Non-Interactive and Non-Malleable Commitment*

Giovanni Di Crescenzo† Yuval Ishai‡ Rafail Ostrovsky§

Abstract

A *commitment* protocol is a fundamental cryptographic primitive used as a basic building block throughout modern cryptography. In STOC 1991, Dolev Dwork and Naor showed that in many settings the implementation of this fundamental primitive requires a strong *non-malleability* property in order not to be susceptible to a certain class of attacks. In this paper, assuming that a common random string is available to all players, we show how to implement non-malleable commitment without any interaction and based on any one-way function. In contrast, all previous solutions required either logarithmically many rounds of interaction or strong algebraic assumptions.

1 Introduction

COMMITMENT: One of the most fundamental cryptographic protocols is the *commitment* protocol. A commitment protocol involves two probabilistic polynomial-time players: the *committer* and the *receiver*. Very informally, it consists of two stages, a *commitment* stage and a *de-commitment* stage. In the commitment stage, the committer with a secret input x engages in a protocol with the receiver. In the end of this protocol, receiver still does not know what x is (i.e. x is computationally hidden), and at the same time, the committer can subsequently (i.e., during the de-commitment stage) open only one possible value of x.

Commitment is used as a sub-protocol in a vast variety of cryptographic applications, including, to name a few, contract signing [8], zero-knowledge proofs for all of NP [11], general multi-party computations [12] and many others. Hence, a more efficient implementation of this protocol (with the right notion of security) is crucial for the efficient implementation of a variety of cryptographic primitives.

THE MODEL AND OUR CONTRIBUTION: We will consider the common random string model (originally introduced in [5] and elaborated in [4], for non-interactive zero-knowledge proofs), a model where a polynomial-length common random string is available to all the users. In this setting, we consider the following problem: users wish to commit (and later de-commit) values to one another, in a so-called *non-malleable* manner [6], where informally, a *non-malleable* commitment requires that given a "committed" value, an attacker can not come-up with a commitment of a "related" value.

In this paper, we exhibit a non-malleable commitment protocol which relies on the existence of *any* one-way function (a necessary and sufficient assumption), *does not* require interaction (i.e., committer sends a single message to receiver for both commitment and de-commitment stages) and *does not* use any costly zero-knowledge proofs. In contrast, despite the fundamental importance of this primitive (formalized by Dolev, Dwork and Naor [6]), all previous work required either logarithmically many rounds of interaction (in the size of identities) and the use of zero-knowledge proofs [6] or very strong assumptions [7, 3].

In the heart of our construction there are a new protocol and a new proof-technique, which allow us to completely avoid many rounds of interaction without sacrificing the generality of the assumption. As with the original work of Dolev, Dwork and Naor [6], our setting does not assume a trusted center, and users do not need to know anything about the number or identity of other users in the system.

In our model, we assume the existence of a common random string, whereas [6] do not. However, our results extend to the case where no such common random string is available (again based on any one-way function and without the use of zero-knowledge).

THE NOTION OF NON-MALLEABILITY: The notion of non-malleable commitment can be best explained with the following motivating example from [6]: suppose there are several players who participate in a *contract bidding* game, where a contract goes to the lowest bidder. First, the players send the *commitments* of their bids, and once all bids have been deposited, they de-commit. In [6] it was observed that even if the commitment scheme is computationally secure against any polynomially-bounded receivers, still a malicious committer can potentially come up with a commitment of a related bid, without any knowledge what the original bid is, but still being able to underbid. The reason is that the standard notion of commitment does not disallow the ability to come-up with the related commitments (for which an attacker does not know the de-commitment at the commitment stage), but for which once the attacker gets the de-commitment of the original value, he can compute the de-commitment of his related de-commitment as well. In fact, in the appendix we show that several standard commitment schemes are provably malleable.

*THIS IS A PRELIMINARY EXTENDED ABSRTACT, A FULL VERSION OF THE PAPER IS BEING WRITTEN, CONTACT THE AUTHORS

†Computer Science and Engineering Department, University of California San Diego, La Jolla, CA, 92093-0114, USA. E-mail: giovanni@cs.ucsd.edu. Part of this work was done while visiting Bellcore.

‡Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: yuvali@cs.technion.ac.il. Part of this work was done while visiting Bellcore.

§Bell Communication Research, Morristown, NJ, 07960-6438, USA. E-mail: rafail@bellcore.com

PREVIOUS WORK: The notion of non-malleability was first formalized and implemented by Dolev, Dwork and Naor in [6]. Their main result is the first implementation of non-malleable commitment based on any one-way function. The drawbacks of their solution are that it requires at least logarithmic number of rounds of interaction between committer and receiver and it uses costly zero-knowledge proofs.

Based on algebraic assumptions, one can build a non-malleable commitment scheme using non-interactive zero-knowledge proofs of knowledge of De Santis and Persiano [7]. That is, [7] implement non-interactive zero-knowledge proof of knowledge assuming the existence of so-called "dense" cryptosystems, which in turn are known to exist only under some strong algebraic assumptions (such as RSA). Moreover, the scheme uses inefficient zero-knowledge sub-protocols.

With even stronger assumption, that of the existence of cryptographic hash functions which behave like random oracles, Bellare and Rogaway [3] showed how to implement non-malleable commitment in an efficient way. However it is not known how to implement (or even define) such random oracles with the properties that they require under any complexity-theoretic assumptions. In practical setting, the implementation substitutes a random oracle with collision-free hash function (like MD5) and relies on an unproven assumption that MD5 or some other function behaves like a random oracle.

In summary, all the previous proposed solutions to this fundamental problem required either very strong assumptions or logarithmic number of rounds of interaction, and relied on inefficient zero-knowledge proofs.

REMARKS AND GENERALIZATIONS: We introduce new techniques for achieving non-malleability, which avoid using proofs of knowledge or zero-knowledge proofs and use weakest possible complexity assumptions (since any bit-commitment protocol implies the existence of a one-way function [16]). Specifically, in a completely anonymous setting, we construct a non-interactive non-malleable string commitment scheme under the (minimal) assumption of the existence of one-way functions.

Several remarks regarding definitions of malleability are in order here. (The concerns are the same as in [6] and these points are addressed there as well, for further discussion the reader is suggested to look there.) One is the issues of *identities*. In a completely anonymous setting, one can not prevent exact copying of the commitments. Thus, the non-malleability definition specifies that if the commitment is not copied exactly, then it is not related according to any *interesting* relation (for further details, see [6]). Assuming user identities, one can prevent exact copying as well.

We remark that our techniques allow polynomially many commitments by using a public random string of fixed size and also generalize to other settings and other non-malleable tasks as well, including non-malleable zero-knowledge and non-malleable commitment without the common random string. They also generalize the assumption needed for a result on interactive arguments in [2]. Finally, we remark that our techniques in fact solve another open problem posed by Beaver [1] – that of the construction of so-called equivocable bit-commitment, which has implications to zero-knowledge proofs as well. We postpone this and other generalizations to the full version of the paper.

2 Definitions

In this section we recall some definitions about indistinguishability, and the definitions of bit-commitment scheme, equivocable bit-commitment scheme and non-malleable bit-commitment scheme in a public random string model.

