a Type 1 forgery, i.e., $\tau^* \notin T'$. Moreover, it must also be $f^* \neq 0^m$, i.e., there must exist an index $\mu^* \in \{1, \ldots, m\}$ such that $f^*_{\mu^*} \neq 0$.

If $f^*_\mu = 0$ or $\tau^* \in T \setminus T'$, then $\mathcal{B}$ aborts the simulation and fails. Otherwise, it continues the simulation. Notice though that $\Pr[\mu = \mu^*] = 1/m$ (as μ is perfectly hidden), and that $\Pr[\tau^* \in T \setminus T'] \leq Q/2^\lambda$. Therefore, $\mathcal{B}$ does not abort with probability at least $1/m(1 - Q/2^\lambda)$. By the validity of the forgery we have:

$$F_\kappa(x^*) = \prod_{i=1}^m H(\tau^*, i)^{f_i^*} \cdot u^{s^*} \prod_{j=1}^n g_j^{M_j^*} = \prod_{i=1}^m F_\kappa(\delta_{\tau^*,i})^{f_i^*} \cdot \rho^{f_\mu^*} \cdot F_\kappa(y_0)^{s^*} \prod_{j=1}^n F_\kappa(y_j)^{M_j^*}$$

Thus, by the homomorphic property of F we obtain:

$$F_\kappa \left(\frac{x^*}{\prod_{i=1}^m \delta_{\tau^*,i}^{f_i^*} y_0^{s^*} \prod_{j=1}^n y_j^{M_j^*}} \right) = \rho^{f_\mu^*}$$

Therefore, $\mathcal{B}$ can output $U = \frac{x^*}{\prod_{i=1}^m \delta_{\tau^*,i}^{f_i^*} y_0^{s^*} \prod_{j=1}^n y_j^{M_j^*}}$ and $d = f^*_\mu$. If $\mathcal{A}$ outputs a Type 1 forgery with non-negligible probability ϵ, then $\mathcal{B}$ breaks the security of the algebraic TDP with non-negligible probability $\frac{\epsilon}{m}(1 - Q/2^\lambda)$.

Type 2. Let $\tau_1, \ldots, \tau_Q$ be the tags of all the datasets queried by the adversary in the signing phase. For a Type 2 adversary we have that $\tau^* = \tau_j$ for some $j \in \{1, \ldots, Q\}$, and $M^* \neq \hat{M} = f^*(M^{(1)}, \ldots, M^{(m)})$ where $(M^{(1)}, \ldots, M^{(m)})$ are the messages of the dataset with tag τ_j. Let $\hat{\sigma} = (\hat{x}, \hat{s}) = \mathsf{Hom.Eval}(\mathsf{VK}, \tau_j, f^*, \boldsymbol{\sigma}, \boldsymbol{M}, 1^m)$ be the signature obtained by correctly applying the $\mathsf{Hom.Eval}$ algorithm on the messages (and signatures) of the dataset τ_j with the function f^*. Since $M^* \neq \hat{M}$, there must exists an index $\nu \in \{1, \ldots, n\}$ such that $M^*_\nu \neq \hat{M}_\nu$. We distinguish the following two mutually exclusive cases:

(a) $s^* - \hat{s} + M^*_\nu - \hat{M}_\nu \neq 0$

(b) $s^* - \hat{s} + M^*_\nu - \hat{M}_\nu = 0$, i.e., $s^* - \hat{s} \neq 0$

where all inequalities are intended over the finite field $\mathbb{K}$.

We provide different simulations for the two cases.

Type 2.a $\mathcal{B}$ takes as input (κ, ρ) where κ is the description of an algebraic TDP $F_\kappa : \mathcal{X}_\kappa \to \mathcal{X}_\kappa$ and $\rho \in \mathcal{X}_\kappa$. The goal of $\mathcal{B}$ is to find values $(x, d) \in \mathcal{X}_\kappa \times \mathbb{K}$ such that $F_\kappa(x) = \rho^d$ and $d \neq 0$. Our simulator $\mathcal{B}$ proceeds as follows.

Key Generation. $\mathcal{B}$ chooses the index $\nu \xleftarrow{\$} \{1, \ldots, n\}$ and group elements $y_1, \ldots, y_n \xleftarrow{\$} \mathcal{X}_\kappa$ uniformly at random. For $j = 1$ to n, $j \neq \nu$, it sets $g_j = F_\kappa(y_j)$, $g_\nu = F_\kappa(y_\nu) \cdot \rho$, and $u = \rho$. It returns the public key $\mathsf{VK} = (\kappa, u, g_1, \ldots, g_n, H)$ and it answers random oracle queries to H as described below. We notice that the simulated public key has the same distribution as the real one.

Hash queries. The simulator maintains a table $\bar{H}$ whose entries, indexed by pairs (τ, i), are triples of the form (δ, β, h). If an entry $\bar{H}[\tau, i]$ is empty we denote it by $\bar{H}[\tau, i] = \perp$. When the adversary makes a query $H(\tau, i)$ the simulator looks for $\bar{H}[\tau, i]$. If $\bar{H}[\tau, i] = (\delta, \beta, h)$, then it returns h. Otherwise, if $\bar{H}[\tau, i] = \perp$, $\mathcal{B}$ chooses $\delta_{\tau,i} \xleftarrow{\$} \mathcal{X}_\kappa$, $\beta_{\tau,i} \xleftarrow{\$} \mathbb{K}$ and computes $h_{\tau,i} = F_\kappa(\delta_{\tau,i}) \cdot \rho^{\beta_{\tau,i}}$. Finally, it returns $h_{\tau,i}$ to $\mathcal{A}$ and stores $\bar{H}[\tau, i] = (\delta_{\tau,i}, \beta_{\tau,i}, h_{\tau,i})$. Notice that since $\delta_{\tau,i}$ is "fresh" (i.e., chosen independently at random) for every query, all answers are uniformly distributed over $\mathcal{X}_\kappa$, and thus the value $\beta_{\tau,i}$ is perfectly hidden.

Signing queries. Let (F, i, M) be a signing query. If this is the first query with filename F, then $\mathcal{B}$ chooses a new tag $\tau \xleftarrow{\$} \{0,1\}^\lambda$. Otherwise, let τ be the tag already chosen for F, and let $\bar{H}[\tau, i] = (\delta_{\tau,i}, \beta_{\tau,i}, h_{\tau,i})$. The simulator sets $s = -(\beta_{\tau,i} + M_\nu) \in \mathbb{K}$, and uses the strong one-wayness property of the algebraic TDP to compute the preimage $\tilde{\rho} = F_\kappa^{-1}(\rho^{s+\beta_{\tau,i}+M_\nu})$ (this can be done efficiently as $s + \beta_{\tau,i} + M_\nu$ is 0 over $\mathbb{K}$). Then it sets $x = \delta_{\tau,i} \prod_{j=1}^n y_j^{M_j} \tilde{\rho}$ and returns $\sigma = (x, s)$. It is not hard to check that the signature is distributed correctly. In particular, it holds $F_\kappa(x) = H(\tau, i) \cdot u^s \prod_{j=1}^n g_j^{M_j}$ and s is uniform in $\mathbb{K}$ as so is $\beta_{\tau,i}$.

Forgery. Let $(\tau^*, M^*, \sigma^*, f^*)$ be the forgery returned by $\mathcal{A}$, and let $\hat{\sigma} = (\hat{x}, \hat{s}) = \mathsf{Hom.Eval}(\mathsf{VK}, \tau_j, f^*, \boldsymbol{\sigma}, \boldsymbol{M}, 1^m)$ be the signature obtained by applying the correct evaluation algorithm with function f^* to the messages of the dataset with tag τ^* (that by definition of Type 2 was asked in the signing phase). If $M^*_\nu - \hat{M}_\nu = 0$, then $\mathcal{B}$ aborts and stops running. Otherwise it continues the simulation. Notice though that since an index ν^* such that $M^*_{\nu^*} - \hat{M}_{\nu^*} \neq 0$ must exist, and the ν chosen by $\mathcal{B}$ is perfectly hidden, then $\Pr[M^*_\nu - \hat{M}_\nu \neq 0] = \Pr[\nu = \nu^*] = 1/n$.

