Algebraic (Trapdoor) One-Way Functions and Their Applications

Dario Catalano[1], Dario Fiore[2,⋆], Rosario Gennaro[3], and Konstantinos Vamvourellis[4]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy
`catalano@dmi.unict.it`

[2] Max Planck Institute for Software Systems (MPI-SWS), Germany
`fiore@mpi-sws.org`

[3] City College of New York, USA
`rosario@cs.ccny.cuny.edu`

[4] New York University, USA
`kv472@nyu.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 *ring-homomorphic*, meaning that it is possible to compute linear operations "in the exponent" over some ring (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 *flexibly 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. These include publicly verifiable secure outsourcing of polynomials, linearly homomorphic signatures and batch execution of Sigma protocols.

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.

⋆ Work entirely done while at NYU.

- *Ring-Homomorphic:* this is a new property saying, intuitively, that it is possible to efficiently perform linear operations "in the exponent" over some ring $\mathbb{K}$. While this property turns out to be equivalent to the homomorphic property for groups of known order n and the ring $\mathbb{K} = \mathbb{Z}_n$, 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 ring.
- *Flexibly One-Way:* We strengthen the usual notion of one-wayness in the following way: given $y = F(x)$ is should be unfeasible to compute (x', d) such that $F(x') = y^d$ and $d \in \mathbb{K}_{\neq 0}$ (in contrast with the traditional definition of one-wayness where d is fixed as 1).

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.

We demonstrate the existence of Algebraic OWFs with three instantiations, the security of which is deduced from the hardness of the Diffie-Hellman problem in groups with bilinear maps and the RSA/Factoring assumptions respectively.

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 rings of *arbitrary* size and characteristic and does not necessarily use bilinear maps.
- *Linearly Homomorphic Signature Schemes* also over *arbitrary* rings, and in particular even small fields such as $\mathbb{F}_2$. The only known constructions for the latter case require assumptions over lattices [8] 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 [20] 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 [25] which yields, to the best of our knowledge, the first batch verifiable Sigma protocol for groups of unknown order, a problem left open in [20].

The application to batch executions of identification protocols is deferred to the full version of this paper [11]. Below, we elaborate in detail about the improvements of our solutions to the remaining two applications.

1.1 Secure Outsourcing of Polynomials

Starting from work by Benabbas *et al.* [6], 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 verifying such a 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 [6], several other papers (e.g. [33,34,16]) have investigated this problem, focusing specifically on the feature of *public verification*, i.e. the proof of correctness of the result provided by the server can be verified by *anyone*. This comes in contrast with the original solution in [6] which obtained only private verification, i.e. the proof of correctness of the result provided by the server can be verified only by the client who initially stored the polynomial.

The popularity of this research problem can be explained by its numerous practical applications including, as discussed in [6], *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 [6,33,16] 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. That means that for the schemes to be secure, p must be large. Therefore up to now, none of the existing schemes could handle small field sizes. The solution recently proposed in [34] 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 [33,34,16] 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 rings of any size and any characteristic, thus allowing for much more flexibility and efficiency. Moreover, the schemes in [34] are based on specific Attribute-Based Encryption schemes (e.g. [28]) whose security relies on "q-type" assumptions, whereas our solution can do so based on the well known RSA/Factoring assumptions.

As in the case of [16] our techniques extend for building a protocol for *Matrix Multiplication*. In this problem (also studied in [30]) 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. [4,24,27,29,23]) and on the more applied arena (e.g. [31,5,37,38]).

Our work follows the "amortized" paradigm introduced in [18] (also adopted in [14,2]) 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. [21]) resulting in solutions of limited practical relevance. Instead, we follow [6] 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 [33] by providing a solution that works for finite rings of arbitrary characteristic (even small fields) and by avoiding the use of bilinear maps. Given that our solution is a generalization of [16] we also inherit all the improvements of that paper. In particular, compared to [33]:

- 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[1].

1.2 Linearly Homomorphic Signatures

Imagine a user Alice owns some data set $m_1, \ldots, m_n \in \mathcal{M}$ that she keeps (signed) in some database stored at a, not necessarily trusted, server. Imagine also that

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

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 [8]. 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 $f : \mathcal{M}^n \rightarrow \mathcal{M}$, one can compute a signature σ_f on the value $f(m_1, \ldots, m_n)$ *without* knowledge of the secret signing key SK. In other words, for a fixed set of original signed messages, it is possible to provide any $y = f(m_1, \ldots, m_n)$ with a proof of correctness σ_f. In particular the creation and the verification of σ_f does not require SK. The security definition is a relaxation over the classical security notion for signatures: it should be impossible to create a signature σ_f for $m \neq f(m_1, \ldots, m_n)$ without knowing SK.

