Cryptographic Primitives Enforcing Communication and Storage Complexity

Philippe Golle*, Stanford University, USA
Stanislaw Jarecki, Intertrust, USA
Ilya Mironov**, Stanford University, USA

Abstract. We introduce a new type of cryptographic primitives which enforce high communication or storage complexity. Intuitively, to evaluate these primitives on a random input one has to engage in a protocol of high communication complexity, or one has to use a lot of storage. Therefore, the ability to compute these primitives constitutes certain "proof of work," because the computing party is forced to contribute a lot of its communication or storage resources to this task. Such primitives can be used in applications which deal with non-malicious but selfishly resource-maximizing parties. For example, they can be useful in constructing peer-to-peer systems which are robust against so called "free riders." In this paper we define two such primitives, a communication-enforcing signature and a storage-enforcing commitment scheme, and we give constructions for both.

Keywords: Communication Complexity and Cryptography, Storage Complexity, Peer-to-Peer Networks

1 Introduction

Peer-to-peer networks have led to an explosive growth of file trading between Internet users. Much effort has been devoted to restricting this freewheeling market, on the assumption that Internet users are eager to trade as much as possible and should be legally prevented from doing so. However, quite apart of legal issues, peer-to-peer networks suffer also from a diametrically opposite problem of so-called "free riders," i.e. users who use the services provided by others but do not provide any services to their peers (e.g. see [AH00,SGG02]). Motivated by this problem, we study the general question of how to construct efficiently-verifiable proofs that a party in some protocol uses adequately high communication or storage resources. We thus propose a novel class of cryptographic primitives which enforce either high communication or high storage complexity on some participating party, where "high" means proportional to the size of a some large input of the protocol, usually the message. These primitives are interesting on theoretical grounds, because they constitute a unique application of cryptographic tools to enforcing linear lower-bounds on communication

* Supported by Stanford Graduate Fellowship.
** Supported by NSF contract #CCR-9732754.

or storage resources. They can be used to generate a proof of work performed by some party, and are therefore applicable to settings that deal with non-malicious but selfish parties which try to minimize their resources. For example, they can be applied to auditing file trading or file storage in peer-to-peer networks and thus making such networks robust against free riders.

In this paper we define and give constructions for two novel primitives: (1) a *communication-enforcing signature*, which is a signature scheme which enforces high communication complexity on the party that signs a message, and (2) a *storage-enforcing commitment*, a commitment scheme which enforces high storage complexity on the party that commits itself to a message. We explain both notions using the following two examples.

Consider an advertising agency that distributes digital ads (e.g., pictures or short videos) to viewers on behalf of advertisers. We assume that viewers are also encouraged to exchange ads among themselves. For example, a viewer might forward an interesting or amusing ad to a friend. Viewers must submit "proofs of download" to the ad agency, which rewards them in proportion to the number of ads they have downloaded. Observe that standard digital signatures are inadequate as proofs of download, since the *hash-and-sign* paradigm on which they are based implies that knowledge of the digest of a message is sufficient to sign the whole message. Viewers could save on bandwidth by exchanging only the hashes of ads rather than the ads themselves, and claim credit for ads they never downloaded. We propose instead to use as proofs of download a new type of digital signature called *communication-enforcing signatures*. Communication-enforcing signatures (CES) share the same security properties as standard digital signatures, but offer the added guarantee that, short of leaking his private-key, the signer can only sign a message after exchanging at least as much data with the source of the message as would be required to send the message. In our example, the ad agency could collect CES from viewers and submit them to advertisers for the purpose of auditing the success of an ad campaign.

Let us now consider another example, to illustrate this time the need for auditing file storage. Assume that a couple entrusts a public notary with their will. The couple does not want to reveal this will to their children, but at the same time they want their children to have the ability to verify that the public notary is storing their will. In essence, these requirements amount to a proof of storage of a secret document. We propose a novel primitive of *storage-enforcing commitment* which can be used to enable auditing file storage. A storage-enforcing commitment (SEC) is a standard commitment scheme with the additional property that the party holding a commitment can ask the party that committed itself to a message to prove that it is still in the possession of this message, or, more precisely, that it uses at least as much storage as required to store that message. In our example, the notary would give to the children a storage-enforcing commitment to the will, which would enable the children to verify in the future that the notary is not saving on storage by erasing the will.

Organization. The rest of the paper is organized as follows. In the rest of this section, we give a brief survey of related work. In section 2, we give a formal definition of communication-enforcing signature (CES) and propose heuristic constructions for CES. We also show that the ability to compute on encrypted data would imply that CES are not possible. This leads us to believe that heuristic constructions for CES is the best we may hope for. In section 3, we define storage-enforcing commitment (SEC) and give a *provably secure* SEC construction.

1.1 Related work

Much work has been devoted to the study of communication complexity in multi-party computations. Mehlhorn et al. showed in [MS82] that there is a relationship between the communication complexity of a boolean function and the rank of the associated matrix. This relationship was also investigated in [Yao83,NW94].

However, there are surprisingly few results on communication complexity under cryptographic assumptions. A notable exception are digital signets introduced by Dwork et al. in [DLN96]. Signets are based on a primitive called incompressible functions. A length-increasing function f is incompressible if, in order to communicate $f(x)$ to Bob, Alice must either reveal her secret x to Bob, or else send Bob a message of length $\Omega(|f(x)|)$.