By the validity of the forgery we have:

$$F_\kappa(x^*) = \prod_{i=1}^m H(\tau^*, i)^{f^*_i} \cdot u^{s^*} \prod_{j=1}^n g_j^{M^*_j}$$

while by the correctness of $\mathsf{Hom.Eval}$ it holds

$$F_\kappa(\hat{x}) = \prod_{i=1}^m H(\tau^*, i)^{f^*_i} \cdot u^{\hat{s}} \prod_{j=1}^n g_j^{\hat{M}_j}$$

So, we can divide the two equations and obtain:

$$F_\kappa(x^*/\hat{x}) = u^{s^*-\hat{s}} \prod_{j=1}^n g_j^{M^*_j - \hat{M}_j} = \rho^{s^*-\hat{s}+M^*_\nu-\hat{M}_\nu} \prod_{j=1}^n F_\kappa(y_j)^{M^*_j - \hat{M}_j}$$

and thus

$$F_\kappa\left(\frac{x^*}{\hat{x}\prod_{j=1}^n F_\kappa(y_j)^{M_j^*-\hat{M}_j}}\right) = \rho^{s^*-\hat{s}+M_\nu^*-\hat{M}_\nu}$$

Therefore, since $s^* - \hat{s} + M_\nu^* - \hat{M}_\nu \neq 0$ over $\mathbb{K}$ by definition of Type 2.a forgery, $\mathcal{B}$ can output $U = \frac{x^*}{\hat{x}\prod_{j=1}^n F_\kappa(y_j)^{M_j^*-\hat{M}_j}}$ and $d = s^* - \hat{s} + M_\nu^* - \hat{M}_\nu$.

If $\mathcal{A}$ outputs a Type 2 forgery with non-negligible probability ϵ, then $\mathcal{B}$ breaks the security of the algebraic TDP with non-negligible probability ϵ/n.

Type 2.b The proof for this case is almost identical to that of Type 2.a except for the following changes. In the Key Generation there is no guess about the index ν, and all values g_j are simulated as $g_j = F_\kappa(y_j)$ for random $y_j \in \mathcal{X}_\kappa$. To answer signing queries, $\mathcal{B}$ sets $s = -\beta_{\tau,i}$. Finally, given the adversary's forgery, it holds

$$F_\kappa\left(\frac{x^*}{\hat{x}\prod_{j=1}^n F_\kappa(y_j)^{M_j^*-\hat{M}_j}}\right) = \rho^{s^*-\hat{s}}$$

Hence, $U = \frac{x^*}{\hat{x}\prod_{j=1}^n F_\kappa(y_j)^{M_j^*-\hat{M}_j}}$ and $d = s^* - \hat{s}$ form a valid solution for breaking the strong one-wayness of the algebraic TDP.

Efficiency and Comparisons. The most attractive feature of our proposal is that it allows for great variability of the underlying message space. In particular our scheme allows to consider finite fields of arbitrary size without sacrificing efficiency[6]. This is in sharp contrast with previous solutions which can either support only large fields (whose size directly depends on the security parameter e.g., [9, 22, 3, 11, 10, 14, 15, 20]) or are much less efficient in practice [11, 10].

Here we discuss in more details the efficiency of our scheme when instantiated with our RSA and Factoring based Algebraic TDP. Since each signature $\sigma = (x, s)$ consists of an element $x \in \mathbb{Z}_N^*$ and a value s in the field $\mathbb{K}$, i.e., its size is $|\sigma| = |N| + |S|$ where $|N|$ is the bit size of the RSA modulus and $|S|$ is the bit size of the cardinality S of the field $\mathbb{K}$. Ignoring the cost of hashing, both signing and verifying require one single multi exponentiation (where all exponents have size $|S|$) and one additional exponentiation. Thus the actual efficiency of the scheme heavily depends on the size of $|S|$. For large values of $|S|$ our scheme is no better than previous schemes (such as the RSA schemes by Gennaro *et al.* [22] and by Catalano, Fiore and Warinschi [15]). For smaller $|S|$, however, our schemes allow for extremely efficient instantiations. If we consider for instance the binary field $\mathbb{F}_2$, then generating a signature costs only (again ignoring the cost of hashing) one square root extraction and a bunch of multiplications. Notice however that for the specific N (i.e. $N = pq$ where $p = 2p' + 1$, $q = 2q' + 1$ and p', q' are both primes) considered in our instantiations, extracting square root costs one single exponentiation (i.e., one just exponentiates to the power $z = 2^{-1} \bmod p'q'$). Verification is even cheaper as it requires (roughly) $m + n$ multiplications.

As mentioned above, the only known schemes supporting small fields are those by Boneh and Freeman [11, 10]. Such schemes are also secure in the random oracle model, but rely on the hardness

[6] In fact, the exact size of the field can be chosen ad-hoc (e.g., according to the desired application) at the moment of instantiating the scheme

of SIS-related problems over lattices. There, a signature is a short vector σ in the lattice, whereas the basic signing operation is computing a short vector in the intersection of two integer lattices. This is done by using techniques from [25, 13]. Even though the algebraic tools underlying our scheme are significantly different with respect to those used in [11, 10] and it is not easy to make exact comparisons, it is reasonable to expect that taking a square root in $\mathbb{Z}_N$ is faster than state-of-the-art preimage sampling for comparable security levels.

6 An efficient Σ protocol for Algebraic One-Way Permutations

Here we propose an efficient Σ protocol for any Algebraic One-Way Permutation (OWP). Let $F_\kappa : \mathcal{X}_\kappa \to \mathcal{X}_\kappa$ be an algebraic one-way permutation. We let RndSp coincide with $\mathcal{X}_\kappa$ and $\mathsf{ChSp} = \mathbb{K}$. Let L be the language $\{\langle y, F_\kappa\rangle : \exists z \in \mathcal{X}_\kappa \, s.t. \, F_\kappa(z) = y\}$ and $\mathcal{R}$ be the corresponding relation (i.e. $(x = \langle y, F_\kappa\rangle, z) \in \mathcal{R}$ iff $F_\kappa(z) = y$).

- $\Sigma.\mathsf{Setup}(1^\lambda, \mathcal{R})$ It runs $\kappa \leftarrow \mathsf{Gen}(1^\lambda)$. Next it chooses at random $z \in \mathcal{X}_\kappa$ and computes $F_\kappa(z) = y$. The statement is set as $x \leftarrow \langle y, F_\kappa\rangle$, while the witness is z.
- $\Sigma.\mathsf{Com}(x; r) \to R$ On input $x = \langle y, F_\kappa\rangle$ and random coins r in RndSp, outputs the first message $R \leftarrow F_\kappa(r)$.
- $\Sigma.\mathsf{Resp}(x, w, r, c) \to s$ Output $s \leftarrow r \cdot z^c \in \mathcal{X}_\kappa$.
- $\Sigma.\mathsf{Ver}(x, R, c, s) \to 0/1$ On input $R \in \mathcal{Y}_\kappa$, $c \in \mathsf{ChSp}$ and $s \in \mathcal{X}_\kappa$, outputs 1 if $F_\kappa(s) = R \cdot y^c$ or 0 otherwise.