The notion of homomorphic signature was introduced by Johnson *et al.* [26] and later refined by Boneh *et al.* [7]. Its main motivation was realizing a linear network coding scheme [1,35] secure against pollution attacks. The construction from [7] uses bilinear groups as the underlying tool and authenticates linear functions on vectors defined over large prime fields. Subsequent works considered different settings as well. In particular, the constructions in [19,12,13] are based on RSA, while [9,8] rely on lattices and can support linear functions on vectors over small fields. A general framework for building homomorphic signatures in the standard model, was recently provided by Freeman [17].

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 rings. While it was already known how to realize linearly homomorphic signatures over small fields [9,8], ours seem to be the first schemes achieving this in a very efficient way and based on simple assumptions such as Factoring and RSA. To give a more concrete idea about the efficiency of our scheme, if we consider the case of messages in $\mathbb{F}_2$, then our signing algorithm is more efficient than that in [8] in the same order of magnitude as taking a square root in $\mathbb{Z}_N^*$ is more efficient than sampling a pre-image in lattice-based trapdoor functions, at comparable security levels.

2 Preliminaries

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 $x \overset{\$}{\leftarrow} S$ the process of selecting x uniformly at random in S. Let $\mathcal{A}$ be a probabilistic algorithm. We denote with $x \overset{\$}{\leftarrow} \mathcal{A}(\cdot)$ the process of running $\mathcal{A}$ on some appropriate input and assigning its output to x.

Below we give informal definitions of verifiable computation and linearly homomorphic signatures. For more formal and precise descriptions, we defer the interested reader to the full version of this paper [11] and to relevant related work [34,17].

Verifiable Computation [34]. A Verifiable Computation scheme $\mathcal{VC}$ enables 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. A $\mathcal{VC}$ scheme for a class of functions $\mathcal{F}$ is defined by the following algorithms. The key generation $\mathsf{KeyGen}(1^\lambda, f)$, given a function $f \in \mathcal{F}$, 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. The problem generation algorithm $\mathsf{ProbGen}(\mathsf{PK}_f, \mathsf{SK}_f, x) \rightarrow (\sigma_x, \mathsf{VK}_x)$ takes a value $x \in \mathsf{Dom}(f)$, and is run by the delegator to produce an encoding σ_x of x, together with a public verification key VK_x. $\mathsf{Compute}(\mathsf{EK}_f, \sigma_x) \rightarrow \sigma_y$ is run by the worker to compute an encoded version of $y = f(x)$. The verification algorithm $\mathsf{Verify}(\mathsf{PK}_f, \mathsf{VK}_x, \sigma_y) \rightarrow y \cup \perp$ takes the public information and an encoded output σ_y, and returns a value y or an error $\perp$.

Intuitively, for security, we require that any PPT worker, with oracle access to $\mathsf{ProbGen}$, should not be able to cheat by producing a proof σ for $y' \neq f(x)$ that correctly verifies for $f(x)$.

Linearly-Homomorphic Signatures. Linearly-homomorphic signatures, as recently formalized in [9,8,17], extend the standard notion of digital signatures as follows. A linearly-homomorphic signature scheme consists of the following algorithms. The key generation $\mathsf{Hom.KG}(1^\lambda, m)$, given a maximum data set size m, outputs a public key PK and a secret key SK. The signing algorithm $\mathsf{Hom.Sign}(\mathsf{SK}, \tau, M, i)$ takes SK, a tag τ identifying a data set, a message M and an index $i \in \{1, 2, \ldots, m\}$, and outputs a signature σ. $\mathsf{Hom.Ver}(\mathsf{VK}, \tau, M, \sigma, f)$ checks whether σ is valid w.r.t. a tag τ, a message M and a function $f \in \mathcal{F}$. The evaluation algorithm $\mathsf{Hom.Eval}(\mathsf{VK}, \tau, f, \boldsymbol{\sigma})$, given a tag τ, a function f and a tuple of signatures $\{\sigma_i\}_{i=1}^m$ (that should be valid for $\{M_i\}_{i=1}^m$ respectively) outputs a new signature σ' that will verify correctly for $f(M_1, \ldots, M_m)$[2].