CES mirror the properties of incompressible functions in reverse: Alice may either send a short message to Bob (her private key) or *receive* from Bob the message to be signed. In [DLN96], the authors conjecture that some functions are incompressible, but leave as an open problem the task of building an incompressible function based on standard cryptographic primitives, or prove that such functions can not exist. Similarly, this paper offers only heuristics for building CES.

We propose a provably secure construction for a storage-enforcing commitment scheme, based conceptually upon Toeplitz matrices. These matrices have also been used for efficient constructions of a MAC [Kra94] and universal hash functions [MNT93].

2 Communication-Enforcing Signature Scheme

In this section we define communication-enforcing signature schemes (CES). We propose a few simple CES with heuristic security. Since we observe that a signature scheme cannot be provably communication-enforcing if computation on encrypted data were feasible, we conclude that the heuristically communication-enforcing schemes are the best we can hope for, and we propose our constructions as good heuristics for enforcing communication complexity as long as there are no (efficient) methods for computing on encrypted data.

2.1 Definition of a CES

A communication-enforcing signature scheme (CES) has all the properties of a regular signature scheme [GMR88], and in addition it has the "communication-enforcing" property, namely that in order to sign a message held by a source S, a signer P must either enable S to sign any message on its behalf, or else engage with S in a protocol of communication complexity equal at least to the length of the message to be signed.

A stronger notion of communication-enforcing would require that the message is computable from the signature, perhaps by an all-powerful machine. However, such a notion could only be satisfied if the length of the signature was at least the length of the message. In contrast, our definition of CES allows for schemes in which the size of the signature is polynomial only in the security parameter, as in standard digital signature schemes. Since CES are motivated by the goal of creating disincentives for the owner of a private key to avoid work associated with downloading a message that it must sign, the weaker notion of communication-enforcing is good enough. Indeed, it forces the signer to download *some* file of size comparable to the message.

Definition 1. *A communication-enforcing signature scheme (CES) is a triple of probabilistic polynomial-time algorithms:*

– *A key-generation algorithm G which outputs a private/public key pair (d, e) for every security parameter k;*
– *A signing algorithm Sig which given a message M and a private key d computes a signature $sig = \mathsf{Sig}(M, d)$;*
– *A verification algorithm Ver s.t. if $(e, d) = G(k)$ and $sig = \mathsf{Sig}(M, d)$ then $\mathsf{Ver}(M, sig, e) = 1$;*

such that for every probabilistic polynomial-time interactive algorithms of a signer P and a message source S,

- *If for every private key d and message M, the protocol $(S(M), P(d))$ outputs a signature $sig = \mathsf{Sig}(M, d)$,*
- *and if, after a repeated interaction with P on polynomially-many messages $M_1, ..., M_n$ of his choice, S cannot forge a signature on some $M \neq M_1, ..., M_n$, except for probability negligible in k*
- *then the communication complexity of the $(P(M), S(d))$ protocol for messages $M \in \{0, 1\}^n$ is at least n.*

We can restate this definition in simpler terms as follows. Whatever protocol the signer P and the message source S engage in to produce the signature $\mathsf{Sig}(M, d)$ on message M, either the communication complexity of this protocol is at least the size of M, or the signing protocol enables the source to forge signatures. Thus the definition above incorporates the unforgeability notion of [GMR88].

2.2 Implications of computing on encrypted data

It was pointed out [San01] that if there exists a method for performing general computation on encrypted data then communicating-enforcing signature schemes cannot exist. The famous problem of computing on encrypted data was posed twenty years ago by Yao [Yao82]. If there existed an efficient solution to this problem, communicating-enforcing signature schemes could not exist because the signer P could send the *encryption* of its signature key to the source S, which would then, having encrypted inputs to the Sig function compute an encryption of the output $\sigma = \mathsf{Sig}(M, d)$, which it could then send back to P, from which P would decrypt the signature σ. Thus P could compute σ without revealing its inputs to S, and the communication complexity of this protocol would be independent of $|M|$. Since there is nothing that we know of which suggests that computation on encrypted data cannot be done, we conclude that we cannot hope for a proof of communicating-enforcing property of any signature scheme. However, even if a method for computing on encrypted data was realized, it is unlikely that its computational complexity would be lower than one cryptographic operation per *gate* of the computer circuit. We thus propose the following constructions as *heuristics* for a CES primitive.

2.3 First heuristic constructing for CES: CBC-RSA

A simple implementation of a CES would be to apply a trapdoor permutation to each block of the message separately. For example, using the RSA function [RSA77] with k-bit modulus and a hash function H with k_1-bit long output size, we divide message M into blocks $M_1 ... M_n$ of size $k - k_1$, and compute $\mathsf{Sig}(M, d) = C_1 | ... | C_n$ where $C_i = H(M_i)^d \bmod N$.[1] However, such scheme is impractical because the size of the resulting signature is longer than the input message.

