Efficient Commitment Schemes with Bounded Sender and Unbounded Receiver *

Shai Halevi
MIT – Laboratory for Computer Science,
545 Technology Square, Cambridge, MA 02139
shaih@theory.lcs.mit.edu

December 24, 1996

Abstract

In this paper we address the problem of constructing commitment schemes where the sender is bounded to polynomial time and the receiver may be all powerful. Many known constructions for such commitment schemes are based on the hardness of factoring large integers. However, these schemes typically use integers of a special form and thus require a rather expensive initialization procedure for establishing these special-form integers. In this paper we present a scheme which is based on the hardness of factoring large integers but avoids the need of a complex initialization procedure.

Key words: Blum-Integers, Commitment Schemes, Factoring, Permutation Pairs.

1 Introduction

In this paper we address the problem of constructing commitment schemes for (possibly long) messages. A commitment scheme is a protocol by which one party (called the Sender) can deliver a message to another party (called the Receiver) without revealing the contents of this message, and while being committed to this message. Such a scheme emulates by means of a protocol the process of delivering the message to the Receiver in a locked box: the Sender wants to prevent the Receiver from knowing anything about the message in the box until such time in the future when the Receiver is given the key to the box. The Receiver, on the other hand, wants

*Preliminary Version appeared in *Advances in Cryptography, Proceedings of Crypto '95*. Lecture Notes in Computer Science, volume 963, Springer-Verlag, 1995. Pages 84–96.

to prevent the Sender from changing the message in the box after he has already received it.

Commitment-schemes are very useful building blocks in the design of larger cryptographic protocols. They are typically used as a mean of flipping fair coins between two players, and also play a crucial part in some zero-knowledge proofs and in various types of signature schemes.

COMMITMENT SCHEMES. A commitment scheme is comprised of two phases. The first phase simulates the delivery of the locked box. When this phase is completed, the Receiver does not know the message yet, but the Sender can not change it any more. The second phase simulates the delivery of the key. The Receiver can now see the message and verify that it is indeed the message to which the Sender is committed.

COMMITMENT SCHEMES WITH COMPUTATIONALLY UNBOUNDED RECEIVER. It is easily seen that if both parties have unlimited computational power, they cannot emulate the above process by just exchanging messages back and forth. Thus, at least one of the two parties must be *computationally bounded,* to make digital implementation possible. Indeed, many cryptographic implementations of commitment schemes have been suggested in the literature. In this work we concentrate on the case where the Sender is computationally bounded, but the Receiver may be all powerful.

1.1 Previous Work

Many commitment schemes in the unbounded-receiver model are known based on number-theoretic constructions. The first such scheme was suggested by Blum [4] in the context of flipping coins over the phone. Blum described a commitment scheme for one bit, which is based on the hardness of factoring large integers. Blum's scheme calls for one or two modular multiplications and a k-bit commitment string for every bit which is being committed to (where k is the size of the composite modulus). A similar construction with the same efficiency parameters was later described by Brassard and Crépeau [1].

A more efficient construction, which is also based on the hardness of factoring large integers, is implicit in the work of Goldwasser, Micali and Rivest [11]. Their claw-free permutation-pairs enables one to commit to long messages using about the same amount of local computation as in Blum's scheme, but to send only a k-bit commitment string, regardless of the length of the message being committed to. This construction was described explicitly in [2] and was used since then in many other works (e.g. [3, 5, 8]).

One common problem of all these constructions, however, is that they all rely on composite numbers of a special form (i.e., product of two primes which are both congruent to 3 mod 4). Thus they require a special initialization procedure in which these special-form numbers are established. It should be noted that we can not let any of the parties pick these numbers on its own, since the security of both parties depend on the proper choice of the numbers. In this paper we describe a method which also uses the GMR construction but avoids the need for this initialization step.

Several other constructions in the literature are based on the difficulty of extracting discrete-logarithms. In particular, [2] also show how to use claw-free permutation-pairs which are based on the hardness of the discrete-log problem for commitment. This scheme, however, requires one modular exponentiation per message bit. Pedersen [21] and Chaum, van-Heijst and Pfitzmann [5], described a scheme in which the Sender can commit to a string of length k (where k is the size of the prime modulus) by performing two modular exponentiations and sending a k-bit commitment string.

There were also a few implementations of commitment-schemes using more generic complexity assumptions. A "folklore" scheme in the bounded receiver (and unbounded sender) model, uses a hard-core bit of any one-way permutation and requires one application of the one-way permutation for each bit which is being committed to (cf. [14]). In the same model, Naor [18] presented a commitment scheme which can be implemented using any pseudo-random generator (or, equivalently, any one-way function [17]). As opposed to the previous schemes, however, Naor's scheme is interactive, and it requires 2 rounds of communication to commit to a string. The Sender in this scheme generates an $O(n)$-bit pseudorandom string and sends an $O(n)$-bits commitment string in order to commit to an n-bit message.