The security notion for linearly-homomorphic signatures is an extension of the classical notion of unforgeability against chosen-message attacks. The adversary $\mathcal{A}$ can ask signatures on triples of the form (τ, M, i) (precisely, the tag is chosen by the challenger), and at the end it should not be able to produce a valid signature σ^* on (τ^*, M^*, f^*) such that: either (1) τ^* is "new", or (2) $\tau^* = \tau$ for some tag τ asked during the game and $M^* \neq f^*(M_1, \ldots, M_m)$, where $M_1, \ldots, M_m$ are the messages in the data set identified by τ. Notice that by definition of $\mathsf{Hom.Eval}$, for any f^* everyone could compute a valid signature on $M = f^*(M_1, \ldots, M_m)$. Thus condition (2) makes sure that no one can do it for $M^* \neq f^*(M_1, \ldots, M_m)$.

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, for any input $x \in \mathcal{X}_\kappa$ it is efficient to compute $y \in \mathcal{Y}_\kappa$ where $y = F_\kappa(x)$. In addition, we assume that κ specifies a finite ring $\mathbb{K}$ that will be used as described below.

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

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 probabilistic polynomial time).

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

Ring-homomorphic: intuitively, this property states that it is possible to evaluate inner product operations in the exponent given some "blinded" bases. Before stating the property formally, we give a high level explanation of this idea by using an example. Assume that one is given values $W_1 = h^{\omega_1}, W_2 = h^{\omega_2} \in \mathcal{X}_\kappa$, $\omega_1, \omega_2 \in \mathbb{Z}$, and wants to compute $h^{(\omega_1\alpha_1 + \omega_2\alpha_2 \bmod q)}$ for some integer coefficients α_1, α_2. If $q \neq |\mathcal{X}_\kappa|$ and the order of $\mathcal{X}_\kappa$ is not known, then it is not clear how to compute such a value efficiently (notice that h is not given). The ring-homomorphic property basically says that with the additional knowledge of $F_\kappa(h)$, such computation can be done efficiently.

More formally, 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).

[2] We remark that, for technical reasons, the realization given in section 5, slightly deviates from the above syntax. In particular, it requires $\mathsf{Hom.Eval}$ to receive, as additional inputs, the vector messages $\boldsymbol{M}$ and the functions $\boldsymbol{f}$ under which the signatures $\boldsymbol{\sigma}$ are supposed to verify.

We say that (Gen, F) is *ring-homomorphic* (for the ring $\mathbb{K}$ 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 ring $\mathbb{K}$. We notice that over all the paper we often abuse notation by treating elements of the ring $\mathbb{K}$ as integers and vice versa. 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 ring $\mathbb{K}$ 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 ring-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 ring $\mathbb{K}$, 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.

Flexibly One-way: finally, we require a family (Gen, F) to be non-invertible in a strong sense. Formally, we say that (Gen, F) is *flexibly 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 ring $\mathbb{K}$) 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 ring $\mathbb{K}$ (d may though be different from zero over the integers), there is an efficient algorithm that computes $x' \in \mathcal{X}_\kappa$ such that $F_\kappa(x') = y^d$.

Notice that flexible one-wayness is stronger than standard one-wayness (in which d is always fixed to 1). Also, our notion is closely related to the notion of *q-one wayness* for group homomorphisms given in [15]. Informally, this latter notion states that for some prime q: (1) f is one-way in the standard sense, (2) there is a polynomial-time algorithm that on input (f, z, y, i) such that $f(z) = y^i$ (for $0 < i < q$) computes x such that $f(x) = y$, and (3) y^q is efficiently invertible.

It is not hard to see that when $q = |\mathbb{K}|$ flexible one-wayness and q-one-wayness are basically equivalent, except for that we do not require the existence of an efficient algorithm that on input (F, z, y, i) such that $F(z) = y^i$ computes x such that $F(x) = y$.

We stress that even though flexible one-wayness 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 ring-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 ring $\mathbb{K}$ is $\mathbb{Z}_p$.

$F_\kappa(x)$**:** 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 ring-homomorphic for $\mathbb{Z}_p$ as p is the order of $\mathbb{G}_1$.

Its security can be shown via the following Theorem. The proof is straightforward and is deferred to the full version.