To reduce the output size we could CBC-chain the above construction. In other words the prover signs the first block $C_1 = H(M_1)^d \bmod N$, and for each i-th block for $i = 2, ..., n$, it computes $C_i = (C_{i-1} \oplus H(M_i))^d \bmod N$, and so on until $C_n = (C_{n-1} \oplus H(M_n))^d \bmod N$. The resulting communicating-enforcing signature is equal to the last block signature C_n. Because the block-wise RSA signature operation we use supports message-recovery, the verifier algorithm can unwind this CBC construction as follows. It computes $C_{i-1} = (C_i^e \bmod N) \oplus H(M_i)$ for every $i = n, ..., 2$, until it computes C_1, at which point it verifies if $C_1^d \bmod N = H(M_1)$.

Such CBC-chaining construction enforces high communication complexity if we model the public-key RSA permutation as a random oracle which can be accessed by any party, with the additional property that if the prover P accesses this oracle then it can also compute the back-door private-key operation.

The above construction of CBC-chaining private-key operations has a drawback of only a heuristic security argument and an usually high computational cost for the signer, who needs to perform one private-key operation per each k'-bit block of a message M, where k' is about 850 bits if we use a 1024-bit RSA and a 160-bit hash function, which means that to sign a 10 MBytes music file the signer would need to perform 100,000 RSA signature operations.

[1] Note that in the random oracle model we create an unforgeable signature on each block of the message separately [BR93].

2.4 Another heuristic construction for CES: two-root RSA

In this section we present a signature scheme, which is secure in the random oracle model and heuristically communicating-enforcing. The scheme is based on a generalization of the RSA signature scheme [RSA77].

Consider the following typical implementation of the RSA signature scheme. A signature on a message M is a solution of the following equation in $\mathbb{Z}_N$:

$$x^3 = H(M) \pmod N,$$

where H is a collision-resistant function and N is the product of two primes.

We generalize this scheme to include more blocks of the message as follows. A signature on a message $M = (M_0, \ldots, M_n)$ is a pair of two distinct roots $x_1, x_2 \in \mathbb{Z}_N$ of the following polynomial in $\mathbb{Z}_N[x]$:

$$H(M_n)x^n + H(M_{n-1})x^{n-1} + \cdots + H(M_1)x + H(M_0) = 0 \pmod N, \quad (1)$$

where H is a full-domain length-preserving collision-resistant function and $N = pq$ is a product of two prime numbers. The roots x_1, x_2 are subject to the condition that $x_1 - x_2$ is coprime with N.

By using the Ben-Or algorithm [Ben81] a root of this polynomial, if it exists, can be found in time $O(n \log n \log\log n \log^2 N)$. By the Chinese Remainder Theorem the polynomial has two roots $x_1, x_2 \in \mathbb{Z}_N$ such that $\gcd(x_1 - x_2, N) = 1$ if and only if it has at least two distinct roots both in $\mathbb{Z}_p$ and $\mathbb{Z}_q$. The existence of at least two roots in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ are independent events, each with probability between 1/3 and 1/2 [Knu75, section 4.6.2]. Therefore a solution to equation (1) exists with probability between 1/9 and 1/4. As n increases the probability approaches $e^{-2} \approx .135$.

We apply this concept to design a signature scheme which is as secure as factoring (in the random oracle model). This is done by introducing chaining between blocks, as follows.

Signing algorithm. The message M is formatted as $(M_1, \ldots, M_n)$, $n > 1$. The following algorithm is executed by the signer until step 5 succeeds (the expected number of attempts depends on n and is between 4 and 9).

Step 1. Randomly choose an initial vector $IV \xleftarrow{R} \mathbb{Z}_N^*$.
Step 2. Compute $C \leftarrow H(M_1, H(M_2, \ldots H(H(0, IV), M_n) \ldots)$.
Step 3. Compute $C_i \leftarrow H(C, M_{i+1})$ for $0 \le i < n$.
Step 4. Define $P(x) \leftarrow x^n + C_{n-1}x^{n-1} + C_{n-2}x^{n-2} + \cdots + C_1 x + C_0$.
Step 5. Find two distinct roots $t_1, t_2 \in \mathbb{Z}_p$ and two distinct roots $s_1, s_2 \in \mathbb{Z}_q$ of $P(x)$.
Step 6. Find $x_1, x_2 \in \mathbb{Z}_N$ satisfying

$$\begin{aligned} x_1 &\equiv t_1 \pmod p \qquad & x_1 &\equiv s_1 \pmod q \\ x_2 &\equiv t_2 \pmod p \qquad & x_2 &\equiv s_2 \pmod q. \end{aligned}$$

Step 7. Output the signature (IV, x_1, x_2).

Verification algorithm. A signature on a message $M = (M_1, \ldots, M_n)$ is parsed as (IV, x_1, x_2). The verifier computes $C, C_1, \ldots, C_n$ and $P(x) \in \mathbb{Z}_N[x]$ as above and checks whether $x_1 \neq x_2 \pmod{N}$. The signature is accepted if $P(x_1) = 0 \pmod{N}$ and $P(x_2) = 0 \pmod{N}$.

In the random oracle model this signature scheme is existentially unforgeable against adaptive chosen message attacks assuming the hardness of factoring. We omit the standard part of the argument in the random oracle model (see [BR93] for an example of a detailed proof) and sketch the reduction from the ability to find two roots of a random polynomial of degree n in $\mathbb{Z}_N[x]$ to factoring N.