In the unbounded receiver model - Naor, Ostrovki, Venkatesan and Yung [20] described a construction which is based on any one-way permutation. Their scheme calls for $2k$ rounds of communication and one application of the one-way permutation for each bit which is being committed to. Finally a commitment scheme which uses collision intractable hash functions was first mentioned by Naor and Yung in [19], and was later improved by Damgård, Pedersen, Pfitzmann in [10] (and by Halevi and Micali in [16]). This scheme calls for one application of the hash function to the message and has a commitment string of size k (where k is the security parameter) regardless of the message length.

1.2 Contributions of This Paper

In this paper we present a modification of the GMR-based commitment scheme. The main difference between our scheme and the original GMR-based scheme is in the system set-up. The GMR-based scheme (as well as most other factoring based schemes) uses a *Blum integer* (i.e., a product of two primes, both congruent to 3 mod 4) as a system parameter. Therefore this scheme requires additional tools (such as using zero-knowledge proofs) to ensure that the system parameter is indeed a Blum integer. These additional tools are typically expensive, so the schemes become less efficient.

We present a new technique which eliminates the need for such expensive initialization steps. Instead, in our scheme we simply let the Receiver choose the system parameter and send it to the Sender. Our scheme is unique in that the form of this system parameter does not effects the security of the Sender.

Efficiency of the scheme. The modified scheme almost maintains the efficiency (of the commit and reveal phases) of the original scheme. In particular, the length of the commitment string does not depend on the length of the message: It is linear in the security parameter of the system. Also, each of the Commit and De-commit phases consists of a single round of communication. In term of local computation, the scheme requires one or two multiplications for each bit in the message (as in the original scheme) and then some additional number of multiplications which depends only on the security parameter of the system.

1.3 Organization of the Paper

The rest of this paper is organized as follows: In Sect. 2 we define the notion of a commitment scheme. In Sect. 3 we recall the factoring-based scheme which uses the claw-free permutation pairs due to [11] and then show how it can be modified to allow simple initialization.

2 Preliminaries

2.1 Commitment schemes

In this paper we do not try to give the most general definition possible for a commitment scheme. Instead, we only define "non-interactive" schemes, in which each phase (except, perhaps, initialization phase) consists of a single message, as these are the ones that we consider in the paper.

The Syntactic Structure of a Commitment Scheme. A commitment scheme is a protocol of three phases (*Initialization, Commit* and *De-commit* phases) between two parties (the Sender and the Receiver). Both parties share a common input, which is the security parameter of the system encoded in unary (we denote this by 1^k). For the later two phases, the Sender also has another input m, which is the message string to which she wants to commit herself. When used inside some other protocol, the parties may also have other inputs which represent their history at the point where the commitment scheme is being invoked.

The parties execute the Initialization phase first, and then the other two phases. It is possible to execute the initialization phase only once, and then to execute the Commit and De-commit phases many times (for many messages). In each of these times, the parties execute first the Commit phase, and at some later time they execute the De-commit phase. Typically, when used in another protocol, there will be some other parts of that protocol between the Commit and the De-commit phases.

After the initialization phase, both parties have a string p (which denotes the system parameters). During the Commit phase the Sender sends to the Receiver a commit-string c and during the De-commit phase the Sender sends to the Receiver a de-commit string d. From p, c and d the Receiver computes a message m and then checks that m is consistent with p, c and d.

Syntactically, we view a commitment scheme as a collection of the following (probabilistic polynomial-time) algorithms:

- A probabilistic INITIALIZE protocol, in which the parties - on input 1^k - compute the system parameters p. We denote the Sender's algorithm during this phase by INIT_S and the Reciever's algorithm by INIT_R.
- A probabilistic algorithm SEND which on input (p, m) outputs a pair (c, d), where c is the commit string and d is the de-commit string.
- A (possibly deterministic) algorithm RECEIVE which on input (p, c, d) outputs either a string m or the special symbol $\perp$ (meaning that the strings c, d are not the commit/de-commit strings for any message).

THE SEMANTICS OF A COMMITMENT SCHEME. The semantics of a commitment scheme should ensure that after the Commit phase the Receiver does not know anything about the message yet, but the Sender can not change it anymore, and that after the De-commit phase the Receiver is able to learn the message.

The definition of what it means for the Receiver "not to know anything about m", and for the Sender "not to be able to alter m" depends on the computational power of the parties. In the context of this paper, the Sender is bounded to probabilistic polynomial-time and the Receiver has unbounded computational power.