Basic notations and definitions.
Basic notations. We use notations for probabilistic algorithms similar to those in [13]. The notation $x \leftarrow S$ denotes the *random process* of selecting element x from set S with uniform probability distribution over S. Similarly, if $\mathcal{D}$ is a distribution, the notation $x \leftarrow S$ denotes the random process of selecting element x according to distribution $\mathcal{D}$. Moreover, the notation $y \leftarrow A(x)$, where A is an algorithm, denotes the random process of obtaining y when running algorithm A on input x, where the probability space is given by the random coins (if any) of algorithm A. A random variable V will be denoted by $\{R_1; \ldots; R_n : v\}$, where v denotes the values that V can assume, and $R_1, \ldots, R_n$ is a sequence of random processes generating value v. By $\Pr[R_1; \ldots; R_n : E]$ we denote the probability of event E, after the execution of random processes $R_1, \ldots, R_n$. We say that a function in n is *negligible* if it is $\leq n^{-c}$, for all constants c and all sufficiently large n.
System model. We will consider a distributed model known as the *public-random-string* model [5, 4], introduced in order to construct non-interactive zero-knowledge proofs (i.e., zero-knowledge proofs which consist of a single message sent from a prover to a verifier). In this model, all parties share a public *reference string* which is assumed to be uniformly distributed. Furthermore, this model is anonymous in a strong sense: parties do not have any knowledge of other parties' identities, or of the network topology. A *sender-receiver pair* (A,B) is a pair of probabilistic polynomial-time Turing machines sharing a communication tape. We will distinguish between the algorithms A,B and the parties S, R that execute such algorithms. We assume all parties share a security parameter 1^n as common input.

Indistinguishability. Following [13, 21], we say that two families of random variables V_0, V_1 are *computationally indistinguishable* if for all efficient non-uniform (distinguishing) algorithms D_n, for every $d > 0$ and all sufficiently large n,

$$\left|\Pr[D_n(V_0(1^n)) = 1] - \Pr[D_n(V_1(1^n)) = 1]\right| < n^{-d}.$$

In the sequel, the index n will usually be omitted when referring to families of random variables.

We say that two families of random variables V_0, V_1 are *perfectly indistinguishable*, if they are identically distributed.

2.1 Bit-commitment schemes

We start by defining the basic bit-commitment primitive, which will be used as a building block for constructing the much stronger primitive of non-malleable string-commitment.

Informally speaking, a *bit-commitment scheme* (A,B) in the public-random-string model is a two-phase interactive protocol between two probabilistic polynomial time parties A and B, called the sender and the receiver, respectively, such that the following is true. In the first phase (the commitment phase), A commits to bit b by computing a pair of keys (com, dec) and sending com (the commitment key) to B. Given just the public random string and the commitment key, the polynomial-time receiver B cannot guess the bit with probability significantly better than $1/2$ (this is the security property). In the second phase (the decommitment phase) A reveals the bit b and the key dec (the decommitment key) to B. Now B checks whether the decommitment key is valid; if not, B outputs a special string $\perp$, meaning that he rejects the decommitment from A; otherwise, B can efficiently compute the bit b revealed by A and is convinced that b was indeed chosen by A in the first phase (this is the binding property).

We remark that the commitment schemes considered in the literature can be divided in two types, according to whether the security property holds with respect to computationally bounded adversaries or to unbounded adversaries. The first (resp., second) type of bit-commitment schemes have been shown to have applications mostly to zero-knowledge proofs (resp., arguments) (see, e.g., [11, 18]). A computationally-secure bit-commitment scheme has been constructed under the minimal assumption of the existence of pseudo-random generators (see [17]). A perfectly-secure bit-commitment scheme has been constructed under the assumption of the existence of one-way permutations (see [18]). In the following, we include both types in the same formal definition.

Definition 1 (Non-interactive bit-commitment)
Let a be a constant, n be an integer and σ be a public random string of length n^a; let (A,B) be a sender-receiver pair. We say that (A,B) is a *computationally-secure bit-commitment scheme* (resp. *perfectly-secure bit-commitment scheme*) in the public-random-string model if the following conditions hold:

1. *Meaningfulness.* For all constants c, each $b \in \{0,1\}$, and all sufficiently large n,

$$\mathbf{Pr}[\quad \sigma \leftarrow \{0,1\}^{n^a}; (com, dec) \leftarrow A(\sigma, b); \\ d \leftarrow B(\sigma, com, dec) : d = b \qquad] \geq 1 - n^{-c}.$$

2. *Security.* The families of random variables A_0 and A_1 are computationally (resp. perfectly) indistinguishable, where $A_b = \{\sigma \leftarrow \{0,1\}^{n^a}; (com, dec) \leftarrow A(\sigma, b) : (\sigma, com)\}$, for $b = 0, 1$.

3. *Binding.* For all algorithms (resp. probabilistic polynomial time algorithms) A', all constants c, and all sufficiently large n,

$$\mathbf{Pr}[\, \sigma \leftarrow \{0,1\}^{n^a}; (com, dec_0, dec_1) \leftarrow A'(\sigma): \\ B(\sigma, com, dec_0) = 0 \ \wedge \ B(\sigma, com, dec_1) = 1 \,] < n^{-c}.$$

We remark that the above definition naturally extends to a definition of string commitment scheme, where the security is formalized using the notion of semantic security [13]. Moreover, for any string $s = s_1 \circ \cdots \circ s_n$, where $s_i \in \{0,1\}$, the scheme obtained by independently committing to each bit s_i using a secure bit-commitment scheme is a secure string commitment scheme.

2.2 Equivocable bit-commitment scheme

Informally speaking, a bit-commitment scheme is equivocable if it satisfies the following additional requirement. There exists an efficient simulator which outputs a transcript leading to a faked commitment such that: (a) the commitment can be decommitted both as 0 and as 1, and (b) the simulated transcript is indistinguishable from a real execution. We now formally define the equivocability property for bit-commitment schemes in the public random string model.

Definition 2 (Non-interactive equivocable bit commitment)
Let a be a constant, n be an integer and σ be a public random string of length n^a; let (A,B) be a bit-commitment scheme in the public random string model. We say that (A,B) is a *non-interactive computationally (resp., perfectly) equivocable bit commitment scheme* in the public random string model if there exists an efficient probabilistic algorithm M which, on input 1^n, outputs a 4-tuple $(\sigma', com', dec_0, dec_1)$, satisfying the following:

1. For $c = 0, 1$, it holds that $\mathrm{B}(\sigma', com', dec_c) = c$.

2. For $b = 0, 1$, the families of random variables $A_0 = \{\sigma \leftarrow \{0,1\}^{n^a}; (com, dec) \leftarrow A(\sigma, b) : (\sigma, com, dec)\}$ and $A_1 = \{\, (\sigma', com', dec_0, dec_1) \leftarrow M(1^n) : (\sigma', com', dec_b)\}$ are computationally (resp., perfectly) indistinguishable.