Claim. The problem of finding two distinct roots of a random polynomial of degree n drawn from the uniform distribution on $\mathbb{Z}_N[x]$ is as difficult as factoring.

Proof Sketch: Suppose there exists an algorithm $\mathcal{A}$ that given $P(x) \stackrel{R}{\leftarrow} \mathbb{Z}_N[x]$ of degree n outputs two distinct roots (x_1, x_2) of $P(x)$ with a non-negligible probability ε. We build an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to factor N as follows:

Step 1. Choose $y_1, y_2 \stackrel{R}{\leftarrow} \mathbb{Z}_N$.
Step 2. Choose $Q(x) \stackrel{R}{\leftarrow} \mathbb{Z}_N[x]$ of degree $n - 2$.
Step 3. Compute $P(x) = (x - y_1)(x - y_2)Q(x)$.
Step 4. Run $\mathcal{A}$ on $P(x)$. Let the output of $\mathcal{A}$ be (x_1, x_2).
Step 5. If either of $x_1 - y_1, x_1 - y_2, x_2 - y_1, x_2 - y_2$ is not zero and not coprime with N, we have found a nontrivial factor of N.

The distribution from which Q is drawn in steps 1–2 is off by at most a factor of n^2 from the uniform distribution on polynomials of degree n with at least two roots. Therefore, the probability of success in step 4 is at least ε/n^2.

Notice that along with known roots y_1, y_2 of $Q(x)$ there exist two unknown roots (y_1', y_2') satisfying

$$\begin{aligned} y_1' &\equiv y_1 \pmod{p} & \qquad y_1' &\equiv y_2 \pmod{q} \\ y_2' &\equiv y_2 \pmod{p} & \qquad y_2' &\equiv y_1 \pmod{q}. \end{aligned}$$

Any one of y_1', y_2' gives away the factorization of N. The probability that $\mathcal{A}$ outputs one of these two values is no less than $\frac{1}{n^2-2}$. Therefore $\mathcal{B}$ succeeds in factoring N with probability at least ε/n^4. □

It is interesting to note that it is not known whether a similar signature scheme in which the signature of a message consists of a single root of a polynomial from $\mathbb{Z}_N[x]$ is secure under any standard assumption.

Heuristically this scheme is communicating-enforcing, since it is deemed to be hard to find a root of a polynomial without full knowledge of this polynomial.

3 Storage-enforcing Commitment

In this section we introduce a primitive called *storage-enforcing commitment scheme*. Its setup is similar to commitment schemes, but the scheme has the additional property that the committer (whom we call the prover) cannot throw away the secret it is supposed to store. This problem is trivial if the storage complexity of the verifier or the complexity of its communications with the prover are unbounded. However, we are able to define a practical storage-enforcing commitment scheme even in a more restricted setting, for which the storage and the amortized communication complexity are independent of the length of the message.

Regular commitment schemes [Blu83] bind the prover to a particular value of a string that is to be kept secret during some stages of the execution of a protocol. These commitment schemes are not designed to permit verification of repeated commitments to the same string. Storage-enforcing commitment schemes on the other hand are multi-round protocols that ensure that the prover neither "forgets" nor alters the secret between rounds. Any prover who is able to answer the verifier's challenges must keep the secret, or at least use as much storage complexity as would be required to store the secret. Formally, we define:

Definition 2 (Storage-enforcing commitment scheme). *A* storage-enforcing commitment scheme *(SEC) is a three-party protocol executed between a message source S, a prover P, and a verifier V. The message source communicates the message M to the prover and the commitment C to the verifier. The verifier V may verify whether the prover is storing the secret by invoking a probabilistic interactive algorithm* $\mathsf{Check}_{P,V}(C, M)$*. This algorithm may be executed an unlimited number of times. Once the message is revealed, the verifier may check the commitment by running the algorithm* $\mathsf{Verify}(C, M)$*.*

The scheme has the following three properties called binding, concealing, *and* storage-enforcing*:*

binding *A coalition of a computationally bounded message source and a prover cannot find two strings M and M' that both pass the test* Verify *with the same commitment C.*

concealing *A computationally unbounded verifier V cannot decide if $M = M_1$ or $M = M_2$ prior to the opening of the commitment with non-negligible advantage over a random guess even if M is known to be either M_1 or M_2 and the strings M_1 and M_2 were chosen by V.*

storage-enforcing *Any prover $\hat{P}$ that passes the test* Check *with probability $\alpha > 0$ has storage complexity $\Omega(|M|)$. The probability is taken over all messages of a fixed length $|M|$ and over the coin tosses of the prover and the verifier.*

To illustrate this definition, recall our example from the introduction. A couple entrusts a public notary with their will. The couple doesn't want their children to learn the will, yet the children should be able to verify that the public notary is properly storing the will. These requirements are satisfied by a storage-enforcing commitment scheme, where the notary is the prover and the children are verifiers. The verification protocol enables the children to verify that the notary is still in possession of the will (or at least, that the notary uses at least as much storage space as would be required to store the will), yet it does not leak any information about the will.