Definition 1: We say that the algorithms INITIALIZE, SEND, RECEIVE comprise a commitment scheme for unbounded receiver is they satisfy the following requirements:

Viability: If both the Sender and the Receiver follow their parts in the protocol, then the message m which the Receiver computes from (p, c, d) after the De-commit phase is equal to the Sender's input message. That is,

$\forall k \in \mathcal{N}, m \in \{0,1\}^*$,
if $p = \text{INITIALIZE}(1^k)$ and $(c, d) = \text{SEND}(p, m)$ then $\text{RECEIVE}(p, c, d) = m$

Secrecy: Let INIT_R^* be any algorithm which is employed by the Receiver during the Initialization phase. We denote by p_S the output of the interaction $\text{INIT}_R^*(1^k) \leftrightarrow \text{INIT}_S(1^k)$ *from the Sender's point-of-view.* Notice that p_S is a random variable.

For any possible value of p_S and any string $m \in \{0,1\}^*$, denote by $C_{p_S}(m)$ the distribution over the commit-strings for m using the system parameters p_S. That is, $C_{p_S}(m)$ is the distribution on the first coordinate of the pair which is obtained by running the algorithm $\text{SEND}(p_S, m)$. Using these notations, we can write the secrecy requirement as follows:

For any algorithm $\textsc{Init}^*_R$ *and any value of* p_S *which has a non-zero probability, and for any two messages* m_1, m_2*, the distributions* $C_{p_S}(m_1)$ *and* $C_{p_S}(m_2)$ *are identical,* $C_{p_S}(m_1) \equiv C_{p_S}(m_2)$*.*

Non-Ambiguity: The Non-ambiguity requirement asserts that it is infeasible for the Sender to "open a commitment in two different ways". We formalized this requirement by considering any probabilistic polynomial time algorithms which are employed by the Sender during the Initialization and the Commit phases, denoted $\textsc{Init}^*_S$ and $\textsc{Send}^*$ respectively.

We consider the probability that the $\textsc{Send}^*$ algorithm generates a commitment-string c and two de-commit strings d, d', and the Receiver accepts both (c, d) and (c, d') as valid commit/de-commit pairs, and computes two different messages from these pairs. The Non-ambiguity requirement is that this probability must be negligible for any probabilistic polynomial time algorithms $\textsc{Init}^*_S$ and $\textsc{Send}^*$.

Formally, we denote by $(p_R, p_{S^*}) \leftarrow [\textsc{Init}_R(1^k) \leftrightarrow \textsc{Init}^*_S(1^k)]$ the event in which, the Receiver's output from the interaction $\textsc{Init}_R(1^k) \leftrightarrow \textsc{Init}^*_S(1^k)$ is p_R and the Sender's output is p_{S^*}.

We require that for any probabilistic polynomial time algorithms $\textsc{Init}^*_S, \textsc{Send}^*$

$$\Pr\left[\begin{array}{l} (p_R, p_{S^*}) \leftarrow [\textsc{Init}_R(1^k) \leftrightarrow \textsc{Init}^*_S(1^k)]; \\ (c, d, d') \leftarrow \textsc{Send}^*(p_{S^*}); \\ \textsc{Receive}(p_R, c, d) \neq \textsc{Receive}(p_R, c, d') \\ \textsc{Receive}(p_R, c, d), \textsc{Receive}(p_R, c, d') \neq \perp \end{array}\right] = \text{negligible}(k)$$

where the probability is taken over the random coin-tosses of the algorithms $\textsc{Init}^*_S$, $\textsc{Init}_R$, $\textsc{Send}^*$ and $\textsc{Receive}$ (whichever of them happens to be probabilistic).

See Appendix A for a discussion on a few choices which we made in the above definition.

2.2 The Factorization Conjecture

The assumption we need for our scheme is that factoring a special type of integers is infeasible. The type of integers we are interested at is composites which are product of two primes of the same size, one of which is equal to 3 mod 8 and the other is equal to 7 mod 8.

Formally, for any integer k denote by $PRIMES^k_3$ the set of all k-bit primes which are congruent to 3 mod 8, and by $PRIMES^k_7$ the set of all k-bit primes which are congruent to 7 mod 8. Then we have

The Factorization Conjecture. For any probabilistic polynomial time algorithm A,

$$\Pr\left[p \in PRIMES^k_3,\ q \in PRIMES^k_7,\ N \leftarrow pq\ :\ A(N) = (p, q)\right] = negligible(k)$$

where the probability is taken over the choices of p, q (which are uniformly selected in $PRIMES_3^k, PRIMES_7^k$ respectively) and over the coin-tosses of the algorithm A.

