Efficient and Non-Interactive Non-Malleable Commitment

Giovanni Di Crescenzo[1] Jonathan Katz[2] Rafail Ostrovsky[1]
Adam Smith[3]

[1] Telcordia Technologies, Inc.
{giovanni,rafail}@research.telcordia.com
[2] Telcordia Technologies and
Department of Computer Science, Columbia University.
jkatz@cs.columbia.edu
[3] Laboratory for Computer Science, MIT.
Work done while the author was at Telcordia Technologies.
asmith@theory.lcs.mit.edu

Abstract. We present new constructions of non-malleable commitment schemes, in the public parameter model (where a trusted party makes parameters available to all parties), based on the discrete logarithm or RSA assumptions. The main features of our schemes are: they achieve *near-optimal communication* for arbitrarily-large messages and are *non-interactive*. Previous schemes either required (several rounds of) interaction or focused on achieving non-malleable commitment based on general assumptions and were thus efficient only when committing to a single bit. Although our main constructions are for the case of perfectly-hiding commitment, we also present a communication-efficient, non-interactive commitment scheme (based on general assumptions) that is perfectly binding.

1 Introduction

Commitment protocols are one of the most fundamental cryptographic primitives, used as sub-protocols in such applications as zero-knowledge proofs (see Goldreich, Micali, and Wigderson [17] and Goldreich [15]), secure multi-party computation (see Goldreich, Micali, and Wigderson [16]), contract signing (see Even, Goldreich, and Lempel [13]), and many others. Commitment protocols can also be used directly; for example, in remote (electronic) bidding. In this setting, parties bid by committing to a value; once bidding is complete, parties reveal their bids by de-committing. In many of these settings, it is required that participants, upon viewing the commitment of one party, be unable to generate a commitment to a related value. For example, in the bidding scenario it is unacceptable if one party can generate a valid commitment to $x + 1$ upon viewing a commitment to x. Note that the value of the original commitment may remain unknown (and thus secrecy need not be violated); in fact, the second party may

[1] This is an extended version of a paper which appears in Eurocrypt 2001.

only be able to decommit his bid after viewing a decommitment of the first. Unfortunately, most known commitment protocols are easily susceptible to these types of attacks.

Two types of commitment schemes have been considered in the literature: perfectly-binding [19] and perfectly-hiding [21] (following [15] we refer to the former as *standard* and the latter as *perfect*). In a standard commitment scheme, each commitment is information-theoretically bound to only one possible (legal) decommitment value; on the other hand, the secrecy of the commitment is guaranteed only with respect to a computationally-bounded receiver. In a perfect commitment scheme, the secrecy of the commitment is information-theoretic, while the binding property guarantees only that a computationally-bounded sender cannot find a commitment which can be opened in two possible ways. The type of commitment scheme to be used depends on the application [15]; it may also depend on assumptions regarding the computational power of the participants. For example, in many protocols certain commitments are never opened; information-theoretic privacy ensures that the committed data will remain hidden indefinitely (for further discussion, see [23, 21]).

Commitment size is an important parameter, particularly when committing to a very large message such as the contents of a database. Unfortunately, *standard* commitment schemes (even malleable ones) require commitment size at least $M + \omega(\log k)$, where M is the message size and k is the security parameter. *Perfect* commitment schemes, on the other hand, offer the opportunity to achieve much shorter commitment lengths. Indeed, the non-malleable, perfect commitment schemes presented here achieve commitment size only $3k$ for arbitrarily-large messages.

Previous Work. Non-malleability was first explicitly considered by Dolev, Dwork, and Naor [11], who define the notion in a number of different settings. They also provide the first construction of a standard commitment scheme which is provably non-malleable. Although their protocol is constructed from the minimal assumption of a one-way function (in particular, without assuming a public random string), it requires a non-constant number of rounds of interaction[2]. Assuming a public random string available to all participants, Di Crescenzo, Ishai, and Ostrovsky [9] construct a *non-interactive*, non-malleable standard commitment scheme. Interestingly, their construction can be modified to give a non-interactive, non-malleable *perfect* commitment scheme. Unfortunately, the resulting commitments are large (i.e., $\mathcal{O}(Mk)$), thus motivating the search for more efficient protocols.

Constructions of non-malleable public-key encryption schemes have also been proposed [11, 6, 25]. In some cases, these constructions give non-malleable standard commitment schemes, in the model where public parameters are published by a trusted party. We discuss this connection in more detail in Section 3.

[2] Furthermore, their protocol allows an adversary to generate a different commitment to an identical value (unless user identities are assumed). Other protocols discussed in this paper (including our own) do not suffer from this drawback.

Two efficient non-malleable commitment schemes, based on stronger (but standard) assumptions, have also been proposed. Like the construction of [9], these protocols both require publicly-available parameters generated by a trusted party (in some cases this can be reduced to the assumption of a public random string). The first can be obtained from an adaptive chosen-ciphertext secure public-key encryption scheme proposed by Cramer and Shoup [6], whose security is based on the decisional Diffie-Hellman problem. More recently, non-malleable perfect commitment schemes based on the discrete logarithm and RSA assumptions were introduced by Fischlin and Fischlin [14]. Though efficient, these protocols require interaction between the sender and receiver.

Our Contribution. We present the first efficient constructions of non-interactive, non-malleable perfect commitment schemes. We work in the same setting as other efficient non-malleable commitment schemes, where public parameters are available to all participants [6, 14] (our discrete logarithm construction can be implemented in the public random string model using standard techniques). Our constructions are based on the discrete logarithm or the RSA assumptions. Previous constructions are either for the case of standard commitment [11, 9, 6] or require interaction [11, 14]. Our constructions allow efficient, perfectly-hiding commitment to arbitrarily-large messages. The schemes described in [14], while able to handle large messages, require modifications which render them less efficient and also result in statistical secrecy only.

Additionally, we discuss the case of non-interactive, non-malleable, standard commitment schemes and prove secure a folklore construction based on trapdoor permutations which is near-optimal in terms of commitment size. The large commitment size of this construction (though near-optimal) serves as motivation for our consideration of perfect commitment schemes. Indeed, for arbitrarily-large messages, our perfect commitment schemes require commitments of size $3k$, where k is the size of RSA or discrete log problems believed to be hard to solve (see Section 5 for improvements which reduce the commitment size even further). Our schemes require only $\mathcal{O}(k)$ bits of public information.

2 Definitions

We discuss the communication models in which we present our constructions, and recall the notions of commitment schemes, equivocable commitment schemes, and finally non-malleable commitment schemes.

Communication models. We will consider two models: the *public-random-string* model of [4, 3], and a slight generalization of it, considered for instance by [14] in the context of commitment schemes, which we call the *public-parameter* model.

The former model was 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. The latter model generalizes the public random string model in the following sense: all parties still share a

public reference string which is now defined as the output of an efficient algorithm (and may therefore have arbitrary distribution).

For a unified treatment, we present our definitions for the public-parameter model, keeping in mind that analogous definitions may be obtained for the public random string model if the algorithm generating the public reference string is replaced by an algorithm which chooses a uniformly distributed string.

Commitment schemes. A *commitment scheme* $(\mathcal{TTP}, \mathcal{S}, \mathcal{R})$ in the public-parameter model is a two-phase protocol between two probabilistic polynomial time parties $\mathcal{S}$ and $\mathcal{R}$, called the sender and the receiver, respectively, such that the following is true. In the first phase (the commitment phase), given the public reference string σ returned by the probabilistic polynomial time algorithm $\mathcal{TTP}$, $\mathcal{S}$ commits to bit b by computing a pair of keys (com, dec) and sending com (the commitment key) to $\mathcal{R}$. Given just σ and the commitment key, the polynomial-time receiver $\mathcal{R}$ cannot guess the bit with probability significantly better than $1/2$ (this is the *hiding property*). In the second phase (the decommitment phase) $\mathcal{S}$ reveals the bit b and the key dec (the decommitment key) to $\mathcal{R}$. Now $\mathcal{R}$ checks whether the decommitment key is valid; if not, $\mathcal{R}$ outputs a special string $\perp$, meaning that he rejects the decommitment from $\mathcal{S}$; otherwise, $\mathcal{R}$ can efficiently compute the bit b revealed by $\mathcal{S}$ and is convinced that b was indeed chosen by $\mathcal{S}$ 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 hiding property holds with respect to computationally bounded adversaries or to unbounded adversaries. Commitment schemes of the first (resp., second) type have been shown to have applications to zero-knowledge proofs (resp., arguments) [17, 21]. A computationally-hiding bit-commitment scheme has been constructed under the minimal assumption of the existence of pseudo-random generators [19]. A perfectly-hiding bit-commitment scheme has been constructed under the assumption of the existence of one-way permutations [21]. Both schemes have been designed in the interactive model (where no public reference string is available to parties); the former, however, can be adapted to run in the public parameter model.