Correctness is obvious by inspection. Special soundness comes from the fact that the function is strongly one-way. Indeed, the extractor $\Sigma.\mathsf{Ext}$, on input x, R, c, s, c', s', works as follows. It sets $x' \leftarrow s \cdot (s')^{-1} (= z^{c-c'})$. Next, it sets $d \leftarrow c - c' \in \mathbb{K}$ where $(c - c') \neq 0$ in $\mathbb{K}$. The extractor outputs (x', d). Clearly such a couple contradicts the strong one wayness of the function as $F_\kappa(x') = F_\kappa(z^d) = y^d$. Honest verifier zero knowledge can be proved as follows. The simulator, on input $(x = \langle y, F_\kappa\rangle, c)$, chooses a random $s \in \mathcal{X}_\kappa$ and sets $R \leftarrow F_\kappa(s) y^{-c}$. The output is (R, s). Clearly $\Sigma.\mathsf{Ver}(x, R, c, s) = 1$ and the probability distribution of (R, c, s) is identical to that obtained by running the real algorithms.

6.1 Efficient Batch Execution of Sigma Protocols

In this section we present a generalization of the above Sigma protocol to the case in which the statement being proven consists of multiple values $x \leftarrow \langle y_1, \ldots, y_\ell, F_\kappa\rangle$, while the witness is the corresponding z_i such that $F_\kappa(z_i) = y_i$ for $i = 1, \ldots, \ell$.

A naive approach would be to compose the original Sigma protocol in parallel ℓ times. In other words the prover would send over ℓ commitments and the verifier would reply with ℓ challenges - one per identity. Note that this scheme has a communication and computation cost that is ℓ times the cost of the original protocol. A possible improvement would be to use the same challenge for all rounds, and apply batch verification techniques (such as the ones in [6]) to the last verification step. Even with these improvements, the communication and computation cost of the whole scheme would still be higher by a factor of ℓ (the prover would still have to send and compute ℓ commitments).

Following [23] we propose a more efficient scheme where the prover sends *one* commitment and the verifier sends one challenge across all identities. The prover's response is generalized from a degree one polynomial to a degree ℓ polynomial formed from the ℓ secret keys. In [23] this approach was applied to the Schnorr's protocol [42]. Using our abstraction of algebraic OWFs, we generalize this approach to the entire family of Sigma protocols described above. In particular for the instantiation of Algebraic OWP based on Factoring/RSA, we obtain an efficient batch execution of the Guillou-Quisquateur protocol [28], which was left as an open problem in [23].

We now describe our protocol $\mathsf{Batch\text{-}Sigma}$:

- $\Sigma.\mathsf{Setup}(1^\lambda, \mathcal{R})$ It runs $\kappa \leftarrow \mathsf{Gen}(1^\lambda)$. Next it chooses at random ℓ values $z_i \in \mathcal{X}_\kappa$ and computes $F_\kappa(z_i) = y_i$. The statement is set as $x \leftarrow \langle y_1, \ldots, y_\ell, F_\kappa \rangle$, while the witness is $\langle z_1, \ldots, z_\ell \rangle$.
- $\Sigma.\mathsf{Com}(x; r) \rightarrow R$ On input x and random coins r in RndSp, outputs the first message $R \leftarrow F_\kappa(r)$.
- $\Sigma.\mathsf{Resp}(x, w, r, c) \rightarrow s$ Output $s \leftarrow r \cdot \prod_{i=1}^{\ell} z_i^{c^i} \in \mathcal{X}_\kappa$ where c^i is computed over the field $\mathbb{K}$ defined by (Gen, F).
- $\Sigma.\mathsf{Ver}(x, R, c, s) \rightarrow 0/1$ On input $R \in \mathcal{Y}_\kappa$, $c \in \mathsf{ChSp}$ and $s \in \mathcal{X}_\kappa$, outputs 1 if $F_\kappa(s) = R \cdot \prod_{i=1}^{\ell} y_i^{c^i}$ or 0 otherwise.

Concretely, to support the batch verification of ℓ statements, we need an algebraic OWP with a field $\mathbb{K}$ of size at least $\ell + 1$[7]. Correctness and Honest verifier zero knowledge can be proven as for the single-statement case. Special soundness clearly does not hold, as two transcripts with the same commitment and two distinct challenges do not yield a sufficient number of equations from which to extract the ℓ witnesses. What we are able to prove, however, is that $\mathsf{Batch\text{-}Sigma}$ is a proof of knowledge, i.e. it is possible to extract the witness from a prover that succeeds with a sufficiently high probability.

Theorem 7. $\mathsf{Batch\text{-}Sigma}$ *is a proof of knowledge of* $\langle z_1, \ldots, z_\ell \rangle$.

Proof. A fraudulent prover can cheat by guessing the correct challenge c ahead of time and sending the commitment R such that the verification equation is satisfied for a randomly chosen s. The probability of success for this attack is at most 2^{-t} where $t = |\mathsf{ChSp}|$.

In the proposition to follow we show that if a prover has probability of success significantly larger than 2^{-t}, then all the witnesses can be "extracted" from such a prover. The basic idea of the proof is that if we can generate $\ell + 1$ transcripts with the same commitment R, then we have enough relationships to compute all the witnesses.

In [23] these relationships were simple linear equations and the witnesses could be easily computed by inverting the matrix of such a system of equations (which is invertible being a Van der Monde matrix). In our case the proof is complicated by the fact that the inverse matrix may not be efficiently computable, yet using the *field-homomorphism* property of the underlying algebraic one-way function we will be able to extract the witnesses.

Let us introduce some notation. Let P' (the fraudulent prover) be any PPT Turing machine that runs on the common input of $\mathsf{Batch\text{-}Sigma}$. Let RP denote the random string of P'. Let success bit $S(RP, c)$ be 1 if P' succeeds with RP on challenge c and 0 otherwise. The success rate S is defined to be the average over $S(RP, c)$ where RP and c are chosen uniformly at random. Let T be the running time of P', note that we may assume T to be independent of RP and c since limiting the time to twice the average running time for successful pairs RP and c decreases the success rate by at most a factor of 2.

We postpone the proof of the following proposition.

[7] This is due to the fact that we need at least $\ell + 1$ distinct values $c_j \in \mathbb{K}$ in order for our Proposition 1 to hold.

Proposition 1. *If the success rate S of* P' *is greater than 2^{-t+1} then there exists a PPT Turing machine* TE *(transcript extractor) which, given black box access to* P'*, runs in expected time $O(d \log d \cdot T/S)$ and computes $\ell+1$ transcripts of the form R, c_j, s_j where all the c_j's are distinct and the transcripts satisfy the verification equation*

$$F_\kappa(s_j) = R \cdot \prod_{i=1}^{\ell} y_i^{c_j^i}$$

Note that if S is non-negligible and T is polynomial, the running time of TE is polynomial.

Therefore we run TE to obtain the above $\ell+1$ transcripts. Consider the Van der Monde matrix $C = (c_j^i)$ and let Δ be an integer such that $\Delta \cdot det(C)$ is also an integer. By using simple linear algebra "in the exponent" we can then recover the values z_i^{Δ}.

We now continue as in the case of the basic Sigma protocol. Compute $d \in \mathbb{K}$ such that $d\Delta = 1_{\mathbb{K}}$ (remember that the challenge space ChSp from where the c_i are chosen is set to $\mathbb{K}$). Of course such a value is guaranteed to exist as long as $\Delta \neq 0$, moreover it can be computed efficiently using the extended euclidean algorithm. Finally, for all i, it runs Eval on input $(\kappa, y_i, z_i^{\Delta}, \Delta, d)$, thus getting $z_i^{\Delta d} = z_i$ which is the required witness.

To finish the proof of Theorem 7 we need to prove Proposition 1.

Proof. Algorithm TE runs as follows:

1. It picks an RP at random and simulates P' using a random challenge, say c_1. If P' fails then it repeats step 1 with a new RP. Otherwise it goes to step 2.
2. Let u be the number of iterations of Step 1. Now hold RP fixed and probe up to $(8u)(\ell+1) \cdot \log(\ell+1)$ random c's while rewinding P' each time to the point after which he sent the initial commitment R. The goal is to find a total of $\ell+1$ distinct c's, $c_1, c_2 \ldots c_{\ell+1}$ on which P' succeeds. If it fails in this attempt to find $\ell+1$ c's it then goes back to step 1.

