Multi-trapdoor Commitments and their Applications to Non-Malleable Protocols*

Rosario Gennaro
IBM T.J.Watson Research Center
P.O.Box 704, Yorktown Heights NY 10598
rosario@watson.ibm.com

November 26, 2004

Abstract

We introduce the notion of *multi-trapdoor* commitments which is a stronger form of trapdoor commitment schemes. We then construct two very efficient instantiations of multi-trapdoor commitment schemes, based on the Strong RSA Assumption and the recently introduced Strong Diffie-Hellman Assumption.

The main applications of our result are *non-malleable* trapdoor commtiments and a *compiler* that takes *any* proof of knowledge and transforms it into one which is secure against a concurrent man-in-the-middle attack. Such a proof of knowledge immediately yields concurrently secure identification protocols.

When using our number-theoretic istantiations, the non-malleable commitment and the compiler are very efficient (require no more than four exponentiations). The latter also maintains the round complexity of the original proof of knowledge; it works in the common reference string model, which in any case is necessary to prove security of proofs of knowledge under this kind of attacks. Compared to previously known efficient solutions, ours is a factor of two faster.

1 Introduction

A *commitment* scheme is the cryptographic equivalent of an envelope. Consider the classic example of sealed bid auctions. Parties who want to bid on an item, place their bids in an envelope, in order to maintain secrecy until the end of the bidding period. At that time all bids are revealed by opening the envelopes, which in particular means that parties cannot alter bids at this point. A commitment scheme plays the role of the envelope: it's a cryptographic protocol composed of two phases: the committing phase and the opening phase. At the end of the first phase, a sender has committed to a message which however remains secret; in the opening phase the sender can only reveal that fixed message.

Commitment schemes play an important role in almost all cryptographic protocols. But the auction example above shows that the secrecy of the committed message at the end of the committing phase, may not be a sufficient requirement. In the real world, even if B sees the envelope containing party A's bid he will not be able to produce an envelope with a bid related to A's bid (for example a slightly higher bid, sufficient for B to win the auction). In the cryptographic world, the basic definition of security for a commitment scheme may not prevent this.

*A preliminary version of this paper appeared in the proceedings of CRYPTO'04, Springer LNCS 3152.

A *non-malleable* commitment (a notion introduced in [23]) requires that the adversary, when given various commitments to messages m_i, should not be able to commit to a related message m.

Proofs of Knowledge. A proof of knowledge allows a Prover to convince a Verifier that he knows some secret information w (for example a witness for an NP-statement y). Since w must remain secret, one must ensure that the proof does not reveal any information about w to the Verifier (who may not necessarily act honestly and follow the protocol). Proofs of knowledge have several applications, chief among them identification protocols where a party, who is associated with a public key, identifies himself by proving knowledge of the matching secret key.

However when proofs of knowledge are performed on an open network, like the Internet, one has to worry about an active attacker manipulating the conversation between honest parties. In such a network, also, we cannot expect to control the timing of message delivery, thus we should assume that the adversary has control also on when messages are delivered to honest parties.

The adversary could play the "man-in-the-middle" role, between honest provers and verifiers. In such an attack the adversary will act as a prover with an honest verifier, trying to make her accept a proof, even if the adversary does not know the corresponding secret information. During this attack, the adversary will have access to honest provers proving other statements. In the most powerful attack, the adversary will start several such sessions at the same time, and interleave the messages in any arbitrary way.

Informally, we say that a proof of knowledge is concurrently non-malleable, if such an adversary will never be able to convince a verifier when she does not know the relevant secret information (unless, of course, the adversary simply relies messages unchanged from an honest prover to an honest verifier).

1.1 Our Contribution

First, we present a new non-malleable commitment scheme. Then we present a general transformation that takes any proof of knowledge and makes it concurrently non-malleable. The transformation preserves the round complexity of the original scheme and it requires a common reference string shared by all parties.

The crucial technical tool is the notion of *multi-trapdoor commitments* which we introduce in this paper. After defining the notion we show two specific number-theoretic constructions based on the Strong RSA Assumption and the recently introduced Strong Diffie-Hellman Assumption. These constructions are very efficient.

As far as we know this is the most efficient construction of concurrently non-malleable proofs of knowledge, gaining at least a factor of two in faster computation and communication over previous proposals (see below).

Multi-Trapdoor Commitments. Recall that a commitment scheme consist of two phases, the first one in which a sender commits to a message and a second one in which the sender reveals the committed message.

A trapdoor commitment scheme allows a sender to commit to a message with information-theoretic privacy. I.e., given the transcript of the commitment phase the receiver, even with infinite computing power, cannot guess the committed message better than at random. On the other hand when it comes to opening the message, the sender is only computationally bound to the committed message. Indeed the scheme admits a *trapdoor* whose knowledge allows to open a commitment in any possible way. This trapdoor should be hard to compute efficiently.

A *multi-trapdoor* commitment scheme consists of a family of trapdoor commitments. Each scheme in the family is information-theoretically private. The family admits a *master* trapdoor whose knowledge allows to open *any* commitment in the family in any way it is desired.

Moreover each commitment scheme in the family admits its own specific trapdoor. The crucial property in the definition of multi-trapdoor commitments is that when given the trapdoor of one scheme in the family it is infeasible to compute the trapdoor of another scheme (unless the master trapdoor is known).

CONCURRENT COMPOSITION IN DETAIL. When we consider a man-in-the-middle attacker for proofs of knowledge we must be careful to define exactly what kind of concurrent composition we allow.

Above we described the case in which the attacker acts as a verifier in several concurrent executions of the proof, with several provers. We call this *left-concurrency* (as usually the provers are on positioned on the left of the picture, see Figure 1). On the other hand *right-concurrency* means that the adversary could start several concurrent executions as a prover with several verifiers.

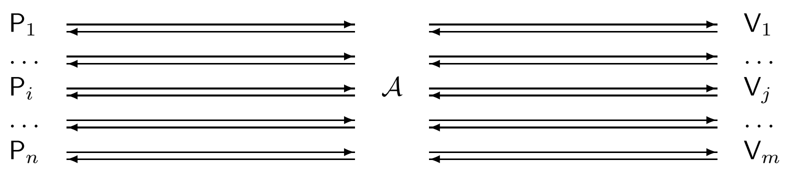

Figure 1: Left and Right Concurrency

Under these attacks, we need to prove that the protocols are zero-knowledge (i.e. simulatable) and also proofs of knowledge (i.e. one can extract the witness from the adversary). When it comes to extraction one also has to make the distinction between *in-line* and *post-protocol* extraction [32]. In an on-line extraction, the witness is extracted as soon as the prover successfully convinces the verifier. In a post-protocol extraction procedure, the extractor waits for the end of all the concurrent executions to extract the witnesses of the successful executions.

In the common reference string it is well known how to fully (i.e. both left and right) simulate proofs of knowledge efficiently, using the result of Damgård [17]. We use his techniques, so our protocols are fully concurrently zero-knowledge. Extraction is more complicated. Lindell in [37] shows how to do post-protocol extraction for the case of right concurrency. We can use his techniques as well. But for many applications what really matters is on-line extraction. We are able to do that only under left-concurrency[1]. This is however enough to build fully concurrently secure applications like identification protocols.

APPLICATIONS: CONCURRENTLY SECURE IDENTIFICATION SCHEMES. The main application of proofs of knowledge is in the design of secure identification schemes [25]. In these schemes a party identifies himself by proving that he knows some secret information. Our result allows the construction of efficient identification schemes that remain secure against an active man-in-the-middle in a concurrent communication model. As far as we know these are the first efficient schemes that can be proven concurrently secure.

[1]However, as we explain later in the Introduction, we could achieve also right-concurrency if we use so-called Ω-protocols

For example, by using Schnorr's identification scheme [44], together with our techniques, we obtain a very efficient concurrently secure identification scheme under both the discrete log and Strong RSA Assumption. By using the Guillou-Quisquater identification scheme [31], we obtain a comparable scheme which can be proven secure using only the Strong RSA Assumption. Both schemes require 3 rounds of communication and only 3 modular exponentiations per party.

Bellare *et al.* in [5] present *resettable* identification schemes, which are in particular also concurrently secure. Our schemes achieve only the weaker property of concurrent composition (and work under a specific assumption) but they are more efficient.

1.2 Prior Work

Previous Work on Non-Malleable Commitments. The problem of malleability in cryptographic algorithms was formalized by Dolev *et al.* in [23]. Starting from [23], there has been a lot of work devoted to non-malleable commitments, e.g. [20, 26, 21]. However the schemes described there satisfy a weaker definition of non-malleability which is not sufficient for our purposes. Indeed in those papers the adversary is limited to seeing a *single* commitment for a message m and is required to produce a commitment to a related message m'.

