Verifiable Delegation of Computation over Large Datasets*

Siavosh Benabbas[1], Rosario Gennaro[2], and Yevgeniy Vahlis[3]

[1] University of Toronto, siavosh@cs.toronto.edu
[2] IBM Research, rosario@us.ibm.com
[3] Columbia University, evahlis@cs.columbia.edu

Abstract. We study the problem of computing on large datasets that are stored on an untrusted server. We follow the approach of *amortized verifiable computation* introduced by Gennaro, Gentry, and Parno in CRYPTO 2010. We present the first practical verifiable computation scheme for high degree polynomial functions. Such functions can be used, for example, to make predictions based on polynomials fitted to a large number of sample points in an experiment. In addition to the many non-cryptographic applications of delegating high degree polynomials, we use our verifiable computation scheme to obtain new solutions for verifiable keyword search, and proofs of retrievability. Our constructions are based on the DDH assumption and its variants, and achieve adaptive security, which was left as an open problem by Gennaro *et al* (albeit for general functionalities).

Our second result is a primitive which we call a *verifiable database* (VDB). Here, a weak client outsources a large table to an untrusted server, and makes retrieval and update queries. For each query, the server provides a response and a proof that the response was computed correctly. The goal is to minimize the resources required by the client. This is made particularly challenging if the number of update queries is unbounded. We present a VDB scheme based on the hardness of the subgroup membership problem in composite order bilinear groups.

1 Introduction

This paper presents very efficient protocols that allow a computationally weak client to securely outsource some computations over very large datasets to a powerful server. Security in this context means that the client will receive an assurance that the computation performed by the server is correct, with the optional property that the client will be able to hide some of his data from the server.

The problem of securely outsourcing computation has received widespread attention due to the rise of *cloud computing:* a paradigm where businesses lease computing resources from a service (the *cloud provider*) rather than maintain their own computing infrastructure [2, 57]. A crucial component of secure cloud

* A full version of this paper is available at http://eprint.iacr.org/2011/132

computing is a mechanism that enforces the integrity and correctness of the computations done by the provider.

Outsourced computations are also increasingly relevant due to the proliferation of mobile devices, such as smart phones and netbooks, computationally weak devices which might off-load heavy computations, e.g., a cryptographic operation or a photo manipulation, to a network server. Here too, a proof of the correctness of the result might be desirable if not necessary.

A crucial requirement in all of these cases is that the computation invested by the (weak) client in order to verify the result of the server's work must be substantially smaller than the amount of computation required to perform the work to begin with. Indeed if that was not the case, the client could perform the computation on its own without interacting with the server! It is also desirable to keep the server's overhead as small as possible: in other words the computation of the server to provide both the result and a correctness proof to the client should be as close as possible to the amount of work needed to simply compute the original function (otherwise, the server, which might provide this service to many clients, may become overwhelmed by the computational load).

This paper initiates a line of research about efficient protocols for verifiable computation of specific functions, in our case the evaluation of polynomials derived from very large datasets. Most of the prior work (reviewed below) has focused on generic solutions for arbitrary functions. So while in "general" the problem we are considering has been solved, by focusing on specific computations we are able to obtain much more efficient protocols. This is similar to the way research over secure multiparty computation has evolved: following generic protocols for the evaluation of arbitrary functions [60, 29, 9, 17], there has been a twenty-plus year effort to come up with efficient distributed protocols for specific computations encountered in practical applications (e.g. the entire work on threshold cryptography [21], or protocols on set intersection and pattern matching such as [34]).

Our Results. This paper focuses on the evaluation of polynomials derived from very large datasets. While the computations themselves are simple, it's the magnitude of data that prevents the client (who cannot even store the entire data) to perform them by itself. In our protocols the client will initially store the data at the server (with the option of encrypting it for confidentiality, if desired), with some authenticating information. The client will only keep a short secret key. Later, every time the client requests the value of a computation over the data, the server will compute the result and return it together with an *authentication code*, which the client will be able to quickly verify with the secret key. This description shows that our problem naturally fits into the *amortized* model for outsourced computation introduced in [27]: the client performs a one-time computationally expensive phase (in our case storing the data with its authentication information) and then quickly verifies the results provided by the server.

Our protocols are very efficient. The computation of the authentication data is comparable to encrypting the file using the ElGamal encryption scheme (i.e.

roughly 2 exponentiations per data block). Verification takes at most a logarithmic (in the number of blocks) number of exponentiations under the DDH Assumption. Additionally, we present a faster protocol (which requires only a *single* exponentiation to verify the result) which is secure under a decisional variant of the Strong Diffie Hellman Assumption in single groups[4].

An immediate application of our results is the ability to verifiably outsource computations to make predictions based on polynomials fitted to a large number of sample points in an experiment.

In the second part of our paper, we present an extension to our protocols, which allows the client to efficiently *update* the data (and its associated authentication information) stored at the server. We also present applications of our protocols to the problems of verifiable *keyword search* (the client stores a large database with the server and it queries if a specific keyword appears in it) and secure *proofs of retrievability* (the client checks that the file stored with the server is indeed retrievable) [50, 36].

VERIFIABLE DELEGATION OF POLYNOMIALS. The basis of all our protocols is verifiable delegation of polynomials. Assume the client has a polynomial $P(\cdot)$ of large degree d, and it wants to compute the value $P(x)$ for an arbitrary inputs x. In our basic solution the client stores the polynomial in the clear with the server as a vector $\boldsymbol{c}$ of coefficients in $\mathbb{Z}_p$. The client also stores with the server a vector $\boldsymbol{t}$ of group elements of the form $g^{ac_i+r_i}$ where $a \in_{\mathrm{R}} \mathbb{Z}_p$ and r_i is the i^{th}-coefficient of a polynomial $R(\cdot)$ of the same degree as $P(\cdot)$. When queried on input x the server returns $y = P(x)$ and $t = g^{aP(x)+R(x)}$ and the client accepts y iff $t = g^{ay+R(x)}$.

If $R(\cdot)$ was a random polynomial, then we can prove that this is a secure delegation scheme in the sense of [27]. However checking that $t = g^{ay+R(x)}$ would require the client to perform computation polynomial in the degree of $P(\cdot)$ – the exact work that we set out to avoid. The crucial point, therefore, is how to perform this verification fast, in time which is independent, or at the very least sublinear in the degree of $P(\cdot)$. We do that by defining $r_i = F_K(i)$ where F is a pseudo-random function (PRF in the following) with a special property which we call *closed form efficiency.* The property is that given the polynomial $R(\cdot)$ defined by the r_i coefficients, the value $R(x)$ (for any input x) can be computed very efficiently (sub-linearly in d) by a party who knows the secret key K for the PRF. Since F is a PRF, the security of the scheme is not compromised (as F is indistinguishable from a random function), and the closed form efficiency of F will allow the client to verify the result in time sub-linear in the degree of the polynomial.

We generalize our result for PRFs with other types of closed form efficiency, which yield efficient and secure delegation protocols not only for single-variable polynomials of degree d, but also for multivariate polynomials with total degree d or of degree d in each variable. We have several different variations of PRFs: the least efficient one is secure under the Decisional Diffie-Hellman assumption, while more efficient ones require a decisional variant of the Strong DH assumption.