To analyze the running time of TE, we need some additional definitions and two auxiliary results. Define $S(RP)$ to be the fraction of c for which $S(RP, c)$ is 1. Define RP to be "good" if $S(RP)$ is at least $S/2$. Let $\#RP$ denote the size of the set of all RP and $\#c$ the size of the set of all c. Note that $\#c = 2^{t+\log \ell}$.

Lemma 3. *With probability at least 1/2,* TE *picks a good RP in step 1.*

Proof. Note that the mean of $S(RP)$, over all RP chosen uniformly at random, is S. Now $\Sigma_{RP} S(RP) = \#RP \cdot S$. But since $\Sigma_{\text{not-good}RP} S(RP) \leq \#RP \cdot S/2$ it follows that $\Sigma_{\text{good}RP} S(RP) > \#RP \cdot S/2$. In other words the set of RP, c for which $S(RP, c)$ is 1 and RP is good is at least half the entire set for which $S(RP, c)$ is 1. Hence, with probability at least 1/2, RP is good.

Lemma 4 (Coupon collector lemma). *With probability at least 1/2, for a good RP,* KE *will succeed in finding a total of $\ell+1$ c's, $c_1, c_2 \ldots c_{\ell+1}$ on which* P' *succeeds, using up to $(4/S)(\ell+1) \cdot \log(\ell+1)$ random probes.*

Proof. Fix the good RP. Observe that since RP is good there must be greater than $S/2$ c's such that $S(RP, c)$ is 1, i.e. there must be greater than $2^{-t} \cdot 2^{t+log\ell} = \ell$ successful c's. Let there be

$k \geq S/2 \cdot 2^{t+log\ell} \geq \ell + 1$ successful c's (i.e. c's for which $S(RP, c)$ is 1). Then the expected number of probes to find $\ell + 1$ distinct successful c's is

$$2^{t+\log \ell} \left(\frac{1}{k} + \frac{1}{k-1} + \ldots + \frac{1}{k-\ell} \right).$$

Since $k \geq S/2 \cdot 2^{t+log\ell}$ the expected number of probes is at most

$$\frac{2k}{S} \left(\frac{1}{k} + \frac{1}{k-1} + \ldots + \frac{1}{k-\ell} \right)$$

which is at most $(2/S)(\ell+1)\log(\ell+1)$. Hence with probability at least $1/2$, TE will succeed using at most twice the expected number of probes.

We now return to the proof of Proposition 1. First observe that the expected number of probes in step 1 is $1/S$. Next, observe that, since the expectation of u is $1/S$, with probability at least $1/2$, $u \geq (1/2)(1/S)$. By Lemma 3 RP is good with probability $1/2$. Hence with probability at least $1/4$, we have that both $u \geq (1/2)(1/S)$ and RP is good. Then by Lemma 4, TE will succeed in step 2 with probability at least $1/2$. Since each probe takes $O(T)$ steps it follows that with probability at least $1/8$, TE succeeds in $O(\ell \log \ell \cdot T/S)$ steps. Hence the expected time is bounded by $((1/8) + (7/8)(1/8) + (7/8)(7/8)(1/8) + \ldots) \cdot O(\ell \log \ell \cdot T/S) = O(\ell \log \ell \cdot T/S)$ steps.

References

1. R. Ahlswede, Ning-Cai, S. Li, and R. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.
2. B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In S. Abramsky, C. Gavoille, C. Kirchner, F. Meyer auf der Heide, and P. G. Spirakis, editors, *ICALP 2010: 37th International Colloquium on Automata, Languages and Programming, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 152–163, Bordeaux, France, July 6–10, 2010. Springer, Berlin, Germany.
3. N. Attrapadung and B. Libert. Homomorphic network coding signatures in the standard model. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 17–34, Taormina, Italy, Mar. 6–9, 2011. Springer, Berlin, Germany.
4. L. Babai. Trading group theory for randomness. In *17th Annual ACM Symposium on Theory of Computing*, pages 421–429, Providence, Rhode Island, USA, May 6–8, 1985. ACM Press.
5. M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, A. Küpçü, and A. Lysyanskaya. Incentivizing outsourced computation. In *Workshop on Economics of Networked Systems – NetEcon*, pages 85–90, 2008.
6. M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 236–250, Espoo, Finland, May 31 – June 4, 1998. Springer, Berlin, Germany.
7. S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 111–131, Santa Barbara, CA, USA, Aug. 14–18, 2011. Springer, Berlin, Germany.
8. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer, Berlin, Germany.
9. D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In S. Jarecki and G. Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 68–87, Irvine, CA, USA, Mar. 18–20, 2009. Springer, Berlin, Germany.

10. D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.
11. D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 1–16, Taormina, Italy, Mar. 6–9, 2011. Springer, Berlin, Germany.
12. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, Dec. 9–13, 2001. Springer, Berlin, Germany.
13. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.
14. D. Catalano, D. Fiore, and B. Warinschi. Adaptive pseudo-free groups and applications. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 207–223, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.
15. D. Catalano, D. Fiore, and B. Warinschi. Efficient network coding signatures in the standard model. In *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, Lecture Notes in Computer Science, pages 680–696. Springer, Berlin, Germany, 2012. Also in Cryptology ePrint Archive `http://eprint.iacr.org/2011/696`.
16. K.-M. Chung, Y. Kalai, and S. P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 483–501, Santa Barbara, CA, USA, Aug. 15–19, 2010. Springer, Berlin, Germany.
17. Y. Dodis, I. Haitner, and A. Tentes. On the instantiability of hash-and-sign rsa signatures. In *TCC 2012: 9th Theory of Cryptography Conference*, pages 112–132, 2012.
18. D. Fiore and R. Gennaro. Improved publicly verifiable delegation of large polynomials and matrix computations. Cryptology ePrint Archive, Report 2012/434, 2012.
19. D. Fiore and R. Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *2012 ACM Conference on Computer and Communication Security*. Full version available at `http://eprint.iacr.org/2012/281`. ACM Press, October 2012.
20. D. M. Freeman. Improved security for linearly homomorphic signatures: A generic framework. In *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, Lecture Notes in Computer Science, pages 697–714. Springer, Berlin, Germany, 2012.
21. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482, Santa Barbara, CA, USA, Aug. 15–19, 2010. Springer, Berlin, Germany.
22. R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin. Secure network coding over the integers. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 142–160, Paris, France, May 26–28, 2010. Springer, Berlin, Germany.
23. R. Gennaro, D. Leigh, R. Sundaram, and W. S. Yerazunis. Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In P. J. Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 276–292, Jeju Island, Korea, Dec. 5–9, 2004. Springer, Berlin, Germany.
24. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, Maryland, USA, May 31 – June 2, 2009. ACM Press.
25. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.
26. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In R. E. Ladner and C. Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 113–122, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.
27. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
28. L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT'88*,

volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Davos, Switzerland, May 25–27, 1988. Springer, Berlin, Germany.
29. D. Hofheinz, T. Jager, and E. Kiltz. Short signatures from weaker assumptions. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 647–666, Seoul, South Korea, Dec. 4–8, 2011. Springer, Berlin, Germany.
30. R. Johnson, D. Molnar, D. X. Song, and D. Wagner. Homomorphic signature schemes. In B. Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262, San Jose, CA, USA, Feb. 18–22, 2002. Springer, Berlin, Germany.
31. A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. http://eprint.iacr.org/.
32. J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *24th Annual ACM Symposium on Theory of Computing*, pages 723–732, Victoria, British Columbia, Canada, May 4–6, 1992. ACM Press.
33. A. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. Crypto 2012, to appear.
34. S. Micali. Cs proofs. In *35th Annual Symposium on Foundations of Computer Science*, Santa Fe, New Mexico, Nov. 20–22, 1994.
35. P. Mohassel. Efficient and secure delegation of linear algebra. Cryptology ePrint Archive, Report 2011/605, 2011.
36. F. Monrose, P. Wyckoff, and A. D. Rubin. Distributed execution with remote audit. In *ISOC Network and Distributed System Security Symposium – NDSS'99*, San Diego, California, USA, Feb. 3–5, 1999. The Internet Society.
37. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, Oct. 19–22, 1997. IEEE Computer Society Press.
38. C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. Cryptology ePrint Archive, Report 2011/587, 2011.
39. B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. TCC 2012, 2012.
40. M. O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Jan. 1979.
41. S.-Y. Robert-Li, R. Y. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.
42. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, Aug. 20–24, 1990. Springer, Berlin, Germany.
43. A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. Comput. Syst.*, 1(1):38–44, 1983.
44. S. W. Smith and S. Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, 31:831–860, 1999.
45. B. Yee. *Using Secure Coprocessors*. PhD thesis, Carnegie Mellon University, 1994.