Equivocable commitment schemes. Informally, an equivocable commitment scheme in the public parameter model is one for which there exists an efficient algorithm, substituting for the trusted third party $(\mathcal{TTP})$, which outputs a set of public parameters and a commitment such that: (a) the distribution of the generated public parameters, the commitment, and any decommitment is exactly equivalent to their distribution in a real execution of the protocol; and (b) the commitment can be opened in more than one possible way.

Definition 1. *Let $(\mathcal{TTP}, \mathcal{S}, \mathcal{R})$ be a perfectly-hiding commitment scheme in the public parameter model over message space $\mathcal{M}$. We say that $(\mathcal{TTP}, \mathcal{S}, \mathcal{R})$ is* perfectly equivocable *if there exists a probabilistic, polynomial time* equivocable commitment generator Equiv *such that:*

1. $\mathsf{Equiv}_1(1^k)$ *outputs* $(\sigma, \mathrm{com}, s)$ *(where s represents state information).*
2. *For all $m \in \mathcal{M}$,* $\mathsf{Equiv}_2(s, m)$ *outputs* dec *such that:*

(a) $\mathcal{R}(\sigma, \mathrm{com}, \mathrm{dec}) = m$.
(b) The following two random variables are identically distributed:

$$\{\sigma \leftarrow \mathcal{TTP}(1^k); (\mathrm{com}, \mathrm{dec}) \leftarrow \mathcal{S}(\sigma, m) : (\sigma, \mathrm{com}, \mathrm{dec})\}$$
$$\{(\sigma, \mathrm{com}, s) \leftarrow \mathsf{Equiv}_1(1^k); \mathrm{dec} \leftarrow \mathsf{Equiv}_2(s, m) : (\sigma, \mathrm{com}, \mathrm{dec})\}. \quad \square$$

The notion of equivocable commitment was first discussed by Beaver [1]. In [9] it was shown that an adaptation of the commitment scheme in [19] is equivocable in the public random string model (this fact was used in the construction of the non-malleable commitment scheme of [9]). Other applications of such schemes include zero-knowledge protocols [10].

Non-malleable commitment schemes. Two definitions of non-malleable commitment have appeared in the literature, both seeking to capture the following intuition of security: if an adversary, after viewing a commitment to x, can produce a commitment to a related value y, then a simulator can perform at least as well without viewing a commitment to x. The difference is in the definition of "producing a commitment". In the original definition [11] (*non-malleability with respect to commitment*), generating a valid commitment of y is sufficient. Note that this definition does not apply to perfectly-hiding commitment schemes since for such schemes the value committed to by a commitment is not well-defined. In the definition of [9] (*non-malleability with respect to opening*), the adversary must also be able to give a (valid) decommitment to y after viewing the decommitment to x. Since our primary constructions are of perfectly-hiding commitment schemes (for which non-malleability with respect to opening is the appropriate notion), we present a formal definition of this variant, and refer the reader elsewhere [11, 14] for definitions of non-malleability with respect to commitment.

Definition 2. *Let $(\mathcal{TTP}, \mathcal{S}, \mathcal{R})$ be a perfectly-hiding commitment scheme, and let k be a security parameter. We say that $(\mathcal{TTP}, \mathcal{S}, \mathcal{R})$ is* ϵ-non-malleable *(following [11])* with respect to opening *if, for all $\epsilon > 0$ and every probabilistic, polynomial time algorithm $\mathcal{A}$, there exists a simulator $\mathcal{A}'$ running in* $\mathrm{poly}(k, 1/\epsilon)$ *time, such that for all poly-time computable, valid relations R (see note below), for all efficiently sampleable distributions $\mathcal{D}$, we have:*

$$\mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k) - \widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k) \leq \epsilon + \mathrm{negl}(k)$$

(for some negligible function negl*); where:*

$$\mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k) \stackrel{\mathrm{def}}{=}$$
$$\Pr\left[\sigma \leftarrow \mathcal{TTP}(1^k); m_1 \leftarrow \mathcal{D}; (\mathrm{com}_1, \mathrm{dec}_1) \leftarrow \mathcal{S}(\sigma, m_1); \mathrm{com}_2 \leftarrow \mathcal{A}(\sigma, \mathrm{com}_1); \right.$$
$$\left. \mathrm{dec}_2 \leftarrow \mathcal{A}(\sigma, \mathrm{com}_1, \mathrm{dec}_1); m_2 \leftarrow \mathcal{R}(\sigma, \mathrm{com}_2, \mathrm{dec}_2) : \right.$$
$$\left. \mathrm{com}_1 \neq \mathrm{com}_2 \ \wedge \ R(m_1, m_2) = 1\right]$$

$$\widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k) \stackrel{\mathrm{def}}{=}$$
$$\Pr\left[m_1 \leftarrow \mathcal{D}; m_2 \leftarrow \mathcal{A}'(1^k, \mathcal{D}) : R(m_1, m_2) = 1\right]. \quad \square$$

DEFINITION OF NON-MALLEABILITY: The definition of security above allows for the possibility that the simulator may do *arbitrarily better than* the adversary. The reason for this is that the adversary may simply refuse to decommit, even when it would have otherwise succeeded[3]. In any case, if a simulator can do better than an adversary who gets to see a commitment to m_1, the scheme still satisfies our intuition of non-malleability.

VALID RELATIONS. In order for relation R to be valid, we impose the following restriction: for all $m \in \mathcal{M}$, we have $R(m, \perp) = 0$. This could also be taken into account by checking that $m_2 \neq \perp$ in the definitions of success, above; however, we find it easier to simply work with valid relations only.

MULTIPLE MESSAGES. The authors of [11] point out that a strictly stronger definition allows the adversary to produce *several* commitments $com_2^{(1)}, com_2^{(2)}, \ldots$, and later several decommitments $dec_2^{(1)}, dec_2^{(1)}, \ldots$ to messages $m_2^{(1)}, m_2^{(2)}, \ldots$. The simulator simply outputs messages $m_2^{(1)}, m_2^{(2)}, \ldots$. The adversary (or simulator) succeeds when a relation $\mathcal{R}(m_1, m_2^{(1)}, m_2^{(2)}, \ldots)$ holds. For simplicity, we use the weaker definition in this paper. However, we stress that all the schemes in this paper are non-malleable with respect to this stronger definition.

HISTORY. The definition of [11] includes the possibility of giving the adversary $hist(m_1)$ (for any computable function $hist$) before he is required to generate his commitment. We note that the current proof of our perfect commitment schemes does not consider this property.

3 Computationally-Hiding Commitment Schemes

We first (briefly) examine the case of standard commitment schemes. Note that the size of a standard, non-interactive commitment (even for malleable schemes) must be at least $M + \omega(\log k)$, where M is the message length and k is the security parameter. Perfect binding implies that the size must be at least M, and semantic security requires, in particular, that each message have $\omega(\text{poly}(k))$ possible commitments associated with it.

The lemma below indicates that we can achieve roughly this bound for standard non-malleable commitment, assuming the existence of trapdoor permutations[4] (in the model with public parameters). The commitment scheme is built from the following components: first, we use a cryptosystem that is secure indistinguishable under an adaptive-chosen-ciphertext attack. Such a scheme can be obtained using a construction in [11], and we denote this scheme by $\mathcal{E}_{\text{pk}}(\cdot)$. Next,

[3] For any relation R, a simulator exists for R as well as for its complement $\bar{R}$, so one might think that this "problem" can be avoided. The difficulty is that there is an asymmetry here, in that both R and $\bar{R}$ must satisfy $R(*, \perp) = \bar{R}(*, \perp) = 0$ (see the note on valid relations).