[4] See e.g., [19] for a survey of the strong DH family of assumptions

Adaptivity: One of the main questions to remain open after the work of GGP [27] is whether we can achieve verifiable delegation even if the malicious server knows whether the verifier accepted or rejected the correctness proof of the value computed by the server. Indeed, the GPV scheme becomes insecure if the server learns this single bit of information after each proof is sent to the verifier. Our constructions are the first to achieve adaptive security in the amortized setting.

Privacy: Our solution allows the client to preserve the secrecy of the polynomial stored with the server, by encrypting it with an additively homomorphic encryption scheme. In this case the server returns an encrypted form of y which the client will decrypt.

Keyword Search: The applications to keyword search without updates is almost immediate. Consider a text file $F = \{w_1, \ldots, w_\ell\}$ where w_i are the words contained in it. Encode F as the polynomial $P(\cdot)$ of degree ℓ such that $P(w_i) = 0$. To make this basic solution efficiently updatable we use a variation of the polynomial delegation scheme which uses bilinear maps. We also present a generic, but less efficient way to make *any* static keyword search protocol updatable which might be of independent interest.

Proof of Retrievability: Again the application of our technique is quite simple. The client encodes the file as a polynomial $F(x)$ of degree d (each block representing a coefficient), and delegates the computation of $F(x)$ to the server. The proof of retrievability consists of the client and the server engaging in our verifiable delegation protocol over a random point r: the client sends r and the server returns the value $F(r)$ together with a proof of its correctness. The client accepts if it accepts the proof that $F(r)$ is the correct value.

VERIFIABLE DATABASES WITH EFFICIENT UPDATES. In the second part of our paper we study the problem of verifiable databases, where a resource constrained client wishes to store an array DB on a server, and to be able to retrieve the value at any cell $DB[i]$, and to update the database by assigning $DB[i] \leftarrow v$ for a new value v. The goal is to achieve this functionality with an additional guarantee that if a server attempts to tamper with the data, the tampering will be detected when the client queries the database.

Simple solutions (based on Message Authentication Codes or Signature Schemes) exist for the restricted case where the database is static – i.e. the client only needs to retrieve data, but does not modify the database. One example is to have the client sign each pair (index,value) that is sent to the server. Clearly, if no updates are performed, the server has no choice but to return the correct value for a given cell. However, the problem becomes significantly harder when efficient updates are needed. One solution is for the client to just keep track of all the changes locally, and apply them as needed, but this contradicts our goal of keeping client state and workload as small as possible. On a high level, the main technical difficulty stems from the fact that the client must revoke any authenticating data that the server has for the previous value of the updated cell. This issue has been addressed in the line of works on cryptographic accumulators [16, 47, 53], and, using different techniques, in the authenticated datastructures literature [48, 41, 52, 59].

We present a verifiable database delegation scheme based on the hardness of the subgroup membership problem in composite order bilinear groups (this assumption was originally introduced in [13]). Our solution allows the client to query any location of the database, and verify the response in time that is independent of the size of the database. The main advantage of our construction is that it allows the client to insert and delete values, as well as update the value at any cell by sending a single group element to the server after retrieving the current value stored in the cell. Prior solutions either rely on non-constant size assumptions (such as variants of the Strong Diffie-Hellman assumption [23, 15]), require expensive generation of primes for each operation (in the worst case), or require expensive "re-shuffling" procedures to be performed once in a while on the data. On the other hand, our construction works in the private key setting, whereas some prior solutions allow public verification (e.g., [16, 47]).

ROADMAP. The rest of the paper is organized as follows. In Section 2 we define the security assumptions used in the paper. Readers interested in the precise definition of Verifiable Computation and its security can find them in Section 3. In Section 4 we introduce our notation of Algebraic Pseudorandom Functions which are the main building block of our constructions. In Section 5 we show how to securely delegate polynomial evaluations to an untrusted server using Algebraic PRFs. In the full version of the paper [7] we show how to use delegation of polynomials to implement verifiable databases, and to obtain new constructions of Proofs of Retrievability.

1.1 Related Work

As mentioned above our work follows the paradigm introduced in [27] which is also adopted in [20, 3]. 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. [28]) resulting in protocols of limited practical relevance. Our protocols on the other hand work for only a very limited class of computations (mostly polynomial evaluations) but are very efficient and easily implementable in practice.

The previous schemes based on fully homomorphic encryption also suffer from the following drawback: if a malicious server tries to cheat and learns if the client has accepted or rejected its answer, then the client must repeat the expensive pre-processing stage. The only alternative way to deal with this problem proposed in these papers is to protect this bit of information from the server (which is a very strong assumption to make). Somewhat surprisingly our scheme remains secure even if a cheating server learns the acceptance/rejection bit of the client, without any need to repeat the pre-processing stage. This is not only conceptually interesting, but also a very practical advantage.

There is a large body of literature, prior to [27], that investigates the problem of verifiably outsourcing the computation of an arbitrary functions (we refer to [27] for an exhaustive list of citations). This problem has attracted the attention of the Theory community, starting from the work on Interactive Proofs [5, 31],

efficient arguments based on probabilistically checkable proofs (PCP) [37, 38], CS Proofs [43] and the *muggles proofs* in [30]. However in PCP-based schemes, the client must store the large data in order to verify the result and therefore these solutions might not be applicable to our setting.

This problem has also been studied by the Applied Security community, with solutions which are practical but whose security holds under some very strong assumptions on the behavior of the adversary. For example, solutions based on audit (e.g. [46, 6]) which typically assume many clients, and require a fraction of them to recompute some of the results provided by the server, but are secure only under the assumption that bad clients do not collude. Another approach is to use secure co-processors (e.g. [56, 61]) which "sign" the computation as correct, under the assumption that the adversary cannot tamper with the processor. Finally, other trust models have been considered. The area of authenticated data structures [58, 41, 54] aims to provide delegation solutions when the owner of the data is decoupled from the client. In this scenario, the owner maintains a large state, and acts as a trusted third party, but delegates his data to an untrusted server that can be queried by weak clients.

For the specific case of outsourcing expensive cryptographic operations, Chaum and Pedersen in [18], describe protocols to allow a client to verify the behavior of a piece of hardware placed on the client's device by a service provider such as a bank. Hohenberger and Lysyanskaya formalize this model [35], and present protocols for the computation of modular exponentiations (arguably the most expensive step in public-key cryptography operations). Their protocol requires the client to interact with *two* non-colluding servers. Other work targets specific function classes, such as one-way function inversion [32].

The application of secure keyword search over a stored file can be handled using *zero-knowledge sets*, [44] which however does not allow for an easy way to update the file. Our protocol for keyword search combines ideas from our polynomial delegation scheme with some machinery inspired by zero-knowledge sets, to obtain a protocol that allows for efficient updates and other additional desirable properties (see full version [7]).

The problem of proof of retrievability was first posed in [50, 36], and subsequent protocols include [4, 55, 22]. A proof of retrievability protocol usually goes like this: after storing a (potentially large) file with the server, the client issues a *query* to receive an assurance that the file is still correctly stored. The server computes an answer based on the query and the file, and finally the client performs some verification procedure on the answer. All of these protocols incur a substantial storage overhead for the server (since the file is stored using an erasure code) and, except for [22], require communication which is quadratic in the security parameter. The protocol in [22] has linear communication complexity but it requires *both* the server and the client to work in time proportional to the size of the file. Our solution achieves linear communication complexity in the security parameter and is very efficient for the client (as its work is sublinear in the size of the file).