A A linearly-homomorphic signature secure in the standard model for a bounded number of datasets

In this section we show that by requiring specific properties on the hash function H, the signature scheme described in the previous section can be proven secure without random oracles, but in a weaker security model in which the adversary is allowed to ask signatures on only Q different datasets, where Q is a (pre-specified) number. We call this notion Q-time security (similarly to the Q-time security of regular signatures). The basic idea for doing the proof is to require H to be a weak programmable hash function, following the definition proposed by Hofheinz, Jager and Kiltz in [29]. However, for some technical details, we extend the original definition as we describe below.

Weak Programmable Hash Functions. A *group hash function* H for $\mathbb{G}$ consists of two algorithms ($\mathsf{PHF.Gen}, \mathsf{PHF.Eval}$) such that: $\mathsf{PHF.Gen}(1^\lambda)$ takes as input the security parameter λ and outputs a description of the function K (which also contains a description of the input space $\mathcal{I}$); $\mathsf{PHF.Eval}(K, X)$ given the key K and a value $X \in \mathcal{I}$ evaluates the function $H_K(X) \in \mathbb{G}$.

Definition 8 (Weak Programmable Hash Functions [29]). *A group hash function* $H = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ *is a* weak (m, n, γ, η)-programmable hash function *(weak PHF, for short) if there exist efficient trapdoor generation* $\mathsf{PHF.TrapGen}$ *and trapdoor evaluation* $\mathsf{PHF.TrapEval}$ *algorithms such that:*

1. $\mathsf{PHF.TrapGen}(1^\lambda, g, h, X_1, \ldots, X_m)$ *is given the security parameter* λ*, two generators* $g, h \in \mathbb{G}$*, and* m *input values* $X_1, \ldots, X_m \in \mathcal{I}$*, and it outputs a key* K *and a trapdoor* t*.*
2. *For all* $g, h \in \mathbb{G}$ *and* $X_1, \ldots, X_m \in \mathcal{I}$*, the keys* $K \xleftarrow{\$} \mathsf{PHF.Gen}(1^\lambda)$ *and* $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h, X_1, \ldots, X_m)$ *are statistically* γ*-close.*
3. *Given* $X \in \mathcal{I}$, $\mathsf{PHF.TrapEval}(t, X)$ *produces two integers* $a_X, b_X \in \mathbb{Z}$ *such that* $\mathsf{PHF.Eval}(K, X) = g^{a_X} h^{b_X}$*.*
4. *For all* $X_1, \ldots, X_m \in \mathcal{I}$*, all* $(K, t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h, X_1, \ldots, X_m)$*, and any* $Z_1, \ldots, Z_n \in \mathcal{I}$ *such that* $Z_i \neq X_j$ *for all* i, j*, it holds*

$$\Pr[a_{X_1} = \cdots = a_{X_m} = 0 \wedge a_{Z_1}, \ldots, a_{Z_n} \neq 0] \geq \eta$$

where $(a_{X_i}, b_{X_i}) = \mathsf{PHF.TrapEval}(t, X_i)$*,* $(a_{Z_j}, b_{Z_j}) = \mathsf{PHF.TrapEval}(t, Z_j)$ *and the probability is taken over the random coins in the generation of* (K, t)*.*

Hofheinz, Jager and Kiltz proposed a refinement of the notion of programmable hash functions, called "evasively PHF" [29]. Roughly speaking, a PHF is evasively secure if in the property 4, the inequality $a_{Zi} \neq 0$ is replaced by $gcd(a_{Z_i}, e) = 1)$, for some prime number e. In what follows, we extend that definition and defining weak PHFs that are $\mathbb{K}$-evasively with respect to a finite field $\mathbb{K}$.

Definition 9 (Weak $\mathbb{K}$-Evasively Programmable Hash Functions). *Let* $\mathbb{K}$ *be a finite field. We say that a group hash function is a* $\mathbb{K}$-evasively *weak* (m, n, γ, η)*-programmable hash function if it satisfies all four properties of Definition 8 except that in property 4 the inequalities* $a_{Z_1}, \ldots, a_{Z_n} \neq 0$ *are required to hold over the finite field* $\mathbb{K}$*, instead of* $\mathbb{Z}$*.*

$\mathbb{K}$ can be any finite field of the form $\mathbb{F}_e$ where $e \geq 2$ is a prime number. In this sense, this definition is almost the same as the one by Hofheinz et al., except that we do not ask any specific requirement on the size of the prime e (in contrast, they require $|X| < e \leq |\mathbb{G}|$).

Finally, for the sake of our application, we define an additional property of a group hash function H, which is a special form of m-wise independence.

Definition 10 (m-wise Independent Group Hash Functions). *Let* $\mathbb{K}$ *be a finite field. A group hash function* $H = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ *is* m-wise independent for $\mathbb{K}$ *if there exist efficient trapdoor generation* $\mathsf{PHF.TrapGen}$ *and trapdoor evaluation* $\mathsf{PHF.TrapEval}$ *algorithms such that:*

1. $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ *is given the security parameter* λ*, two generators* $g, h \in \mathbb{G}$*, and it outputs a key* K *and a trapdoor* t*.*
2. *For all* $g, h \in \mathbb{G}$*, the keys* $K \xleftarrow{\$} \mathsf{PHF.Gen}(1^\lambda)$ *and* $(K', t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h)$ *are statistically close.*
3. *Given* $X \in \mathcal{I}$*,* $\mathsf{PHF.TrapEval}(t, X)$ *produces two integers* a_X, b_X *such that* $\mathsf{PHF.Eval}(K, X) = g^{a_X} h^{b_X}$*.*

4. *For all $g, h \in \mathbb{G}$, all $(K, t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h)$, for all distinct $X_1, \ldots, X_m \in \mathcal{I}$, and $\forall d_1, \ldots, d_m \in \mathbb{K}$, it holds*

$$\Pr[a_{X_1} = d_1 \wedge \cdots \wedge a_{X_m} = d_m] = \frac{1}{|\mathbb{K}|^m}$$

where $(a_{X_i}, b_{X_i}) = \mathsf{PHF.TrapEval}(t, X_i)$ and the probability is taken over the generation of (K, t).

An instantiation for the group of quadratic residues $\mathbb{QR}_N$. Hofheinz *et al.* already propose in [29] a weak evasively PHF for the group $\mathbb{QR}_N$ of quadratic residues. Here we show that the same construction, with a small adaptation, satisfies our $\mathbb{K}$-evasively extension for all finite fields $\mathbb{F}_e$ for $e \geq 2$ prime.

The construction is the following:

- $\mathsf{PHF.Gen}(1^\lambda)$ outputs $K = (h_0, \ldots, h_\ell)$ where $h_i \xleftarrow{\$} \mathbb{QR}_N$ for $i = 0$ to m.
- $\mathsf{PHF.Eval}(K, X)$: on input a key K and $X \in \{0,1\}^l$, compute

$$H_K(X) = \prod_{j=1}^{\ell} h_i^{X^i}$$

where X is interpreted as an integer in the canonical way.