Remarks and history. As for ordinary commitment, we remark that the above definition naturally extends to a definition of equivocable string commitment scheme, and that for any string $s = s_1 \circ \cdots \circ s_n$, where $s_i \in \{0,1\}$, the scheme obtained by independently committing to each bit s_i using an equivocable bit-commitment scheme is an equivocable string commitment scheme. Equivocable bit-commitment schemes have been first discussed in [1], who observed the seemingly paradoxical requirement that such schemes need to satisfy (in the case of computational security). Precisely, the existence of an efficient simulator which is able to construct a commitment key, that can be opened in two ways, seems to be in contrast with the binding property of the scheme, requiring that an infinitely powerful committer is not allowed to do so in a real execution of the scheme. In [1] the construction of an equivocable commitment scheme is left as an open problem. In this paper, we show the existence of an equivocable commitment scheme in the public-random-string model, and use it to construct a non-malleable commitment scheme.

2.3 Non-malleable commitment scheme

We present the definition of non-malleable commitment schemes, introduced in [6]. Here, we present an adaptation of that definition to the public random string model.

Let k be an integer and let $\mathcal{D}$ be an efficiently sampleable distribution over the set of k-bit strings (represented by its generator). Let R be a *relation approximator*, that is, an efficient probabilistic algorithm that, given two strings, returns a binary output (algorithm R is supposed to measure the correlation between the two input strings). Also, given a committer algorithm, we say that $\mathcal{A}'$ is an *adversary simulator* if, on input $\mathcal{D}$, it outputs a string in $\{0,1\}^k$ (algorithm $\mathcal{A}'$ is supposed to simulate the behavior of an adversary who is not given a commitment as input).

Now, consider two experiments: an *a-posteriori* experiment, and an *a-priori* one.

In the a-posteriori experiment, given a commitment com_1 to a string s_1, an efficient non-uniform adversary $\mathcal{A}$ tries to compute a commitment $com_2 \neq com_1$ which, later, when he is given the decommitment of com_1, can be decommitted as a string s_2, having some correlation with string s_1.

In the a-priori experiment, an adversary simulator $\mathcal{A}'$ commits to a string s_2, given only the knowledge of $\mathcal{D}$.

We consider a non-malleable commitment scheme as a commitment scheme in which for any relation approximator R and for any adversary $\mathcal{A}$, there exists an adversary simulator $\mathcal{A}'$ which succeeds "almost as well" as $\mathcal{A}$ in returning strings which make R evaluate to 1.

Definition 3 (Non-interactive non-malleable string commitment) Let a be a constant, let (A,B) be a non-interactive string commitment scheme in the public random string model. We say that (A,B) is a *non-interactive non-malleable string commitment scheme* in the public random string model if for every efficient non-uniform algorithm $\mathcal{A}$, there exists an efficient non-uniform adversary simulator $\mathcal{A}'$, such that for all relation approximators R, for all efficiently sampleable distributions $\mathcal{D}$, for all constants c and all sufficiently large n, it holds that $p(\mathcal{A}, R) - p'(\mathcal{A}', R) \leq n^{-c}$, where the probabilities $p(\mathcal{A}, R)$ and $p'(\mathcal{A}', R)$ are defined as

$$p(\mathcal{A}, R) = \mathbf{Pr}[\quad \sigma \leftarrow \{0,1\}^{n^a}; s \leftarrow \mathcal{D}; \\ (com_1; dec_1) \leftarrow A(\sigma, s); \\ com_2 \leftarrow \mathcal{A}(\sigma, com_1); \\ dec_2 \leftarrow \mathcal{A}(\sigma, com_1, com_2, dec_1) : \\ B(\sigma, com_1, dec_1) = s \ \wedge \\ B(\sigma, com_2, dec_2) = t \ \wedge \\ com_2 \neq com_1 \ \wedge \ R(s,t) = 1 \quad].$$

$$p'(\mathcal{A}', R) = \mathbf{Pr}[\, s \leftarrow \mathcal{D}; t \leftarrow \mathcal{A}'(\mathcal{D}) : R(s,t) = 1 \,].$$

Remarks. Notice that the definition of non-interactive non-malleable bit commitment can be easily derived from the above. For sake of clarity, we will first describe our construction of a non-interactive non-malleable bit commitment (in Section 4), and then give a technique transforming any non-interactive

non-malleable bit commitment scheme into a non-interactive non-malleable string commitment scheme (notice that simple repetition does not work, see Section 5).

Moreover, we see that in the above definition the adversary succeeds only if he generates a different commitment key; i.e., if $com_1 \neq com_2$. In other words, we are ruling out the situation in which the committer S_2' simply copies the commitment string sent by committer S_1. The reason for this is that, as also observed by [6], this situation provably cannot be avoided in a setting of fully anonymous parties, while, on the other hand, it can always be avoided in a setting in which parties have verifiable identities.

Finally, we notice that the above definition considers an adversary that uses the same commitment scheme as the original committer. We can generalize the definition to require that a scheme is non-malleable if the adversary, using *any* commitment scheme, is not successful as formalized above. We note that our schemes satisfy this stronger definition as well. We have also investigated several alternative but equivalent definitions for non-malleable commitment, which we will further explore in the full version.

3 Non-interactive equivocable bit-commitment

In this section we show that in the public random string model any non-interactive bit-commitment scheme can be transformed into a non-interactive equivocable bit-commitment scheme. Precisely, we show that the bit-commitment scheme from [17], when implemented in the public random string model, can be shown to be equivocable. Since the scheme in [17] is based on the existence of pseudo-random generators, and pseudo-random generators are known to exist under the assumption of existence of a one-way function (using [15]), we obtain that there exists a non-interactive equivocable commitment scheme under the minimal assumption of the existence of a one-way function. Observing that one-way functions can be constructed from a non-interactive bit-commitment scheme (using [16]), we obtain the following

Theorem 1 In the public random string model, given a non-interactive commitment scheme it is possible to construct a non-interactive equivocable commitment scheme.

Notice that it is enough to prove the above theorem for the case of single bit-commitment, since, as already remarked, this would extend to strings using simple independent repetition. Now, we start by briefly recalling the bit-commitment scheme in [17], and its properties, and then prove that the implementation of this scheme in the public random string model is equivocable.

Bit commitment from any pseudo-random generator [17]. Let $n > 0$ be an integer, and $G : \{0,1\}^n \to \{0,1\}^{3n}$ be a pseudo-random generator agreed upon by the committer A and the receiver B.

Commitment phase. First B sends a $3n$-bit uniformly chosen string $R = r_1 \circ \cdots \circ r_{3n}$, where each $r_i \in \{0,1\}$. Then A uniformly chooses an n-bit seed s and computes $G(s) = t_1 \circ \cdots \circ t_{3n}$, where each $t_i \in \{0,1\}$. Then, in order to commit to bit b, for $i = 1, \ldots, 3n$, the committer computes bit $c_i = t_i$ if $r_i = 0$ or bit $c_i = t_i \oplus b$ if $r_i = 1$. The commitment key is then string $com = c_1 \circ \cdots \circ c_{3n}$, and the decommitment key is $dec = s$. Then A sends the commitment key to B.

Decommitment phase. A sends the decommitment key to B. The receiver B, given R,com and s, performs the following test: If $com = G(s)$, B outputs 0; if $com = G(s) \oplus R$, B outputs 1; otherwise, B outputs $\perp$.

The analysis in [17] shows the two basic properties of this scheme: 1) a probabilistic polynomial time receiver breaking the (computational) security property of the scheme can be turned into a probabilistic polynomial algorithm which breaks the pseudo-random generator; 2) the probability (over the random choice of R) that an infinitely powerful committer can output a commitment key which can be decommitted both as 0 and as 1 is negligible.