Our verifiable database construction is closely related to Memory Checkers (see e.g. [10, 26, 1, 24, 50]). However, our setting differs from the memory checking setting in that we allow the server to be an arbitrary algorithm, whereas a memory checker interacts with a RAM (an oracle that accepts store/retrieve queries). In this context, our construction would yield a memory checker with poor performance since it would require the checker to issue a number of queries that is linear in the size of the memory. In contrast, we focus on optimizing the communication and the work of the client when the server can perform arbitrary computation on its data. Our construction requires the server to perform a linear amount of work to answer one type of queries (update/retrieve), while the other type of queries requires only a constant amount of work. Finally, we note that the work on accumulators [16, 47, 53] and authenticated data structures [48, 41, 52, 59] can be used to construct verifiable databases with similar efficiency under different assumptions.

2 Assumptions

In this work we rely on the following assumptions about computational groups.

Decisional Diffie Hellman. The standard Decisional Diffie-Hellman Assumption (DDH) is defined as follows. For every PPT distinguisher A there exists a negligible function $neg(\cdot)$ such that for all n,

$$|\Pr[A(1^n, g, g^x, g^y, g^{xy}) = 1] - \Pr[A(1^n, g, g^x, g^y, g^z) = 1]| \leq neg(n)$$

where g is a generator of a group $\mathbb{G}$ of order p where p is a prime of length approximately n, and $x, y, z \in_{\mathrm{R}} \mathbb{Z}_p$.

Strong Diffie Hellman. The strong Diffie-Hellman family of assumptions allows an adversary to obtain group elements $g, g^x, g^{x^2}, \ldots, g^{x^d}$, and requires the adversary to compute or distinguish a related group element from a random one. Computational variants of the problem appeared as early as the work of Mitsunari *et al* [45]. More recently, bilinear versions of the assumptions, starting with the works of Boneh and Boyen [11, 12], were used in several applications (e.g. [23, 15]). Boneh and Boyen gave a proof of the bilinear assumptions in the generic group model. In one of our constructions, we achieve high efficiency by relying on a decisional version of the strong DH assumption in single groups.

The d-SDDH assumption is stated as follows. For every PPT distinguisher A there exists a negligible function $neg(\cdot)$ such that for all n,

$$|\Pr[A(1^n, g, g^x, g^{x^2}, \ldots, g^{x^d}) = 1] - \Pr[A(1^n, g, g^{x_1}, g^{x_2}, \ldots, g^{x_d}) = 1]| \leq neg(n)$$

where g is a generator of a group $\mathbb{G}$ of order p where p is a prime of length approximately n, and $x, x_1, \ldots, x_d \in_{\mathrm{R}} \mathbb{Z}_p$.

Subgroup membership assumption in composite order bilinear groups. The subgroup membership assumption in composite order bilinear groups first

appeared in [13], and has seen many recent applications in the areas of Identity Based Encryption (IBE), Hierarchical IBE, and others [25, 13, 8]. The assumption we rely on (for our verifiable database delegation scheme) is the following.

For every PPT distinguisher A there exists a negligible function $neg(\cdot)$ such that for all n,

$$|\Pr[A(1^n, g_1g_2, u_2, (g_1g_2)^x) = 1] - \Pr[A(1^n, g_1g_2, u_2, u_2^x) = 1]| \leq neg(n)$$

where $\mathbb{G}$ is a group of order $N = p_1p_2$ where p_1 and p_2 are primes of length approximately n, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of $\mathbb{G}$ of orders p_1 and p_2 respectively, $g_1 \in_R \mathbb{G}_1$, $g_2, u_2 \in_R G_2$, and $x \in_R \mathbb{Z}_N$.

In addition, we require the existence of an efficiently computable pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ where $\mathbb{G}_T$ is a group of order N. We shall make use of the following property of pairings over composite order groups: for $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, $e(g_1, g_2) = 1_{\mathbb{G}_T}$. This property holds for every bilinear pairing over composite order groups (as shown e.g. in [39]).

Bilinear sub-group projection assumption. In the analysis of our verifiable database scheme we first show the security of the scheme based on the following new assumption. We then apply Lemma 1 (given below) to obtain a reduction to the subgroup membership problem. The Bilinear Sub-Group Projection Assumption (BSGP) is stated as follows: for every PPT adversary A, there exists a negligible function $neg(\cdot)$ such that for all n,

$$\Pr[A(1^n, (g_1g_2), (h_1h_2), u_2) = e(g_1, h_1)] \leq neg(n)$$

where $\mathbb{G}$ is a group of order $N = p_1p_2$ where p_1 and p_2 are primes of length approximately n, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of $\mathbb{G}$ of orders p_1 and p_2 respectively, $g_1, h_1 \in_R \mathbb{G}_1$, and $g_2, h_2, u_2 \in_R G_2$. The following lemma shows that the BSGP assumption is implied by the standard sub-group membership assumption in composite order bilinear groups.

Lemma 1. *The subgroup membership assumption in composite order bilinear groups reduces to the BSGP assumption.*

The proof of the lemma, as well as its application to delegation of data structures, is given in the full version of this paper [7].

3 Verifiable Computation

A verifiable computation scheme is a two-party protocol between a *client* and a *server*. The client chooses a function and an input which he provides to the server. The latter is expected to evaluate the function on the input and respond with the output together with a *proof* that the result is correct. The client then verifies that the output provided by the worker is indeed the output of the function computed on the input provided.

The goal of a verifiable computation scheme is to make such verification very efficient, and particularly much faster than the computation of the function itself.

We adopt the *amortized* model of Gennaro *et al.* [27]: for each function F, the client is allowed to invest a one-time expensive computational effort (comparable to the effort to compute F itself) to produce a public/secret key pair, which he will use to efficiently (e.g. in linear-time) verify the computation of F by the server on many inputs.

We prove our results in a stronger version of the [27] definition of verifiable computation scheme. As we discussed in the Introduction, the main difference is that in our protocols the server is allowed to learn if the client accepts or rejects the output of a particular computation (in [27] and following works in the amortized model, leaking this bit of information to the server would help him cheat in the following executions).

We refer the reader to the full version of this paper [7] for the precise definition of a verifiable computation scheme.

4 Algebraic Pseudorandom Functions

Our main technical tool is a new way of viewing pseudo-random functions (PRF) with algebraic properties to achieve efficient verification of server certificates in the delegation setting. Intuitively, we rely on the fact that certain pseudo-random functions (such as the Naor-Reingold PRF [49]) have outputs that are members of an abelian group, and that certain algebraic operations on these outputs can be computed significantly more efficiently if one possesses the key of the pseudo-random function that was used to generate them. In this section we present an abstraction of the said property, and several constructions achieving different trade-offs between the types of functions that can be efficiently evaluated given the key, and the assumption that is needed to guarantee pseudo-randomness.