This definition is clearly not sufficient in a concurrent scenario, as there the adversary may be interacting with several honest parties at the same time, and thus see several commitments before trying to output a commitment to a related message. This stronger notion was considered by Damgård and Groth in [18] (they call it *reusability*) and it is the notion we will use in this paper.

Damgård and Groth present a generic paradigm to construct (reusable) non-malleable commitments based on secure signature schemes, including a very efficient istantiation of this paradigm based on the Strong RSA Assumption. When comparing our scheme with theirs, it is is easy to see that our proposal is more efficient in computation (about twice as fast) though produces much longer commitment and opening strings.

Non-Malleable Proofs of Knowledge. Zero-knowledge protocols were introduced in [29]. The notion of proof of knowledge (already implicit in [29]) was formalized in [25, 6].

Concurrent zero-knowledge was introduced in [24]. They point out that the typical simulation paradigm to prove that a protocol is zero-knowledge fails to work in a concurrent model. This work sparked a long series of papers culminating in the discovery of non-constant upper and lower bounds on the round complexity of concurrent zero-knowledge in the black-box model [13, 42], unless extra assumptions are used such as a common reference string. Moreover, in a breakthrough result, Barak [2] shows a constant round non-black-box concurrent zero-knowledge protocol, which however is very inefficient in practice.

If one is willing to augment the computational model with a common reference string, Damgård [17] shows how to construct very efficient 3-round protocols which are concurrent (black-box) zero-knowledge.

However all these works focus only on the issue of zero-knowledge, where one has to prove that a verifier who may engage with several provers in a concurrent fashion, does not learn any information. Our work focuses more on the issue of *malleability* in proofs of knowledge, i.e. making sure that a man-in-the-middle who may start concurrent sessions really knows the witness he claims to know.

The first non-malleable zero-knowledge proof was presented in [23]. This protocol, however, is only *sequentially* non-malleable, i.e. the adversary can only start sessions sequentially (and non concurrently) with the prover. Barak in [3] shows a constant round non-malleable ZK proof in the non-black-box model (and thus very inefficient).

Using the taxonomy introduced by Lindell [36], we can think of concurrent composition as the most general form of composition of a protocol *with itself* (i.e. in a world where only this protocol is run). On the other hand it would be desirable to have protocols that arbitrarily compose, not only with themselves, but with any other "secure" protocol in the environment they run in. This is the notion of *universal composable security* as defined by Canetti [11]. Universally composable zero-knowledge protocols are in particular concurrently non-malleable. In the common reference string model (which is necessary as proven in [11]), a UCZK protocols for Hamiltonian Cycle was presented in [12]. Thus UCZK protocols for any NP problem can be constructed, but they are usually inefficient in practice since they require a reduction to the Hamiltonian Cycle problem.

As it turns out, the common reference string model is necessary also to achieve concurrent non-malleability (see [37]). In this model, the first theoretical solution to our problem was presented in [19]. Following on the ideas presented in [19] more efficient solutions were presented in [32, 27, 38]. Katz [32] presents efficient proofs of plaintext knowledge (a special type of proofs of knowledge) which are sequentially non-malleable. These proofs can be made concurrently non-malleable if one makes some limitations on the power of the adversary in arbitrarily delaying messages, (usually called *timing assumptions*). Finally, reasonably efficient proofs of knowledge, which are concurrently non-malleable are presented by Garay, MacKenzie and Yang in [27, 38]. A detailed comparison between our work and [32, 27, 38] appears below.

COMPARISON WITH [32, 27]. Our result uses Katz's protocol [32] as a starting point and modifies it to be left-concurrently secure without timing assumptions. The most important change is to use multi-trapdoor commitments which will allows us to defeat a concurrent man-in-the-middle[2].

Garay *et al.* in [27] introduce the notion of *simulation-sound* commitments (SSC), which was later refined and improved in [38]. They show generic constructions of SSC and specific direct constructions based on the Strong RSA Assumption and the security of the DSA signature algorithm. They use SSC's to compile (in a way similar to ours) any Σ-protocol into one which is left-concurrently non-malleable. Also they introduce Ω-protocols which dispense of the need for rewinding when extracting and thus can be proven to be left and right-concurrently non-malleable (and with some extra modification even universally composable).

The concept of SSC is related to ours, though we define a notion of commitment which is in some ways stronger and in some others weaker than SSC (we elaborate on the difference in Section 4). The end result is that multi-trapdoor commitments are less general, but more efficient than SSC. Indeed our Strong RSA solution is a factor of 2 faster than the one presented in [38]. This efficiency improvement is inherited by the concurrently non-malleable proof of knowledge, since in both cases the computation of the commitment is the whole overhead.

It should be noted that if we apply our transformation to the so-called Ω-protocols introduced by [27], then we obtain on-line extraction under both left and right concurrency. However we do not know how to construct efficient direct constructions of Ω-protocols for any relationship, except knowledge of discrete logarithms, and even that is not particularly efficient. Since for the applications we had in mind left-concurrency was sufficient, we did not follow this path in this paper.

1.3 Organization of the paper

In the next section we recall the notion of proofs of knowledge and the definition of non-malleability under concurrent composition. We also recall some cryptographic tools that we will use in our main

[2]Some of the other modifications we make (which borrow from Damgård's techniques in [17]) result in a slightly more efficient protocol even in the non-concurrent case, in terms of communication complexity.

construction. In Section 3 we recall non-malleable trapdoor.

Our main contribution is presented in Section 4 where we introduce the notion of multi-trapdoor commitments and present specific number-theoretic constructions. This is followed by Section 5 where we show how to build non-malleable commitments from multi-trapdoor ones.

Section 6 describes our main protocol and its proof of security. In Section 7 we instantiate the generic protocol with the Strong RSA based commitment scheme, and we introduce a few number-theoretic tricks to improve the efficiency.

In Section 8 we present the main application of our result.

2 Preliminaries

In the following we say that function $f(n)$ is negligible if for every polynomial $Q(\cdot)$ there exists an index n_Q such that for all $n > n_Q$, $f(n) \leq 1/Q(n)$.

In the course of the paper we will refer to probabilistic polynomial time Turing machines as *efficient* algorithms.

Also if $A(\cdot)$ is a randomized algorithm, with $a \leftarrow A(\cdot)$ we denote the event that A outputs the string a. With $Prob[A_1; \ldots; A_k : B]$ we denote the probability of event B happening after $A_1, \ldots, A_k$.

2.1 One-time Signatures

Our construction requires a one-time signature scheme which is secure against chosen message attack. Informally this means that the adversary is given the public key and the signature on a message of her choice (chosen after seeing the public key). Then it is infeasible for the adversary to compute the signature of a different message. The following definition is adapted from [30].

Definition 1 $(\mathsf{SG}, \mathsf{Sig}, \mathsf{Ver})$ *is a* one-time secure signature *if for every probabilistic polynomial time forger $\mathcal{F}$, the following*

$$Prob\left[\begin{array}{c} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SG}(1^n) \ ; \ M \leftarrow \mathcal{F}(\mathsf{vk}) \ ; \\ \mathsf{sig} \leftarrow \mathsf{Sig}(M, \mathsf{sk}) \ ; \ \mathcal{F}(M, \mathsf{sig}, \mathsf{vk}) = (M', \mathsf{sig}') \ : \\ \mathsf{Ver}(M', \mathsf{sig}', \mathsf{vk}) = 1 \ \ \mathit{and} \ \ M \neq M' \end{array}\right]$$

is negligible in n.

The main construction requires the signature scheme to also be *strong* (though some of the applications do not require this). Informally this means that it is infeasible for the adversary to also find a *different* signature on the message on which she received one signature already[3]. This is an issue if the signature is randomized, and a message may have many valid signatures.

Definition 2 $(\mathsf{SG}, \mathsf{Sig}, \mathsf{Ver})$ *is a* strong *one-time secure signature if for every probabilistic polynomial time forger $\mathcal{F}$, the following*

$$Prob\left[\begin{array}{c} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SG}(1^n) \ ; \ M \leftarrow \mathcal{F}(\mathsf{vk}) \ ; \\ \mathsf{sig} \leftarrow \mathsf{Sig}(M, \mathsf{sk}) \ ; \ \mathcal{F}(M, \mathsf{sig}, \mathsf{vk}) = (M', \mathsf{sig}') \ : \\ \mathsf{Ver}(M', \mathsf{sig}', \mathsf{vk}) = 1 \ \ \mathit{and} \\ (M \neq M' \ \ \mathit{or} \ \ \mathsf{sig} \neq \mathsf{sig}') \end{array}\right]$$

is negligible in n.