[4] Recall that [9] achieves a non-interactive, non-malleable computationally-hiding commitment using only one-way functions. However, their scheme requires commitment size $\mathcal{O}(kM)$.

we use a symmetric-key cryptosystem (with secret key of length k) which is indistinguishable under adaptive chosen-ciphertext attack (which can be obtained using, e.g., the construction of [11]), and we denote this scheme by $\mathcal{E}^*_K(\cdot)$. The commitment scheme works as follows: public parameters consist of a public key pk for the public-key cryptosystem. Commitment is done by choosing a random secret key for the symmetric-key system, encrypting this secret key using the public key, and then encrypting the committed message using the secret key. A commitment to message m is then computed as:

$$\mathcal{E}_{\mathrm{pk}}(K) \circ \mathcal{E}^*_K(m). \tag{1}$$

Decommitment consists of revealing m and the random bits used to form the commitment. Commitment verification is done in the obvious way.

Although the proof of the lemma is relatively straightforward (and is a "folk lemma" for the case of encryption), the result below was not widely known for the case of commitment. Indeed, there are some complications which require care to get right. A sketch of the proof can be found in Appendix A.

Lemma 1. *Assuming the existence of trapdoor permutations, there exists a computationally-hiding commitment scheme in the public parameter model that is non-malleable with respect to commitment and has commitment size $M + \mathrm{poly}(k)$, where M is the size of the committed message and k is a security parameter.*

Note that this lemma immediately implies the security (under the decisional Diffie-Hellman assumption) of the above construction when using the efficient public-key cryptosystem of [6] for $\mathcal{E}$ and any adaptive chosen-ciphertext-secure private-key cryptosystem $\mathcal{E}^*$. Finally, we note that the security requirements for $\mathcal{E}$ and $\mathcal{E}^*$ can be relaxed. One can show that $\mathcal{E}$ is only required to be non-malleable under a chosen-plaintext attack (NM-CPA) and $\mathcal{E}^*$ need only be indistinguishable under a P0 plaintext attack and an adaptive chosen-ciphertext attack (IND-PO-C2); see [2, 18] for formal definitions). This allows for much greater efficiency since NM-CPA-secure public-key cryptosystems can be constructed more efficiently than IND-CCA2 schemes [12] and IND-P0-C2-secure private-key schemes may be deterministic. We remark that the result in the lemma applies to the public random string model when so-called dense public-key encryption schemes [8, 7] are used.

4 Perfectly-Hiding Commitment Schemes

The computationally-hiding commitment scheme presented in Section 3 achieves near-optimal commitment size $M + \mathrm{poly}(k)$. We cannot hope to improve this by much (since computationally-hiding commitments have size at least M). In this section we present perfectly-hiding commitment schemes that improve significantly on the commitment length, achieving commitment size $3k$ for arbitrarily-large messages (see Section 5 for modifications allowing further reductions in the commitment size).

Both of our perfectly-hiding commitment schemes build on the paradigm established in [9], with changes which substantially improve the efficiency. A commitment consists of three components $\langle A, B, Tag \rangle$. The first component A is a commitment to parameters r_1 and r_2 for a one-time "message authentication code" (MAC) for B. The second component B contains the actual commitment to the message m, using public parameters which depend upon the first component A. Finally, $Tag = \text{MAC}_{r_1,r_2}(B)$. An adversary who wishes to generate a commitment to a related value has two choices: he can either re-use A or use a different A'. If he re-uses A, with high probability he will be unable to generate a correct Tag for a different B', since he does not know the values r_1, r_2. On the other hand, if he uses a different A', the public parameters he is forced to use for his commitment B' will be different from those used for the original commitment; thus, the adversary will be able to decommit in only one way, regardless of how the original B is decommitted. In particular, if it is possible to equivocate B for a particular choice of A, an adversary who uses a different A' will be unable to equivocate B' (without breaking some computational assumption). We refer the reader to [9] for further discussion.

In [9], the dependence (upon A) of the public parameters used for commitment B was achieved via a "selector function"[5], which results in public parameters of size dependent on the length of the committed message (as a consequence, the scheme can be efficient only in the case of commitment to a single bit). Here, we exploit algebraic properties to drastically reduce the size of the public parameters and obtain a more efficient scheme, even in the case of large messages.

4.1 Construction Based on the Discrete Logarithm Problem

The schemes discussed in this paper work over *any* group G of prime order for which extracting discrete logarithms is hard but multiplication is easy. However, for concreteness we will always assume that p, q are prime with $q|p-1$ and the group $G \subseteq \mathbb{Z}_p^*$ is the set of elements of order q.

Our starting point is the perfect commitment scheme of Pedersen [24]. Let g, h be generators of G. To commit to a message $m \in \mathbb{Z}_q$, choose random $r \in \mathbb{Z}_q$ and output $com = g^m h^r$. This scheme achieves information-theoretic secrecy, since com is uniformly distributed in G; furthermore, it is computationally binding as long as the discrete logarithm problem is hard. Note that a simple extension of the scheme (which we refer to as *extended-Pedersen*) allows commitment to two messages: simply let g_1, g_2, g_3 be generators of G, and to commit to messages $m_1, m_2 \in \mathbb{Z}_q$, choose random r and output $com = {g_1}^{m_1} {g_2}^{m_2} g_3^r$. This scheme retains perfect secrecy; furthermore, computational binding of the extended-Pedersen scheme can be proved via a reduction to the standard Pedersen scheme (see [5]). Note further that the Pedersen and extended-Pedersen schemes are perfectly equivocable (one simply chooses public parameters with known discrete logarithms).

The public parameters, output by $\mathcal{TTP}(1^k)$, are primes p, q with $q|(p-1)$ and $|p| = k$, along with random generators g_1, g_2, g_3 of G. Additionally, a random

[5] A different implementation of this technique first appeared in [11].

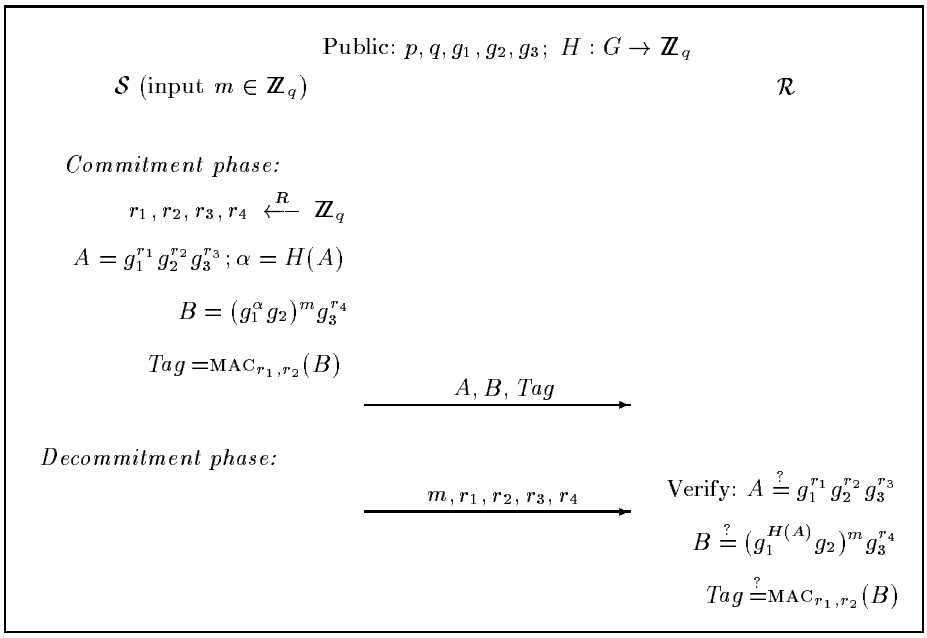

Fig. 1. DLog-based, NM perfect commitment scheme.

function H is chosen from a family of universal one-way hash (UOWH) functions [20]. Commitment is as shown in Figure 1.

Theorem 1. *Assuming the hardness of the discrete logarithm problem in the underlying group, the protocol of Figure 1 is an ϵ-non-malleable perfectly-hiding commitment scheme in the public-parameter model.*

Proof It is clear that the protocol is perfectly-hiding since B is uniformly distributed in group G independently from the distribution of every other component of the commitment. Computational binding of the protocol is also easy to show (proof omitted).