An algebraic pseudorandom function (PRF) consists of algorithms $\mathcal{PRF} = \langle \mathsf{KeyGen}, F, \mathsf{CFEval} \rangle$ where KeyGen takes as input a security parameter 1^n and a parameter $m \in \mathbb{N}$ that determines the domain size of the PRF, and outputs a pair $(K, param) \in \mathcal{K}_n$, where $\mathcal{K}_n$ is the key space for security parameter n. K is the secret key of the PRF, and $param$ encodes the public parameters. F takes as input a key K, public parameters $param$, an input $x \in \{0,1\}^m$, and outputs a value $y \in Y$, where Y is some set determined by $param$.

We require the following properties:

- *(Algebraic)* We say that $\mathcal{PRF}$ is algebraic if the range Y of $F_K(\cdot)$ for every $n \in \mathbb{N}$ and $(K, param) \in \mathcal{K}_n$ forms an abelian group. We require that the group operation on Y be efficiently computable given $param$. We are going to use the multiplicative notation for the group operation.
- *(Pseudorandom)* $\mathcal{PRF}$ is pseudorandom if for every PPT adversary A, and every polynomial $m(\cdot)$, there exists a negligible function $neg : \mathbb{N} \to \mathbb{N}$, such that for all $n \in \mathbb{N}$:

$$|\Pr[A^{F_K(\cdot)}(1^n, param) = 1] - \Pr[A^{R(\cdot)}(1^n, param) = 1]| \leq neg(n)$$

  where $(K, param) \leftarrow_{\mathrm{R}} \mathsf{KeyGen}(1^n, m(n))$, and $R : \{0,1\}^m \to Y$ is a random function.

– *(Closed form efficiency)* Let N be the order of the range sets of F for security parameter n. Let $z = (z_1, \ldots, z_l) \in (\{0,1\}^m)^l$, $k \in \mathbb{N}$, and efficiently computable $h : \mathbb{Z}_N^k \to \mathbb{Z}_N^l$ with $h(x) =< h_1(x), \ldots, h_l(x) >$. We say that (h, z) is *closed form efficient* for $\mathcal{PRF}$ if there exists an algorithm $\mathsf{CFEval}_{h,z}$ such that for every $x \in \mathbb{Z}_N^k$,

$$\mathsf{CFEval}_{h,z}(x, K) = \prod_{i=1}^{l} [F_K(z_i)]^{h_i(x)}$$

and the running time of CFEval is polynomial in n, m, k but sublinear in l. When $z = (0, \ldots, l)$ we will omit it from the subscript, and write $\mathsf{CFEval}_h(x, K)$ instead.

The last condition (which distinguishes our notion from traditional PRFs) allows to compute a "weighted product" of l PRF values much more efficiently than by computing the l values separately and then combining them. Indeed, given *param*, h, x, and $F_K(z)$, one can always compute the value $\prod_{i=1}^{l} [F_K(z_i)]^{h_i(x)}$ in time linear in l (this follows from the algebraic property of the PRF). The purpose of the closed form efficiency requirement is therefore to capture the existence of a more efficient way to compute the same value given the secret key K.

Note that closed form efficiency can be defined for PRFs over arbitrary input spaces. In particular, it is a non-trivial condition to attain even when the input space is polynomial in the security parameter[5]. In the constructions needed for our delegation scheme, this will be the case.

4.1 Small Domain Algebraic PRFs From Strong DDH

Construction 1 *Let $\mathcal{G}$ be a computational group scheme. The following construction $\mathcal{PRF}_1$ is an algebraic PRF with polynomial sized domains.*

$\mathsf{KeyGen}(1^n, m)$: *Generate a group description $(p, g, \mathbb{G}) \leftarrow_R \mathcal{G}(1^n)$. Choose $k_0, k_1 \in_R \mathbb{Z}_p$. Output param $= (m, p, g, \mathbb{G})$, $K = (k, k')$.*

$F_K(x)$: *Interpret x as an integer in $\{0, \ldots, D = 2^m\}$ where D is polynomial in n. Compute and output $g^{k_0 k_1^x}$.*

Closed form efficiency for polynomials. We now show an efficient closed form for $\mathcal{PRF}_1$ for polynomials of the form

$$p(x) = F_K(0) + F_K(1)x + \cdots + F_K(d)x^d$$

[5] When the input space is polynomial in the security parameter traditional PRFs exist unconditionally: if the input space has ℓ elements $\{x_1, \ldots, x_\ell\}$, define the key as ℓ random values $y_1, \ldots, y_\ell$ and $F_K(x_i) = y_i$. Notice however that this function does not have closed-form efficiency

where $d \leq D$. Let $h : \mathbb{Z}_p \to \mathbb{Z}_p^{d+1}$, be defined as $h(x) \stackrel{\mathsf{def}}{=} (1, x, \ldots, x^d)$. Then, we can define

$$\mathsf{CFEval}_h(x, K) \stackrel{\mathsf{def}}{=} g^{\frac{k_0(1-k_1^{d+1}x^{d+1})}{1-k_1 x}}$$

Let us now write the $\prod_{i=0}^{d}[F_K(z_i)]^{h_i(x)}$ where $(z_0, \ldots, z_d) = (0, \ldots, d)$:

$$\prod_{i=0}^{d}[F_K(z_i)]^{h_i(x)} = \prod_{i=0}^{d}[g^{k_0 k_1^i}]^{x^i} = g^{k_0 \sum_{i=0}^{d} k_1^i x^i}$$

Applying the identity $\sum_{i=0}^{d} k_0 k_1^i x^i = \frac{k_0(1-(k_1 x)^{d+1})}{1-k_1 x}$ we obtain the correctness of $\mathsf{CFEval}_h(x)$.

Theorem 1. *Suppose that the D-Strong DDH assumption holds. Then, $\mathcal{PRF}_1$ is a pseudorandom function.*

Proof. The input to the reduction is a description $(p, g, \mathbb{G})$ of a group, and a challenge $t_1, \ldots, t_d$ where t_i is either a random member of $\mathbb{G}$, or $g^{k_1^i}$, and $k_1 \in \mathbb{Z}_p$ is randomly chosen once for the entire challenge. The reduction then chooses $k_0 \in_{\mathrm{R}} \mathbb{Z}_p$, and computes the function $H(i) = t_i^{k_0}$ for $0 \leq i \leq d$. Clearly, H is a random function if the t_i are random, and is equal to $F_K(\cdot)$ for $K = (k_0, k_1)$ if the t_i are determined by k_1.