Verifier		**Prover**		**Message Source**
C	$\xleftarrow{\qquad\qquad C \qquad\qquad}$			commit
		M	$\xleftarrow{M}$	
`accept` or `reject`	$\xleftrightarrow{\mathsf{Check}}$	M		
	$\vdots$			
`accept` or `reject`	$\xleftrightarrow{\mathsf{Check}}$	M		
M	$\xleftarrow{M}$			
$\mathsf{Verify}(C, M)$				

Table 1. Storage-enforcing commitment scheme

Unsatisfactory solutions. We explain here why other possible commitment schemes do not meet the requirements of our example. Consider first that we could have the message source send to the verifier a non-compressing commitment to each block of the message. The verifier stores hashes of all these commitments. To execute the protocol Check, the verifier requests from the prover a commitment to a random block, hashes it and compares it to the hash previously stored. The problem with this approach is that it requires the verifier to store an amount of data (in the form of commitments) which is proportional to the amount of data the prover has to store. This defeats the point of having the prover store the data on behalf of the verifier.

Alternately, instead of storing hashes of all the commitments, the verifier could store a single hash computed on all the commitments concatenated. This approach however requires the prover to send *all* the commitments each time the protocol Check is executed. This leads to a scheme with unacceptable communication complexity.

3.1 Our storage-enforcing commitment scheme

We work in a group G of prime order p with generator g. The order of the group depends on the security parameter λ. Our scheme is based on a variant of the Decisional Diffie-Hellman assumption, which we call the n-Power Decisional Diffie-Hellman (n-PDDH) assumption.

n-Power Decisional Diffie-Hellman (n-PDDH) assumption. No polynomial-time algorithm can distinguish between the following two distributions with non-negligible advantage over a random guess:

Distribution $\mathcal{P}^n$: $(g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^n})$, where $x \overset{R}{\leftarrow} \mathbb{Z}_p$,

and

Distribution $\mathcal{R}^n$: $(g_1, g_2, \ldots, g_n)$ where $g_1, g_2, \ldots, g_n \overset{R}{\leftarrow} G$.

Notice that 2-PDDH is the same as the Decisional Square Exponent Diffie-Hellman assumption [BDS98,SS01]. We also need a weaker, computational assumption defined below.

n-Power Computational Diffie-Hellman (n-PCDH) assumption. No probabilistic polynomial-time algorithm can compute g^{x^n} given $g^x, g^{x^2}, \ldots, g^{x^{n-1}}$ with non-negligible probability.

The 2-PCDH assumption is equivalent to the Computational Diffie-Hellman assumption. It is unknown whether n-PCDH implies $(n+1)$-PCDH (the converse is obviously true).

Design of the scheme. We limit the size of a message to m blocks, where blocks are interpreted as elements of $\mathbb{Z}_p$. Longer messages can be broken into pieces and committed to simultaneously. Let $n = 2m + 1$ and assume that n-PDDH holds in the group G.

The **secret key** of the verifier is x randomly chosen from $\mathbb{Z}_p$. The corresponding **public key** is

$$PK = (g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^n}) = (g_1, \ldots, g_n).$$

The *commitment phase* consists of the following three steps:

Step 1. The verifier publishes the public key PK.

Step 2. The verifier gives a zero-knowledge proof to P that the key is properly constructed, i.e. for any index $1 \leq i < n$ the quadruple (g, g_1, g_i, g_{i+1}) is a DH-tuple [CP92].

Step 3. The message source S formats the message as an m-tuple $M = (M_1, \ldots, M_m) \in \mathbb{Z}_p^m$. S chooses a random element $M_{m+1} \in \mathbb{Z}_p^*$ and appends it to the message.

Step 4. S computes $f_0 = \prod_{i=1}^{m+1} g_i^{M_i}$ and sends it to the verifier. $f_0 \in G$ is the commitment to the message.

From the verifier's point of view the blinding block M_{m+1} is considered as an integral part of the message. For the purpose of computing the storage complexity we include this block into the message.

To check that the prover still has the message, the verifier initiates the following Check protocol:

Step 1. The verifier sends to the prover a challenge $0 \leq k \leq m$.

Step 2. The prover computes $f_k = \prod_{i=1}^{m+1} g_{i+k}^{M_i}$, where M_i is the i^{th} block of the message and sends f_k back to the verifier.

Step 3. The verifier checks whether $f_0^{x^k} = f_k$. If the equality holds, the verifier accepts the proof, otherwise the proof is rejected.

The *verification phase* is trivial—given the message $\hat{M}$, the verifier computes $\hat{f}_0$ and compares it to the f_0 received from the message source during the commitment phase.

The storage complexity of the verifier is one group element f_0 and $x \in \mathbb{Z}_p$. Since the verifier's public key can be reused for multiple messages, the scheme's storage overhead approaches zero on a per-message basis.

3.2 Proof of this commitment scheme

The commitment scheme described above is unconditionally concealing and satisfies the binding property against a computationally-bounded adversary under the n-PCDH assumption. Our proofs are generalizations of [Ped91].