The proof of non-malleability is more involved; however, we provide some intuition here. As mentioned in Sec. 2, we prove non-malleability with respect to a single commitment output by the adversary; however, the same proof technique suffices to prove non-malleability with respect to multiple commitments. The simulator (which will do as well as the adversary without seeing the commitment) works as follows. First, it generates public parameters which are distributed identically to the real experiment, but for which the simulator knows some trapdoor information which allows it to perfectly equivocate its commitment (cf. Definition 1). The simulator generates a commitment *com* to a random message, gives this commitment to the adversary, and the adversary produces its commitment com_2. The simulator now tries to get the adversary to open com_2 (this will be the message output by the simulator). To do this, the simulator

decommits *com* to a random message and gives the decommitment to the adversary, and repeats this step (rewinding the adversary each time) sufficiently many times until the adversary opens[6] com_2. Since the simulator can perfectly equivocate its commitment, the adversary's view is equivalent to its view in the original experiment. Furthermore, we show that the adversary itself is unable to equivocate its commitment com_2 (under the discrete logarithm assumption). A complete proof follows.

Assume an adversary $\mathcal{A}$ which, given commitment $\langle A, B, Tag \rangle$, generates commitment $\langle A', B', Tag' \rangle$. Given decommitment $\langle m, r_1, r_2, r_3, r_4 \rangle$, the adversary gives decommitment $\langle m', r_1', r_2', r_3', r_4' \rangle$. Following the proof structure of [9], we distinguish the following sub-cases:

CASE 1. $A' = A$. If this occurs, there are two possibilities: either $\langle r_1, r_2, r_3 \rangle = \langle r_1', r_2', r_3' \rangle$, or not. If they are equal, since r_1 and r_2 are information-theoretically hidden from the adversary when giving his commitment (and assuming the security of the MAC), the adversary will have been unable (except with negligible probability) to generate $B' \neq B$ and Tag' such that $Tag' = \text{MAC}_{r_1, r_2}(B')$. If $\langle r_1, r_2, r_3 \rangle \neq \langle r_1', r_2', r_3' \rangle$, we can construct an adversary $\mathcal{C}$ which, given oracle access to $\mathcal{A}$, can violate the computational binding property of the extended-Pedersen scheme (via a standard reduction). Thus, the success probability of $\mathcal{A}$ in this case must be negligible.

CASE 2. $A' \neq A$ but $H(A') = H(A)$. If this happens, the security of the family of universal one-way hash functions is violated. Simply choose p, q along with random generators g_1, g_2, g_3. Then, select random m, r_1, r_2, r_3, r_4, generate the commitment $\langle A, B, Tag \rangle$, and output A. Upon being given a random member H from the UOWH family, run $\mathcal{A}$ on input the public parameters and the generated commitment. The first component of the commitment generated by $\mathcal{A}$ will then give the desired collision.

CASE 3. $A' \neq A$ and $H(A') \neq H(A)$. This is the most interesting case to consider. Fix ϵ, $\mathcal{D}$, and R, and assume adversary $\mathcal{A}$. Denote the process of selecting group parameters, as run by $\mathcal{TTP}$, by $p, q, G \leftarrow \mathcal{G}(1^k)$ (i.e., this selects primes p, q with $q|p-1$ and $|p| = k$). We describe an equivocable commitment generator

[6] If the adversary never opens its commitment, the simulator outputs $\perp$.

Equiv which will be used as a subroutine of simulator $\mathcal{A}'$:

$\mathsf{Equiv}_1(1^k)$
$p, q, G \leftarrow \mathcal{G}(1^k)$
$g_1, g_3 \leftarrow G; H \leftarrow$ UOWH
$r, s, t \leftarrow \mathbb{Z}_q$
$A = g_1^r g_3^s; \alpha = H(A)$
$g_2 = g_1^{-\alpha} g_3^t$
$\sigma = \langle p, q, g_1, g_2, g_3, H \rangle$
$r_2, u \leftarrow \mathbb{Z}_q$
$r_1 = r + \alpha r_2; r_3 = s - t r_2$
$B = g_3^u$; $Tag = \mathrm{MAC}_{r_1, r_2}(B)$
$com = \langle A, B, Tag \rangle$
$s = \langle r_1, r_2, r_3, t, u \rangle$
Output (σ, com, s)

$\mathsf{Equiv}_2(\langle r_1, r_2, r_3, t, u \rangle, m)$
$r_4 = u - tm$
$dec = \langle m, r_1, r_2, r_3, r_4 \rangle$
Output dec

Note that Equiv satisfies Definition 1. In particular, the distributions of the public parameters output by Equiv and those of the real protocol are the same; they differ only in the "trapdoor" information stored by Equiv. Furthermore, note that p, q, g_1, g_3 can be chosen at random and given to Equiv; knowledge of $\log_{g_3} g_1$ is not necessary. This will be crucial for the proof of security. We now describe the simulator $\mathcal{A}'$:

$\mathcal{A}'(1^k, \mathcal{D})$
$(\sigma, com, s) \leftarrow \mathsf{Equiv}(1^k)$
Fix random coins ω
$com_2 = \mathcal{A}(\sigma, com; \omega)$
Repeat at most $2\epsilon^{-1} \ln 2\epsilon^{-1}$ times:
$m_1 \leftarrow \mathcal{D}$
$dec = \mathsf{Equiv}(s, m_1)$
$dec_2 = \mathcal{A}(\sigma, com, dec)$
$m_2 = \mathcal{R}(\sigma, com_2, dec_2)$
if $m_2 \neq \perp$ break
Output m_2

We show that the difference $\mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k) - \widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k)$ (with terms as defined in Definition 2) is negligible. Straightforward manipulation, using the fact that Equiv is a perfectly equivocable commitment generator and $(\mathcal{TTP}, \mathcal{S}, \mathcal{R})$ is a perfect commitment scheme, gives:

$$\mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k) =$$
$$\Pr\left[\sigma \leftarrow \mathcal{TTP}(1^k); m_1 \leftarrow \mathcal{D}; \omega \leftarrow \Omega; r_1, r_2, r_3 \leftarrow \mathbb{Z}_q;\right.$$
$$(com_1, dec_1) \leftarrow \mathcal{S}(\sigma, m_1; r_1, r_2, r_3);$$
$$\left. m_2 = \mathcal{R}(\sigma, \mathcal{A}(\sigma, com_1; \omega), \mathcal{A}(\sigma, com_1, dec_1; \omega)) : R(m_1, m_2) = 1\right]$$

and

$$\begin{aligned}
&\widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k) = \\
&\Pr\left[\sigma \leftarrow \mathcal{TTP}(1^k); m_1 \leftarrow \mathcal{D}; \omega \leftarrow \Omega; r_1, r_2, r_3 \leftarrow \mathbb{Z}_q;\right. \\
&(com_1, dec_1) \leftarrow \mathcal{S}(\sigma, m_1; r_1, r_2, r_3); \\
&\left. m_2^* = \mathcal{R}(\sigma, \mathcal{A}(\sigma, com_1; \omega), \mathcal{A}(\sigma, com_1, dec^*; \omega)) : R(m_1, m_2^*) = 1\right].
\end{aligned}$$

The notation dec^* represents the fact that the decommitment given to $\mathcal{A}$ was produced according to algorithm $\mathcal{A}'$. In particular, dec^* represents either the first decommitment given to $\mathcal{A}$ which resulted in $m_2 \neq \perp$, or the $(2\epsilon^{-1}\ln 2\epsilon^{-1})^{\text{th}}$ decommitment given to $\mathcal{A}$ (if all decommitments up to then had $m_2 = \perp$).