[3]The non-malleable protocols in [32, 27] also require one-time strong signatures, but the strong requirement was actually overlooked.

One-time signatures can be constructed more efficiently than general signatures since they do not require public key operations (see [7, 8, 35]). Virtually all the efficient one-time signature schemes are strong.

2.2 The Strong RSA Assumption.

Let N be the product of two primes, $N = pq$. With $\phi(N)$ we denote the Euler function of N, i.e. $\phi(N) = (p-1)(q-1)$. With Z_N^* we denote the set of integers between 0 and $N-1$ and relatively prime to N.

Let e be an integer relatively prime to $\phi(N)$. The RSA Assumption [43] states that it infeasible to compute e-roots in Z_N^*. I.e. given a random element $s \in_R Z_N^*$ it is hard to find x such that $x^e = s \bmod N$.

The Strong RSA Assumption (introduced in [4]) states that given a random element s in Z_N^* it is hard to find $x, e \neq 1$ such that $x^e = s \bmod N$. The assumption differs from the traditional RSA assumption in that we allow the adversary to freely choose the exponent e for which she will be able to compute e-roots.

We now give formal definitions. Let $RSA(n)$ be the set of integers N, such that N is the product of two $n/2$-bit primes.

Assumption 1 *We say that the RSA Assumption holds, if for all probabilistic polynomial time adversaries $\mathcal{A}$ the following probability*

$$Prob[\, N \leftarrow RSA(n)\ ;\ e \leftarrow Z_{\phi(N)}^*\ ;\ s \leftarrow Z_N^*\ :\ \mathcal{A}(N, s, e) = x \ \ s.t. \ \ x^e = s \bmod N\,]$$

is negligible in n. We say that the Strong RSA Assumption holds, if for all probabilistic polynomial time adversaries $\mathcal{A}$ the following probability

$$Prob[\, N \leftarrow RSA(n)\ ;\ s \leftarrow Z_N^*\ :\ \mathcal{A}(N, s) = (x, e) \ \ s.t. \ \ x^e = s \bmod N\,]$$

is negligible in n.

The RSA Assumption can be strengthened to make it hold for a specific e, rather than a randomly chosen one.

A more efficient variant of our protocol requires that N is selected as the product of two *safe* primes, i.e. $N = pq$ where $p = 2p' + 1$, $q = 2q' + 1$ and both p', q' are primes. We denote with $SRSA(n)$ the set of integers N, such that N is the product of two $n/2$-bit safe primes. In this case the assumptions above must be restated replacing $RSA(n)$ with $SRSA(n)$.

2.3 The Strong Diffie-Hellman Assumption

We now recall the Strong Diffie-Hellman (SDH) Assumption, recently introduced by Boneh and Boyen in [9].

Let G be cyclic group of prime order q, generated by γ. The SDH Assumption can be thought as an equivalent of the Strong RSA Assumption over cyclic groups. It basically says that no attacker on input $G, \gamma, \gamma^y, \gamma^{y^2}, \gamma^{y^3}, \ldots$, for some random $y \in Z_q$, should be able to come up with a pair (e, ξ) such that $\xi^{y+e} = \gamma$.

Assumption 2 *We say that the ℓ-SDH Assumption holds over a cyclic group G of prime order q generated by γ, if for all probabilistic polynomial time adversaries $\mathcal{A}$ the following probability*

$$Prob[\ y \leftarrow Z_q\ :\ \mathcal{A}(\gamma, \gamma^x, \gamma^{x^2}, \ldots, \gamma^{x^\ell}) = (e \in Z_q, \xi \in G)\ \ s.t.\ \ \xi^{y+e} = \gamma\ ]$$

is negligible in $n = |q|$.

Note that, depending on the group G, there may not be an efficient way to determine if $\mathcal{A}$ succeeded in outputting (e, ξ) as above. Indeed in order to check if $\xi^{y+e} = \gamma$ when all we have is γ^{y^i}, we need to solve the Decisional Diffie-Hellman (DDH) problem on the triple $(\gamma^y\gamma^e, \xi, \gamma)$.

Our construction of multi-trapdoor commitments that uses the SDH Assumption requires an efficient way to perform such testing. Thus, although Assumption 2 is well defined on any cyclic group G, we are going to use it on the so-called *gap-DDH* groups, i.e. groups in which there is an efficient test to determine (with probability 1) on input $(\gamma^a, \gamma^b, \gamma^c)$ if $c = ab \bmod q$ or not. Gap-DDH groups where Assumption 2 is believed to hold can be constructed using bilinear maps introduced in the cryptographic literature by [10, 9].

2.4 Definition of Concurrent Proofs of Knowledge

Polynomial Time Relationships. Let $\mathcal{R}$ be a polynomial time computable relationship, i.e. a language of pairs (y, w) such that it can be decided in polynomial time in $|y|$ if $(y, w) \in \mathcal{R}$ or not. With $\mathcal{L}_\mathcal{R}$ we denote the language induced by $\mathcal{R}$ i.e. $\mathcal{L}_\mathcal{R} = \{y\ :\ \exists w\ :\ (y, w) \in \mathcal{R}\}$.

More formally an ensemble of polynomial time relationships $\mathcal{PTR}$ consists of a collection of families $\mathcal{PTR} = \cup_n \mathcal{PTR}_n$ where each $\mathcal{PTR}_n$ is a family of polynomial time relationships $\mathcal{R}_n$. To an ensemble $\mathcal{PTR}$ we associate a randomized *instance generator* algorithm IG that on input 1^n outputs the description of a relationship $\mathcal{R}_n$. In the following we will drop the suffix n when obvious from the context.

The following definition captures the notion of a *hard* relationship, i.e. one in which given $y \in \mathcal{L}_\mathcal{R}$ it is hard to compute the corresponding w such that $(y, w) \in \mathcal{R}$. With $(y, w) \leftarrow \mathcal{R}$ we denote the process of choosing w at random and then computing the matching y.

Definition 3 *We say that an ensemble of polynomial time relationships is* hard *if for every probabilistic polynomial time machine $\mathcal{A}$ we have that*

$$Prob[\ \mathcal{R}_n \leftarrow \mathsf{IG}(1^n)\ ;\ (y, w) \leftarrow \mathcal{R}_n\ :\ \mathcal{A}(y) = w\ ]$$

is negligible in n.

In the following we abuse the notation and we say that $\mathcal{R}$ is hard if the above is satisfied.

Proofs of Knowledge. In a proof of knowledge for a relationship $\mathcal{R}$, two parties, Prover P and Verifier V, interact on a common input y. P also holds a secret input w, such that $(y, w) \in \mathcal{R}$. The goal of the protocol is to convince V that P indeed knows such w. Ideally this proof should not reveal any information about w to the verifier, i.e. be zero-knowledge.

The protocol should thus satisfy certain constraints. In particular it must be *complete*: if the Prover knows w then the Verifier should accept. It should be *sound*: for any (possibly dishonest) prover who does not know w, the verifier should almost always reject. Finally it should be *zero-knowledge*: no (polytime) verifier (no matter what possibly dishonest strategy she follows during the proof) can learn any information about w.

Σ-PROTOCOLS. Many proofs of knowledge belong to a class of protocols called Σ-protocols. These are 3-move protocols for a polynomial time relationship $\mathcal{R}$ in which the prover sends the first message a, the verifier answers with a random challenge c, and the prover answers with a third message z. Then the verifier applies a local decision test on y, a, c, z to accept or not.

Σ-protocols satisfy two special constraints:

Special soundness A cheating prover can only answer *one* possible challenge c. In other words we can compute the witness w from two accepting conversations of the form (a, c, z) and (a, c', z').

Special zero-knowledge Given the statement y and a challenge c, we can produce (in polynomial time) an accepting conversation (a, c, z), with the same distribution of real accepting conversations, without knowing the witness w. Special zero-knowledge implies zero-knowledge with respect to the honest verifier.

All the most important proofs of knowledge used in cryptographic applications are Σ-protocols (e.g. [44, 31]) (see Appendix A for a description of Schnorr's Σ-protocol for knowledge of discrete logarithms.)

We will denote with $a \leftarrow \Sigma_1[y, w]$ the process of selecting the first message a according to the protocol Σ. Similarly we denote $c \leftarrow \Sigma_2$ and $z \leftarrow \Sigma_3[y, w, a, c]$.