Equivocability of the implementation in the public random string model. First of all, notice that the scheme in [17] can be executed in the public random string model, as follows. The step in which B sends the $3n$-bit random string $R = r_1 \circ \cdots \circ r_{3n}$ to A is replaced as follows: A just sets R equal to the first $3n$ bits from the public reference string. The remaining steps are the same as in the original scheme (see above description). We obtain:

Lemma 1 The implementation in the public random string model of the bit-commitment scheme in [17] results in an equivocable bit-commitment scheme.

Proof: We need to show an efficient simulator M, which on input 1^n, generates a 4-tuple $(\sigma', com', dec_0, dec_1)$ satisfying properties 1 and 2 of Definition 2.

The algorithm M. On input 1^n, M uniformly chooses two seeds $s_0, s_1 \in \{0,1\}^n$, and computes $u = G(s_0)$ and $v = G(s_1)$. Then it sets the faked random string as $\sigma' = R = u \oplus v$, the faked commitment key as $com' = u$, the decommitment key opening com' as b will be string $dec_b = s_b$, for $b = 0, 1$.

M can open both as 0 and as 1. Clearly, string s_0 is a valid decommitment key of the commitment key com' as 0. Now, to see that s_1 is a valid decommitment key of $com' = u$ as 1, we write strings R, u, v as $R = r_1 \circ \cdots \circ r_{3n}$, $u = u_1 \circ \cdots \circ u_{3n}$, and $v = v_1 \circ \cdots \circ v_{3n}$. From the construction of M, it holds that $r_i = u_i \oplus v_i$, for $i = 1, \ldots, 3n$, and therefore, in order to open R as 1, M has to present a seed s such that $t = G(s) = t_1 \circ \cdots \circ t_{3n}$, where $t_i = u_i$ if $r_i = 0$ and $t_i = u_i \oplus b$ if $r_i = 1$. Since $b = 1$, we obtain $t = u \oplus R = v$, and therefore $s = s_1$.

M's output is indistinguishable from a real execution. Let us recall the definition of the two random variables in Definition 2: $A_0 = \{\sigma \leftarrow \{0,1\}^{n^a}, (com, dec) \leftarrow A(\sigma, b) : (\sigma, com, dec)\}$, and $A_1 = \{(\sigma', com', dec_0, dec_1) \leftarrow M(1^n) : (\sigma', com', dec_b))\}$. Assume, for the sake of contradiction, that there exists a probabilistic polynomial time algorithm D, which distinguishes A_0 from A_1 with probability at least n^{-c}, for some constant c and infinitely many n. We show the existence of a probabilistic polynomial time algorithm E, which, using D as a subroutine, is able to distinguish the output of pseudo-random generator G from a totally random string with probability at least n^{-c}, for some constant c and infinitely many n. Algorithm E works as follows: on input a string y, it randomly chooses a seed $s \in \{0,1\}^n$ and sets $u = G(s)$ and $R = u \oplus y$. Now, it randomly chooses $v \leftarrow \{y, u\}$, and runs algorithm D on input (R, v, s). Algorithm D returns a bit c, denoting that it guesses that the triple (R, v, s) is distributed according to A_c. Finally, algorithm E outputs 'pseudo-random' if $c = 1$ and 'random' if $c = 0$. By observing that the triple (R, v, s) is distributed as A_0 if y is totally random or as A_1 if y is output by G, we derive that the probability that algorithm E distinguishes whether y is random or pseudo-random is the same as the probability that algorithm D distinguishes A_0 from A_1. ■

4 Non-interactive non-malleable bit-commitment

In this section we show a transformation from any non-interactive bit-commitment scheme to a non-interactive *non-malleable* bit-commitment scheme. The transformation is done in the common random string model and does not make use of any additional assumption. We obtain:

Theorem 2 In the common random string model, for any computationally secure non-interactive commitment scheme, it is possible to construct a computationally secure non-interactive non-malleable commitment scheme.

Using [15], we obtain as a corollary that there exists a non-interactive non-malleable commitment scheme under the minimal assumption of the existence of one-way functions. Moreover, our theorem extends to the case of perfect security. In

order to simplify the presentation, in this section we will only deal with bit-commitment, and explain the (non-trivial) extension to string commitment in Section 5. In Section 4.1 we describe our construction of the non-interactive non-malleable bit-commitment scheme, and in Section 4.2 we prove that our construction meets the requirements of Definition 3.

4.1 The construction

Now we have all the necessary tools to present our construction in the public random string model of a non-interactive non-malleable bit-commitment scheme. We show a transformation which, given a non-interactive equivocable bit-commitment scheme, returns a non-interactive non-malleable bit-commitment scheme. We obtain:

Lemma 2 In the public random string model, given a non-interactive bit-commitment scheme (A,B) and a non-interactive equivocable bit-commitment scheme (C,D), it is possible to construct a non-interactive non-malleable bit-commitment scheme (Alice,Bob). Furthermore, if (C,D) is computationally secure (resp., perfectly secure) then (Alice,Bob) is also computationally secure (resp., perfectly secure).

Clearly, the results in Lemma 2 and Theorem 1 are enough to prove the result in Theorem 2. We start with an informal description of the ideas behind the transformation and then present a formal description of scheme (Alice,Bob) and a proof for Lemma 2.