Our adaptation consists into restricting the input space $\mathcal{I}$ to being the set of integers X of a certain fixed length l such that $X = 0$ over $\mathbb{K}$ if and only if $X = 0$ over $\mathbb{Z}$. We will show later in this section that given any integer X it is possible to map X to another integer X' such that X' has the desired property, i.e., $X' = 0$ over $\mathbb{K}$ iff $X = 0$ over $\mathbb{Z}$.

Theorem 8. *Let e be a prime number and let $\mathbb{K}$ be the finite field $\mathbb{F}_e$. The construction above is a weak $\mathbb{K}$-evasively $(\ell, 1, \gamma, \eta)$-programmable hash function for the group $\mathbb{QR}_N$ of quadratic residues modulo $N = pq$, where N is a Blum integer, product of safe primes, $\gamma = (\ell+1)/\sqrt{N}$ and $\delta = 1$.*

Proof. The proof of this theorem is essentially the same as the proof given in [29] except for a few observations that are needed to prove that it is $\mathbb{K}$-evasive. For completeness, we recall most of the proof pointing out the main differences.

- The trapdoor generation algorithm $\mathsf{PHF.TrapGen}(1^\lambda, g, h, X_1, \ldots, X_m)$ samples random values $\beta_0, \ldots, \beta_\ell \xleftarrow{\$} \{1, \ldots, N^2\}$ and computes the integer coefficients $\alpha_0, \ldots, \alpha_\ell$ of the polynomial

$$\alpha(w) = \sum_{j=0}^{\ell} \alpha_i \cdot w^i = \prod_{j=1}^{\ell} (w - X_j)$$

  Then it outputs $K = (h_0, \ldots, h_\ell)$ where $h_i = g^{\alpha_i} h^{\beta_i}$, for $i = 0$ to m, and the trapdoor $t = (\alpha_0, \beta_0, \ldots, \alpha_\ell, \beta_\ell)$.
  We defer the reader to [29] for the proof that the trapdoor key K is statistically $(\ell+1)/\sqrt{N}$-close to the real one.
- $\mathsf{PHF.TrapEval}(t, X)$ outputs $a_X = \sum_{j=0}^{\ell} \alpha_i \cdot X^i$, $b_X = \sum_{j=0}^{\ell} \beta_i \cdot X^i$.
  It is easy to see that the trapdoor evaluation algorithm is correct. To prove that it is $\mathbb{K}$-evasive for $\mathbb{K} = \mathbb{F}_e$ we first observe that by definition of the polynomial $\alpha(w)$ we have that

$a_{X_1} = \cdots = a_{X_m} = 0$. Next, for $Z \neq X_i$ for all $i = 1, \ldots, m$, we have that $a_Z = \alpha(Z) \neq 0$ over the integers. It is left to observe that $a_Z \neq 0$ even over the field $\mathbb{K}$. This is true as all values X_i and Z are assumed to be zero over $\mathbb{K}$ only if zero over $\mathbb{Z}$, hence it holds $\alpha(Z) = \prod_{j=1}^{\ell}(Z - X_j) \neq 0$ over $\mathbb{K}$ with probability $\eta = 1$.

Then we prove that for the same group hash function there are trapdoor algorithms so that it can be shown to satisfy our m-wise independence property.

Theorem 9. *Let $\mu = e^t$ for $e \geq 2$ prime and let $\mathbb{K}$ be the finite field $\mathbb{F}_\mu$. The construction above is a ℓ-wise independent group hash function for the group $\mathbb{QR}_N$ of quadratic residues modulo $N = pq$, where N is a Blum integer, product of safe primes.*

Proof. The first part of the proof is similar to that of Theorem 8. Recall that $\mathbb{K} = \mathbb{F}_\mu$, where $\mu = e^t$ is a prime power. We distinguish between the two cases, when $\ell \leq \mu$ and when $\ell > \mu$.

For $\ell \leq \mu$ we show the following algorithms.

- The trapdoor generation algorithm $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ samples random values $\alpha_0, \ldots, \alpha_\ell \xleftarrow{\$} \mathbb{K}$, $\beta_0, \ldots, \beta_\ell \xleftarrow{\$} \{1, \ldots, N^2\}$ and outputs $K = (h_0, \ldots, h_\ell)$ where $h_i = g^{\alpha_i} h^{\beta_i}$, for $i = 0$ to m, and the trapdoor $t = (\alpha_0, \beta_0, \ldots, \alpha_\ell, \beta_\ell)$.
  By the same argument as in [29], the trapdoor key K is statistically $(\ell + 1)/\sqrt{N}$-close to the real one.
- $\mathsf{PHF.TrapEval}(t, X)$ outputs $a_X = \sum_{j=0}^{\ell} \alpha_i \cdot X^i, \;\; b_X = \sum_{j=0}^{\ell} \beta_i \cdot X^i$.
  To see that the function is m-wise independent we observe that: (1) given the key K produced by $\mathsf{PHF.TrapGen}$, the coefficients α_i of the polynomial $\alpha(w)$ are information theoretically hidden, and (2) when $\alpha(w)$ is reduced over the finite field $\mathbb{K}$ (e.g., it is reduced $\pmod e$) it is well known that the evaluation of the polynomial on up to m distinct points has the desired m-wise independence property.

In the case when $\ell > \mu$, we can use the above algorithms with some small changes. First, we define $\mathbb{K}'$ to be the extension field $\mathbb{F}_{\mu^s}$ of $\mathbb{F}_\mu$ where s is the smallest integer such that $\mu^s > \ell$. Second, the trapdoor generation algorithm $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ is changed so that the exponents α_i are now taken uniformly at random in $\mathbb{K}'$. Being $\mu^s > \ell$, by the same observation (2) above, it follows that the value $\alpha(w)$ is uniformly distributed over $\mathbb{K}'$. Hence, if we define $a_X = \alpha(X) \bmod \mu$ we get a_X to be uniformly distributed over $\mathbb{K} = \mathbb{F}_\mu$, which completes the proof.

Finally, before proving the security of our linearly-homomorphic signature scheme using weak PHFs, we show that our restriction on the input space of the function H can be efficiently realized via the following function.

Proposition 2. *Let $X \in \mathbb{Z}$ be an integer between 0 and n, and let $e \geq 2$. Let n' be the smallest multiple of e greater than n. We define the function $f_{e,n'}(x)$ as follows:*

- *if $x = 0$, set $f_{e,n'}(x) = 0$* **end**
- *else*   *if $x \neq 0 \bmod e$ set $f_{e,n'}(x) = x$* **end**
  else express $x = a_k e^k + a_{k-1} e^{k-1} + \ldots a_1 e$
- *if $a_1 \neq 0$ set $f(x) = n' + (a_k e^{k-1} + \ldots a_1)$* **end**
  else $n' = n' + n'/e, y = a_k e^{k-1} + \ldots a_1$ run $f_{e,n'}(y)$

Then we claim that the function $f(X)$ defined above satisfies the following properties:

1. $f(x) \equiv 0 \pmod{e}$ *iff* $x = 0$ *(over $\mathbb{Z}$)*
2. $f(x)$ *is injective*
3. $f(x) \in \{0, \ldots, 2n'\}$

Proof. 1. To see $x = 0 \Rightarrow f(x) \equiv 0 \pmod{e}$, this follows by definition, i.e., $f(0) = 0$. To see $f(x) \equiv 0 \pmod{e} \Rightarrow x = 0$, observe that $f(0) = 0$, and for any $x \neq 0$, this follows from the injectivity property (see below).