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

RSA (over $\mathbb{QR}_N$). This construction is an algebraic trapdoor permutation, and it allows to explicitly choose the ring $\mathbb{K}$ as $\mathbb{Z}_e$ for any prime $e \geq 3$.

$\mathsf{Gen}(1^\lambda, e)$**:** 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: $x^c \bmod N$.

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

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

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

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

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

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

To prove the theorem, we simply observe that since $d \neq 0$ and $d \in \mathbb{Z}_e$, it holds $gcd(e, d) = 1$. Therefore, it is possible to apply the well known Shamir's trick [36] 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. This construction also allows to explicitly choose the ring $\mathbb{K}$, which can be $\mathbb{Z}_{2^t}$ 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 $\omega^{(j)} = \omega^{(j)'} + 2^t \cdot \omega^{(j)''}$. Finally, output

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

The algebraic and homomorphic properties are easy to check. To see that the function is ring-homomorphic for $\mathbb{Z}_{2^t}$, 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. For lack of space, its proof appears in the full version of this paper.

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

4 Our Verifiable Computation Schemes

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

An Overview of Our Solutions. Our starting point is the protocol of [6]: 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 [6] 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 [6] 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 [6] was basically a variant of the Naor-Reingold PRF [32] which can be easily istantiated over RSA moduli assuming the DDH assumption 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 together with algebraic PRFs based on the decision linear problem [16].

Our second observation was to note that the scheme in [6] 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 relies on these assumptions and that support polynomials over a large variety of finite rings: $\mathbb{Z}_e$ for any prime $e \geq 3$, $\mathbb{Z}_{2^t}$ for any integer $t \geq 1$. Previously known solutions [33,16] could support only polynomials over $\mathbb{Z}_p$ 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 allows us to compute polynomials over $\mathbb{Z}_e$ for any prime $e \geq 3$, where e is the prime used by the RSA function. Using the Rabin function allows us to handle polynomials over $\mathbb{Z}_{2^t}$ for any integer $t \geq 1$.

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 [6,16], we define the function $h : \mathbb{K}^m \to \mathbb{K}^l$ 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 $1 \leq j \leq l$, use a canonical ordering to write $j = (i_1, \ldots, i_m)$ with $0 \leq i_k \leq d$, and 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 with closed form efficiency for polynomials whose notion is recalled below.

Closed-Form Efficient PRFs. The notion of closed form efficient pseudorandom functions, firstly introduced by Benabbas *et al.* [6] and later refined by Fiore and Gennaro [16], is defined as follows.

The function 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 $(K, \mathsf{pp}) \overset{\$}{\leftarrow} \mathsf{PRF.KG}(1^\lambda)$, 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 $R_1, \ldots, R_l \in \mathcal{Y}$ and a vector of m arbitrary values $\boldsymbol{x} = (x_1, \ldots, x_m)$, and assume that the best algorithm to compute $\mathsf{Comp}(R_1, \ldots, R_l, x_1, \ldots, x_m)$ takes time T. Let $z = (z_1, \ldots, z_l)$ 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

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

and its running time is $o(T)$. For $z = (1, \ldots, l)$ 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 the PRF. 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).

Our scheme. Our verifiable computation scheme works generically for any family of functions $\mathcal{F}$ that is the set of m-variate polynomials of degree d over a finite ring $\mathbb{K}$ such that: (1) the algebraic one-way function $F_\kappa : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa$ is ring-homomorphic for $\mathbb{K}$, and (2) there exists a PRF whose range is $\mathcal{X}_\kappa$, and that has closed form efficiency relative to the computation of polynomials, i.e., for the algorithm $\mathsf{Poly}(\boldsymbol{R}, \boldsymbol{x}) = \sum_{j=1}^{l} R_j^{h_j(\boldsymbol{x})}$.

If we instantiate these primitives with the CDH-based algebraic OWF of Section 3.1 and the PRFs based on Decision Linear described in [16], then our generic construction captures the verifiable computation scheme of Fiore and Gennaro [16]. Otherwise we can obtain new schemes by using our algebraic OWFs based on RSA and Factoring described in Section 3.1. They have input and output space $\mathcal{X}_\kappa = \mathcal{Y}_\kappa = \mathbb{QR}_N$, the subgroup of quadratic residues in $\mathbb{Z}_N^*$. So, 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 [6] 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 $\mathbb{QR}_N$.