An informal discussion. Intuitively it might seem that the security property of a bit-commitment scheme is enough to guarantee that an adversary observing a commitment key com_1 to a bit b is not able to efficiently compute a commitment key com_2 to, say, the same bit, with some sufficiently high probability. In fact, this is not the case, since the adversary, by looking at com_1, can come up with a commitment key com_2 for which he knows the associated decommitment key dec_2 only after he sees the decommitment key dec_1 associated to com_1. A key idea in our construction is to overcome this situation by constructing a scheme such that the commitment key com_1 does not contain any 'useful' information to the adversary. One way we achieve this in our scheme is as follows: first we simulate the execution of the first commitment protocol and produce a commitment key com'_1, two decommitment keys dec'_0, dec'_1 and a public reference string τ' such that 1) for each $b = 0, 1$, the triple (τ', com', dec'_b) is computationally indistinguishable from the triple (τ, com, dec) which is seen by the receiver in an execution of a commitment to bit b of the real protocol; 2) for each $b = 0, 1$, the decommitment key dec'_b is a valid decommitment key as bit b for the commitment key com'. Notice that these are precisely the properties of equivocable bit commitment schemes, which we know how to construct from any bit-commitment schemes, as shown in Theorem 1. Now, assume that an efficient adversary $\mathcal{A}$, after seeing τ' and a commitment com' to some bit b, is able to compute a commitment com_2 to a bit d such that $R(b, d) = 1$, for some relation approximator R. Also, assume that for such R, there is no adversary simulator $\mathcal{A}'$ which closely simulates $\mathcal{A}$ when committing to a bit. We observe that the above mentioned property 2) guarantees that the adversary $\mathcal{A}$ can derive no 'useful' information when he receives the commitment key com' and, later, any among the two possible decommitment keys. In order to simplify the discussion, consider, as an example, the case in which the algorithm R outputs 1 on input bits b, d if and only if $b = d$. Then, if $\mathcal{A}$ succeeds with high probability in committing to d such that $R(b, d) = 1$, then he succeeds in "copying" the bit committed by com'. However, notice that this is impossible, since in the simulated commitment key com', the bit b can be opened both as 0 and as 1 *after* $\mathcal{A}$ commits to d, and therefore the adversary $\mathcal{A}$ would be able to open bit d both as 0 and as 1 as well. Now, we would like to use this fact to contradict the binding property of the original bit-commitment scheme. The above fact alone, however, is not enough to contradict the binding property. The reason for this is that the adversary is able to open a commitment in two ways, given a *not* totally random public reference string τ'. Instead, the binding property says that an efficient committer cannot open a commitment in two ways, given a *totally random* string. We will overcome this problem with another modification of our commitment scheme: instead of running a single execution of the commitment scheme, we will run many executions of commitments to the same bit, each on a different portion of the public reference string. In particular, the portions will be chosen in such a way that with high probability the adversary will be forced to choose at least one portion which was left unused by the honest committer. We achieve this using the following authentication procedure. Specifically, the committer chooses the seed for a key of an authentication scheme, and commits to it using an ordinary non-interactive commitment scheme. Then, the bits of this committed key are used to determine the portions of the reference string on which the equivocable bit-commitment scheme will be used. Finally, the authentication key is used to "seal" all commitments output by the equivocable scheme, giving the following property: either (a) the adversary entirely copies the commitment to the seed for the authentication key, or (b) he will run an execution of the equivocable commitment scheme using a portion of the reference string which was not used by the committer. By the use of authentication, (a) will happens only with negligible probability, unless the entire commitment is copied. On the other hand, if (b) happens, then the above properties 1) and 2) are enough to show that if the scheme is not non-malleable then we can contradict the binding property of the scheme itself. This gives an intuition on how a proof would work for the mentioned specific example of a relation approximator R. Later, in the proof for our scheme, we deal with the more general case of any relation approximator R; in one case we will exhibit an adversary simulator $\mathcal{A}'$ which closely simulates the adversary $\mathcal{A}$ for any R. A formal description of our scheme is in Figure 1.

4.2 Sketch of proof of Lemma 2

The meaningfulness, security and binding properties of the above scheme (Alice,Bob) follow directly from the same properties of the bit-commitment scheme (C,D). We now turn to proving non-malleability.

Let us assume for the sake of contradiction that (Alice,Bob) is malleable (i.e., not non-malleable). This means that there exists a relation approximator R and an efficient adversary $\mathcal{A}$ such that for all adversary simulators $\mathcal{A}'$, the difference $p(\mathcal{A}, R) - p'(\mathcal{A}', R)$ is noticeable.

The first step of the proof consists in constructing an algorithm Q which will play the role of the committer but will run a modified version of algorithm Alice having the following properties: (1) the output of algorithm Q is computationally indistinguishable from the output of algorithm Alice; and (2) the commitment key output by algorithm Q can be opened in two ways. Now, we will show that if there exists an efficient algorithm $\mathcal{A}$ contradicting the non-malleability property of (Alice,Bob), then this algorithm will either distinguish the output of algorithm Q from the output of algorithm Alice (which contradicts property (1)) or, using property (2), be able to output a commitment to a bit which does not depend from the commitment made by algorithm Q (this implies that there exists an $\mathcal{A}'$ who can simulate some behaviour of adversary $\mathcal{A}$).

The algorithm Q. Now we formally describe algorithm Q.

Input to Q:

- A security parameter 1^n;
- a non-interactive bit-commitment scheme (A,B);
- a non-interactive equivocable bit-commitment scheme (C,D).
- a pseudo-random generator G.

Instructions for Q.

Q.1 (*Simulate commitment to the authentication key.*)
Uniformly choose a seed $s \in \{0,1\}^n$;
let $s_1 \circ \cdots \circ s_n$ be its binary expansion;

for $i = 1, \ldots, 2n$,
uniformly choose string α_i;
run algorithm A on input (α_i, s_i),
let $(A\text{-}com_i, A\text{-}dec_i)$ be its output;
set $A\text{-}com = A\text{-}com_1 \circ \cdots \circ A\text{-}com_n$;
let $d_1 \circ \cdots \circ d_m$ be its binary expansion;
for $j = 1, \ldots, m$,
run algorithm M on input 1^n;
let $(\tau_j, C\text{-}com_j, C\text{-}dec_{j,0}, C\text{-}dec_{j,1})$ be its output;
set $\beta_{j,d_j} = \tau_j$ and uniformly choose $\beta_{j,1-d_j}$;
set $\beta = \beta_{1,0} \circ \beta_{1,1} \circ \cdots \circ \beta_{m,0} \circ \beta_{m,1}$;
set $\sigma' = \alpha_1 \circ \cdots \circ \alpha_n \circ \beta$;

Q.2 (*Simulate authentication phase.*)
set $C\text{-}com = C\text{-}com_1 \circ \cdots \circ C\text{-}com_m$ and $q = 2^{|C-com|}$;
compute $G(s) = a \circ b$, for $a, b \in \{0,1\}^m$;
compute $tag' = a \cdot (C\text{-}com) + b$ (over $GF(q)$);
let $Q\text{-}com' = (A\text{-}com, C\text{-}com, tag)$;
let $A\text{-}dec = A\text{-}dec_1 \circ \cdots \circ A\text{-}dec_n$;
let $C\text{-}dec_i = C\text{-}dec_{1,i} \circ \cdots \circ C\text{-}dec_{m,i}$, for $i = 0, 1$;
set $Q\text{-}dec'_i = (A\text{-}dec, C\text{-}dec_i)$, for $i = 0, 1$;

Q.3 (*Output in the commitment phase.*)
Output: $(\sigma', Q\text{-}com')$.

Q.4 (*Output in the decommitment phase.*)
For $b = 0, 1$, in order to decommit string com as b output: $Q\text{-}dec'_b$.

In the following two lemmas, we show that algorithm Q satisfies the above discussed two properties (1) and (2). The first property says that algorithm Q outputs a commitment key for which he can provide two decommitment keys, one opening it as 0 and the other as 1. Its proof follows directly from the Property 1 of equivocable commitment schemes. The second property of algorithm Q says that the output of algorithm Q is indistinguishable from a real execution of the protocol.

Lemma 3 Let (C,D) be an equivocable bit commitment scheme. Then the output of algorithm Q satisfies the following. For each $j = 1, \ldots, m$, and $c = 0, 1$, it holds that $\text{Bob}(\beta_{j,d_j}, Alice\text{-}com_j, Alice\text{-}dec_{j,c}) = c$.

Lemma 4 Let (A,B) be a commitment scheme, and (C,D) be an equivocable commitment scheme. Also, let us denote by $V_0 = (\sigma, Alice\text{-}com, Alice\text{-}dec)$ the view of algorithm Bob when receiving messages from algorithm Alice in the commitment and decommitment phase, where the input to Alice is $(1^n, b)$. Similarly, let us denote by $V_1 = (\sigma', Q\text{-}com', Q\text{-}dec'_b)$ the view of algorithm Bob when receiving messages from algorithm Q in the commitment and decommitment phase, where the input to Q is (1^n). If the scheme (C,D) is computationally equivocable then V_0 and V_1 are computationally indistinguishable.