3 A Factoring-Based Implementation

3.1 The Goldwasser-Micali-Rivest Claw-Free Permutation Pairs

Let p and q be two primes such that $p = 3 \pmod 8$, $q = 7 \pmod 8$, and denote $N \stackrel{\text{def}}{=} p \cdot q$. We start by defining two functions

$$f_{N,0}(x) \stackrel{\text{def}}{=} x^2 \pmod N \quad \text{and} \quad f_{N,1}(x) \stackrel{\text{def}}{=} 4x^2 \pmod N$$

Then, for any string $s = b_1 b_2 \cdots b_n$ we define $f_{N,s}(x) \stackrel{\text{def}}{=} f_{N,b_1}(\cdots f_{N,b_n}(x)\cdots)$. It is easy to see that both $f_{N,0}$ and $f_{N,1}$ are permutations over the quadratic residues mod N, which implies that for any s the function $f_{N,s}$ is also a permutations over the quadratic residues mod N.

3.2 Using the GMR Construction for Commitment

The following is a simple commitment scheme that uses the GMR construction, which is essentially the same as the scheme which is described in [2]. We assume that the Sender and the Receiver uses some standard (prefix free) encoding function Enc so that for no two messages $m \neq m'$ is $Enc(m)$ a prefix of $Enc(m')$. For example, $Enc(b_1 b_2 \cdots b_n) = b_1 0 b_2 0 \cdots b_n 1$ has this property (though there are better prefix-free encoding schemes).

Initialization: The Sender and the Receiver "choose at random" a composite N with k bits of the above form. We discuss this phase in more details below.

Commit phase: Given a message m, the Sender computes $s = Enc(m)$. Then she picks a random element $x \in Z_N^*$ and sends $y = f_{N,s}(x^2)$ to the Receiver.

Reveal Phase: The Sender sends both m and x to the Receiver. the Receiver computes $s = Enc(m)$ and verifies that $y = f_{N,s}(x^2)$.

It is obvious from the description that this scheme satisfies the viability requirement in Definition 1. To show that it also satisfies the other two conditions we prove two claims:

Claim 3.1: The scheme above satisfies the Secrecy requirement in Definition 1 if the composite N is a product of two primes which are both congruent to 3 mod 4.

Proof: If N is indeed of that form, then both $f_{N,1}$ and $f_{N,0}$ are permutations, and therefore so is $f_{N,s}$ for any s. Thus, for every y (which is a quadratic residue mod N) and every s there exists exactly one quadratic residue mod N, denoted x, such that $y = f_{N,s}(x)$. This means that for any s and any possible N, we have for all $y \in Z_N^*$

$$\Pr_{x \in Z_N^*}[f_{s,N}(x^2) = y] = \begin{cases} \frac{4}{\varphi(N)} & \text{if } y \text{ is a quadratic-residue mod } N \\ 0 & \text{otherwise} \end{cases}$$

which is independent of s, as needed. □

Claim 3.2: If the Factorization Conjecture holds, then the above scheme satisfies the Non-ambiguity requirement in Definition 1, provided that the composite N is chosen by a trusted third party.

This claim was proven in [11] (Theorem 1) and a generalization of it was proven in [6] (Theorem 2.8). The main idea in the proof is that given two quadratic residues x, y mod N and two strings $s \neq s'$ (none of which is a prefix of the other) so that $f_{N,s}(x) = f_{N,s'}(y)$, we can find two quadratic residues u, v mod N for which $u^2 = 4v^2$ mod N. Since N is chosen so that ± 2 are *not* quadratic residues mod N, it can be shown that $gcd(u \pm 2v, N)$ is the prime factorization of N. □

3.3 Efficiency of the Scheme

The amount of communication in the commit phase is independent of the message m. the Sender always send exactly k bits to the Receiver (where k is the number of bits in N). In the reveal phase, the Sender sends the message m and k more bits.

In terms of running time, to compute the commitment string the Sender needs to perform one or two modular multiplications for every bit in s (which presumably has about the same length as m). Using a construction similar to [6], we can use larger families of permutations to reduce the number of multiplication to one or two per byte of s. The idea is to use 256 different permutations rather than just two, and to view m as a sequence of bytes, where each byte specifies one permutation. Of course, for every r we can use the same idea to get one or two multiplications per r bits of s using 2^r permutations. Clearly, we pay for this saving in running-time by having to keep many more bits to describe these larger families of permutations, and by having to choose one of these families in the initialization phase. Moreover, the security properties of this scheme are weaker that those of the original scheme. In particular, in the original scheme we could convert an algorithm with probability ε of braking the Non-ambiguity requirement into an algorithm which factors n with the same probability. In the new scheme, instead, the success probability of the factorization algorithm will only be something like $\frac{\varepsilon}{2^{r-1}}$.

3.4 Implementing the Initialization Phase

The main problem with the above scheme is the implementation of the initialization phase. Clearly, it is important to choose the composite number N in such a way that the Sender will not be able to factor it easily. Notice that it doesn't matter whether the Receiver knows the factorization of N or not.

One idea is to let the Receiver choose N in the appropriate way and send it to the Sender. But if the Sender doesn't know the factorization of N, how can she verify that N it is really a product of two primes which are 3 mod 4 (which is the property that makes the functions $f_{N,0}, f_{N,1}$ permutations) ?