Define the tuple $(\sigma; \omega; r_1, r_2, r_3; com_1)$ as *good* if the following holds:

$$\Pr\left[m_1 \leftarrow \mathcal{D} : \mathcal{R}(\sigma, \mathcal{A}(\sigma, com_1; \omega), \mathcal{A}(\sigma, com_1, dec_1; \omega)) \neq \perp\right] \geq \epsilon/2,$$

(the above probability is over choice of m_1 only; note that once the tuple is fixed, choice of m_1 determines r_4, and hence dec_1). Furthermore, define event Good as occurring when the tuple generated by the experiment is *good*. We now have (for brevity, we denote generation of a random tuple by $\gamma \leftarrow \Gamma(1^k)$; also, we denote $m_2 = \mathcal{R}(\sigma, \mathcal{A}(\sigma, com; \omega), \mathcal{A}(\sigma, com, dec; \omega))$ by $m_2 = \mathcal{A}(\sigma, com, dec)$):

$$\begin{aligned}
&\mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k) - \widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k) = \\
&\Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2 = \mathcal{A}(\sigma, com_1, dec_1) : R(m_1, m_2) \ \wedge\ \mathsf{Good}\right] \\
&+ \Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2 = \mathcal{A}(\sigma, com_1, dec_1) : R(m_1, m_2) \ \wedge\ \overline{\mathsf{Good}}\right] \\
&- \Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2^* = \mathcal{A}(\sigma, com_1, dec^*) : R(m_1, m_2^*) \ \wedge\ \mathsf{Good}\right] \\
&- \Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2^* = \mathcal{A}(\sigma, com_1, dec^*) : R(m_1, m_2^*) \ \wedge\ \overline{\mathsf{Good}}\right],
\end{aligned}$$

from which we derive (by definition of event $\overline{\mathsf{Good}}$):

$$\begin{aligned}
&\mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k) - \widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k) \leq \\
&\Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2 = \mathcal{A}(\sigma, com_1, dec_1) : R(m_1, m_2) \ \wedge\ \mathsf{Good}\right] \\
&+ \epsilon/2 \\
&- \Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2^* = \mathcal{A}(\sigma, com_1, dec^*) : R(m_1, m_2^*) \ \wedge\ \mathsf{Good}\right].
\end{aligned}$$

But this, in turn, implies:

$$\begin{aligned}
&\mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k) - \widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k) \leq \\
&\Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2 = \mathcal{A}(\sigma, com_1, dec_1); m_2^* = \mathcal{A}(\sigma, com_1, dec^*) :\right. \\
&\left. R(m_1, m_2) \ \wedge\ \overline{R(m_1, m_2^*)} \ \wedge\ \mathsf{Good}\right] + \epsilon/2,
\end{aligned}$$

which can be re-written as:

$$\mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k) - \widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k) \leq$$
$$\Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2 = \mathcal{A}(\sigma, com_1, dec_1); m_2^* = \mathcal{A}(\sigma, com_1, dec^*) : R(m_1, m_2) \wedge \overline{R(m_1, m_2^*)} \wedge m_2^* = \perp \wedge \mathsf{Good}\right] \quad (2)$$
$$+ \Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D}; m_2 = \mathcal{A}(\sigma, com_1, dec_1); m_2^* = \mathcal{A}(\sigma, com_1, dec^*) : R(m_1, m_2) \wedge \overline{R(m_1, m_2^*)} \wedge m_2^* \neq \perp \wedge \mathsf{Good}\right] \quad (3)$$
$$+ \epsilon/2.$$

We now bound probabilities (2) and (3). First, notice that expression (2) is bounded from above by the probability that $m_2^* = \perp$. However, definition of event Good and a straightforward probability calculation show that:

$$\Pr\left[\gamma \leftarrow \Gamma(1^k); m_2^* \leftarrow \mathcal{A}(\sigma, com_1, dec^*) : m_2^* = \perp \wedge \mathsf{Good}\right] \leq$$
$$\Pr\left[\gamma \leftarrow \Gamma(1^k); m_2^* \leftarrow \mathcal{A}(\sigma, com_1, dec^*) : m_2^* = \perp \,|\mathsf{Good}\right] \leq \epsilon/2.$$

Finally, notice that for the event in expression (3) to occur, we must have $m_2 \neq \perp$ and $m_2 \neq m_2^*$. But this then gives a Pedersen commitment com_2 (using generators g_3 and $g_1^{\alpha'} g_2 = g_1^{(\alpha' - \alpha)} g_3^t$) which is decommited in two different ways. This would allow determination of $\log_{g_3} g_1$ (recall that $\alpha' \neq \alpha$ since we are dealing with Case 3). The experiment is as follows: choose random ω and r_1, r_2, r_3 and run Equiv using the given values g_1, g_3 to generate σ' and com_1 (recall that knowledge of $\log_{g_3} g_1$ is not necessary to run Equiv). The adversary $\mathcal{A}$ then produces a commitment com_2. Following the description of $\mathcal{A}'$, run $\mathcal{A}$ to obtain a decommitment to message m_2^*. Then, decommit once more to a randomly selected m_1 and give this as input to $\mathcal{A}$ to obtain a decommitment to m_2. If $m_2 \neq \perp$ and $m_2^* \neq \perp$ and $m_2 \neq m_2^*$ (which we call event Success), then $\log_{g_3} g_1$ can be calculated, as discussed above. But the probability of Success is bounded from below by expression (3); by assumption, however, the discrete logarithm problem is intractable and thus:

$$(3) \leq \Pr[\mathsf{Success}] \leq negl(k).$$

Putting everything together gives the desired result. □

Note that the proof of non-malleability is exactly the same even if the message is hashed before commitment. Equiv can still perfectly equivocate to any (random) message M by first computing $m = \mathcal{H}(M)$ and then running the identical Equiv_2 algorithm. The simulator $\mathcal{A}'$ is also identical (messages will be longer, but this does not affect the analysis). The hash function must be collision resistant for the binding property to hold, but no other assumptions about the hash function are necessary, and the scheme is still perfectly secret[7]. The

[7] This can be compared to [14] which requires added complications when using an arbitrary hash function and achieves only statistical secrecy.

present scheme therefore gives a practical method for committing to arbitrarily long messages.

We remark that by making minor modifications to the above protocol, it can be proven secure in the public random string model as well.

We give an alternate proof of Theorem 1 in App. B. This proof, while more complicated than the proof given above, achieves a slightly stronger security guarantee by using a simulator which runs in expected polynomial time.

4.2 Construction Based on RSA

We have also developed an efficient non-interactive, non-malleable perfect commitment scheme based on the RSA assumption. Since the ideas underlying this construction, as well as the proof of security, are substantially similar to the scheme presented above, we include only a brief description of the scheme in Appendix C.

5 Extensions

There are extensions of our scheme which may be of practical value:

REDUCING THE COMMITMENT SIZE. Our schemes produce commitments $com = (A, B, Tag)$ of size $3k$, where k is the length of the string representing a group element. However, inspection of the proof of Thm. 1 reveals that one can replace this with any string that uniquely *binds* the sender to com. At least two modifications in this vein seem useful:

- Using a *collision-resistant* hash-function h, we can replace the commitment com with $h(com)$. The decommitment phase is the same as before. This does not increase the computational cost of the protocol by very much. The resulting commitment size is the output length of a hash function believed to be collision-resistant, e.g. SHA or MD5. In particular, this allows us to achieve optimal commitment size $\mathcal{O}(\omega(\log k))$, assuming an appropriate hash function. Note that this approach (hashing the commitment) does not seem to give provable security for general non-malleable commitment schemes, yet it *does* work (as can be seen by careful examination of the proof) for the particular construction given here.
- By adding one more public parameter and making appropriate (small) modifications to the scheme, we can set the commitment to the *product* of A, B and Tag (assuming Tag is computed as $B^{r_1} g_3^{r_2}$, which serves as an information-theoretically secure MAC). This reduces the commitment length to k. We defer a proof of security to the full version of the paper.

UNIQUE IDENTIFIERS. As mentioned in [11], in many situations there is a unique identifier (ID) associated to each user and using them can improve the efficiency of non-malleable primitives. This is also true of our scheme. If each user in the system has ID $id \in \mathbb{Z}_q$, we can simplify the scheme by replacing α with id. An adversary who attempts to generate related commitments must do so with respect to *his* identifier $id' \neq id$. The public parameters are p, q and three

generators g_1, g_2, g_3. The commitment is $B = (g_1^{id} g_2)^m g_3^{r_3}$ (the components A and Tag are no longer needed, since their only role in the original protocol was to force an adversary to change α). The proof of non-malleability is the same as for the original scheme except there is no need to handle cases 1 and 2.