Proof: Let us compare the distribution of strings $(\sigma, Alice\text{-}com, Alice\text{-}dec)$ and $(\sigma', Q\text{-}com', Q\text{-}dec'_b)$, sent by algorithm Alice on input b, and algorithm Q, respectively. Notice that we can write $\sigma = (\alpha_1, \ldots, \alpha_n, \beta)$, $Alice\text{-}com = (A\text{-}com, C\text{-}com, tag)$, $\sigma' = (\alpha'_1, \ldots, \alpha'_n, \beta')$(and $Q\text{-}com' = (A\text{-}com', C\text{-}com', tag')$. We see that the probability distribution of the triple $((\alpha_1, \ldots, \alpha_n), A\text{-}com, tag)$ conditioned by the value of the triple $(\beta, C\text{-}com, C\text{-}dec)$ is the same as the distribution of the triple $((\alpha'_1, \ldots, \alpha'_n), A\text{-}com', tag')$ conditioned by the value of the triple $(\beta', C\text{-}com', C\text{-}dec')$. Namely, the triple $((\alpha_1, \ldots, \alpha_n), A\text{-}com, tag)$ is computed as follows: $A\text{-}com$ is a commitment to a randomly chosen seed s, using $\alpha_1, \ldots, \alpha_n$ as public random strings, and tag is a valid authentication of string $C\text{-}com$ using the key (a, b) obtained as $G(s) = a \circ b$. The same is true for triple $((\alpha'_1, \ldots, \alpha'_n), A\text{-}com', tag')$, conditioned by the value of the triple $(\beta', C\text{-}com', C\text{-}dec')$. Then the only difference in Bob's view in the two cases might be between the triples $(\beta, C\text{-}com, C\text{-}dec)$ and $(\beta', C\text{-}com', C\text{-}dec')$. Here, notice that the first triple is a transcript of an execution of the equivocable commitment scheme (C,D), and the second triple is the output of the simulator M of such scheme. Therefore, the two triples are computationally indistinguishable by Property 2 of equivocable commitment schemes if (C,D) is computationally equivocable. ■

Now we use algorithm Q to show that the assumption that the scheme (Alice,Bob) is malleable brings us to a contradiction. First of all, define probability $q(\mathcal{A}, R)$ as

$$
\begin{aligned}
q(\mathcal{A}, R) = \mathbf{Pr}[\quad & (\sigma, com, dec_0, dec_1) \leftarrow Q; \\
& com' \leftarrow \mathcal{A}(\sigma, com); \\
& b \leftarrow \mathcal{D}; dec' \leftarrow \mathcal{A}(\sigma, com, com', dec_b); \\
& Bob(\sigma, com, dec_b) = b \wedge \\
& Bob(\sigma, com', dec') = d \wedge \\
& com' \neq com \wedge R(b, d) = 1 \quad].
\end{aligned}
$$

Intuitively, q measures the probability that $\mathcal{A}$ succeeds when facing the simulator Q.
Recall that by our contradiction assumption, there exist a relation approximator R and an efficient algorithm $\mathcal{A}$ such that for all adversary simulators $\mathcal{A}'$, the difference $p(\mathcal{A}, R) - p'(\mathcal{A}', R)$ is at least n^{-c}, for some constant c and infinitely many n. Now, since we can write $p(\mathcal{A}, R) - p'(\mathcal{A}', R) = (p(\mathcal{A}, R) - q(\mathcal{A}, R)) + (q(\mathcal{A}, R) - p'(\mathcal{A}', R))$, we have that at least one of the two differences $(p(\mathcal{A}, R) - q(\mathcal{A}, R))$ and $(q(\mathcal{A}, R) - p'(\mathcal{A}', R))$ is at least n^{-c}, for some constant c and infinitely many n. We then derive two cases which we analyze in the rest of the proof.

Case 1. In the first case we assume that there exist a relation approximator R and an efficient algorithm $\mathcal{A}$ such that the difference $p(\mathcal{A}, R) - q(\mathcal{A}, R)$ is at least n^{-c} for infinitely many n and some constant c. Now, consider the definitions of the two random experiments involved in probabilities $p(\mathcal{A}, R)$ and $q(\mathcal{A}, R)$. We see that the only difference is that the first experiment uses algorithm *Alice*, while the second one uses algorithm Q. Therefore algorithm $\mathcal{A}$ can be used to efficiently distinguish the view of Bob when receiving messages from algorithm Q from the view of Bob when receiving messages from algorithm Alice in the commitment and decommitment phase with

Input to Alice and Bob:

- A security parameter 1^n;
- an n^a-bit reference string σ, for some constant a;
- a non-interactive bit-commitment scheme (A,B);
- a non-interactive equivocable bit-commitment scheme (C,D).
- a pseudo-random generator G;

Input to Alice: A bit b.

Instructions for Alice:

A.1 (*Commitment to a seed for an authentication key.*)
Write σ as $\sigma = \alpha_1 \circ \cdots \circ \alpha_n \circ \beta$;
uniformly choose a seed $s \in \{0,1\}^n$;
let $s_1 \circ \cdots \circ s_n$ be its binary expansion;
for $i = 1, \ldots, n$,
run algorithm A on input (α_i, s_i), and let $(A\text{-}com_i, A\text{-}dec_i)$ be its output;
set $A\text{-}com = A\text{-}com_1 \circ \cdots \circ A\text{-}com_n$ and let $d_1 \circ \cdots \circ d_m$ be its binary expansion;
write β as $\beta = \beta_{1,0} \circ \beta_{1,1} \circ \cdots \circ \beta_{m,0} \circ \beta_{m,1}$.

A.2 (*Bit commitment and commitment authentication.*)
For $j = 1, \ldots, m$,
run algorithm C on input (β_{j,d_j}, b), and let $(C\text{-}com_j, C\text{-}dec_j)$ be its output;
set $C\text{-}com = C\text{-}com_1 \circ \cdots \circ C\text{-}com_m$;
set $q = 2^{|C\text{-}com|}$ and $z = G(s)$;
write z as $z = a \circ b$, where $a, b \in GF(q)$;
compute $tag = a \cdot (C\text{-}com) + b$ (over $GF(q)$).

A.3 (*Output.*)
Let $Alice\text{-}com = (A\text{-}com, C\text{-}com, tag)$;
set $A\text{-}dec = A\text{-}dec_1 \circ \cdots \circ A\text{-}dec_n$ and $C\text{-}dec = C\text{-}dec_1 \circ \cdots \circ C\text{-}dec_m$;
set $Alice\text{-}dec = (A\text{-}dec, C\text{-}dec)$;
output $(Alice\text{-}com, Alice\text{-}dec)$.