At first glance this may not look like a real problem. After all, the Sender can choose the starting point x at random, so she may be able to hide m from the Receiver even if these functions are not permutations. Unfortunately, this is not the case. Consider for example $N = 5$ and a message of one bit b. It is easy to see that for any element $x \in Z_5^*$ we have $f_{5,0}(x^2) = 1$ and $f_{5,1}(x^2) = 4$. Thus the Receiver can recover the message from the commitment string.

Intuitively, we can solve this problem by having the Receiver choose N and then prove (by means of a zero-knowledge proof) to the Sender that it is of the right form. Notice, however, that this will not satisfy our (stringent) definition of secrecy, since there is always a non-zero probability that the Sender will accept a composite N which is not of the right form. More importantly, this zero-knowledge proof can be expensive in terms of both running time and communication. It will therefore be desirable to have a system where choosing a "bad N" does not help the Receiver getting any information about m. We present such a system below.

3.5 A Modification of the GMR-Based Scheme

The only difference between the following scheme and the previous one is that after computing $y = f_{N,s}(x^2)$, the Sender squares y for k more times (where k is the number of bits in N) and sends the result to the Receiver. Note that this is equivalent to computing $f_{N,0^k s}(x^2)$. The new scheme is:

Initialization: The Receiver picks at random a k-bit composite number N which is a product of two large primes, one congruent to 3 mod 8 and the other congruent to 7 mod 8. The Receiver sends N to the Sender, who just verifies that N is of the right length.

Commit phase: Given a message m, the Sender computes $s = Enc(m)$. Then she picks a random element $x \in Z_N^*$ and sends $y = f_{N,0^k s}(x^2)$ to the Receiver.

Reveal Phase: The Sender sends both m and x to the Receiver. The Receiver computes $s = Enc(m)$ and verifies that $y = f_{N,0^k s}(x^2)$.

It is easy to see that if the Receiver picks N according to the protocol then it is still infeasible for the Sender to find two different messages with the same commitment string (if factoring is hard). The harder part is to show that even if the Receiver tries to "cheat" by picking a "bad" N, he still does not get any information about m from the commitment string.

3.6 Proof of Secrecy for the Modified Scheme

We need to show that the modified scheme satisfies the secrecy requirement for any integer N (even if N is not "of the right form"). To do that we prove the following lemma

Lemma 3.3: Let N be any integer and denote the number of bits in N by k, and let s_1, s_2 be any two strings. Then, for any element $y \in Z_n^*$ we have

$$\Pr_{x \in Z_N^*}\left[\left(f_{N,s_1}(x^2)\right)^{2^k} = y\right] = \Pr_{x \in Z_N^*}\left[\left(f_{N,s_2}(x^2)\right)^{2^k} = y\right]$$

Below we give an elegant proof for Lemma 3.3 which is due to Damgård [9]. A longer proof for the same claim can be found in the preliminary version of this paper [15]. For this proof, we need to review a few facts. The first fact about the function $f_{s,N}(\cdot)$ was first observed by Goldreich:

Fact 3.4: [12] For any integer N, any string s and any element $x \in Z_N^*$, $f_{s,N}(x) = 2^{2\hat{s}} \cdot x^{2^{|s|}}$, where $\hat{s}$ is the integer whose binary representation is s.

We now need some facts about the structure of the group Z_N^*. We start with a definition and a few notations:

Definition 2: Let N be an integer, $N > 1$, and let x be an element in Z_N^*. *The order of* x, denoted $ord(x)$, is the smallest positive integer e so that $x^e = 1 \pmod N$. We denote by O_N the subset containing all the elements of odd order in Z_N^*. That is,

$$O_N \stackrel{\text{def}}{=} \{x \in Z_N^* \ : \ ord(x) \text{ is odd}\}$$

Fact 3.5: Let N be an integer, $N > 1$. Then O_N is a subgroup of Z_N^*.

Proof: O_N is closed under multiplication since for any $x, y \in Z_N^*$, $ord(xy)$ must divide $ord(x) \cdot ord(y)$. Since $ord(x), ord(y)$ are both odd, then so is $ord(x) \cdot ord(y)$ and thus so is $ord(xy)$. $\square$

Fact 3.6: For any integer N, squaring mod N is a permutation over O_N.

Proof: It is sufficient to show that for every $x \in O_N$ there exists $y \in O_N$ so that $x = y^2 \pmod N$. So let $x \in O_N$ and denote the order of x by $2r+1$. Then if we set $y = x^{r+1} \pmod N$ we have $y^2 = x^{2r+2} = x \cdot x^{2r+1} = x \pmod N$. Finally, by Fact 3.5, $y = x^{r+1} \in O_N$. $\square$

Fact 3.7: For any integer N, any element $x \in Z_N^*$ and any $\ell \geq |N|$, $x^{2^\ell} \in O_N$.