Construction 2 *Let $\mathcal{G}$ be a computational group scheme. We define $\mathcal{PRF}_{2,d}$, for $d \in \mathbb{N}$, as follows:*

$\mathsf{KeyGen}(1^n, m)$: *Generate a group description $(p, g, \mathbb{G}) \leftarrow_R \mathcal{G}(1^n)$. Choose $k_0, k_1, \ldots, k_m \in_R \mathbb{Z}_p$. Output param $= (m, p, g, \mathbb{G})$, $K = (k_0, k_1, \ldots, k_m)$.*

$F_K(x)$: *Interpret x as a vector $(x_1, \ldots, x_m) \in \{0, \ldots, d\}^m$. Compute and output $g^{k_0 k_1^{x_1} \cdots k_m^{x_m}}$.*

Closed form for m-variate polynomials of total degree at most d. We describe an efficient closed form for $\mathcal{PRF}_{2,d}$ for computing polynomials of the form

$$p(x_1, \ldots, x_m) = \sum_{\substack{i_1, \ldots, i_m \\ i_1 + \cdots + i_m \leq d}} F_K(i_1, \ldots, i_m) x_1^{i_1} x_2^{i_2} \cdots x_m^{i_m}.$$

Let $h : \mathbb{Z}_p^m \to \mathbb{Z}_p^l$, where $l = \binom{m+d}{d}$, be defined as

$$h(x_1, \ldots, x_m) \stackrel{\mathsf{def}}{=} \left(\binom{d}{i_1 \ldots, i_m} \prod_{j=1}^{m} x_j^{i_j} \right)_{i_1 + \cdots + i_m \leq d}$$

Let $z = [z_1, \ldots, z_l] = [(i_1, \ldots, i_m)]_{i_1 + \cdots + i_m \leq d} \in \mathbb{Z}_d^{m \times l}$. We can now define

$$\mathsf{CFEval}_{h,z}(x_1, \ldots, x_m, K) \stackrel{\mathsf{def}}{=} g^{k_0(1 + k_1 x_1 + \cdots + k_m x_m)^d}$$

Correctness follows by algebraic manipulation and is given in the full version of this paper.

Theorem 2. *Let $d \in \mathbb{N}$, and suppose that the d-Strong DDH assumption holds. Then, $\mathcal{PRF}_{2,d}$ is a pseudorandom function.*

We refer the reader to the full version of this paper [7] for the proof of Theorem 2.

Remark 1. It is interesting to note that the Naor-Reingold PRF is a special case of Construction 2 obtained by setting $d = 1$. Therefore, our construction provides a tradeoff between the security assumption and the size of the key of the PRF: to operate on binary inputs of length n our construction requires $n/\log_2(d+1)$ elements of $\mathbb{Z}_p$ in the key.

Remark 2. One can change the above construction so that it becomes slightly less efficient but secure under the *standard DDH* assumption. We call this modified version $\mathcal{PRF}_{4,d}$ and refer the reader to the full version for its exact definition. The reader can also get a glimpse of this PRF's parameters in Table 1.

4.2 Small Domain Algebraic PRFs from DDH

Construction 3 *Let $\mathcal{G}$ be a computational group scheme. We define $\mathcal{PRF}_3$ as follows:*

$\mathsf{KeyGen}(1^n, m)$: *Generate a group description $(p, g, \mathbb{G}) \leftarrow_R \mathcal{G}(1^n)$. Choose k_0, $k_{1,1}, \ldots, k_{1,s}, \ldots, k_{m,1}, \ldots, k_{m,s} \in_R \mathbb{Z}_p$. Output $param = ((m, s), p, g, \mathbb{G})$, $K = (k_0, k_{1,1}, \ldots, k_{1,s}, \ldots, k_{m,1}, \ldots, k_{m,s})$.*

$F_K(x)$: *Interpret $x = (x_1, \ldots, x_m)$ with each $x_i = [x_{i,1}, \ldots, x_{i,s}]$ as an s-bit string. Compute and output $g^{k_0 k_{1,1}^{x_{1,1}} \cdots k_{1,s}^{x_{1,s}} \cdots k_{m,1}^{x_{m,1}} k_{m,s}^{x_{m,s}}}$.*

Closed form for polynomials of degree d in each variable. We describe an efficient closed form for $\mathcal{PRF}_3$ for computing polynomials of the form

$$p(x_1, \ldots, x_m) = \sum_{i_1, \ldots, i_m \leq d} F_K(i_1, \ldots, i_m) x_1^{i_1} \cdots x_m^{i_m}$$

where the PRF F is initialized with m and $s = \lceil \log d \rceil$. Let $h : \mathbb{Z}_p^m \to \mathbb{Z}_p^l$, where $l = md$, be defined as $h(x_1, \ldots, x_m) = (x_1^{i_1} \cdots x_m^{i_m})_{i_1, \ldots, i_m \leq d}$. Let $z = [z_1, \ldots, z_l] = [(i_1, \ldots, i_m)]_{i_1, \ldots, i_m \leq d}$ then

$$\mathsf{CFEval}_{h,z}(x_1, \ldots, x_m, K) \stackrel{\mathsf{def}}{=} g^{k_0 \prod_{j=1}^{m} (1 + k_{j,1} x_j)(1 + k_{j,2} x_j^2) \cdots (1 + k_{j,s} x_j^{2^s})}$$

Correctness follows directly by expanding the expression in the exponent.

Remark 3. Note that for $m = 1$ we obtain an alternative construction for single-variable polynomials of degree d. Below we prove that Construction 3 is a PRF under the DDH Assumption. Therefore compared to Construction 1, this construction relies on a weaker assumption (DDH vs. D-strong DDH). However the efficiency of the closed form computation in Construction 1 is better: constant

vs. $O(\log d)$ in Construction 3. Jumping ahead this will give us two alternative ways to delegate the computation of a single-variable polynomial of degree d with the following tradeoff: either one assumes a weaker assumption (DDH) but verification of the result will take $O(\log d)$ time, or one assumes a stronger assumption to obtain constant verification time.

Closed form for 1-out-of-2 multivariate polynomials of degree 1. We now consider polynomials of the form

$$p(x_1, y_1, \ldots, x_m, y_m) = \sum_{s \in \{0,1\}^m} F_K(s) x_1^{s_1} y_1^{1-s_1} \cdots x_m^{s_m} y_m^{1-s_m}$$

In such polynomials, each monomial contains exactly one of x_i and y_i for $1 \leq i \leq m$. We initialize the PRF F with m and $s = 1$ (and for simplicity we drop the double subscript and denote the key $k_{i,1}$ as k_i). Specifically, let $h : \mathbb{Z}_p^{2m} \to \mathbb{Z}_p^l$, where $l = 2^m$, be defined as $h(x_1, y_1, \ldots, x_m, y_m) = (x_1^{s_1} y_1^{1-s_1} \cdots x_m^{s_m} y_m^{1-s_m})_{s \in \{0,1\}^m}$. We can then define

$$\mathsf{CFEval}_h(x_1, y_1, \ldots, x_m, y_m) \stackrel{\mathsf{def}}{=} g^{k_0(x_1 + k_1 y_1) \cdots (x_m + k_m y_m)}$$

Correctness is straightforward by expanding the expression in the exponent. The proof of the following theorem was given in [49];

Theorem 3. *[49] Suppose that the DDH assumption holds for $\mathcal{G}$. Then, $\mathcal{PRF}_3$ is a pseudorandom function.*

5 Verifiable Delegation of Polynomials

The basic idea of our construction is the following. The client stores the polynomial $P(\cdot)$ in the clear with the server as a vector $\boldsymbol{c}$ of coefficient in $\mathbb{Z}_p$. The client also stores with the server a vector $\boldsymbol{t}$ of group elements of the form $g^{ac_i + r_i}$ where $a \in_{\mathrm{R}} \mathbb{Z}_p$ and r_i is the i^{th}-coefficient of a polynomial $R(\cdot)$ of the same degree as $P(\cdot)$. When queried on input x the server returns $y = P(x)$ and $t = g^{aP(x)+R(x)}$ and the client accepts y iff $t = g^{ay+R(x)}$.