MAN-IN-THE-MIDDLE ATTACKS. Consider now an adversary $\mathcal{A}$ that engages with a verifier V in a proof of knowledge. At the same time $\mathcal{A}$ acts as the verifier in another proof with a prover P. Even if the protocol is a proof of knowledge according to the definition in [6], it is still possible for $\mathcal{A}$ to make the verifier accept even without knowing the relevant secret information, but by using P as an oracle. Of course $\mathcal{A}$ could always copy the messages from P to V, but it is not hard to show that she can actually prove even a different statement to V (see for example [32] or Appendix A where we show such an attack for Schnorr's protocol.)

In a *concurrent* attack, the adversary $\mathcal{A}$ is activating several sessions with several provers, in any arbitrary interleaving. We call such an adversary a *concurrent man-in-the-middle*. We say that a proof of knowledge is concurrently non-malleable if such an adversary fails to convince the verifier in a proof in which he does not know the secret information. In other words a proof of knowledge is concurrently non-malleable, if for any such adversary that makes the verifier accept with non-negligible probability we can extract a witness.

Since we work in the common reference string model we define a proof system as tuple $(\mathsf{crsG}, \mathsf{P}, \mathsf{V})$, where crsG is a randomized algorithm that on input the security parameter 1^n outputs the common reference string crs. In our definition we limit the prover to be a probabilistic polynomial time machine, thus technically our protocols are *arguments* and not proofs. But for the rest of the paper we will refer to them as proofs.

If $\mathcal{A}$ is a concurrent man-in-the-middle adversary, let $\pi_{\mathcal{A}}(n)$ be the probability that the verifier V accepts. That is

$$\pi_{\mathcal{A}} = Prob[\ \mathcal{R}_n \leftarrow \mathsf{IG}(1^n)\ ;\ crs \leftarrow \mathsf{crsG}(1^n)\ ;\ [\mathcal{A}^{\mathsf{P}(y_1),\ldots,\mathsf{P}(y_k)}(crs, y), \mathsf{V}(crs, y)] = 1\ ]$$

where the statements $y, y_1, \ldots, y_k$ are adaptively chosen by $\mathcal{A}$. Also with $\mathsf{View}[\mathcal{A}, \mathsf{P}, \mathsf{V}]_{crs}$ we denote the view of $\mathcal{A}$ at the end of the interaction with P and V on common reference string crs.

Definition 4 *We say that* $(\mathsf{crsG}, \mathsf{P}, \mathsf{V})$ *is a concurrently non-malleable proof of knowledge for a relationship* $(\mathcal{PTR}, \mathsf{IG})$ *if the following properties are satisfied:*

Completeness *For all* $(y, w) \in \mathcal{R}_n$ *(for all* $\mathcal{R}_n$*) we have that* $[\mathsf{P}(y, w), \mathsf{V}(y)] = 1$.

Witness Extraction *There exist a probabilistic polynomial time* knowledge extractor KE*, a function* $\kappa : \{0,1\}^* \rightarrow [0,1]$ *and a negligible function* ϵ*, such that for all probabilistic polynomial time concurrent man-in-the-middle adversary* $\mathcal{A}$*, if* $\pi_{\mathcal{A}}(n) > \kappa(n)$ *then* KE*, given rewind access to* $\mathcal{A}$*, computes* w *such that* $(y, w) \in \mathcal{R}_n$ *with probability at least* $\pi_{\mathcal{A}}(n) - \kappa(n) - \epsilon(n)$.

Zero-Knowledge *There exist a probabilistic polynomial time* simulator $\mathsf{SIM} = (\mathsf{SIM}_1, \mathsf{SIM}_\mathsf{P}, \mathsf{SIM}_\mathsf{V})$, *such that the two random variables*

$$Real(n) = [\ crs \leftarrow \mathsf{crsG}(1^n)\ ,\ \mathsf{View}[\mathcal{A}, \mathsf{P}, \mathsf{V}]_{crs}\ ]$$

$$Sim(n) = [\ crs \leftarrow \mathsf{SIM}_1(1^n)\ ,\ \mathsf{View}[\mathcal{A}, \mathsf{SIM}_\mathsf{P}, \mathsf{SIM}_\mathsf{V}]_{crs}\ ]$$

are indistinguishable.

Notice that in the definition of zero-knowledge the simulator does *not* have the power to rewind the adversary. This will guarantee that the zero-knowledge property will hold in a concurrent scenario. Notice also that the definition of witness extraction assumes only left-concurrency (i.e. the adversary has access to many provers but only to one verifier).

3 Trapdoor Commitment Schemes

In this section we recall the notion of trapdoor commitment scheme, both in the "plain" and *non-malleable* versions.

A trapdoor commitment scheme allows a sender to commit to a message with information-theoretic privacy. I.e., given the transcript of the commitment phase the receiver, even with infinite computing power, cannot guess the committed message better than at random. On the other hand when it comes to opening the message, the sender is only computationally bound to the committed message. Indeed the scheme admits a *trapdoor* whose knowledge allows to open a commitment in any possible way (we will refer to this also as *equivocate* the commitment). This trapdoor, or ever just the equivocation for a single commitment, should be hard to compute efficiently.

More fomally a (non-interactive) trapdoor commitment scheme consists of the following algorithms:

- KG, key generation algorithm. On input the security parameter 1^k, it outputs a pair pk, tk where pk is the public key and tk is called the *trapdoor* (or secret key).

- Com, the commitment algorithm. On input pk and a message M and R a random string in $\{0,1\}^k$, it computes $[C(M), D(M)] = \mathsf{Com}(\mathsf{pk}, M, R)$. $C(M)$ is the commitment string and is the output, while $D(M)$ is the opening of $C(M)$ and is kept secret.

- Ver, the verification algorithm. On input pk, a message M and two strings C, D it outputs 1 or 0.

- Equiv is the algorithm that opens a commitment in any possible way given the trapdoor information. It takes as input the public key pk, a commitment $C(M)$ and its opening $D(M)$, a message $M' \neq M$ and a string T. If $T = \mathsf{tk}$ then Equiv outputs D' such that $[C(M), D'] = \mathsf{Com}(\mathsf{pk}, M', R')$ for some R', with the same distribution as if R' has been chosen at random.

We require the following properties. Assuming pk is chosen according to the distributions induced by KG:

Correctness For all messages M, if $[C(M), D(M)] = \mathsf{Com}(\mathsf{pk}, M, R)$, then $\mathsf{Ver}(\mathsf{pk}, M, C(M), D(M)) = 1$.

Information Theoretic Secrecy For every message pair M, M' the distributions $C(M)$ and $C(M')$ are statistically close.

Secure Binding Let $\mathcal{A}$ be an algorithm that on input pk wins if it outputs strings C, M, D, M', D' such that $M \neq M'$, and $\mathsf{Ver}(\mathsf{pk}, M, C, D) = \mathsf{Ver}(\mathsf{pk}, M', C, D') = 1$. Then for all efficient algorithms $\mathcal{A}$ the probability that $\mathcal{A}$ wins is negligible in the security parameter.

3.1 Non-Malleable Trapdoor Commitments

We now recall the definition of (reusable) non-malleable commitments from [18]. The informal, intuitive, security condition for non-malleable commitments, is that the adversary should not be able to commit to messages that are somewhat related to messages which are contained in commitments produced by honest parties.

We are going to focus on a definition in which the adversary sees many commitments, and tries to output a commitment to a related message. This is what Damgård and Groth term *reusability.* Surprisingly this definition is not equivalent to the one in which the adversary sees only one commitment. In the rest of the paper we will omit the word "reusable" and all non-malleable commitments have to be intendended with respect to this stronger definition.

Think of the following game. The adversary, after seeing a tuple of commitments produced by honest parties, outputs his own tuple of committed values. At this point the honest parties decommit their values and now the adversary tries to decommit his values in a way that his messages are related to the honest parties' ones[4]. Intuitively, we say that a commitment scheme is non-malleable if the adversary fails at this game.

However the adversary could succeed by pure chance, or because he has some a priori information on the distribution of the messages committed by the honest parties. So when we formally define non-malleability for commmitments we need to focus on ruling out that the adversary receives any help from seeing the committed values. This can be achieved by comparing the behavior of the adversary in the above game, to the one of an adversary in a game in which the honest parties' messages are not committed to and the adversary must try to output related messages without any information about them.