Proof: Denote $ord(x) = 2^i \cdot r$ where r is an odd integer. Since $ord(x) < \varphi(N) < 2^{|N|}$ then $i < |N| \leq \ell$. Moreover, we have $ord(x^{2^i}) = r$, so $x^{2^i} \in O_N$, and since O_N is closed under squaring (Fact 3.5), then also $x^{2^\ell} = (x^{2^i})^{2^{\ell-i}} \in O_N$. $\square$

Fact 3.8: Let N be any integer. If we uniformly select an element $x \in Z_N^*$ and square it $|N|$ times or more mod N, then we get a uniformly distributed element in O_N. Namely, for any $\ell \geq |N|$ and $o \in O_N$

$$\Pr_{x \in Z_N^*}\left[x^{2^\ell} = o \pmod{N}\right] = \frac{1}{|O_N|}$$

Proof: Denote the prime factorization of N by $N = q_1 \cdot q_2 \cdots q_m$, where the q_i's are powers of distinct primes ($q_i = p_i^{e_i}$ for some prime p_i and positive integer e_i). Recall that Z_N^* is homomorphic to $Z_{q_1}^* \times Z_{q_2}^* \cdots \times Z_{q_m}^*$, and this homomorphism induces a homomorphism between O_N and $O_{q_1} \times O_{q_2} \cdots \times O_{q_m}$. Thus, it is sufficient to show that for each of the q_i's, uniformly selecting an element $x \in Z_{q_i}^*$ and squaring it ℓ times yields a uniformly distributed element in O_{q_i}. We distinguish between two "types" of q_i's:

Type 1: q_i is a power of two. In this case the only element of odd order in $Z_{q_i}^*$ is 1, so $|O_{q_i}| = 1$. Indeed, since $\ell \geq |N| \geq |q_i|$, then by Fact 3.7 we have

$$\Pr_{x \in Z_{q_i}^*}\left[x^{2^\ell} = 1 \pmod{q_i}\right] = \Pr_{x \in Z_{q_i}^*}\left[x^{2^\ell} \in O_{q_i} \pmod{q_i}\right] = 1$$

Type 2: q_i is a power of an odd prime. In this case $Z_{q_i}^*$ is a cyclic group. So let g be a generator in this group and we denote $ord(g) = \varphi(q_i) = r \cdot 2^t$ where r is an odd integer and $t \leq |q_i|$. Also, denote $h = g^{2^t} \pmod{q_i}$. Then the odd-order elements in $Z_{q_i}^*$ are $h, h^2, h^3, \cdots, h^r = 1$, so we have $|O_{q_i}| = r$.

Let now $o = h^e$ be some element of O_{q_i}, and we compute the probability that $x^{2^t} = o \pmod{q_i}$ when x is chosen at random in $Z_{q_i}^*$. Picking a random element in $Z_{q_i}^*$ is equivalent to picking an exponent at random $e' \in \{1, 2, \cdots r2^t\}$ and computing $x = g^{e'} \pmod{q_i}$. Moreover, we have

$$x^{2^t} = (g^{e'})^{2^t} = h^{e'} = h^{e' \bmod r} \pmod{q_i}$$

So we have $x^{2^t} = o \pmod{q_i}$ if and only if $e = e' \pmod{r}$. Thus we get

$$\Pr_{x \in Z_{q_i}^*}\left[x^{2^t} = o \pmod{q_i}\right] = \Pr_{e' \in \{1, \cdots, r2^t\}}\left[e' = e \pmod{r}\right] = \frac{1}{r} = \frac{1}{|O_{q_i}|}$$

Moreover, since (by Fact 3.6) squaring is a permutation over O_{q_i} then also for every $\ell \geq |q_i| \geq t$ we have $\Pr_{x \in Z_{q_i}^*}\left[x^{2^\ell} = o \pmod{q_i}\right] = \frac{1}{|O_{q_i}|}$. $\square$

Proof: (of Lemma 3.3) Armed with Facts 3.4-3.8, we can now prove Lemma 3.3. Let s be any string, let N be any integer, let x be a random element in Z_N^* and denote $k = |N|$, $z = (f_{s,N}(x^2))^{2^k}$. Then from Fact 3.4 we have

$$z = \left(f_{s,N}(x^2)\right)^{2^k} = \left(2^{2\hat{s}}(x^2)^{2^{|s|}}\right)^{2^k} = \left(2^{2\hat{s}}\right)^{2^k} \cdot x^{2^{(|s|+k+1)}}$$

From Fact 3.7 we have that $\left(2^{2\hat{s}}\right)^{2^k} \in O_N$, and from Fact 3.8 we have that $x^{2^{|s|+k+1}}$ is a uniformly distributed element in O_N (which is independent of s). Thus, z is a uniformly distributed element in O_N, regardless of s. This concludes the proof of Lemma 3.3. $\square$

Acknowledgments

I thank Oded Goldreich and Silvio Micali for many helpful ideas, discussions and comments, Shafi Goldwasser for several helpful comments, Ivan Damgård for the elegant proof of Lemma 3.3 and Birgit Pfitzmann for pointing out to me several references.