If $R(\cdot)$ was a random polynomial, then our proof below shows that this is a secure delegation scheme. However checking that $t = g^{ay+R(x)}$ would require the client to evaluate the random polynomial $R(\cdot)$, which is just as inefficient as evaluating the original polynomial $P(\cdot)$. Moreover, the client would have to store a long description of a random polynomial that is as long as the original polynomial[6] $P(\cdot)$. The crucial point, therefore, is how to make this computation fast. We do that by defining $r_i = F_K(i)$ for an algebraic PRF that has a closed form efficient computation for polynomials, such as the ones described in the previous

[6] Alternatively, the client could generate the coefficients of R using a standard PRF, thereby avoiding storing a large polynomial. However, this would still require the client to recompute all the coefficients of R each time a verification needs to be performed

section. Since F is a PRF, the security of the scheme is not compromised, and the closed form efficiency of F will allow the client to verify the result in time sub-linear in the degree of the polynomial.

The result is described in general form, using algebraic PRFs. It therefore follows that we obtain efficient and secure delegation protocols not only for single-variable polynomials of degree d, but also for multivariate polynomials with total degree d or of degree d in each variable. The relevant parameters of the resulting protocols for each of these cases can be seen in Table 1.

Finally at the end of the section we show how to protect the privacy of the polynomial, by encrypting it with an homomorphically additive encryption scheme.

Polynomial Type	Setup	C. Query	S. Query	Assumption	$\mathcal{PRF}$
1-variable, degree d	$O(d)$	$O(1)$[7] (1)	$O(d)$	d-Strong DDH	$\mathcal{PRF}_1$
n-variable, variable degree d	$O((d+1)^n)$	$O(n \log d)$ (1)	$O((d+1)^n)$	DDH	$\mathcal{PRF}_3$
n-variable, total degree d	$O(\binom{n+d}{d})$	$O(n \log d)$ (1)	$O(\binom{n+d}{d})$	d-Strong DDH	$\mathcal{PRF}_{2,d}$
n-variable, total degree d	$O((n+1)^d)$	$O(nd)$ (1)	$O((n+1)^d)$	DDH	$\mathcal{PRF}_{4,d}$

Table 1. Parameters of different protocols for verifiable delegation of polynomials. Numbers inside the parenthesis show the number of group operations. Columns starting with "C." are the client's requirements and the ones starting with "S." are the server's. In each case the server's query runtime (resp. space requirements) is asymptotically the same as evaluating (resp. storing) the polynomial. Note that $(\frac{n+1}{\sqrt{d+1}})^d \leq \binom{n+d}{d} \leq (n+d)^d$, and in particular for constant d it is $\Theta(n^d)$.

5.1 Construction Based on Algebraic PRFs

We describe a verifiable delegation scheme for functions of the form $f_{\boldsymbol{c},h}(\boldsymbol{x}) = \langle h(\boldsymbol{x}), \boldsymbol{c} \rangle$, where $\boldsymbol{c}$ is a (long) vector of coefficients, $\boldsymbol{x}$ is a (short) vector of inputs, and h expands $\boldsymbol{x}$ to a vector of the same length of $\boldsymbol{c}$. Our construction is generic based on any algebraic PRF that has closed form efficiency relative to h.

Protocol Delegate-Polynomial($\boldsymbol{c}$)

KeyGen$(\boldsymbol{c}, n)$**:** Generate $(K, param) \leftarrow_{\mathrm{R}} \mathsf{KeyGen}(1^n, \lceil \log d \rceil)$. Parse $\boldsymbol{c}$ as a vector $\boldsymbol{c} = (c_0, \ldots, c_d) \in \mathbb{Z}_p^{d+1}$. Let $\mathbb{G}$ be the range group of F_K, and let g be a generator for that group. Compute $g_i \leftarrow F_K(i)$ for $0 \leq i \leq d$, choose $a \in_{\mathrm{R}} \mathbb{Z}_p$, and set $\boldsymbol{t} = [t_0, \ldots, t_d] \leftarrow (g_0 g^{ac_0}, \ldots, g_d g^{ac_d})$. Output $PK \leftarrow (param, \boldsymbol{c}, \boldsymbol{t})$, and $SK \leftarrow (K, a)$.

ProbGen$(\boldsymbol{SK}, \boldsymbol{x})$**:** Output $(\sigma_x, \tau_x) = (\boldsymbol{x}, \boldsymbol{x})$.

Compute$(\boldsymbol{PK}, \sigma_x)$**:** Parse PK as $(param, \boldsymbol{c}, \boldsymbol{t})$, $\boldsymbol{c}$ as $c_0, \ldots, c_d$, and σ_x as $\boldsymbol{x}$. Compute $\boldsymbol{w} \leftarrow h(\boldsymbol{x}) = [h_0(\boldsymbol{x}), \ldots, h_d(\boldsymbol{x})] \in \mathbb{Z}_p^{d+1}$, $y \leftarrow \sum_{i=0}^{d} c_i h_i(\boldsymbol{x})$, and $t \leftarrow \prod_{i=0}^{d} t_i^{h_i(\boldsymbol{x})}$. Output $\nu_x = (y, t)$.

[7] The client needs to do two exponentiations.

Verify$(\boldsymbol{SK}, \tau_x, \nu_x)$**:** Parse SK as (K, a), τ_x as $\boldsymbol{x}$, and ν_x as $(y, t) \in \mathbb{Z}_p \times \mathbb{G}$. Compute $z \leftarrow \mathsf{CFEval}_h(\boldsymbol{x}, K)$, and accept if $t \stackrel{?}{=} z \cdot g^{a \cdot y}$. Otherwise, reject.

Correctness. The correctness of the above scheme follows straightforwardly from the correctness of CFEval for the algebraic PRF F.

The security analysis of the above scheme, as well as an extension allowing the client to preserve the privacy of the polynomial, are given in the full version of the paper [7].

6 Verifiable Database Queries with Efficient Updates

We have shown a general framework that allows any resource constrained client to verifiably delegate large polynomials to a server. As we have already mentioned in the introduction, this immediately gives a verifiable delegation solution to many natural practical applications (such as prediction using fitted polynomials). In this Section we present an application of our techniques to the problem of efficient verification of the result to queries posed to a dynamic database. In other words the client stores a database with the server together with some authentication informatiom. It then needs to be able to efficiently verify that the results of its queries are correct, and also to efficiently update the database and its associated authenticator. The techniques we developed for delegation of polynomials are at the basis of the solution we present, which however requires other novel and interesting technical ideas.

The protocol uses ideas borrowed from our polynomial verification scheme: the authenticator for every database entry can be efficiently reconstructed by the client using a PRF with closed form efficiency. However as we pointed out in the Introduction the main challenge comes with the updates: the client must revoke any authenticating data that the server has for the previous value of the updated cell. We deal with this problem by "masking" the authenticator with another closed-form efficient PRF. This "mask" can be efficiently removed by the client to perform the authentication and can also be efficiently updated so that old masked values cannot be reused. The technical details are somewhat involved and are described below.

Handling large payloads. For simplicity we consider databases of the form (i, v_i) where i is the index and v_i the payload data. The construction we describe below allows data values to be only polynomially large (in the security parameter). Before proceeding to describe our protocol, we show a simple transformation that allows us to support databases with arbitrary payload sizes.