Acknowledgments

Thanks to Yuval Ishai for helpful discussions and to Marc Fischlin and Roger Fischlin for a pre-print of the full version of [14] and for close readings of preliminary drafts of this work. Thanks to Yehuda Lindell for pointing out that an NM-CPA scheme is sufficient in the construction of Section 3. We also appreciate the nice suggestion of Ronald Cramer (mentioned in Section 5) to extend our constructions to the setting in which users have unique IDs.

References

1. D. Beaver. Adaptive Zero-Knowledge and Computational Equivocation. FOCS '96.
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. CRYPTO '98.
3. M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge. SIAM Journal of Computing, vol. 20, no. 6, Dec 1991, pp. 1084–1118.
4. M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Applications. STOC '88.
5. S. Brands. Rapid Demonstration of Linear Relations Connected by Boolean Operators. Eurocrypt '97.
6. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Chosen Ciphertext Attack. CRYPTO '98.
7. A. De Santis, G. Di Crescenzo, and G. Persiano. Necessary and Sufficient Assumptions for Non-Interactive Zero-Knowledge Proofs of Knowledge for All NP Relations. ICALP '00.
8. A. De Santis and G. Persiano. Zero-Knowledge Proofs of Knowledge Without Interaction. FOCS '92.
9. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. STOC '98.
10. G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge with Preprocessing. CRYPTO '99.
11. D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. SIAM J. Comp. 30 (2) 391–437, 2000. A preliminary version appears in STOC '91.
12. C. Dwork. The Non-Malleability Lectures. Available from the author.
13. S. Even, O. Goldreich, A. Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM* 28(6), 637–647, 1985.
14. M. Fischlin and R. Fischlin. Efficient Non-Malleable Commitment Schemes. CRYPTO 2000.
15. O. Goldreich. *Foundations of Cryptography, Fragments of a Book*, 1998.
16. O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority. STOC '87.
17. O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP have Zero-Knowledge Proof Systems. *J. ACM* 38(3): 691–729 (1991).

18. J. Katz and M. Yung. Complete Characterization of Security Notions for Probabilistic Private-Key Encryption. STOC '00.
19. M. Naor. Bit Commitment Using Pseudorandomness. J. Crypto. 4(2): 151–158 (1991).
20. M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. STOC '89.
21. M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP can be based on general complexity assumptions. J. Cryptology, 11(2):87–108, 1998 (also CRYPTO '92).
22. T. Okamoto. Provable Secure and Practical Identification Schemes and Corresponding Signature Schemes. CRYPTO '92.
23. R. Ostrovsky, R. Venkatesan, and M. Yung. Fair games against an all-powerful adversary. AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 13 pp. 155-169, 1993.
24. T.P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. CRYPTO '91.
25. A. Sahai. Non-Malleable Non-Interactive Zero-Knowledge and Adaptive Chosen-Ciphertext Security. FOCS '99.

A Proof of Lemma 1

Sketch of Proof First note that for (1) to be binding, we require that the decryption algorithms for both the public-key and symmetric-key systems have zero probability of decryption error[8]. Thus, revealing the randomness used to generate the commitment perfectly binds the sender to the message.

The proof of non-malleability with respect to commitment will imply that the scheme is semantically secure (this has been noted previously for the case of encryption [2, 18], but a similar result holds for the case of commitment). Note that if we can prove that (1) constitutes a non-malleable (public-key) *encryption* scheme, we are done. Using the results of [2], it suffices to prove that (1) is secure under adaptive chosen-ciphertext attack.

Consider an adversary $\mathcal{A}$ who has non-negligible advantage in attacking (1) under an adaptive chosen-ciphertext attack. Define adversary $\mathcal{B}$ which uses $\mathcal{A}$ as a black box to break $\mathcal{E}_{\mathrm{pk}}$ under an adaptive chosen-ciphertext attack (the notation $\mathcal{D}(\cdot)$ means that $\mathcal{B}$ is given access to a decryption oracle for $\mathcal{E}_{\mathrm{pk}}$):

Algorithm $\mathcal{B}_1^{\mathcal{D}(\cdot)}(1^k, \mathrm{pk})$
  $(M_0, M_1, s) \leftarrow \mathcal{A}_1^{\tilde{\mathcal{D}}(\cdot)}(1^k, \mathrm{pk})$
  $K \leftarrow \{0,1\}^k$
  $b \leftarrow \{0,1\}$
  $C \leftarrow \mathcal{E}_K^*(M_b)$
  return $(K, 0^k, (C, s))$

Algorithm $\mathcal{B}_2^{\mathcal{D}(\cdot)}(y, (C, s))$
  $b' \leftarrow \mathcal{A}_2^{\tilde{\mathcal{D}}(\cdot)}(y \circ C, s)$
  if $b' = b$ return 1
  else return 0

[8] This can be relaxed slightly, but since many commonly-used encryption schemes already have this property, we assume it here for simplicity of exposition.

The notation $\widetilde{\mathcal{D}}(\cdot)$ means that decryption oracle queries of $\mathcal{A}$ are handled by $\mathcal{B}$ in the following way: in the first stage, when $\mathcal{A}$ submits ciphertext $y' \circ C'$ to its decryption oracle, $\mathcal{B}$ submits y' to its decryption oracle for $\mathcal{E}_{\mathrm{pk}}$, receives key K', and then computes $M' := \mathcal{D}_{K'}(C')$. In the second stage, $\mathcal{B}$ answers as before except that $\mathcal{A}$ might submit a ciphertext $y \circ C'$. Note that $\mathcal{B}$ would *not* be allowed to submit y to its decryption oracle, since he cannot ask for decryption of the challenge ciphertext. Instead, $\mathcal{B}$ "assumes" that y is an encryption of K, and computes the response $M := \mathcal{D}_K(C)$. Adaptive chosen-ciphertext security of $\mathcal{E}_{\mathrm{pk}}$ implies that the advantage of $\mathcal{B}$ is negligible.

We now consider the following adversary which uses $\mathcal{A}$ as a black box to break $\mathcal{E}^*$ under an adaptive chosen-ciphertext attack. Here, the notation $\mathcal{D}(\cdot)$ means that $\mathcal{C}$ is given access to a decryption oracle for $\mathcal{E}^*_K$ (where K is some secret key unknown to $\mathcal{C}$). We let Gen denote the algorithm which selects public and private keys for $\mathcal{E}$.

Algorithm $\mathcal{C}_1^{\mathcal{D}(\cdot)}(1^k)$	Algorithm $\mathcal{C}_2^{\mathcal{D}(\cdot)}(C, (y, \mathrm{sk}, s))$
$(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{Gen}(1^k)$	$b' \leftarrow \mathcal{A}_2^{\widetilde{\mathcal{D}}(\cdot)}(y \circ C, s)$
$(M_0, M_1, s) \leftarrow \mathcal{A}_1^{\widetilde{\mathcal{D}}(\cdot)}(1^k, \mathrm{pk})$	return b'
$y \leftarrow \mathcal{E}_{\mathrm{pk}}(0^k)$	
return $(M_0, M_1, (y, \mathrm{sk}, s))$	

Here, the notation $\widetilde{\mathcal{D}}(\cdot)$ means that decryption oracle queries of $\mathcal{A}$ are handled by $\mathcal{C}$ in the following way: in the first stage, when $\mathcal{A}$ submits ciphertext $y' \circ C'$ to its decryption oracle, $\mathcal{C}$ decrypts y' to get K' (it knows the secret key) and then computes $M' := \mathcal{D}_{K'}(C')$. In the second stage, however, $\mathcal{C}$ answers as before *unless* $\mathcal{A}$ submits a ciphertext $y \circ C'$. In this case, $\mathcal{C}$ submits C' to its decryption oracle for $\mathcal{E}^*_K$ and returns the result to $\mathcal{A}$. Adaptive chosen-ciphertext security of $\mathcal{E}^*$ implies that the advantage of $\mathcal{C}$ is negligible.