With these instantiations we obtain new verifiable computation schemes that support polynomials over a *large* variety of finite rings: $\mathbb{Z}_e$ for any prime $e \geq 3$, $\mathbb{Z}_{2^t}$ for any integer $t \geq 1$. Previously known solutions [33,16] could support only polynomials over $\mathbb{Z}_p$ where p must be a *large* prime whose size strictly depends on the security parameter 1^λ. In contrast, our factoring and RSA solutions allow for much more flexibility.

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 $F_\kappa : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa$ that is ring-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* $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{PubVer}}(\mathcal{VC}_{\mathsf{Poly}}, \mathcal{F}, q, \lambda)$.

Proof (Sketch). Here we provide a proof sketch of Theorem 4. We defer the interested reader to the the full version of this work for the formal proof.

Consider the following hybrid games:

Game 0: this is the real security game.

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 $\mathsf{VK}_x = \prod_{i=1}^{l} \mathsf{PRF.F}_K(i)^{h_i(\boldsymbol{x})}$.
By correctness of $\mathsf{PRF.CFEval}$, Game 1 is identically distributed as Game 0.

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.
By the security of the pseudorandom function, it is not hard to see that Game 2 is negligibly-close to Game 1.

To complete the proof of the theorem it remains to show that by the flexible one-wayness of the algebraic OWF, any PPT adversary has at most negligible advantage of winning in Game 2.

Assume by contradiction there exists a PPT adversary $\mathcal{A}$ that has non-negligible probability ϵ of winning in Game 2. We show that from such $\mathcal{A}$ it is possible to construct an efficient algorithm $\mathcal{B}$ that breaks the flexible one-wayness of the algebraic one-way function with the same probability ϵ.

$\mathcal{B}$ receives the pair (κ, A) as its input, where $A \in \mathcal{Y}_\kappa$, and 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 $\mathsf{PK}_f = A$. 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 each query $\boldsymbol{x}$ as follows: it computes $\mathsf{VK}_x = \prod_{i=1}^{l} Z_i^{h_i(\boldsymbol{x})}$ and returns VK_x. 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 $\boldsymbol{x}^*, \hat{\sigma}_y = (\hat{y}, \hat{V})$ be the output of $\mathcal{A}$ at the end of the game such that $\mathsf{Verify}(\mathsf{PK}_f, \mathsf{VK}_{x^*}, \hat{\sigma}_y) = \hat{y}$, $\hat{y} \neq \perp$ 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 we have that $F_\kappa(V) = A^y \cdot \mathsf{VK}_{x^*}$. 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 flexible one-wayness of $F_\kappa(A)$.

Extensions of our Protocols. The techniques showed above can be further extended in order to provide efficient solutions for the class of polynomials in m variables and maximum degree d in each monomial, and for matrix multiplications. We leave the description of these extensions for the full version of this work [11].

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, ring-homomorphicity and flexible 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 ring-homomorphic for a ring $\mathbb{K}$, then our signature scheme supports the message space $\mathbb{K}^n$ (for some integer $n \geq 1$) and all linear functions over this ring. 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 rings, i.e., $\mathbb{Z}_{2^t}$ or $\mathbb{Z}_e$, for any $t \geq 1$ and any prime e. As we will detail at the end of this section, previous solutions (e.g., [7,19,3,9,8,12,13,17]) 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

[9,8] 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 rings, 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 ring-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. Moreover we stress that $\mathsf{Hom.Eval}$ needs to receive both f and $\boldsymbol{f}$ as otherwise it would not be able to correctly perform the homomorphic operations. Notice, however, that the value of the produced message does not depend on $\boldsymbol{f}$ (this is needed essentially to run the Eval algorithm correctly).

Since our scheme follows the FDH paradigm, its security holds in the random oracle model, however, following similar results for FDH signatures, in the full version we propose 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 follows from the following theorem. For lack of space, its proof appears in the full version.

Theorem 5. *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.*

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 rings of arbitrary size without sacrificing efficiency[3]. 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., [7,19,3,9,8,12,13,17]) or are much less efficient in practice [9,8].

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 $\mathbb{K}$. 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.* [19] and by Catalano, Fiore and Warinschi [13]). 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)

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

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 [9,8]. Such schemes are also secure in the random oracle model, but rely on the hardness 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 [22,10]. Even though the algebraic tools underlying our scheme are significantly different with respect to those used in [9,8] 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 pre-image sampling for comparable security levels.