Input to Bob: $Alice\text{-}com = (A\text{-}com, C\text{-}com, tag)$, $Alice\text{-}dec = (A\text{-}dec, C\text{-}dec)$;

Instructions for Bob:

B.1 (*Verify the correctness of the decommitment.*)
For $i = 1, \ldots, n$,
verify that $\mathrm{B}(\alpha_i, A\text{-}com_i, A\text{-}dec_i) \neq \perp$;
for $i = 1, \ldots, n$,
let $s_i = \mathrm{B}(\alpha_i, A\text{-}com_i, A\text{-}dec_i)$, and let $s = s_1 \circ \cdots \circ s_n$.
let $d_1 \circ \cdots \circ d_m$ be the binary expansion of $A\text{-}com$.
verify that there exists $b \in \{0,1\}$ such that
$\mathrm{D}(\beta_{j,d_j}, C\text{-}com_j, C\text{-}dec_j) = b$ for $j = 1, \ldots, m$.
set $q = 2^{|C\text{-}com|}$ and $z = G(s)$;
write z as $z = a \circ b$, where $a, b \in GF(q)$;
verify that $tag = a \cdot (C\text{-}com) + b$ (over $GF(q)$);

B.2 (*Output.*)
If any verification is not satisfied then output $\perp$ and halt
else output the bit b.

Figure 1: The non-interactive non-malleable commitment scheme (Alice,Bob).

probability n^{-c_0}, for some related constant c_0. This contradicts Lemma 4.

Case 2. In the second case, assume that there exist a relation approximator R and an efficient algorithm $\mathcal{A}$ such that for all adversary simulators $\mathcal{A}'$, the difference $q(\mathcal{A}, R) - p'(\mathcal{A}', R)$ is at least n^{-c} for infinitely many n and some constant c. We distinguish two sub-cases, according to whether the strings $A\text{-}com$ and $A\text{-}com'$ contained in the commitment keys by Alice and $\mathcal{A}$, respectively, have equal or different binary expansion.

Consider first the case $A\text{-}com = A\text{-}com'$. We would like to show that this case happens only with negligible probability or some contradiction is derived. Also, from our previous assumption we derive that $q(\mathcal{A}, R)$ is at least n^{-c} for infinitely many n and some constant c. To see this, first notice that in this case $\mathcal{A}$ is copying the commitment to the seed s for the authentication key (a, b) without knowing the seed s itself. Then, by a standard hybrid argument, either $\mathcal{A}$ is able to break the security property of (A,B), or he is able to distinguish a pseudo-random string (a, b) from a totally random one, or $\mathcal{A}$ will be able to provide a string tag' such that $(tag', C\text{-}com') \neq (tag, C\text{-}com)$ and $tag' = a \cdot (C\text{-}com') + b$ (by the properties of the authentication scheme the latter can happen only with negligible probability).[1]

Now, consider the case $A\text{-}com \neq A\text{-}com'$. For $b = 0, 1$, define probability q_b as

$$q_b = \mathbf{Pr}[\ (\sigma, com, dec_0, dec_1) \leftarrow Q; com' \leftarrow \mathcal{A}(\sigma, com); dec'_b \leftarrow \mathcal{A}(\sigma, com, com', dec_b) : Bob(\sigma, com, dec_b) = b \wedge Bob(\sigma, com', dec'_b) = d_b \wedge com' \neq com \wedge R(b, d_b) = 1 \ ].$$

Namely, q_b is the probability $q(\mathcal{A}, R)$ conditioned by the fact that distribution $\mathcal{D}$ has output bit b. Therefore we can write $q(\mathcal{A}, R) = Pr(0 \leftarrow \mathcal{D}) \cdot q_0 + Pr(1 \leftarrow \mathcal{D}) \cdot q_1$. Now, notice that Lemma 3 implies that the two strings dec_0, dec_1 that are output by algorithm Q satisfy $Bob(\sigma, com, dec_0) = 0$ and $Bob(\sigma, com, dec_1) = 1$. Moreover, we claim that the probability that $d_0 \neq d_1$ is negligible. To see that the latter claim is true, assume by contradiction that this is not the case. Then observe that since $A\text{-}com \neq A\text{-}com'$, by the construction of our scheme (Alice,Bob) (specifically, the authentication phase), Alice and $\mathcal{A}$ will choose at least one different portion α_{j,d_j} of the public random string σ on which to run the equivocable commitment scheme. Then this implies that Alice will use all the portions α_{j,d_j} of string σ prepared by algorithm Q (which are all distributed according to some pseudo-random distribution), while algorithm $\mathcal{A}$ will use at least one out of all the remaining portions (which are all distributed according to the uniform distribution). Therefore algorithm $\mathcal{A}$ will run at least one execution of the commitment scheme (C,D) on a truly uniformly distributed string. Moreover, by our assumption that $d_0 \neq d_1$ we obtain that with probability at least n^{-c}, for some constant c, algorithm $\mathcal{A}$ is able to compute a commitment key com' and later decommit com' both as a 0 and as a 1. This clearly contradicts the binding property of the bit-commitment scheme (C,D).

Now, using that $Bob(\sigma, com, dec_b) = b$ for $b = 0, 1$, that $d_0 \neq d_1$ with negligible probability, and that $q(\mathcal{A}, R) = Pr(0 \leftarrow \mathcal{D}) \cdot q_0 + Pr(1 \leftarrow \mathcal{D}) \cdot q_1$, we obtain that $q(\mathcal{A}, R) - q_0$ and $q(\mathcal{A}, R) - q_1$ are negligible. Furthermore, with probability at least $1 - n^{-c}$, for all constants c, it holds that $R(0, d_0) = R(1, d_1) = 1$. Intuitively, this suggests that the a-posteriori experiment played by $\mathcal{A}$ can be the same as the a-priori experiment played by $\mathcal{A}'$. In fact, we can define distribution $\mathcal{D}'$ over $\{0, 1\}$ as:

$$[(\sigma, com, dec_0, dec_1) \leftarrow Q; com' \leftarrow \mathcal{A}(\sigma, com); dec'_0 \leftarrow \mathcal{A}(\sigma, com, com', dec_0) : Bob(\sigma, com', dec'_0)\ ].$$

Notice that since algorithms Q and $\mathcal{A}$ are efficient, distribution $\mathcal{D}'$ is efficiently samplable and therefore algorithm $\mathcal{A}'$ is also efficient. Now, since $d_0 \neq d_1$ with negligible probability and $R(0, d_0) = R(1, d_1) = 1$, the probability that distribution $\mathcal{D}'$ returns bit d is exactly equal to d_0, since the random processes in the definition of $\mathcal{D}'$ turn out to coincide to those in the definition of q_0 (we could have similarly obtained q_1 here). Therefore it holds that $\Pr(d \leftarrow \mathcal{D}') = p'(\mathcal{A}', R) = q_0$, and, using the fact that $q(\mathcal{A}, R) - q_0$ is negligible, we obtain that $q(\mathcal{A}, R) - p'(\mathcal{A}', R)$ is also negligible. This negates our contradiction assumption in this case.

[1] The specific authentication scheme "$tag = aM + b$" is used for reasons of concreteness; it can be replaced by other authentication schemes.

5 Non-interactive Non-Malleable String Commitment

Repetition does not preserve non-malleability. As already remarked, given a secure bit-commitment scheme, it is possible to obtain a string commitment scheme by repeating independent executions of the original bit-commitment scheme. It should be observed that this transformation does *not* preserve the non-malleability property. Let us try to get convinced that this is indeed the case. First, let (A,B) be a bit-commitment scheme, consider a string s of two bits $s_0.s_1$, and a 2-bit-commitment scheme (C,D) constructed as a double repetition of (A,B), each having as input a different bit of s. We see that even if (A,B) is non-malleable then (C,D) is malleable. Specifically, consider the algorithm C' that, after seeing the commitment $com_0.com_1$ to s_0, s_1 made by C, outputs $com_1.com_0$, namely, he just swaps the two single bit-commitment keys. Clearly, string $s_1.s_0$ is 'related' to the original string s, and therefore algorithm C' shows that scheme (C,D) is not non-malleable. Notice that our reasoning does not depend on whether we are in the interactive or non-interactive setting.