Informally, define the following probabilities of success:

$$
\begin{aligned}
p_{0,r} &\stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}^{\mathcal{D}(\cdot)}(\mathcal{E}_{\mathrm{pk}}(K) \circ \mathcal{E}^*_K(M_0)) = 0] \\
p_{0,f} &\stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}^{\widetilde{\mathcal{D}}(\cdot)}(\mathcal{E}_{\mathrm{pk}}(0^k) \circ \mathcal{E}^*_K(M_0)) = 0] \\
p_{1,r} &\stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}^{\mathcal{D}(\cdot)}(\mathcal{E}_{\mathrm{pk}}(K) \circ \mathcal{E}^*_K(M_1)) = 1] \\
p_{1,f} &\stackrel{\mathrm{def}}{=} \Pr[\mathcal{A}^{\widetilde{\mathcal{D}}(\cdot)}(\mathcal{E}_{\mathrm{pk}}(0^k) \circ \mathcal{E}^*_K(M_1)) = 1].
\end{aligned}
$$

$\mathcal{B}$'s advantage is given by $1/2\,\{1/2(1 - p_{0,f}) + 1/2(1 - p_{1,f})\} + 1/4(p_{0,r} + p_{1,r})$. $\mathcal{C}$'s advantage is given by $1/2(p_{o,f} + p_{1,f})$. Note that these are both negligible, by the arguments advanced above. Finally, the advantage of $\mathcal{A}$ in the original experiment is given by $1/2(p_{0,r}+p_{1,r})$. Simple algebra implies that $\mathcal{A}$'s advantage must be negligible.

Note that adaptive-chosen-ciphertext-secure private-key encryption schemes can be constructed using a one-way function, while non-malleable public-key encryption schemes (with 0 probability of decryption error) are known to exist assuming trapdoor permutations [11, 25]. This completes the proof. □

B Alternate Proof of Theorem 1

In this section, we present an alternate proof of Theorem 1, which in fact gives us a stronger security guarantee. First, notice that in the previous proof, the simulator had to cut off the simulation after $2\epsilon^{-1}\ln 2\epsilon^{-1}$ steps. This is because for some values of the initial setup γ, it is possible that the adversary would not decommit at all, and thus that the simulation would never terminate. This is an essential problem with the sort of simulation described above: even if the fraction of "bad setups γ" were barely noticeable, the expected running time of the simulation might be infinite!

Instead, we give a simulation which always runs in expected polynomial time, *provided that the adversary succeeds with noticeable probability.* To do so, we adapt the proof technique of DIO [9]. Unfortunately, one cannot apply their proof directly here since their proof relies on the fact that the DIO commitment scheme is statistically binding.

Let $p_{\mathcal{A}}$ be the success probability of the adversary in the original basic experiment for non-malleability with respect to opening, i.e. $p_{\mathcal{A}} = \mathsf{Succ}^{\mathrm{NM}}_{\mathcal{A},\mathcal{D},R}(k)$. For a given simulator A', let $\widetilde{p}_{\mathcal{A}'}$ denote the simulator's success probability, i.e. $\widetilde{p}_{\mathcal{A}'} = \widetilde{\mathsf{Succ}}_{\mathcal{A}',\mathcal{D},R}(k)$. We will construct a simulator $\mathcal{A}'$ such that $p_{\mathcal{A}} - \widetilde{p}_{\mathcal{A}'} \leq negl(k)\left(\frac{1}{p_{\mathcal{A}}}\right)$ and the expected running time of $\mathcal{A}'$ is polynomial in $\frac{1}{p_{\mathcal{A}}}$. Notice in particular that when the adversary's probability of success is noticeable, our simulation does (essentially) at least as well as the original adversary, and runs in expected polynomial time.

The simulator $\mathcal{A}'$ is simple: it runs the adversary in the basic non-malleability experiment until the adversary succeeds; it then outputs whatever m_2 the adversary succeeded with. We describe two equivalent formulations of this simulator below. The first simulation generates all its parameters honestly; the second simulation uses the equivocator of the previous section.

$\mathcal{A}'_1(1^k, \mathcal{D})$
  $m_1, m_2 := \perp$
  $com_1, com_2 := 0$
  Repeat until $R(m_1, m_2) = 1$ and $com_1 \neq com_2$:
    $\sigma \leftarrow \mathcal{TTP}(1^k)$
    $m_1 \leftarrow \mathcal{D}$
    $(com_1, dec_1) \leftarrow \mathcal{S}(\sigma, m_1)$
    $com_2 \leftarrow \mathcal{A}(\sigma, com_1)$
    $dec_2 \leftarrow \mathcal{A}(\sigma, com_1, dec_1)$
    $m_2 \leftarrow \mathcal{R}(\sigma, com_2, dec_2)$
  Output m_2

$\mathcal{A}'_2(1^k, \mathcal{D})$
  $m_1, m_2 := \perp$
  $com_1, com_2 := 0$
  Repeat until $R(m_1, m_2) = 1$ and $com_1 \neq com_2$:
    $(\sigma, com_1, s) \leftarrow \mathsf{Equiv}(1^k)$
    Fix random coins ω
    $com_2 := \mathcal{A}(\sigma, com_1; \omega)$
    $m_1 \leftarrow \mathcal{D}$
    $dec_1 := \mathsf{Equiv}(s, m_1)$
    $dec_2 := \mathcal{A}(\sigma, com_1, dec_1)$
    $m_2 := \mathcal{R}(\sigma, com_2, dec_2)$
  Output m_2

From the point of view of the adversary, both of these simulations are equivalent, since the equivocator creates a public string σ and a commitment com which are from the same distribution as the "real" strings σ and com_1. Thus,

the output distribution of the two simulations is the same, and hence so is their probability of success. Moreover, both simulations expect to make $\frac{1}{p_{\mathcal{A}}}$ calls to $\mathcal{A}$, and thus their expected running times are essentially the same.

The only difference between the two simulations is that in the first simulation $\mathcal{A}'_1$, the simulator knows no more than the adversary about the relationship of the public parameters $g_1, g_2, g_3, \mathcal{H}$, and so all three of these values could come from an outside source. In the second simulation $\mathcal{A}'_2$, only g_1 and g_3 can come from an outside source; g_2 and the other parameters are carefully constructed.

As before, we now consider the three possible cases.

CASES 1 AND 2. Consider the first simulator $\mathcal{A}'_1$. As mentioned above, it does not choose the public parameters g_1, g_2, g_3, H, and so the analysis from the previous proof of cases 1 and 2 tells us that the probability of either of these cases is negligible. (Otherwise, the simulator would break either the computational binding of the Pedersen scheme or the intractability of finding collisions for the hash function).

Hence, we can assume *even in the second simulation* that whenever the adversary generates a new commitment to which he decommits, we have $H(A') \neq H(A)$.

CASE 3. As in the previous proof, we denote the generation of a tuple $\gamma = (\sigma, com, s, \omega)$ by the shorthand $\gamma \leftarrow \Gamma(1^k)$. Note that these variables uniquely determine both com_1 and com_2. Moreover, once the simulator chooses m_1, the decommitted message m_2 is completely determined by m_1 and γ. For conciseness we write simply $m_2 = \mathcal{A}(m_1, \gamma)$. By convention, we will take $\mathcal{A}(m_1, \gamma) = \perp$ whenever the adversary refuses to decommit or simply copies the commitment (i.e. $com_2 = com_1$).

On one hand, we can calculate the adversary's success probability, using the properties of the equivocator:

$$p_{\mathcal{A}} = \Pr\left[\gamma \leftarrow \Gamma(1^k); m_1 \leftarrow \mathcal{D} \ : \ m_2 = \mathcal{A}(m_1, \gamma) \text{ and } R(m_1, m_2)\right].$$

We can also calculate the success probability of the simulator (second formulation):

$$\begin{aligned}\widetilde{p}_{\mathcal{A}'} &= \Pr\left[\gamma \leftarrow \Gamma(1^k);\ m_1, m'_1 \leftarrow \mathcal{D} \ : \quad R(m'_1, \mathcal{A}(m_1, \gamma)) \ \middle|\ R(m_1, \mathcal{A}(m_1, \gamma))\right] \\ &= \frac{\Pr\left[\gamma \leftarrow \Gamma(1^k);\ m_1, m'_1 \leftarrow \mathcal{D} \ : \quad R(m'_1, \mathcal{A}(m_1, \gamma)) \text{ and } R(m_1, \mathcal{A}(m_1, \gamma))\right]}{p_{\mathcal{A}}}\end{aligned}$$

The numerator in the last expression can be interpreted as the success probability of the following experiment:

> Choose m_1 at random, and run the simulation to obtain a decommitment to a message m_2. Then pick a new message m'_1 at random and see if both $R(m_1, m_2)$ and $R(m'_1, m_2)$ hold.