On a high level, we will use a small payload protocol, such as the one described below, to store a database of the form (i, s_i) where s_i is a counter that counts the number of times index i has been updated. The server will also store the MAC of the tuple (i, s_i, v_i) where v_i is the (possibly large) payload.

When the client queries i, it will receive the value s_i through the verifiable database protocol. The security of this protocol will guarantee to the client that

s_i is correct. Then the server will also answer with v_i and the MAC on (i, s_i, v_i) and the client will accept v_i if the MAC is correct. To update index i, the client will first query i and retrieve the current tuple (i, s_i, v_i). It will then update s_i on the verifiable database by setting $s'_i = s_i + 1$. This can be done since s_i is bounded by the running time of the client and therefore polynomial. Finally it will store the new v'_i together with a MAC on (i, s'_i, v'_i). Since s_i will only ever increase, the server will not be able to re-use old MACs once an index has been updated.

Therefore for now we will focus on databases with small (polynomial) payloads. The protocol is described in detail below.

Relation to Merkle Trees. Merkles trees [42] are a standard technique for efficient authentication of data. Each element is represented as a leaf of a binary tree. The internal nodes contain hashes of their two children, and the owner of the data keeps only the root, which is essentially a hash of the entire tree. Retrieval queries can now be answered and verified in logarithmic time: the server simply sends the hashes along the path from the root to the desired leaf (along with any hashes within distance 1 of the path), and the client uses these values to compute the hash at the root of the tree. The client then checks that the stored hash value of the root is equal to the recomputed hash. Updating the tree is also quite efficient – only the hashes along the path to the updated leaf must be recomputed. In comparison, our scheme requires the client to perform a constant amount of work both during retrieval and updates, while the server must choose one of the two types of queries where he will do a linear amount of work (the other type of queries requires a constant amount of work from the server as well).

Our protocol. We give a fully detailed protocol in the full version [7]. Here, we present a high level overview of the approach. The basic tools that we use are computational bilinear groups of composite order. In this setting, a pair of groups $\mathbb{G}, \mathbb{G}_T$ are generated, along with a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Here each of the groups are of some order $N = p_1 p_2$ where p_1 and p_2 are large primes. The cryptographic assumption is that it is infeasible to distinguish a random member of $\mathbb{G}$ (or $\mathbb{G}_T$) from a random member of the subgroup of order p_1 or p_2. Such groups have recently been used to solve several important open problems in the areas of identity based encryption, leakage resilient cryptography, and other related problems (see e.g. [13]).

The basic approach of our construction (leaving some details to the full description below) can be described as follows: each entry (i, v_i) in the database (where i is an index and v_i is data) is encoded as a composite group element of the form

$$t_i = g_1^{r_i + a v_i} g_2^{w_i}.$$

Here, g_1 and g_2 are generators of the subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$ of $\mathbb{G}$, and the values r_i, w_i are generated using a pseudo-random function. To retrieve the value of the database at index i, we will have the server compute (given keys that we shall

describe in a moment) the value

$$t = g_1^{r_i + a v_i} g_2^{\sum_i w_i}.$$

Forgetting (for now) how the server should compute these values, the client can easily strip off the $\mathbb{G}_2$ masking by keeping the single group element $g_2^{\sum w_i}$ in his private key. It is now easy to see that if we replace the r_i's with random values, then our scheme is secure before any updates are performed. This follows from the fact that each entry in the database is MAC'ed with an information theoretically secure MAC (the $\mathbb{G}_2$ part hasn't played a role so far), and so the server must return the correct value in the $\mathbb{G}_1$ part of each entry. The difficulty is in allowing updates that do not require the client to change his keys for the pseudo-random functions, which in turn would require the server to obtain new MACs for all the entries in the database.

A naive solution to change the value of index i from v_i to v_i' can be for the client to send to the server a new encoding $g_1^{r_i + b v_i} g_2^{w_i}$. However, the server can then easily recover the MAC keys r_i and a by dividing the new group element that he receives during the update by the previous encoding that he already has. Our solution is therefore to randomize the new encoding by having the client send

$$t_x' = g_1^{r_i + a\delta} g_2^{w_i'},$$

where $\delta = v_i' - v_i$, and w_i' is a new pseudorandom value (generated by using a counter). Intuitively, this allows the client to send t_x' as an update token that the server can multiply into his existing group element t_i to obtain $g_1^{r_i + a v_i'} g_2^{w_i + w_i'}$. Notice that the $\mathbb{G}_1$ part is a MAC of the value v_i' using the same key that was previously used to MAC v_i. We show, relying on the subgroup membership assumption, that the random mask $g_2^{w_i + w_i'}$ effectively makes the MAC in the $\mathbb{G}_1$ of the token indistinguishable from a new MAC using fresh keys.

We now arrive at the problem of allowing the server to compute the value t, which requires stripping the $\mathbb{G}_1$ part of all the tokens except the token that corresponds to index i, without compromising security. We achieve this by issuing to the server random group elements $\hat{t}_1$ from $\mathbb{G}$, and $\hat{t}_0$ from $\mathbb{G}_2$. The server then computes the response to query i as

$$t = e(t_i, \hat{t}_1) \prod_{j \neq i} e(t_j, \hat{t}_0).$$

A remaining technical issue is the fact the in the above discussion we haven't mentioned anything about how the client should remember the new masked value w_i' after an update. Our solution is to compute it pseudo-randomly as $F_k(i, s_i)$ where s_i is a counter that is incremented with each update and is stored together with the payload v_i. This guarantees that a fresh pseudo-random value is used after each update, which in turn allows us to substitute the pseudo-random w_i's by random ones in the security analysis.

Acknowledgments. Rosario Gennaro's research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints of this work for Government purposes, notwithstanding any copyright notation hereon.

References

1. M. Ajtai, "The invasiveness of off-line memory checking", in *STOC*, pp. 504–513, 2002.
2. Amazon Elastic Compute Cloud. Online at http://aws.amazon.com/ec2.
3. B. Applebaum, Y. Ishai, E. Kushilevitz, "From Secrecy to Soundness: Efficient Verification via Secure Computation", in *ICALP*, pp.152–163, 2010.
4. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores", in *ACM CCS*, pp. 598–609, 2007.
5. L. Babai, "Trading group theory for randomness", in *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, (New York, NY, USA), pp. 421–429, ACM, 1985.
6. M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, A. Küpçü, and A. Lysyanskaya, "Incentivizing outsourced computation", in *Proceedings of the Workshop on Economics of Networked Systems (NetEcon)*, (New York, NY, USA), pp. 85–90, ACM, 2008.
7. S. Benabbas, R. Gennaro, Y. Vahlis, "Verifiable Delegation of Computation over Large Datasets", *Cryptology ePrint Archive, Report 2011/132*, http://eprint.iacr.org/.
8. M. Bellare, B. Waters, and S. Yilek, "Identity-based encryption secure against selective opening attack." To appear in TCC 2011, 2011.
9. M. Ben-Or, S. Goldwasser and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation", in *STOC*, pp.1–10, 1988.
10. M. Blum, W. Evans, P. Gemmell, S. Kannan, M. Naor, "Checking the correctness of memories", Foundations of Computer Science, Annual IEEE Symposium on, pp. 90-99, 32nd Annual Symposium of Foundations of Computer Science (FOCS 1991), 1991
11. D. Boneh and X. Boyen, "Efficient selective-id secure identity-based encryption without random oracles", in *EUROCRYPT*, pp. 223–238, 2004.
12. D. Boneh and X. Boyen, "Short signatures without random oracles and the sdh assumption in bilinear groups", *J. Cryptology*, vol. 21, no. 2, pp. 149–177, 2008.
13. D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts", in *TCC*, pp. 325–341, 2005.
14. D. Boneh, H. Montogomery, and A. Raghunathan, "Algebraic pseudorandom functions with improved efficiency from the augmented cascade", in *Proc. of ACM CCS'10*, 2010.
15. D. Boneh, A. Sahai, and B. Waters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys", in *EUROCRYPT*, pp. 573–592, 2006.