Acknowledgements. The second author did the present work while at NYU supported by NSF grant CNS-1017471. The research of the third author was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Ahlswede, R., Ning-Cai, Li, S., Yeung, R.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: From Secrecy to Soundness: Efficient Verification via Secure Computation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 152–163. Springer, Heidelberg (2010)
3. Attrapadung, N., Libert, B.: Homomorphic Network Coding Signatures in the Standard Model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011)
4. Babai, L.: Trading group theory for randomness. In: 17th ACM STOC, Providence, Rhode Island, USA, May 6-8, pp. 421–429. ACM Press (1985)
5. Belenkiy, M., Chase, M., Erway, C.C., Jannotti, J., Küpçü, A., Lysyanskaya, A.: Incentivizing outsourced computation. In: Workshop on Economics of Networked Systems – NetEcon, pp. 85–90 (2008)
6. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable Delegation of Computation over Large Datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)

7. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a Linear Subspace: Signature Schemes for Network Coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
8. Boneh, D., Freeman, D.M.: Homomorphic Signatures for Polynomial Functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
9. Boneh, D., Freeman, D.M.: Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011)
10. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
11. Catalano, D., Fiore, D., Gennaro, R., Vamvourellis, K.: Algebraic (trapdoor) one-way functions and their applications. Cryptology ePrint Archive, Report 2012/434 (2012); Full version
12. Catalano, D., Fiore, D., Warinschi, B.: Adaptive Pseudo-free Groups and Applications. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 207–223. Springer, Heidelberg (2011)
13. Catalano, D., Fiore, D., Warinschi, B.: Efficient Network Coding Signatures in the Standard Model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer, Heidelberg (2012)
14. Chung, K.-M., Kalai, Y., Vadhan, S.: Improved Delegation of Computation Using Fully Homomorphic Encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010)
15. Cramer, R., Damgård, I.: Zero-Knowledge Proofs for Finite Field Arithmetic or: Can Zero-Knowledge Be for Free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998)
16. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: 2012 ACM Conference on Computer and Communication Security. ACM Press (October 2012), Full version available at `http://eprint.iacr.org/2012/281`
17. Freeman, D.M.: Improved Security for Linearly Homomorphic Signatures: A Generic Framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012)
18. Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
19. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure Network Coding over the Integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)
20. Gennaro, R., Leigh, D., Sundaram, R., Yerazunis, W.S.: Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authorization and Low-Bandwidth Communication Devices. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 276–292. Springer, Heidelberg (2004)
21. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC, Bethesda, Maryland, USA, May 31-June 2, pp. 169–178. ACM Press (2009)
22. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, Victoria, British Columbia, Canada, May 17-20, pp. 197–206. ACM Press (2008)

23. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, Victoria, British Columbia, Canada, May 17-20, pp. 113–122. ACM Press (2008)
24. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
25. Guillou, L.C., Quisquater, J.-J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988)
26. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic Signature Schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
27. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: 24th ACM STOC, Victoria, British Columbia, Canada, May 4-6, pp. 723–732. ACM Press (1992)
28. Lewko, A., Waters, B.: New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012)
29. Micali, S.: Cs proofs. In: 35th FOCS, Santa Fe, New Mexico, November 20-22, New (1994)
30. Mohassel, P.: Efficient and secure delegation of linear algebra. Cryptology ePrint Archive, Report 2011/605 (2011)
31. Monrose, F., Wyckoff, P., Rubin, A.D.: Distributed execution with remote audit. In: NDSS 1999, San Diego, California, USA, February 3-5. The Internet Society (1999)
32. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS, Miami Beach, Florida, October 19-22, pp. 458–467. IEEE Computer Society Press (1997)
33. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. Cryptology ePrint Archive, Report 2011/587 (2011)
34. Parno, B., Raykova, M., Vaikuntanathan, V.: How to Delegate and Verify in Public: Verifiable Computation from Attribute-Based Encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)
35. Robert-Li, S.-Y., Yeung, R.Y., Cai, N.: Linear network coding. IEEE Transactions on Information Theory 49(2), 371–381 (2003)
36. Shamir, A.: On the generation of cryptographically strong pseudorandom sequences. ACM Trans. Comput. Syst. 1(1), 38–44 (1983)
37. Smith, S.W., Weingart, S.: Building a high-performance, programmable secure coprocessor. Computer Networks 31, 831–860 (1999)
38. Yee, B.: Using Secure Coprocessors. PhD thesis, Carnegie Mellon University (1994)