Now intuitively, we expect that for any given γ the adversary can only decommit to one valid message. We want to use that intuition to show that the success probability of the experiment above is no worse than the following:

Choose m_1 and obtain m_2 as before. Now, for the same setup γ, pick a new message m_1' and run the simulation to get m_2'. Output a success if both $R(m_1, m_2)$ and $R(m_1', m_2')$ hold.

This intuition is captured in the following lemma:

Lemma 2. *Let $m_2 = \mathcal{A}(m_1, \gamma)$, and $m_2' = \mathcal{A}(m_1', \gamma)$, where γ, m_1, m_1' are chosen as in the previous discussion. Then we have:*

$$\Pr\left[R(m_1, m_2) \wedge R(m_1', m_2)\right] > \Pr\left[R(m_1, m_2) \wedge R(m_1', m_2')\right] - negl(k).$$

Proof For any two events A and B, we have $P(A) - P(B) \leq P(A \setminus B)$. Thus:

$$\Pr\left[R(m_1, m_2) \wedge R(m_1', m_2')\right] - \Pr\left[R(m_1, m_2) \wedge R(m_1', m_2)\right] \\ \leq \Pr\left[R(m_1, m_2) \wedge R(m_1', m_2') \wedge \overline{R(m_1', m_2)}\right].$$

Now this last event occurs only when m_2 and m_2' are different, yet both of them are valid messages. However, such an event allows extraction of the discrete log of g_3 with respect to g_1, even in the setting of the second simulation. Since $\mathcal{A}$ is polynomial time, this probability must be negligible in k. □

Using the lemma and the shorthand notation set up in the lemma, we get:

$$\widetilde{p}_{\mathcal{A}'} \geq \frac{\Pr\left[R(m_1, m_2) \wedge R(m_1', m_2)\right]}{p_{\mathcal{A}}} \geq \frac{\Pr\left[R(m_1, m_2) \wedge R(m_1', m_2')\right] - negl(k)}{p_{\mathcal{A}}}.$$

Recall that once γ and m_1 are fixed, m_2 is also fixed. Similarly, m_2' is fixed once γ and m_1' are fixed. Thus, we can write:

$$\widetilde{p}_{\mathcal{A}'} \geq \frac{\left(\sum_\gamma \Pr\left[\gamma = \Gamma(1^k)\right] \cdot \Pr\left[R(m_1, m_2) \mid \gamma\right] \cdot \Pr\left[R(m_1', m_2') \mid \gamma\right]\right) - negl(k)}{p_{\mathcal{A}}} \\ = \frac{\left(\sum_\gamma \Pr\left[\gamma = \Gamma(1^k)\right] \cdot \Pr\left[R(m_1, m_2) \mid \gamma\right]^2\right) - negl(k)}{p_{\mathcal{A}}}.$$

For any random variable X we have $E(X^2) \geq (E(X))^2$. Applying this to the numerator we get:

$$\widetilde{p}_{\mathcal{A}'} \geq \frac{\left(\sum_\gamma \Pr\left[\gamma = \Gamma(1^k)\right] \cdot \Pr\left[R(m_1, m_2) \mid \gamma\right]\right)^2 - negl(k)}{p_{\mathcal{A}}}.$$

But the numerator is simply $(p_{\mathcal{A}})^2$. Thus $\widetilde{p}_{\mathcal{A}'} \geq p_{\mathcal{A}} - \frac{negl(k)}{p_{\mathcal{A}}}$. □

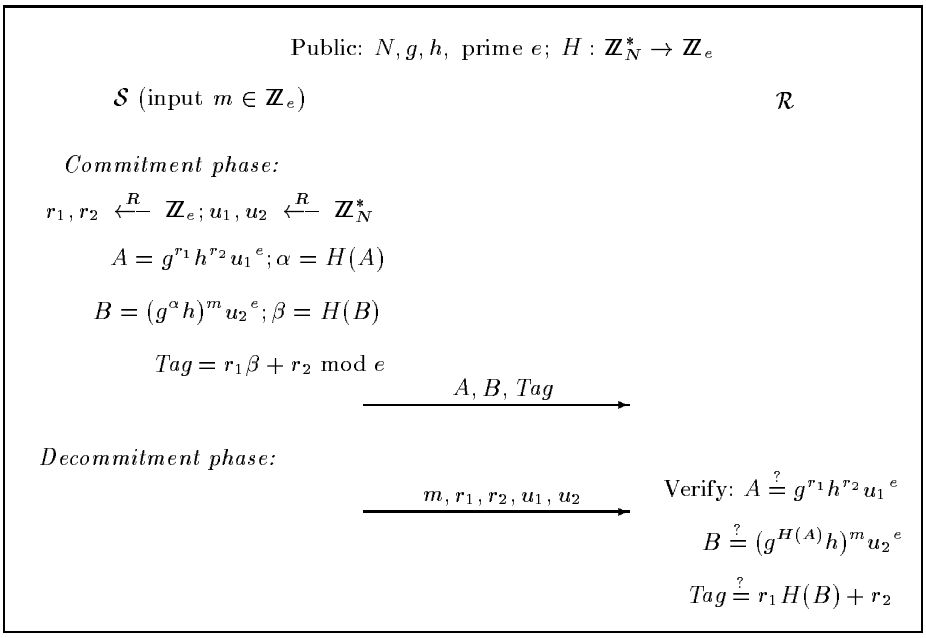

Fig. 2. RSA-based, NM perfect commitment scheme.

C Perfect Commitment Scheme Based on RSA

We briefly discuss an RSA-based implementation of non-malleable commitment. The starting point is the basic RSA-based perfect commitment scheme of Okamoto [22], which works as follows. Let N be a product of two primes, and let $g \in \mathbb{Z}_N^*$ and e a prime number be given. Then, a commitment to a message $m \in \mathbb{Z}_e$ is generated by choosing a random $u \in \mathbb{Z}_N^*$ and forming the commitment $g^m u^e$. This scheme achieves information-theoretic secrecy; furthermore, it is computationally binding under the RSA assumption (which states that it is infeasible to compute $g^{1/e}$ for random $g \in \mathbb{Z}_N^*$, e prime). We also make use of an simple extension (which we call *extended-Okamoto*) which allows commitment to two messages. Here, given $g, h \in \mathbb{Z}_N^*$ and prime e, the commitment to (m_1, m_2) is given by $g^{m_1} h^{m_2} u^e$. This scheme retains perfect secrecy; moreover, the computational binding can be proved via a reduction to the basic Okamoto scheme (see [5]). Finally, note that the Okamoto scheme and extended Okamoto scheme are equivocable (even when factorization of N is unknown and pre-specified e) — simply generate public parameters r^e, e.

A complete description of the protocol appears in Figure 2. The intuition for the protocol is exactly the same as that for the discrete logarithm-based protocol given in Section 4.1. The public parameters consist of N, a product of two large primes with $|N| = k$, along with a prime e and two random elements $g, h \in \mathbb{Z}_N^*$. It is important here that $|e| = \mathcal{O}(poly(k))$; for concreteness, we simply assume that $|e| = k$. Also included is a function H chosen from a family of universal one-

way hash functions. To commit to a message $m \in \mathbb{Z}_e$, the sender first chooses random $r_1, r_2 \in \mathbb{Z}_e$. The sender forms the first component A by "committing" to r_1, r_2 using the extended-Okamoto scheme; these values will later be used to authenticate the second component. The sender calculates $\alpha = H(A)$, and then commits to message m via a "basic" Okamoto scheme, with one important difference: the element used for this commitment depends on α. That is, the sender performs basic Okamoto commitment with prime e and random element $g^{\alpha}h$. As before, this will be essential to the proof of security. Finally, a *Tag* of B is computed using r_1 and r_2 from before. Note again that the probability of forging a *Tag* for a new message, after seeing the *Tag* for only one previous message, is information-theoretically limited to $1/e$ (and thus negligible).

We leave the formal proof of security for this protocol to the final version of the paper since it is substantially equivalent to the proof of Theorem 1.