Let's try to be more formal. We have a publicly known distribution $\mathcal{M}$ on the message space and a randomly chosen public key pk (chosen according to the distribution induced by KG). We assume that

Define Game 1 (the real game) as follows. We think of the adversary $\mathcal{A}$ as two separate efficient algorithms $\mathcal{A}_1, \mathcal{A}_2$. We choose t messages according to this distribution, compute the corresponding commitments and feed them to the adversary $\mathcal{A}_1$. The adversary $\mathcal{A}_1$ outputs a

[4]We are considering *non-malleability with respect to opening* in which the adversary is allows to see the decommitted values, and is required to produce a related decommittment. A stronger security definition *(non-malleability with respect to commitment*) simply requires that the adversary cannot produce a commitment to a related message after being given just the committed values of the honest parties. However for information-theoretic commitments (like the ones considered in this paper) the latter definition does not make sense. Indeed information-theoretic secrecy implies that given a commitment string, *any* message could be a potential decommitment. What specifies the meaning of the commitment is a valid opening of it.

vector of u commitments, with the only restriction that he cannot copy any of the commitments presented to him. $\mathcal{A}_1$ also transfers some internal state to $\mathcal{A}_2$. We now open our commitments and run $\mathcal{A}_2$, who will open the u commitments prepared by $\mathcal{A}$ (if $\mathcal{A}_2$ refuses to open some commitment we replace the opening with $\perp$). We then invoke a distinguisher $\mathcal{D}$ on the two vectors of messages. $\mathcal{D}$ will decide if the two vectors are related or not (i.e. $\mathcal{D}$ outputs 1 if the messages are indeed related). We denote with $\mathsf{Succ1}_{\mathcal{D},\mathcal{A},\mathcal{M}}$ the probability that $\mathcal{D}$ outputs 1 in this game, i.e.

$$\mathsf{Succ1}_{\mathcal{D},\mathcal{A},\mathcal{M}}(k) = Prob \left[\begin{array}{c} \mathsf{pk}, \mathsf{tk} \leftarrow \mathsf{KG}(1^k) \; ; \; m_1, \ldots, m_t \leftarrow \mathcal{M} \\ r_1, \ldots, r_t \leftarrow \{0,1\}^k \; ; \; [c_i, d_i] \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_i, r_i) \\ (\omega, \hat{c}_1, \ldots, \hat{c}_u) \leftarrow \mathcal{A}_1(\mathsf{pk}, c_1, \ldots, c_t) \text{ with } \hat{c}_j \neq c_i \forall i, j \\ (\hat{m}_1, \hat{d}_1, \ldots, \hat{m}_u, \hat{d}_u) \leftarrow \mathcal{A}_2(\mathsf{pk}, \omega, m_1, d_1, \ldots, m_t, d_t) \\ \text{s.t. } \mathsf{Ver}(\mathsf{pk}, m_i, \hat{c}_i, \hat{d}_i) = 1 \text{ or } \hat{m}_i = \perp \\ \mathcal{D}(m_1, \ldots, m_t, \hat{m}_1, \ldots, \hat{m}_u) = 1 \end{array} \right]$$

Define now Game 2 as follows. We still select t messages according to $\mathcal{M}$ but this time feed nothing to the adversary $\mathcal{A}$. The adversary now has to come up with u messages on its own. Again we feed the two vectors of messages to $\mathcal{D}$ and look at the output. We denote with $\mathsf{Succ2}_{\mathcal{D},\mathcal{A}}$ the probability that $\mathcal{D}$ outputs 1 in this game, i.e.

$$\mathsf{Succ2}_{\mathcal{D},\mathcal{A},\mathcal{M}}(k) = Prob \left[\begin{array}{c} \mathsf{pk}, \mathsf{tk} \leftarrow \mathsf{KG}(1^k) \; ; \; m_1, \ldots, m_t \leftarrow \mathcal{M} \\ (\hat{m}_1, \ldots, \hat{m}_u) \leftarrow \mathcal{A}(\mathsf{pk}) \\ \text{s.t. } \hat{m}_i \in \mathcal{M} \cup \{\perp\} \\ \mathcal{D}(m_1, \ldots, m_t, \hat{m}_1, \ldots, \hat{m}_u) = 1 \end{array} \right]$$

Finally we say that a distinguisher $\mathcal{D}$ is admissible, if for any input $(m_1, \ldots, m_t, \hat{m}_1, \ldots, \hat{m}_u)$, its probability of outputting 1 does not increase if we change any message $\hat{m}_i$ into $\perp$. This prevents the adversary from artificially "winning" the game by refusing to open its commitments.

We say that the commitment scheme is (t, u) (reusably) non-malleable if for every message space distribution $\mathcal{M}$, every efficient admissible distinguisher $\mathcal{D}$, and for every efficient adversary $\mathcal{A}$, there is an efficient adversary $\mathcal{A}'$ such that the following difference

$$\mathsf{Succ1}_{\mathcal{D},\mathcal{A},\mathcal{M}}(k) - \mathsf{Succ2}_{\mathcal{D},\mathcal{A}',\mathcal{M}}(k)$$

is negligible in the security parameter. In other words $\mathcal{A}'$ fares almost as well as $\mathcal{A}$ in outputting related messages.

We show a new construction of reusable non-malleable commitments in Section 5.

4 Multi-Trapdoor Commitments

A *multi-trapdoor* commitment scheme consists of a family of trapdoor commitments. Each scheme in the family is information-theoretically private. We require the following properties from a multi-trapdoor commitment scheme:

1. The family admits a *master* trapdoor whose knowledge allows to open *any* commitment in the family in any way it is desired.

2. Each commitment scheme in the family admits its own specific trapdoor, which allows to equivocate that specific scheme.

3. For any commitment scheme in the family, it is infeasible to open it in two different ways, unless the trapdoor is known. However we do allow the adversary to equivocate on a few schemes in the family, by giving it access to an oracle that opens a given committed value in any desired way. The adversary must selects this schemes, *before* seeing the definition of the whole family. It should remain infeasible for the adversary to equivocate any other scheme in the family.

More formally, a (non-interactive) multi-trapdoor commitment scheme consists of the following algorithms:

- KG, the master key generation algorithm. On input the security parameter it outputs a pair PK, TK where PK is the master public key associated with the family of commitment schemes, and TK is called the *master trapdoor*.

- Sel selects a commitment in the family. On input PK it outputs a specific public key pk that identifies one of the schemes.

- Tkg, the specific trapdoor generation algorithm. On input PK,TK,pk it outputs the specific trapdoor information tk relative to pk.

- Com is the commitment algorithm. On input PK,pk and a message M it computes $[C(M), D(M)] = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R)$ where R are the coin tosses. $C(M)$ is the commitment string and is the output, while $D(M)$ is the opening of $C(M)$ and is kept secret.

- Ver, the verification algorithm. On input PK,pk, a message M and two strings C, D it outputs 1 or 0.

- Equiv, the algorithm that opens a commitment in any possible way given the trapdoor information. It takes as input the keys PK,pk, a commitment $C(M)$ and its opening $D(M)$, a message $M' \neq M$ and a string T. If $T = \mathsf{TK}$ or $T = \mathsf{tk}$ then Equiv outputs D' such that $[C(M), D'] = \mathsf{Com}(\mathsf{pk}, M', R')$ for some R', with the same distribution as if R' has been chosen at random.

We require the following properties. Assume PK and all the pk's are chosen according to the distributions induced by KG and Sel.

Correctness For all messages M, if $[C(M), D(M)] = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R)$, then $\mathsf{Ver}(\mathsf{PK}, \mathsf{pk}, M, C(M), D(M)) = 1$.

Information Theoretic Secrecy For every message pair M, M', and every specific public key pk, the distributions $C(M)$ and $C(M')$ are statistically close.

Secure Binding Consider the following game. The adversary $\mathcal{A}$ selects ℓ strings $(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$. It is then given a public key PK for a multi-trapdoor commitment family, generated with the same distribution as the ones generated by KG. Also, $\mathcal{A}$ is given access to an oracle $\mathcal{EQ}$ (for Equivocator), which is queried on the following string $(C, D) = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R), M, R, \mathsf{pk}$ and a message $M' \neq M$. If $\mathsf{pk} = \mathsf{pk}_i$ for some i, and is a valid public key, then $\mathcal{EQ}$ answers with $D' = \mathsf{Equiv}(\mathsf{PK}, \mathsf{pk}, C, D, \mathsf{tk})$ (i.e. D' such that $[C, D'] = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$) otherwise it outputs *nil*.

We say that $\mathcal{A}$ wins if it outputs $C, M, D, M', D', \mathsf{pk}$ such that $\mathsf{Ver}(\mathsf{PK}, \mathsf{pk}, M, C, D) = \mathsf{Ver}(\mathsf{PK}, \mathsf{pk}, M', C, D') = 1$. $M \neq M'$ and $\mathsf{pk} \neq \mathsf{pk}_i$ for all i. We require that for all efficient algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ wins is negligible in the security parameter.