Computational binding. Suppose that the message source can find two messages that may be committed to the same string C. It means that S can find for a given public key $g_1, \ldots, g_n$ two messages $M = (M_1, \ldots, M_m)$ and $M' = (M'_1, \ldots, M'_{m'})$ such that

$$g_1^{M_1} g_2^{M_2} \ldots g_m^{M_m} = g_1^{M'_1} g_2^{M'_2} \ldots g_{m'}^{M'_{m'}}.$$

It follows that the following equation is satisfied:

$$M_1 x + M_2 x^2 + \cdots + M_m x^m = M'_1 x + M'_2 x^2 \cdots + M'_{m'} x^{m'} \pmod p.$$

Since $M_m, M_{m'} \neq 0 \pmod p$ and $M \neq M'$, the two sides of the equation are not identical. Therefore the equation can be solved for x in quasi-linear time [Ben81]. This violates the n-PCDH assumption.

Perfect concealing. The concealing property holds because the only information the verifier learns about the message prior to the opening phase is determined by the value of $\sum_{i=1}^{m+1} M_i x^i \bmod p$, which is independent of the message due to the blinding block M_{m+1}. This property is unconditional and holds against a computationally unbounded verifier.

Storage-enforcing. The following theorem proves that the scheme is storage-enforcing.

Theorem 1. *Any prover $\hat{P}$ that has probability $\alpha > 0$ of passing the test* Check *has storage complexity at least $\alpha|M|$ under the n-PDDH assumption. The probability is taken over messages drawn from the uniform distribution on $\{0,1\}^{|M|}$, the public key of the verifier, and all coin tosses.*

Proof. We present the proof in two parts. First, we show (under the n-PDDH assumption) that the storage complexity of the prover and his probability of success is independent on whether the public key is chosen from $\mathcal{R}^n$ or $\mathcal{P}^n$. Second, we demonstrate that in order to pass the test with probability α when the key comes from $\mathcal{R}^n$, the prover must have storage complexity at least $\alpha|M|$. The second part of the proof is information-theoretic and unconditional.

We model the prover at different points in time as two separate algorithms $\hat{P}_1$ and $\hat{P}_2$. These two algorithms communicate by means of a bit string output by $\hat{P}_1$ and received by $\hat{P}_2$. The length of this string is at least the storage complexity of the prover.

Suppose we are given an instance of the n-PDDH problem: we must decide whether the tuple $(g_1, \ldots, g_n) \in G^n$ belongs to $\mathcal{P}^n$ or $\mathcal{R}^n$. The claim is that a prover with storage complexity less than $\alpha|M|$ can be used as a distinguisher between these two distributions. We set the public key of the verifier to the n-tuple $(g_1, \ldots, g_n)$ and simulate the prover's view such that the only difference between this game and a real execution of the scheme be the type of the tuple.

The zero-knowledge proof of the key's correctness can be simulated using standard techniques. We assume that the transcript of the proof is given to $\hat{P}$.

Let the message M be a random string consisting of $m{+}1$ blocks. The message is known and therefore we can check the responses of $\hat{P}$ by a direct computation of $f_k = \prod_{i=1}^{m+1} g_{i+k}^{M_i}$ without knowing x or even without being aware of whether the public key has the right structure. This perfectly simulates the prover's view *and* enables us to check his output.

If the prover has probability α of passing the test when the public key is constructed correctly but has a different probability of success when the key is a random n-tuple, this prover can be used as a distinguisher between the two distributions. Therefore his probability of success in either case is negligibly close to α.

Similarly, the storage complexity, modelled as the communication complexity between $\hat{P}_1$ and $\hat{P}_2$, is an observable characteristic. Therefore it must be the same for the keys drawn from $\mathcal{R}^n$ and $\mathcal{P}^n$. Otherwise we could use the observable difference between the storage complexity to break the n-PDDH assumption.

We have just proved that $\hat{P}$ must be able to pass the test Check with the same probability and storage complexity regardless of whether PK is properly constructed or drawn from $PK \stackrel{R}{\leftarrow} \mathcal{R}^n$.

We now show that in order to pass the test with a key chosen from $\mathcal{R}^n$ the prover must have storage complexity at least $\alpha|M|$, where α is the probability of success.

Consider the following random tuple: $g_1 = g^{a_1}, \ldots, g_n = g^{a_n}$. To pass the test the prover must compute $f_k = \prod_{i=1}^{m+1} g_{i+k}^{M_i} = \prod_{i=1}^{m+1} g^{a_{i+k} M_i}$ for a random k. We claim that the vector $(f_0, \ldots, f_m)$ has guessing entropy $|M|$. For $\hat{P}_2$ to reproduce a value from this list with probability at least α its input must be at least $\alpha|M|$ bit long.

Let the discrete logarithms of $(f_0, \ldots, f_m)$ to the base g be $(b_0, \ldots, b_m)$. It is easy to see that

$$b_i = (a_i, \ldots, a_{i+m}) \cdot (M_1, \ldots, M_{m+1}),$$

where $\cdot$ denotes the scalar product of two vectors from $\mathbb{Z}_p^n$. Therefore the entire vector $(b_0, \ldots, b_m)$ can be computed as

$$\begin{pmatrix} a_1 & a_2 & \ldots & a_m & a_{m+1} \\ a_2 & a_3 & \ldots & a_{m+1} & a_{m+2} \\ & & \ddots & & \\ a_m & a_{m+1} & \ldots & a_{n-2} & a_{n-1} \\ a_{m+1} & a_{m+2} & \ldots & a_{n-1} & a_n \end{pmatrix} \cdot \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_m \\ M_{m+1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \\ b_m \end{pmatrix}.$$

Denote the matrix from the left-hand side of the equation by A. This is a Toeplitz matrix defined by the values $(a_1, \ldots, a_n)$. The following lemma proves that this matrix is non-singular with overwhelming probability[2].