16. J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials", in *CRYPTO*, pp. 6176, 2002.
17. D. Chaum, C. Crepeau and I. Damgard, "Multiparty unconditionally secure protocols", in *STOC*, pp.11–19, 1988.
18. D. Chaum and T. Pedersen, "Wallet databases with observers", in *Proceedings of CRYPTO*, 1992
19. J. H. Cheon, "Security analysis of the strong diffie-hellman problem", in *EUROCRYPT*, pp. 1–11, 2006.
20. K.M. Chung, Y. Kalai, S.P. Vadhan, "Improved Delegation of Computation Using Fully Homomorphic Encryption", in *CRYPTO*, pp. 483–501, 2010.
21. Y. Desmedt, "Threshold Cryptography", in *Encyclopedia of Cryptography and Security*, 2005.
22. Y. Dodis, S.P. Vadhan, D. Wichs, "Proofs of Retrievability via Hardness Amplification", in *TCC*, pp.109-127, 2009.
23. Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys", in *Public Key Cryptography*, pp. 416–431, 2005.
24. C. Dwork, M. Naor, G. N. Rothblum, V. Vaikuntanathan, "How Efficient Can Memory Checking Be?", in *TCC*, pp. 503–520, 2009.
25. D. M. Freeman, "Converting pairing-based cryptosystems from composite-order groups to prime-order groups", in *EUROCRYPT*, pp. 44–61, 2010.
26. P. Gemmell, M. Naor, "Codes for Interactive Authentication", in *CRYPTO*, pp. 355–367, 2003.
27. R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers", in *CRYPTO*,pp. 465–482, 2010.
28. C. Gentry, "Fully homomorphic encryption using ideal lattices", in *Proceedings of the ACM Symposium on the Theory of Computing (STOC)*, 2009.
29. O. Goldreich, S. Micali and A. Wigderson, "How to play ANY mental game", in *STOC*, pp.218–229, 1987.
30. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles", in *Proceedings of the ACM Symposium on the Theory of Computing (STOC)*, 2008.
31. S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems", *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.
32. P. Golle and I. Mironov, "Uncheatable distributed computations", in *Proceedings of the RSA Conference*, 2001.
33. E. Hall, C. Jutla, "Parallelizable Authentication Trees", in *Selected Areas in Cryptography*, pp.95-109, 2005.
34. C. Hazay and Y. Lindell, "Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries", in *J. Cryptology*, 23(3): 422-456 (2010)
35. S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations", in *Proceedings of TCC*, 2005.
36. A. Juels, B.S. Kaliski Jr., "Pors: proofs of retrievability for large files", in*ACM Conference on Computer and Communications Security*, pp.584-597, 2007
37. J. Kilian, "A note on efficient zero-knowledge proofs and arguments (extended abstract)", in *Proceedings of the ACM Symposium on Theory of computing (STOC)*, (New York, NY, USA), pp. 723–732, ACM, 1992.
38. J. Kilian, "Improved efficient arguments (preliminary version)", in *Proceedings of the International Cryptology Conference on Advances in Cryptology*, (London, UK), pp. 311–324, Springer-Verlag, 1995.

39. A. Lewko, Y. Rouselakis, and B. Waters, "Achieving leakage resilience through dual system encryption." To appear in TCC, 2011.
40. A. B. Lewko and B. Waters, "Efficient pseudorandom functions from the decisional linear assumption and weaker variants", in *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, (New York, NY, USA), pp. 112–120, ACM, 2009.
41. C. Martel and G. Nuckolls and P. Devanbu and M. Gertz and A. Kwong and S. G. Stubblebine, "A general model for authenticated data structures", *Algorithmica,* vol. 39, no. 1, pp. 21–31, 2004.
42. R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function", in *CRYPTO*, pp. 369–378, 1987.
43. S. Micali, "CS proofs (extended abstract)", in *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1994.
44. S. Micali, M.O. Rabin, J. Kilian, "Zero-Knowledge Sets", in *FOCS*, pp.80–91, 2003.
45. S. Mitsunari, R. Sakai, and M. Kasahara, "A new traitor tracing", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 85, no. 2, pp. 481–484, 2002.
46. F. Monrose, P. Wyckoff, and A. Rubin, "Distributed execution with remote audit", in *Proceedings of ISOC Network and Distributed System Security Symposium (NDSS)*, Feb. 1999.
47. L. Nguyen, "Accumulators from bilinear pairings and applications", in *CT-RSA 2005*, vol. 3376, pp. 275292, 2005
48. M. Naor and K. Nissim, "Certificate revocation and certificate update", *USENIX Security*, pp. 17–17, 1998.
49. M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudorandom functions", *J. ACM*, vol. 51, pp. 231–262, March 2004.
50. M. Naor and G. N. Rothblum, "The Complexity of Online Memory Checking", in *FOCS*, pp.573–584, 2005.
51. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes", in *EUROCRYPT*, pp. 223–238, 1999.
52. C. Papamanthou, R. Tamassia, "Time and space efficient algorithms for two-party authenticated data structures", in *ICICS 07*, vol. 4861, pp. 115, 2007
53. C. Papamanthou, R. Tamassia, N. Triandopoulos, "Authenticated hash tables", in CCS, pp. 437448, Oct 2008
54. C. Papamanthou and R. Tamassia and N. Triandopoulos, "Optimal Authentication of Operations on Dynamic Sets", *Cryptology ePrint Archive, Report 2010/455*, http://eprint.iacr.org/.
55. H.Shacham and B. Waters, *Compact Proofs of Retrievability*, in *ASIACRYPT*, pp.90–107, 2008.
56. S. Smith and S. Weingart, "Building a high-performance, programmable secure coprocessor", *Computer Networks (Special Issue on Computer Network Security)*, vol. 31, pp. 831–960, 1999.
57. Sun Utility Computing. Online at http://www.sun.com/service/sungrid/index.jsp.
58. R. Tamassia, "Authenticated data structures, *European Symp. on Algorithms,* pp. 2–5, 2003.
59. R. Tamassia, N. Triandopoulos, "Certification and authentication of data structures", in Proc. Alberto Mendelzon Workshop on Foundations of Data Management. Cite-seer (2010)
60. A. Yao, "Protocols for secure computations", in *FOCS*, pp. 1982.
61. B. S. Yee, *Using Secure Coprocessors.* PhD thesis, Carnegie Mellon University, 1994.