A properly modified repetition preserves non-malleability. We now show a non-interactive non-malleable string commitment scheme, obtained by a careful repetition of our non-interactive non-malleable bit commitment scheme. Let (A,B) be an (ordinary) bit-commitment scheme and let (C,D) be an equivocable bit-commitment scheme. A non-malleable string commitment scheme (Alice,Bob) can be constructed by properly modifying the scheme obtained as an independent repetition of scheme (C,D). The high-level idea of the modification consists in using the commitment authentication technique as done in the scheme in Section 4. In particular, since the committer will authenticate the commitment key of all commitments to the string, then the above swapping attack is not possible any more. We postpone details and proof to the full version.

Acknowledgment

We wish to thank Moni Naor for his remarks.

References

[1] M. Beaver, *Adaptive Zero-Knowledge and Computational Equivocation*, in Proc. of FOCS 96.

[2] M. Bellare, R. Impagliazzo and M. Naor, *Does Parallel Repetition Lower the Error in Computationally Sound Protocols ?*, in Proc. of FOCS 97.

[3] M. Bellare and P. Rogaway, *Random Oracles are Practical: A paradigm for Designing Efficient Protocols*, in Proc. of ACM Conference on Computer and Communication Security, 1993.

[4] M. Blum, A. De Santis, S. Micali, and G. Persiano, *Noninteractive Zero-Knowledge*, in SIAM Journal of Computing, vol. 20, no. 6, Dec. 1991, pp. 1084–1118.

[5] M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge and its Applications*, in Proc. of STOC 88.

[6] D. Dolev, C. Dwork, and M. Naor, *Non-Malleable Cryptography*, in Proc. of STOC 91.

[7] A. De Santis and G. Persiano, *Zero-Knowledge Proofs of Knowledge Without Interaction*, in Proc. of FOCS 92.

[8] S. Even, O. Goldreich, and A. Lempel, *A Randomized Protocol for Signing Contracts*, in Communications of the ACM, vol. 28, No. 6, 1985, pp. 637-647.

[9] R. Gennaro, *Achieving Independence Efficiently and Securely*, in Proc. of PODC 95.

[10] S. Goldreich and L. Levin, *A Hard-Core Predicate for all One-Way Functions*, in Proc. of FOCS 89.

[11] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems*, in Journal of the ACM, vol. 38, n. 1, 1991, pp. 691–729.

[12] O. Goldreich, S. Micali, and A. Wigderson, *How to Play Any Mental Game*, in Proc. of 19th STOC, 1987, pp. 218-229.

[13] S. Goldwasser and S. Micali, *Probabilistic Encryption*, in Journal of Computer and System Sciences, vol. 28, n. 2, 1984, pp. 270–299.

[14] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of interactive Proof-Systems*, in SIAM Journal on Computing, vol. 18, n. 1, February 1989.

[15] J. Hastad, R. Impagliazzo, L. Levin and M. Luby, *Pseudo-Random Generation from any One-way Function*, to appear on SIAM Journal on Computing (previous versions in Proc. of FOCS 89 and STOC 90).

[16] R. Impagliazzo and M. Luby, *One-way Function are Essential for Complexity-Based Cryptography*, in Proc. of FOCS 89.

[17] M. Naor, *Bit Commitment using Pseudo-Randomness*, in Proc. of CRYPTO 89.

[18] M. Naor, R. Ostrovsky, R. Venkatesan and M. Yung, *Perfect Zero-Knowledge Arguments can be based under General Complexity Assumptions*, in Proc. of CRYPTO 92.

[19] M. Naor and M. Yung, *Public-Key Cryptosystems Secure against Chosen Ciphertext Attack*, in Proc. of STOC 90.

[20] C. Rackoff and D. Simon, *Non-Interactive Zero-Knowledge Proofs of Knowledge and Chosen-Ciphertext Attack*, in Proc. of CRYPTO 91.

[21] A. Yao, *Theory and Applications of Trapdoor Functions*, in Proc. of FOCS 82.

A Malleability of some known commitment schemes

We observe that for many secure bit-commitment schemes presented in the literature, it can be provably seen that the non-malleability property is not satisfied.

Commitments based on random-self-reducible problems.
Typically, bit-commitment schemes that are constructed using random-self-reducible languages as quadratic residuosity and discrete log, are of the following form. The committer sends a string y which belongs to the language if he wants to commit to a 1, or does not, if he wants to commit to a 0. The random-self-reducibility property allows an attacker, who is given y, to obtain a string y' such that y' is in the language if and only if y is. Then y' is a commitment to the same bit as y is.

Commitment based on one-way permutations.
A well-known bit-commitment scheme using any one-way permutation can be constructed using the result in [10], as follows. Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a one-way permutation; in order to commit to a bit b, the committer randomly chooses two string $x, r \in \{0,1\}^n$ such that $x \odot r = b$ (where $\odot$ denotes inner product betweeen strings), and outputs the commitment key $com = (f(x), r)$. The decommitment key is string x. Here, we observe that this scheme is malleable.

Claim 1 There exists a one-way permutation g such that the above bit-commitment scheme, based on g, is malleable.

Proof: For any one-way permutation $f : \{0,1\}^n \rightarrow \{0,1\}^n$, consider the one-way permutation $g : \{0,1\}^{n+1} \rightarrow \{0,1\}^{n+1}$ such that $g(x) = f(x') \circ c$, where $x = x_1 \circ \cdots \circ x_{n+1}$, $x' = x_1 \circ \cdots \circ x_n$, and $x_{n+1} = c$. Clearly, if f is a one-way permutation then so is g. Now, let $com = (y, r)$, where $y = y_1 \circ \cdots \circ y_{n+1}$ and $r = r_1 \circ \cdots \circ r_{n+1}$, be a commitment key to a bit b using the above scheme, based on the one-way permutation g. An attacker can commit to bit $d = b$ by sending $com' = (y', r') \neq com$, for $y' = y_1 \circ \cdots \circ y_n \circ c'$, and $r' = r_1 \circ \cdots \circ r_n \circ r'_{n+1}$, where the pair (c', r'_{n+1}) is chosen so that $c' \wedge r'_{n+1} = c \wedge r_{n+1}$, and $(c', r'_{n+1}) \neq (c, r_{n+1})$. Later, when he sees the decommitment key $dec = x = x_1 \circ \cdots \circ x_n \circ b$, he can return a valid decommitment key $dec = x = x_1 \circ \cdots \circ x_n \circ c'$ for com. ■

Commitment based on pseudo-random generators.
The bit-commitment scheme in [17] is based on pseudo-random generators (see Section 3 for a description of the scheme). Here, we observe that this scheme is malleable.

Claim 2 The bit-commitment scheme in [17] is malleable.

Proof: Given a random string R and a commitment com to a bit b, an attacker can commit to bit $1 - b$ by sending the commitment $com' = com \oplus R$. The decommitment key dec opening commitment key com as b also opens commitment key com' as $1 - b$. ■