2. It is obvious that for $x \neq 0 \bmod e$ there cannot be collisions. Now let $A = \{a | a = 0 \bmod e\}$. For contradiction let's assume that $f(x) = f(x')$, where $x, x' \in A$. If $x \neq x'$ then without loss of generality $x < x'$. Let

$$x = a_k e^k + \ldots + a_1 e, \quad x' = b_n e^n + \ldots + b_1 e.$$

The coefficient a_i and b_i uniquely define the numbers x and x', since the polynomial representation of any number on a prime base is unique. For the case when $a_1 \neq 0$, we claim that $x < x'$ implies that $f(x) < f(x')$. We have that $f(x) = a_k e^{k-1} + a_{k-1} e^{k-2} + \ldots + a_2 e + a_1$. Now observe that $f(x') \geq b_k e^{k-1} + b_{k-1} e^{k-2} + \ldots + b_2 e + b_1$ by the construction of the function. Since $x < x'$ it follows that

$$a_k e^{k-1} + a_{k-1} e^{k-2} + \ldots + a_2 e + a_1 < b_k e^{k-1} + b_{k-1} e^{k-2} + \ldots + b_2 e + b_1$$

hence $f(x) < f(x')$.

The tricky case is when $a_1 = 0$ in which case the function will need at least an additional step to converge to the final value $f(x)$. The function firstly calculates

$$y = a_k e^{k-1} + a_{k-1} e^{k-2} + \ldots + a_2 e + a_1,$$

practically same as before, only now we assumed $a_1 = 0$ (which forces $y = 0 \bmod e$) hence y cannot be assigned to $f(x)$. To calculate $f(x)$ the function is called again on y, but this time we replace n' with $N' = n' + \frac{n'}{e}$. This replacement of n' forces $f(x) > N'$.

Now if $b_1 \neq 0$ then by the explanation above $f(x')$ will be mapped in the area between $n' + \frac{n'}{e}$. This area is big enough to fit all the multiples of e less than n' since there are n'/e of them. In this case $N' > f(x')$, and by above $f(x) > N'$, hence $f(x) > f(x')$ and we are done. If $b_1 = 0$ the function needs at least one more step to calculate $f(x')$. The function will first calculate

$$y' = b_k e^{k-1} + b_{k-1} e^{k-2} + \ldots + b_2 e + b_1$$

and call the function again on y', replacing n' with $N' = n' + \frac{n'}{e}$. Since $x < x'$ implies $y < y'$, and the function is now called on y and y', by induction we deduce that $f(x) < f(x')$. Final thing to check is that the algorithm indeed concludes in finite number of steps, which is easily seen since at every step we reduce the powers of the representations, hence it has to hit one of the end points.

Hence we see that in all the cases above we arrive at $f(x) < f(x')$ or $f(x) > f(x')$. This contradicts the initial assumption of $f(x) = f(x')$ and the claim follows.

3. The element that gets mapped further to the right, is the one that takes the highest number of steps to be calculated. That element is e^i where i is the largest integer such that $e^i < n$. See that $f(e^i) = n' + n'/e + n/e^2 + \ldots + n'/e^i + 1$. In fact the range of all but n'/e^2 multiples of e lie on $[n', n' + n'/e]$, the rest but n'/e^3 multiples of e lie on $[n' + n'/e, n' + n'/e^2]$, and so on. So if k is the smallest k such that $n' < e^k$ then the size of the range has size less than $\sum_{i=0}^{k} n'/e^i < 2n'$. Hence the range of f is contained in $\{0, \ldots, 2n'\}$.

Security of the linearly-homomorphic signature in the standard model. Once we have provided all relevant definitions of programmable hash functions we can now show that the linearly-homomorphic signature scheme of Section 5 is Q-time secure in the standard model. Precisely, we consider the scheme in which the hash function $H(\tau, i)$ is defined as $H_{K_i}(\tau)$ for $i = 1, \ldots, m$, where $(K_1, \ldots, K_m)$ are m independent instances of weak $\mathbb{K}$-evasive programmable hash functions for $\mathcal{X}_\kappa$, where the group $\mathcal{X}_\kappa$ and the finite field $\mathbb{K}$ are those defined the by the algebraic TDP.

Theorem 10. *If (Gen, F) is a family of algebraic TDP and H is a group hash function for $\mathcal{X}_\kappa$ such that H is $\mathbb{K}$-evasively weak $(Q, 1, \gamma, \eta)$-programmable and H is Q-wise independent for $\mathbb{K}$ (where $\mathcal{X}_\kappa$ and $\mathbb{K}$ are the group and the finite field defined by F), then the linearly-homomorphic signature scheme described above is Q-time secure.*

Proof. The proof of the theorem is similar to that of the random oracle proof, except that here we use the programmability of the group hash function. Assume there exists an efficient adversary $\mathcal{A}$ that has non-negligible probability ϵ of winning the unforgeability game by querying at most Q datasets to the signing oracle. Let $(\tau^*, M^*, \sigma^*, f^*)$ be the valid forgery returned by the adversary, i.e., such that $\mathsf{Verify}(\mathsf{VK}, \tau^*, M^*, \sigma^*, f^*) = 1$. According to whether the forgery is of Type 1 or Type 2, we distinguish two types of adversaries. For every such adversary $\mathcal{A}$ we describe a simulation in which we reduce $\mathcal{A}$ to an algorithm $\mathcal{B}$ that breaks the strong one-wayness of the algebraic TDP with non-negligible probability.

Type 1. $\mathcal{B}$ takes as input (κ, ρ) where κ is the description of an algebraic TDP $F_\kappa : \mathcal{X}_\kappa \to \mathcal{X}_\kappa$ and $\rho \in \mathcal{X}_\kappa$. The goal of $\mathcal{B}$ is to find values $(x, d) \in \mathcal{X}_\kappa \times \mathbb{K}$ such that $F_\kappa(x) = \rho^d$ and $d \neq 0$. Our simulator $\mathcal{B}$ proceeds as follows.

Key Generation. $\mathcal{B}$ chooses in advance all tags $\tau_1, \ldots, \tau_Q \xleftarrow{\$} \{0,1\}^\lambda$ that it will use in the signing queries. Let T be the set of all such tags. $\mathcal{B}$ chooses $\mu \xleftarrow{\$} \{1, \ldots, m\}$ and $y_0, y_1, \ldots, y_n \xleftarrow{\$} \mathcal{X}_\kappa$ uniformly at random. For $j = 1$ to n, it sets $g_j = F_\kappa(y_j)$, and $u = F_\kappa(y_0)$. Then, for $i = 1$ to m, $i \neq \mu$ it chooses random generators $\rho_i = F_\kappa(\omega_i)$ and $h_i = F_\kappa(\delta_i)$ for $\omega_i, \delta_i \xleftarrow{\$} \mathcal{X}_\kappa$, and it sets $\rho_\mu = \rho$ and $h_\mu = F_\kappa(\delta_\mu)$ for $\delta_\mu \xleftarrow{\$} \mathcal{X}_\kappa$. It defines $X_1 = \tau_1, X_2 = \tau_2, \ldots, X_Q = \tau_Q$, and runs $(K_i, t_i) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, \rho_i, h_i, X_1, \ldots, X_\ell)$. It gives the public key $\mathsf{VK} = (\kappa, u, g_1, \ldots, g_n, K_1, \ldots, K_m)$ to the adversary. We notice that by the property of the PHF, the distribution of the simulated public key is negligibly close to that generated by the real key generation algorithm.

Signing queries. Let (F, i, M) be a signing query. If this is the first query with filename F, then $\mathcal{B}$ takes the next unused tag τ from T. Otherwise, let τ be the tag already chosen for F. Let $X = \tau$, and $(a_X^{(i)}, b_X^{(i)}) \leftarrow \mathsf{PHF.TrapEval}(K_i, X)$. By correctness of $\mathsf{PHF.TrapEval}$ it holds $H_{K_i}(X) = \rho_i^{a_X^{(i)}} h_i^{b_X^{(i)}} = F_\kappa(\delta_i^{b_X^{(i)}})$ as $a_X^{(i)} = 0$ (with probability 1) by property 4 of the PHF. For every $i \in \{1, \ldots, m\}$ (in both cases $i = \mu$ or $i \neq \mu$), $\mathcal{B}$ can easily simulate the signature by picking $s \xleftarrow{\$} \mathbb{K}$ at random, and computing $x = \delta^{b_X^{(i)}} y_0^s \prod_{j=1}^{n} y_j^{M_j}$ (in particular, for $i = \mu$, the exponent of $\rho_\mu = \rho$ is 0). It is easy to see that $\sigma = (x, s)$ is correctly distributed.