We can define a stronger version of the **Secure Binding** property by requiring that the adversary $\mathcal{A}$ receives the trapdoors tk_i's matching the public keys pk_i's, instead of access to the equivocator oracle $\mathcal{EQ}$. In this case we say that the multi-trapdoor commitment family is *strong*[5].

Remark: *Comparisons with SSC.* First let us describe how MTC impose weaker requirements than SSC. SSC can be also thought as a family of commitments (one for each *tag* to use the terminology in [38]). In the Secure Binding game, SSC are defined in the following way: the public keys (i.e. tags) on which the adversary is allowed to equivocate are chosen *after* seeing the master public key PK. Clearly our "static" approach in which the adversary fixes the specific public keys in advance, is weaker and it indeed accounts for the factor of 2 speedup of our number-theoretic schemes compared to SSC.

On the other hand MTC require something stronger as we enforce the commitments in the family to be information-theoretically secret, and that the trapdoor allows to equivocate *any* commitment. We also require a *master* trapdoor which is not present in SSC. Simulation-Sound Commitments are defined in a somewhat more general way: the commitments are not necessarily information-theoretically secure, and not all of them can be equivocated. However knowledge of a trapdoor allows the simulator to prepare "fake" commitments that can be later equivocated in any way. It can be shown that this weaker requirement is sufficient for constructions of non-malleable commitments and proofs of knowledge. The net advantage is that you can build SSC out of very weak assumptions such as the mere existence of one-way functions. We do not know how to build MTC's out of one-way functions (and indeed we do not know how to build simple non-interactive trapdoor commitments out of one-way functions).

However there are applications in which the requirement of information-theoretic secrecy and the presence of a master trapdoor seems to be required, see [22].

4.1 A scheme based on the Strong RSA Assumption.

In this section we recall a commitment scheme based on the RSA Assumption. This commitment scheme has been widely used in the literature before (e.g. [14, 16]).

We show how under the Strong RSA Assumption this scheme is actually a multi-trapdoor commitment.

Let N be the product of two large primes p, q. Let e be a prime such that $GCD(e, \phi(N)) = 1$, and s a random element of Z_N^*. The triple (N, s, e) are the public parameters. The commitment scheme is defined over messages in $[1..e-1]$.

We denote the commitment scheme with $\mathsf{Com}_{N,s,e}(\cdot,\cdot)$ and we drop the indices when obvious from the context. To commit to $a \in [1..e-1]$ the sender chooses $r \in_R Z_N^*$ and computes $A = \mathsf{Com}(a, r) = s^a \cdot r^e \bmod N$. To decommit the sender reveals a, r and the previous equation is verified by the receiver.

Proposition 1 *Under the RSA Assumption the scheme* Com *described above is an unconditionally secret, computationally binding trapdoor commitment scheme. The trapdoor is the value* $x = s^{\frac{1}{e}} \bmod N$.

Proof: We first prove that the scheme is unconditionally secret. Given a value $A = s^a \cdot r^e$ we note

[5]This was actually our original definition of multi-trapdoor commitments. Phil MacKenzie suggested the possibility of using the weaker approach of giving access to an equivocator oracle (as done in [38]) and we decided to modify our main definition to the weaker one, since it suffices for our application. However the strong definition may also have applications, so we decided to present it as well.

that for each value $a' \neq a$ there exists a unique value r' such that $A = s^{a'}(r')^e$. Indeed this value is the e-root of $A \cdot s^{-a'}$.

We now show that the scheme is computationally binding under the RSA Assumption. We show that an adversary $\mathcal{A}$ who is able to open the commitment scheme in two ways can be used to compute e-roots. The proof uses Shamir's GCD-trick [45]. Given as input the values (N, s, e) we want to compute integer x such that $x^e = s \bmod N$. We place (N, s, e) as the public parameters of the commitment scheme and run $\mathcal{A}$. The adversary returns a commitment A and two distinct openings of it (a, r) and (a', r'). Thus

$$A = s^a r^e = s^{a'}(r')^e \implies s^{a-a'} = \left(\frac{r'}{r}\right)^e \tag{1}$$

Let $\delta = a - a'$. Since $a, a' < e$ we have that $GCD(\delta, e) = 1$. We can find integers α, β such that $\alpha\delta + \beta e = 1$. Now we can compute

$$s = s^{\alpha\delta+\beta e} = (s^\delta)^\alpha \cdot s^{\beta e} = \left(\frac{r'}{r}\right)^{\alpha e} s^{\beta e} \tag{2}$$

where we used Eq.(1). By taking e-roots on both sides we find that $x = \left(\frac{r'}{r}\right)^\alpha s^\beta$.

Finally we show that if we know x then we can open a commitment (of which we already know one opening), in any way we want. Assume we computed A as $\mathsf{Com}(a, r) = s^a r^e$, and later we want to open it as a'. All we need to do (as show in the proof of unconditional security) is to compute the e-root of

$$A \cdot s^{-a'} = s^{a-a'} \cdot r^e \bmod N$$

which clearly is $x^{a-a'} \cdot r \bmod N$. □

Remark 1: The commitment scheme can be easily extended to any message domain $\mathcal{M}$, by using a collision-resistant hash function H from $\mathcal{M}$ to $[1..e-1]$. In this case the commitment is computed as $\mathsf{Com}(a, r) = s^{H(a)} r^e$. In our application we will use a collision resistant function like SHA-1 that maps inputs to 160-bit integers and then choose e's larger than 2^{160}.

Remark 2: How to make the scheme into a multi-trapdoor commitment. Notice that the scheme above is really a family of commitment schemes, one for each prime e. The master trapdoor is the factorization of N. The specific trapdoor of each scheme is $s^{1/e} \bmod N$.

We only need to show that the Secure Binding condition holds, under the Strong RSA Assumption. Assume we are given a Strong RSA problem istance N, σ. Let's now run the **Secure Binding** game.

The adversary is going to select k public keys which in this case are k primes, $e_1, \ldots, e_k$. We set $s = \sigma^{\prod_{i=1}^k e_i} \bmod N$ and return N, s as the public key of the multi-trapdoor commitment family. Now we need to show how to simulate the oracle $\mathcal{EQ}$. But that's easy, as we know the e_i-roots of s, so we actually know the trapdoor of the schemes in the family identified by e_i.

Assume now that the adversary equivocates a commitment scheme in the family identified by a prime $e \neq e_i$. Using the above observation we can then compute $\rho = s^{1/e}$. In turn this allows us to solve the Strong RSA problem istance N, σ by computing an e-root of σ as follows. Let $E = \prod_{i=1}^k e_i$. Then $GCD(e, E) = 1$ which means that we can find integers α, β such that $\alpha e + \beta E = 1$. Then

$$\sigma = \sigma^{\alpha e + \beta E} = [\sigma^\alpha \rho^\beta]^e$$

We note that our scheme satisfies the *strong* version of the **Secure Binding** property in the definition of multi-trapdoor commitment, since under the Strong RSA assumption, it is still infeasible for the adversary to equivocate a different commitment even when given the trapdoors $s_i = \sqrt[e_i]{\sigma} \bmod N$ of a bunch of commitments in the family.

4.2 A scheme based on the SDH Assumption

Let G be a cyclic group of prime order q generated by g. We assume that G is a *gap-DDH* group, i.e. a group such that deciding Diffie-Hellman triplets is easy. More formally we assume the existence of an efficient algorithm $\mathsf{DDH\text{-}Test}$ which on input a triplet (g^a, g^b, g^c) of elements in G outputs 1 if and only if, $c = ab \bmod q$. We also assume that Assumption 2 holds in G.

The master key generation algorithm selects a random $x \in Z_q$ which will be the master trapdoor. The master public key will be the pair g, h where $h = g^x$ in G. Each commitment in the family will be identified by a specific public key pk which is simply an element $e \in Z_q$. The specific trapdoor tk of this scheme is the value f_e in G, such that $f_e^{x+e} = g$.

To commit to a message $a \in Z_q$ with public key $\mathsf{pk} = e$, the sender runs Pedersen's commitment [41] with bases g, h_e, where $h_e = g^e \cdot h$. I.e., it selects a random $r \in Z_q$ and computes $A = g^a h_e^r$. The commitment to a is the value A.

To open a commitment the sender reveals a and $F = g^r$. The receiver accepts the opening if $\mathsf{DDH\text{-}Test}(F, h \cdot g^e, A \cdot g^{-a}) = 1$.

Proposition 2 *Under the SDH Assumption the scheme described above is a multi-trapdoor commitment scheme.*

Proof: Each scheme in the family is easily seen to be unconditionally secret.