References

[1] G. Brassard and C. Crèpeau. Nontransitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond. In *Proc. 27th IEEE Symp. on Foundations of Comp. Science*, IEEE, 1986. pages 188–195.

[2] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *JCSS*, 37(2):156–189, 1988.

[3] G. Bleumer, B. Pfitzmann and M. Waidner. A Remark on a Signature Scheme where Forgery can be Proved. In I.B. Damgård, editor, *Proc. of Eurocrypt'90*, Lecture Notes in Computer Science, volume 473, Springer-Verlag, 1990. Pages 441–445.

[4] M. Blum. Coin flipping by telephone. In *Proc. IEEE Spring COMPCOM*, IEEE, 1982. Pages 133–137.

[5] D. Chaum, E. van Heijst and B. Pfitzmann. Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer. In J. Feigenbaum, editor, *Proc. Crypto '91*, Lecture Notes in Computer Science, volume 576, Springer-Verlag, 1992. Pages 470–484.

[6] I.B. Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *Proceedings of EUROCRYPT 87*. Lecture Notes in Computer Science No. 304. Springer-Verlag, 1988. Pages 203–216.

[7] I.B. Damgård. On the existence of a bit commitment schemes and zero-knowledge proofs. In G. Brassard, editor, *Proceedings CRYPTO 89*. Lecture Notes in Computer Science No. 435. Springer-Verlag, 1990. Pages 17–29.

[8] I.B. Damgård, Practical and Provably Secure Release of a Secret and Exchange of Signatures. T. Helleseth, editor, *Proc. EuroCrypt '93*, Lecture Notes in Computer Science, volume 765, Springer-Verlag, 1994. pages 200–217.

[9] I.B. Damgård. Private communication.

[10] I.B. Damgård, T.P. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In Douglas R. Stinson, editor, *Proceedings CRYPTO 93*. Lecture Notes in Computer Science No. 773. Springer, 1994. Pages 250–265.

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

[12] O. Goldreich. Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme. In A.M.Odlyzko, editor, *Proceedings CRYPTO 86*, Lecture Notes in Computer Science No. 263. Springer-Verlag, 1987. Pages 104–110.

[13] O. Goldreich. A Uniform Complexity Treatment of Encryption and Zero-Knowledge. *Journal of Cryptology*, Vol. 6, No. 1 (1993), Pages 21–53.

[14] O. Goldreich. Foundations of Cryptography – Fragments of a Book (1995)

[15] S. Halevi, Efficient commitment with bounded sender and unbounded receiver. In D. Coppersmith, editor, *Proc. Crypto '95*. Lecture Notes in Computer Science, volume 963, Springer-Verlag, 1995. Pages 84–96.

[16] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In N. Koblitz, editor, *Proc. Crypto '96*. Lecture Notes in Computer Science, volume 1109, Springer-Verlag, 1996. pages 201–215.

[17] J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. Construction of Pseudo-random Generator from any One-Way Function. Manuscript, 1993. (To appear in *SICOMP*.) See preliminary versions by Impagliazzo et. al. in *21st STOC* and J. Håstad in *22nd STOC*.

[18] M. Naor. Bit commitment using pseudo-randomness. In G. Brassard, editor, *Proceedings CRYPTO 89*, Lecture Notes in Computer Science No. 435. Springer-Verlag, 1990. Pages 128–137.

[19] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, 1989. Pages 33-43.

[20] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for *NP* can be based on general complexity assumptions. In Ernest F. Brickell, editor, *Proceedings CRYPTO 92*, Lecture Notes in Computer Science No. 740. Springer-Verlag, 1992. Pages 196–214.

[21] T.P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In J. Feigenbaum, editor, *Proc. Crypto '91*, Lecture Notes in Computer Science, volume 576, Springer-Verlag, 1992. pages 129–140.

A Remarks on the Definition of Commitment-Schemes

Recall our definition of (non-interactive) commitment schemes. We say that the algorithms Initialize, Send, Receive comprise a commitment scheme for unbounded receiver is they satisfy the following requirements:

Viability: $\forall k \in \mathcal{N}, m \in \{0,1\}^*$,
if $p = \textsc{Initialize}(1^k)$ and $(c,d) = \textsc{Send}(p,m)$ then $\textsc{Receive}(p,c,d) = m$

Secrecy: For any algorithm $\textsc{Init}_R^*$, any value of the system parameter p_S which has a non-zero probability, and for any two messages m_1, m_2, the distributions $C_{p_S}(m_1)$ and $C_{p_S}(m_2)$ are identical.