Lemma 1. *The Toeplitz matrix A defined as above has full rank with probability at least $1 - \frac{n}{p}$ over all tuples $(a_1, \ldots, a_n)$ randomly chosen from $\mathbb{Z}_p^n$.*

[2] Using more advanced techniques [KL96] showed that a Toeplitz matrix is non-singular with probability $1 - 1/p$.

Proof. We use the Schwartz Lemma [Sch80], which can be briefly stated as follows:

> For a non-trivial multivariate polynomial of degree n defined on variables from $\mathbb{Z}_p$, a random assignment to the variables evaluates it to zero with probability at most $1 - n/p$.

The determinant of A is a polynomial of n variables $(a_1, \ldots, a_n)$. This polynomial is not identically zero, since the matrix defined by $a_m = 1$ and $a_i = 0$ for all other $i \neq m$ has determinant $(-1)^{n-1}$. Therefore, by the Schwartz Lemma, the determinant of the Toeplitz matrix with randomly chosen entries is non-zero with probability at least $1 - n/p$. □(Lemma)

When the matrix A is full-rank, the vector $(b_0, \ldots, b_m)$ can take any value from $\mathbb{Z}_p^{m+1}$ independently of the public key. Since we assume that messages are uniformly distributed, the resulting distributions of $(b_0, \ldots, b_m)$ and the prover's responses $(f_0, \ldots, f_m)$ are also uniform. In this case the guessing entropy is equal to the Shannon entropy, which is $|M|$ bits. Therefore the prover must possess at least $\alpha|M|$ bits to give a correct response with probability α. This completes the proof of the theorem. □

3.3 Heuristic extensions

Apart from two non-classical assumptions (n-PDDH and n-PCDH), the scheme presented in the previous section is provably secure in the standard model of computation. We propose here various heuristic extensions to make the scheme more practical.

Random oracles. Recall that the first step of Commit is a zero-knowledge proof given by V that the key is properly constructed. We can make this step non-interactive by applying the Fiat-Shamir heuristic [FS87,BR93] to the 3-round Chaum-Pedersen protocol. This non-interactive proof can be part of the verifier's public key. In the random oracle model we can attach this proof even to a random tuple, thus preserving the validity of the n-PDDH assumption.

DDH-oracle. We may conjecture that our scheme is secure under a weaker assumption then the decisional variant of the Diffie-Hellman assumption. Assume for a moment that there exists a group in which there is a DDH-oracle that tests a tuple (g, g^a, h, h^b) for the equality $a = b$ but in which the computational n-PCDH problem is hard. The binding and concealing properties of the scheme hold and the scheme may still be storage-enforcing albeit the proof of Theorem 1 is no longer valid. In this case additional optimizations of the scheme are possible. We briefly sketch the resulting scheme.

> With a DDH-oracle, the main improvement to the scheme comes from the fact that all receivers can share the *same* public-key:
>
> $$(g_1, \ldots, g_n) = (g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^n}).$$

This **common parameter** may be generated once and for all during a set-up phase by a trusted authority. The value x used to generate this common parameter can then be discarded and the scheme requires no further involvement of the trusted authority.

The **commitment** is computed as follows:

Step 1. The message source S formats the message as an m-tuple $M = (M_1, \dots, M_m) \in \mathbb{Z}_p^m$. S chooses a random element $M_{m+1} \in \mathbb{Z}_p^*$ and appends it to the message.

Step 2. S computes $f_0 = \prod_{i=1}^{m+1} g_i^{M_i}$ and sends it to the verifier. $f_0 \in G$ is the commitment to the message.

The Check protocol queries the DDH-oracle, which we denote as DDH:

Step 1. The verifier sends to the prover a challenge $0 \leq k \leq m$.

Step 2. The prover computes $f_k = \prod_{i=1}^{m+1} g_{i+k}^{M_i}$ and sends it back to the verifier.

Step 3. The verifier queries $\mathsf{DDH}(g, g_k, f_0, f_k)$. If the tuple is a Diffie-Hellman tuple, the proof is accepted.

The verification of the revealed message is trivial: The verifier recomputes f_0 from the received message.

Since the verifier does not have secret data any more, the scheme can be used in a scenario where the verifier outsources his entire storage to the prover (the public key and the commitment must be signed and come with certificates).

This scheme is provably secure in the generic algorithm model [Sho97]. This is a much weaker security argument, but for many practical protocols it is often the only guarantee we have (see [Sch00] for a survey of this situation).

The first example of a group in which the Decisional Diffie-Hellman problem is easy while its computational counterpart is hard was recently proposed by Joux and Nguyen [JN01]. It is a group of points on an elliptic curve of a special type, in which the Weil pairing is non-trivial and efficiently computable. We refer the interested reader to [BF01] for details of building a practical cryptographic scheme from the Weil pairing.