Forgery. Let $(\tau^*, M^*, \sigma^*, f^*)$ be the forgery returned by $\mathcal{A}$, and let $T' = \{\tau_1, \ldots, \tau_{Q'}\}$ be the set of all tags used in the signing queries. Notice that $T' \subseteq T$, $|T \setminus T'| \leq Q$ and that all unrevealed tags are completely unpredictable. By our assumption in this case of the proof, this is a Type 1 forgery, i.e., $\tau^* \notin T'$. Moreover, it must also be $f^* \neq 0^m$, i.e., there must exist an index $\mu^* \in \{1, \ldots, m\}$ such that $f^*_{\mu^*} \neq 0$.
If $f^*_\mu = 0$ or $\tau^* \in T \setminus T'$, then $\mathcal{B}$ aborts the simulation and fails. Otherwise, it continues the simulation. Notice though that $\Pr[\mu = \mu^*] = 1/m$ (as μ is perfectly hidden), and that $\Pr[\tau^* \in T \setminus T'] \leq Q/2^\lambda$. Therefore, $\mathcal{B}$ does not abort with probability at least $1/m(1 - Q/2^\lambda)$.
Let $Z = \tau^*$. By the validity of the forgery we have:

$$F_\kappa(x^*) = \prod_{i=1}^{m} H_{K_i}(\tau^*)^{f_i^*} \cdot u^{s^*} \prod_{j=1}^{n} g_j^{M_j^*} = \prod_{i=1, i\neq\mu}^{m} F_\kappa(\omega_i^{a_Z^{(i)}} \cdot \delta_i^{b_Z^{(i)}})^{f_i^*} (F_\kappa(\delta_\mu^{b_Z^{(\mu)}}) \cdot \rho^{a_Z^{(\mu)}})^{f_\mu^*} F_\kappa(y_0)^{s^*} \prod_{j=1}^{n} F_\kappa(y_j)^{M_j^*}$$

Thus, by the homomorphic property of F_κ we obtain:

$$F_\kappa\left(\frac{x^*}{\prod_{i=1, i\neq\mu}^{m}(\omega_i^{a_Z^{(i)}} \cdot \delta_i^{b_Z^{(i)}})^{f_i^*} \cdot \delta_\mu^{b_Z^{(\mu)} f_\mu^*} \cdot y_0^{s^*} \prod_{j=1}^{n} y_j^{M_j^*}}\right) = \rho^{a_Z^{(\mu)} f_\mu^*}$$

Therefore, $\mathcal{B}$ can output $U = \left(\frac{x^*}{\prod_{i=1, i\neq\mu}^{m}(\omega_i^{a_Z^{(i)}} \cdot \delta_i^{b_Z^{(i)}})^{f_i^*} \cdot \delta_\mu^{b_Z^{(\mu)} f_\mu^*} \cdot y_0^{s^*} \prod_{j=1}^{n} y_j^{M_j^*}}\right)$ and $d = a_Z^{(\mu)} f_\mu^*$ (which is $\neq 0$ over $\mathbb{K}$, as so are $a_Z^{(\mu)}$ and f_μ^* by assumption). If $\mathcal{A}$ outputs a Type 1 forgery with non-negligible probability ϵ, then $\mathcal{B}$ breaks the security of the algebraic TDP with non-negligible probability $\frac{\epsilon}{m}(1 - Q/2^\lambda)$.

Type 2. For a Type 2 adversary we have that $\tau^* = \tau_j$ for some $j \in \{1, \ldots, Q\}$, and $M^* \neq \hat{M} = f^*(M^{(1)}, \ldots, M^{(m)})$ where $(M^{(1)}, \ldots, M^{(m)})$ are the messages of the dataset with tag τ_j. Let $\hat{\sigma} = (\hat{x}, \hat{s}) = \mathsf{Hom.Eval}(\mathsf{VK}, \tau_j, f^*, \boldsymbol{\sigma}, \boldsymbol{M}, 1^m)$ be the signature obtained by correctly applying the $\mathsf{Hom.Eval}$ algorithm on the messages (and signatures) of the dataset τ_j with the function f^*. Since $M^* \neq \hat{M}$, there must exists an index $\nu \in \{1, \ldots, n\}$ such that $M^*_\nu \neq \hat{M}_\nu$. Then we distinguish the following two mutually exclusive cases:

(a) $s^* - \hat{s} + M^*_\nu - \hat{M}_\nu \neq 0$
(b) $s^* - \hat{s} + M^*_\nu - \hat{M}_\nu = 0$, i.e., $s^* - \hat{s} \neq 0$

where all inequalities are intended over $\mathbb{K}$.

We provide different simulations for the two cases.

Type 2.a $\mathcal{B}$ takes as input (κ, ρ) where κ is the description of an algebraic TDP $F_\kappa : \mathcal{X}_\kappa \to \mathcal{X}_\kappa$ and $\rho \in \mathcal{X}_\kappa$. The goal of $\mathcal{B}$ is to find values $(x, d) \in \mathcal{X}_\kappa \times \mathbb{K}$ such that $F_\kappa(x) = \rho^d$ and $d \neq 0$. Our simulator $\mathcal{B}$ proceeds as follows.

Key Generation. $\mathcal{B}$ chooses $\nu \xleftarrow{\$} \{1, \ldots, n\}$ and $y_1, \ldots, y_n \xleftarrow{\$} \mathcal{X}_\kappa$ uniformly at random. For $j = 1$ to n, $j \neq \nu$, it sets $g_j = F_\kappa(y_j)$, $g_\nu = F_\kappa(y_\nu) \cdot \rho$, and $u = \rho$. Then, for $i = 1$ to m, it chooses random generators $h_i = F_\kappa(\delta_i)$ for $\delta_i \xleftarrow{\$} \mathcal{X}_\kappa$, and it runs $(K_i, t_i) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, \rho, h_i)$. It gives the public key $\mathsf{VK} = (\kappa, u, g_1, \ldots, g_n, K_1, \ldots, K_m)$ to the adversary. We notice that by the property of the PHF, the distribution of the simulated public key is negligibly close to that generated by the real key generation algorithm.

Signing queries. Let (F, i, M) be a signing query. If this is the first query with filename F, then $\mathcal{B}$ chooses a new tag $\tau \xleftarrow{\$} \{0,1\}^\lambda$. Otherwise, let τ be the tag already chosen for F. Let $X = \tau$, $(a^{(i)}, b_X^{(i)}) \leftarrow \mathsf{PHF.TrapEval}(t_i, X)$. The simulator sets $s = -(b_X^{(i)} + M_\nu) \in \mathbb{K}$, and uses the strong one-wayness property of the algebraic TDP to compute the pre-image $\tilde{\rho} = F_\kappa^{-1}(\rho^{s+b_X^{(i)}+M_\nu})$ (this can be done efficiently as $s + b_X^{(i)} + M_\nu$ is 0 over $\mathbb{K}$). Then it sets $x = \delta_i \prod_{j=1}^n y_j^{M_j} \tilde{\rho}$ and returns $\sigma = (x, s)$. It is not hard to check that the signature is distributed correctly. In particular, it holds $F_\kappa(x) = H(\tau, i) \cdot u^s \prod_{j=1}^n g_j^{M_j}$ and s is uniform in $\mathbb{K}$ as so is $b_X^{(i)}$ by the Q-wise independence property of H.

Forgery. This part of the simulation is identical to that in Theorem 6.

Type 2.b This case of the proof is obtained by applying the same changes suggested in the corresponding case of the proof of Theorem 6 to the simulation of Type 2.a described above.