It is not hard to see that opening a commitment in two different ways for a specific e is equivalent to finding f_e. Indeed, assume that we can open a committment $A = g^\alpha$ in two ways $a, F = g^\beta$ and $a', F' = g^{\beta'}$ with $a \neq a'$. The $\mathsf{DDH\text{-}Test}$ tells us that $\alpha - a = \beta(x + e)$ and $\alpha - a' = \beta'(x + e)$, thus $a - a' = (\beta' - \beta)(x + e)$ or

$$g^{(a-a')} = \left(\frac{F'}{F}\right)^{(x+e)} \implies f_e = \left(\frac{F'}{F}\right)^{(a-a')^{-1}}$$

By the same reasoning, if we know f_e and we have an opening F, a and we want to open it as a' we need to set $F' = F \cdot f_e^{a-a'}$.

The proof of the **Secure Binding** property follows from the proof of Lemma 1 in [9], where it is proven that the trapdoors f_e can be considered "weak signatures". In other words the adversary can obtain several $f_{e_1}, \ldots, f_{e_\ell}$ for values $e_1, \ldots, e_\ell$ chosen before seeing the public key g, h, and still will not be able (under the $(\ell + 1)$-SDH) to compute f_e for a new $e \neq e_i$. We recall it for completeness. We are going to show that the family is a *strong* multi-tradpoor commitment scheme.

We want to use such an adversary to contradict Assumption 2. So we assume that γ is a random generator of G and $y \in_R Z_q$. We run on input $a_0, a_1, \ldots, a_{\ell+1} \in G$ where $a_i = \gamma^{y^i}$ in G. We want to compute $e \in Z_q$ and $\xi \in G$ such that $\xi^{y+e} = \gamma$. We run the **Secure Binding** game and the adversary produces ℓ public keys $e_1, \ldots, e_\ell \in Z_q$. Now we need to come up with a master public key and the trapdoors f_{e_j} related to the e_j's.

Consider the polynomial $v(y) = \prod_{i=0}^{l}(y + e_i)$ which we can expand as $v(y) = \sum_{i=0}^{\ell+1} \alpha_i y^i$. We define the master public key as follows:

$$g = \gamma^{v(y)} = \prod_{i=0}^{\ell+1} \gamma^{\alpha_i y^i} = \prod_{i=0}^{\ell+1} a_i^{\alpha_i} \quad \text{and} \quad h = g^y = \gamma^{y v(y)} = \prod_{i=1}^{\ell+1} a_i^{\alpha_{i-1}}$$

According to this master public key we have that the trapdoor f_{e_j} is computed as

$$f_{e_j} = g^{\frac{1}{y+e_j}} = \gamma^{\frac{v(y)}{y+e_j}} = \gamma^{\prod_{i \neq j}(y+e_i)} = \gamma^{\sum_{i=0}^{\ell} \beta_{ij} y^i} = \prod_{i=0}^{\ell} a_i^{\beta_{ij}}$$

for some easily computable coefficients β_{ij}.

At this point the adversary outputs (e, f_e) such that $f_e^{y+e} = g$, where $e \neq e_i$. By the definition of the public key we have that

$$f_e = \gamma^{\frac{v(y)}{y+e}} \tag{3}$$

Using long division we can write $v(y) = \lambda(y)(y+e) + \mu$ for some polynomial $\lambda(y) = \sum_{i=0}^{\ell} \lambda_i y^i$ and $\mu \in Z_q$. Thus we can write the exponent in Equation (3) as

$$\frac{v(y)}{y+e} = \sum_{i=0}^{\ell} \lambda_i y^i + \frac{\mu}{y+e}$$

and thus

$$f_e = \gamma^{\frac{\mu}{y+e}} \prod_{i=0}^{\ell} a_i^{\lambda_i}$$

from where we solve for

$$\xi = \gamma^{\frac{1}{y+e}} = \left(f_e \prod_{i=0}^{\ell} a_i^{-\lambda_i} \right)^{\mu^{-1}}$$

which contradicts Assumption 2 as desired. □

5 Non-Malleable Commitments from Multi-Trapdoor ones

In this Section we present NMCom a (reusable) non-malleable trapdoor commitment based on multi-trapdoor ones.

Definition of NMCom:

- *Key Generation:* The public key of the non-malleable scheme includes three elements: (i) the master public key PK for a a multi-trapdoor commitment scheme; (ii) the description of a one-time signature scheme; (iii) a collision-resistant hash function H from the set of verification keys vk of the one-time signature scheme, to the set of public keys pk in the multi-trapdoor commitment scheme determined by the master public key PK.

  The trapdoor of the scheme is TK the master trapdoor of the multi-trapdoor family.

- *Commitment Algorithm:* To commit to a message M, the sender chooses a key pair (sk, vk) for a one-time signature scheme and computes pk= H(vk). Then the sender computes $[C(M), D(M)]$ = Com(PK, pk,M, R) where R is chosen at random (as prescribed by the definition of Com). The commitment string is vk, $C(M)$.

  To decommit the sender reveals $M, D(M)$ and sig, where sig is the one-time signature on $C(M)$.

- *Verification Algorithm:* On input a commitment vk, C, the receiver accepts the decommitment M, D, sig if after computing pk= H(vk), it holds that Ver(PK, pk, M, C, D) = 1 and the signature is valid.

- *Equivocation Algorithm:* The equivocation algorithm runs on input $M \neq M'$, vk, $C(M), D(M)$, sig a signature on $C(M)$ under vk, and the master trapdoor TK. It runs the equivocation algorithm Equiv from the multi-trapdoor family to compute D' such that $[C(M), D'] =$ Com(PK, pk, M', R') and outputs M', D' and sig.

Theorem 1 NMCom *is a (reusable) non-malleable trapdoor commitment scheme.*

Proof: Recall the definition of non-malleability presented in Section 3.1. Let $\mathcal{A} = [\mathcal{A}_1, \mathcal{A}_2]$ be an adversary for Game 1. Denote with $t_{\mathcal{A}}(k)$ an upper bound on its running time.

We want to build an adversary $\mathcal{A}'$ for the second game such that for every polynomial $P(\cdot)$ we have that

$$\mathsf{Succ1}_{\mathcal{D},\mathcal{A},\mathcal{M}}(k) - \mathsf{Succ2}_{\mathcal{D},\mathcal{A}',\mathcal{M}}(k) \leq \epsilon(k) = 1/P(k)$$

Let's fix such polynomial $P(\cdot)$ and consequently $\epsilon(k)$.

First of all $\mathcal{A}'$ selects a public key for the commitment scheme, and stores the secret key. We recall that for our scheme, the public key includes the master public key PK of the multi-trapdoor commitment scheme. Thus $\mathcal{A}'$ knows TK, and can equivocate any commitment.

$\mathcal{A}'$ selects t messages according to the distribution $\mathcal{M}$ and computes the commitments $c_1, \ldots, c_t$ accordingly. It then feeds the public key PK and the commitments to $\mathcal{A}_1$. Let $\omega, \hat{c}_1, \ldots, \hat{c}_u$ be the output of $\mathcal{A}_1$.

Notice that up to this point, the view of $\mathcal{A}_1$ in this "simulated" game, is identical to the view that $\mathcal{A}_1$ would have had in Game 1, no matter what the hidden messages $m_1, \ldots, m_t$ are. Thus the distribution of $\mathcal{A}_1$'s output will be the same in both games.

Now $\mathcal{A}'$ runs $kt_{\mathcal{A}}(k)\epsilon^{-1}(k)$ copies of $\mathcal{A}_2$. On each copy, the internal state input will be set to the ω output by $\mathcal{A}_1$. However on each copy, $\mathcal{A}'$ will provide a different opening as follows. For each copy, $\mathcal{A}'$ samples t messages from $\mathcal{M}$ and uses its knowledge of TK to equivocate the commitments accordingly.

$\mathcal{A}'$ collects the openings that $\mathcal{A}_2$ provides in all the copies. By looking at the i^{th} position opening, $\mathcal{A}'$ sets $\hat{m}_i$ to the first such opening that is different from $\perp$. If in all the openings, the i^{th} position is $\perp$, $\mathcal{A}'$ sets $\hat{m}_i = \perp$.

Consider each individual commitment created by $\mathcal{A}_1$: we say that such a commitment is *bad*, if the probability that $\mathcal{A}_2$ opens it (i.e. opens to a non $\perp$ value) is less than $\epsilon(k)t_{\mathcal{A}}^{-1}(k)$; otherwise we say it is good. Since $t_{\mathcal{A}}(k)$ is an upper bound on u, we can bound with $\epsilon(k)$ the probability that any bad commitment will be correctly opened by $\mathcal{A}_2$.