Non-Ambiguity: For any probabilistic polynomial time algorithms $\textsc{Init}_S^*, \textsc{Send}^*$

$$\Pr\left[\begin{array}{l} (p_R, p_{S^*}) \leftarrow [\textsc{Init}_R(1^k) \leftrightarrow \textsc{Init}_S^*(1^k)]; \\ (c,d,d') \leftarrow \textsc{Send}^*(p_{S^*}); \\ \textsc{Receive}(p_R,c,d) \neq \textsc{Receive}(p_R,c,d') \\ \textsc{Receive}(p_R,c,d), \textsc{Receive}(p_R,c,d') \neq \perp \end{array}\right] = \text{negligible}(k)$$

where the probability is taken over the random coin-tosses of the algorithms $\textsc{Init}_S^*$, $\textsc{Init}_R$, $\textsc{Send}^*$ and Receive (whichever of them happens to be probabilistic).

A few points about this definition are worth noting

Stringent Secrecy. The *Secrecy* requirement above is rather stringent. First, we require that the distributions $C_{p_S}(m_1), C_{p_S}(m_2)$ be identical (rather than, say, very close). Moreover, we require that this happens for *every possible value* of p_S, and even when the $\textsc{Init}_R^*$ algorithm is not restricted in terms of computational complexity. Most other schemes in the literature do not achieve this requirement for all possible values of p_S, and instead only satisfy it with high probability over the executions of

INIT_S. Nonetheless, the scheme which we presented in this paper satisfies the above requirement.

AUXILIARY INPUTS. The definition as it is written above does not take into account auxiliary inputs which may be available to the parties during the scheme. When a commitment scheme is used inside a larger protocol the parties may have auxiliary inputs from this higher level protocol, so we should require that the scheme remains secure in the presence of such auxiliary inputs. It is not hard to see that this does not affect the *Secrecy* requirement, since the commitment string is statistically independent of the message which is being committed to, regardless of any auxilary inputs.

As for the *Non-ambiguity* requirement, there are two ways in which the auxiliary inputs can be incorporated into the definition. One way is to augment the requirement so that it holds for any auxiliary input z of length polynomial in k (which is given as an extra input to the $\text{INIT}^*, \text{SEND}^*$ algorithms). This, however, has the effect of making $\text{INIT}^*, \text{SEND}^*$ non-uniform. Therefore, any computational assumption that we make must hold in the non-uniform model. In particular, in the scheme which we described in this work we must assume that no polynomial-size circuit can find the prime factorization of Blum-integers.

If we want to stay in the uniform model, we must somehow restrict the type of auxiliary inputs which we allow, to those which can be generated by probabilistic polynomial-time machines. Informally, this means that although some auxiliary inputs may help a cheating Sender to open its commitment in two different ways, it is infeasible for any probabilistic polynomial-time machine to find these inputs. We do not formalize this intuition here. The reader is referred to [13] for a formal treatment of these problems (for the case of encryption and Zero-Knowledge Proofs).

GENERATION OF DE-COMMIT STRINGS. The *Non-ambiguity* condition in our definition may seem weak in that it requires that the de-commit strings d, d' be generated at the same time as the commit string c. To clarify this point, recall that when a commitment scheme is used inside a higher-level protocol, there are typically some other parts of this higher-level protocol between the Commit and the De-commit phases. Therefore, it is conceivable that the Sender will have more information when it sends the de-commit string than it had when it generated the commit string. In principle, the definition should require that the Sender is unable to generate d, d' even after getting all this extra information.

It is worth stressing that we can not simply treat this extra information as an auxiliary input, as it may depend on the behavior of the Receiver between the Commit and De-commit phases, which in turn may depend on the commitment string c and the system parameters p_R. Note that as long as the legitimate Receiver is probabilistic polynomial-time, anything it sends which only depends on c can be simulated by the Sender, and therefore can not help the Sender in cheating. However, the Receiver also knows p_R (which may be hidden from the Sender), and if it sends messages which depends on p_R then these messages may help the Sender.

There are two ways in which this issue can be dealt with in the definition. One way is to leave the definition as is, and require that in any protocol which uses the commitment scheme, the Receiver does not use its knowledge of p_R before the De-commit phase. This requirement may make sense in some cases since p_R can be thought of as an internal parameter of the commitment scheme which should not be used by the higher-level protocol. However, there are cases where this requirement can not be met (specifically when we use the same value of p_R for several executions of the Commit/De-commit phases).

Alternatively, we could strengthen the *Non-ambiguity* requirement so that is holds even when p_R is given as input to the SEND^* algorithm. We note that the scheme we presented in this work satisfies this strengthened requirements (as do all other schemes in the literature).