4 Conclusion and Further Work

We have introduced cryptographic primitives enforcing communication and storage complexity and give constructions for them. A general direction for future work would be to further investigate the connections between communication complexity and cryptographic assumptions.

5 Acknowledgements

The authors would like to thank Dan Boneh, Moni Naor, Ron Rivest, Tomas Sander, and Victor Shoup for many helpful discussions on the subject of this paper. We are also indebted to Vitaly Shmatikov for introducing us to each other.

References

[AH00] E. Adar and B. Huberman, "Free Riding on Gnutella," First Monday, 5(10), 2000

[BG92] M. Bellare and O. Goldreich, "On defining proofs of knowledge," Proc. of CRYPTO'92, pp. 390–420, 1992.

[BKR94] M. Bellare, J. Killian, and P. Rogaway, "The security of the cipher block chaining message authentication code," Proc. of CRYPTO'94, pp. 341–358.`http://www.cs.ucdavis.edu/~rogaway`, 1994.

[BR93] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," Proc. of ACM CCS'93, pp. 62–73, 1993.

[Ben81] M. Ben-Or, "Probabilistic algorithms in finite fields," Proc. of FOCS'81, pp. 394–398, 1981.

[Blu83] M. Blum, "Coin flipping by telephone," Proc. of CRYPTO'81, pp. 11–15, 1981.

[BF01] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," Proc. of CRYPTO'01, pp. 213–229, 2001.

[BDS98] M. Burmester, Y. Desmedt, and J. Seberry, "Equitable key escrow with limited time span (or, How to enforce time expiration cryptographically)," Proc. of Asiacrypt'98, pp. 380–391, 1998.

[CCKM00] C. Cachin, J. Camenish, J. Kilian, and J. Muller, "One-round secure computation and secure autonomous mobile agents," Proc. of ICALP'00, pp. 512–523, 2001.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology revisited," Proc. of STOC'98, pp. 209–218, 1998.

[CP92] D. Chaum and T. Pedersen, "Wallet databases with observers," Proc. of CRYPTO'92, pp. 89–105, 1992.

[DLN96] C. Dwork, J. Lotspiech, and M. Naor, "Digital signets: self-enforcing protection of digital content," Proc. of STOC'96, pp. 489–498, 1996.

[FS87] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," Proc. of CRYPTO'86, pp. 186–194, 1987.

[Gol98] O. Goldreich, "Secure multi-party computation," On-line manuscript, `http://www.wisdom.weizmann.ac.il/~oded`, 1998.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," Proc. of STOC'87, pp. 218–229, 1987. (See also [Gol98]).

[GMR88] S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM J. on Computing, 17(2), pp. 281–308, 1988.

[JN01] A. Joux and K. Nguyen, "Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups," Cryptology ePrint Archive, Report 2001/003, available form `http://eprint.iacr.org/`, 2001.

[KL96] E. Kaltofen and A. Lobo, "On rank properties of Toeplitz matrices over finite fields," Proc. of ISSAC'96, pp. 241–249, 1996.

[Knu75] D. Knuth, *The Art of Computer Programming, v. 2, Seminumerical Algorithms,* 2nd Ed., Addison-Wesley, 1975.

[Kra94] H. Krawczyk, "LFSR-based hashing and authentication," In Proc. of CRYPTO'94, pp. 129–139, 1994.

[MNT93] Y. Mansour, N. Nisan, and P. Tiwari, "The computational complexity of universal hash functions," Theoretical Computer Science, v. 107(1), pp. 121–133, 1993.

[MS82] K. Mehlhorn and E. Schmidt, "Las Vegas is better than determinism in VLSI and distributed computing," Proc. of STOC'82, pp. 330–337, 1982.

[NW94] N. Nisan and A. Wigderson, "On rank vs. communication complexity," Proc. of FOCS'94, pp. 841–836, 1994.

[Ped91] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," Proc. of CRYPTO'91, pp. 129–140, 1991.

[RSA77] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Comm. of ACM, 21, pp. 120–126, 1977.

[SS01] A. Sadeghi and M. Steiner, "Assumptions related to discrete logarithms: why subtleties make a real difference," Proc. of EUROCRYPT'01, pp. 244–261, 2001.

[San01] T. Sander, private communications, 2001.

[SGG02] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," Proc. of Multimedia Computing and Networking 2002, January, 2002.

[Sch00] C. Schnorr, "Security of DL-encryption and signatures against generic attacks—A survey," Proc. of PKC&CNTC'2000, 2000.

[Sch80] J. Schwartz, "Probabilistic algorithms for verification of polynomial identities," J. of ACM, v. 27, pp. 701–717, 1980.

[Sho97] V. Shoup, "Lower bounds for discrete logarithms and related problems," Proc. of Eurocrypt'97, pp. 256–266, 1997.

[Yao82] A.-C. Yao, "Protocols for secure computations," Proc. of FOCS'82, pp. 160–164, 1982.

[Yao83] A.-C. Yao, "Lower bounds by probabilistic arguments," Proc. of FOCS'83, pp. 420-428, 1983.