On the other hand since $\mathcal{A}'$ runs $kt_{\mathcal{A}}(k)\epsilon^{-1}(k)$ copies of $\mathcal{A}_2$, we know that the probability that it opens correctly all good commitments is overwhelming. Notice that $\mathcal{A}'$ may contain correct openings for some bad commitments as well, but that does not matter as $\mathcal{D}$ is an admissible distinguisher.

Thus except than with probability negligibly close to $\epsilon(k)$, we have that (i) $\mathcal{A}_2$ does not open any bad commitment in the real life Game 1; (ii) $\mathcal{A}'$ opens all good commitments in Game 2. So now all we have to prove is that $\mathcal{D}$ outputs 1 with almost the same probability on both outputs. But for $\mathcal{D}$ to output 1 with substantially more probability in Game 1, rather than in Game 2, it means that for at least one (good) commitment, $\mathcal{A}_2$ must provide a different opening in real life Game 1 than in the simulation done during Game 2.

The Lemma below proves that this happens only with negligible probability, which concludes the proof of the Theorem. □

Lemma 1 *For any adversary $\mathcal{A} = [\mathcal{A}_1, \mathcal{A}_2]$ the following probability*

$$Prob\left[\begin{array}{c} \mathsf{PK}, \mathsf{TK} \leftarrow \mathsf{KG}(1^k)\ ;\ m_1, \ldots, m_t \leftarrow \mathcal{M} \\ r_1, \ldots, r_t \leftarrow \{0,1\}^k\ ;\ [c_i, d_i] \leftarrow \mathsf{Com}_{\mathsf{PK}}(m_i, r_i) \\ (\omega, \hat{c}_1, \ldots, \hat{c}_u) \leftarrow \mathcal{A}_1(\mathsf{PK}, c_1, \ldots, c_t)\ \textit{with}\ \hat{c}_j \neq c_i \forall i, j \\ m'_1, \ldots, m'_t \leftarrow \mathcal{M}\ ;\ d'_i \leftarrow \mathsf{Equiv}(\mathsf{PK}, \mathsf{TK}, c_i, m_i, d_i, m'_i) \\ (\hat{m}_1, \hat{d}_1, \ldots, \hat{m}_u, \hat{d}_u) \leftarrow \mathcal{A}_2(\mathsf{PK}, \omega, m_1, d_1, \ldots, m_t, d_t) \\ (\hat{m}'_1, \hat{d}'_1, \ldots, \hat{m}'_u, \hat{d}'_u) \leftarrow \mathcal{A}_2(\mathsf{PK}, \omega, m'_1, d'_1, \ldots, m'_t, d'_t) \\ \exists i : \hat{m}_i \neq \hat{m}'_i\ \textit{and}\ \mathsf{Ver}(\mathsf{PK}, \hat{m}_i, \hat{c}_i, \hat{d}_i) = \mathsf{Ver}(\mathsf{PK}, \hat{m}'_i, \hat{c}_i, \hat{d}'_i) = 1 \end{array}\right]$$

is negligible in k.

Proof: Let's first translate the meaning of the Lemma in plain English. Consider the following process. Feed $\mathcal{A}_1$ with t commitments, and wait for it to output u commitments of his own. At this point split the process in the two separate processes where we feed $\mathcal{A}_2$ with distinct openings of the t commitments (where in both cases, the messages have been selected at random according to the distribution $\mathcal{M}$). Let $\hat{m}_i$ and $\hat{m}'_i$ (for $i = 1, \ldots, u$) be the openings of $\mathcal{A}$'s commitments in the two processes. Then the probability that there are two *distinct* and *correct* openings is negligible.

The proof is by reduction to the security of the multi-trapdoor commitment family and the security of the one-time signature scheme. Assume that $\mathcal{A}$ has a non-negligible probability of opening one of the u commitments in two different ways, in the two separate processes. We show how to use it to contradict the **Secure Binding** condition in the definition of multi-trapdoor commitment schemes, or the unforgeability of the one-time signature scheme.

We prepare in advance the t one-time signature keys in advance $(\mathsf{sk}_1, \mathsf{vk}_1), \ldots, (\mathsf{sk}_m, \mathsf{vk}_m)$, to use in the t commitments we present to $\mathcal{A}$. Then we compute $\mathsf{pk}_i = H(\mathsf{vk}_i)$ and start the **Secure-Binding** game, thus obtaining a public key PK for the MTC family and access to the equivocator oracle $\mathcal{EQ}$.

Now we run $\mathcal{A}_1$ on PK,H. We sample t messages according to the distribution $\mathcal{M}$ and compute their commitments using the vk_i chosen in advance (and thus using the resulting pk_i's). Let $[\mathsf{vk}_i, c_i]$ be the resulting commitment strings. At this point $\mathcal{A}_1$ will output u commitments.

Now we simulate the two processes. In one we open the commitments correctly. In the other one, we sample t more messages according to $\mathcal{M}$ and use the equivocator oracle $\mathcal{EQ}$ to open the commitments accordingly.

By hypothesis, with non-negligible probability $\mathcal{A}_2$ will open one of the u commitments in two different ways in the two processes. Let $[\widehat{\mathsf{vk}}, \hat{c}]$ be this commitment string.

Assume for now that the key $\widehat{\mathsf{vk}}$ is different from all the keys vk_i we used. Then $\widehat{\mathsf{pk}} \neq \mathsf{pk}_i$ for all $i = 1, \ldots, t$ (by the collision-resistance of H). Thus we have contradicted the **Secure Binding** condition in the definition of multi-trapdoor commitments.

Assume now that $\widehat{\mathsf{vk}} = \mathsf{vk}_j$ for some j. Recall, however, that $\mathcal{A}_1$ is not allowed to copy any of our commitments, so it must be that $\hat{c} \neq c_j$. But since $\mathcal{A}_2$ successfully decommits, it must produce a valid signature using vk_i. But then we can use it to break the one-time signature scheme as follows.

We run on input a verification key vk. We are given access to a signing oracle which will sign a single message for us using the matching secret key sk. Our goal is to come up with a valid signature for a different message. We run the above procedure, except that we choose j at random between 1 and t, and we set the corresponding verification key $\mathsf{vk}_j = \mathsf{vk}$.

We can now produce the t commitments for $\mathcal{A}_1$ which will respond with his own. With probability $1/t$ then $\mathcal{A}_1$ will chose $\widehat{\mathsf{vk}} = \mathsf{vk}$. Now we only run one of the two processes (no need to use the equivocator oracle) and use the signature oracle to produce a correct opening of the jth commitment.

When $\mathcal{A}_2$ decommits, since $\hat{c} \neq \hat{c}_j$ we have a different message signed under vk as desired, thus contradicting the security of the one-time signature scheme. □

6 Non-malleable Proofs of Knowledge

In this section we describe our construction of non-malleable proofs of knowledge secure under concurrent composition using multi-trapdoor commitments.

Informal Description. We start from a Σ-protocol as described in Section 2. That is the prover P wants to prove to a verifier V that he knows a witness w for some statement y. The prover sends a first message a. The verifier challenges the prover with a random value c and the prover answers with his response z.

We modify this Σ-protocol in the following way. We assume that the parties share a common reference string that contains the master public key PK for a multi-trapdoor commitment scheme. The common reference string also contains a collision-resistant hash function H from the set of verification keys vk of the one-time signature scheme, to the set of public keys pk in the multi-trapdoor commitment scheme determined by the master public key PK.

The prover chooses a key pair $(\mathsf{sk}, \mathsf{vk})$ for a one-time strong signature scheme. The prover computes $\mathsf{pk}= H(\mathsf{vk})$ and $A = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, a, r)$ where a is the first message of the Σ-protocol and r is chosen at random (as prescribed by the definition of Com). The prover sends vk, A to the verifier. The crucial trick is that we use the verification key vk to determine the value pk used in the commitment scheme.

The verifier sends the challenge c. The prover sends back a, r as an opening of A and the answer z of the Σ-protocol. It also sends sig a signature over the whole transcript, computed using sk. The verifier checks that a, r is a correct opening of A, that sig is a valid signature over the transcript using vk and also that (a, c, z) is an accepting conversation for the Σ-protocol. The protocol is described in Figure 2.

Theorem 2 *If multi-trapdoor commitments exist, if H is a collision-resistant hash function, and if* $(\mathsf{SG}, \mathsf{Sig}, \mathsf{Ver})$ *is a strong one-time signature scheme, then* $\mathsf{CNM\text{-}POK}$ *is a concurrently non-malleable proof of knowledge (see Definition 4).*

Proof: Completeness is obvious. Zero-knowledge follows pretty easily from the results in [17]. We focus on witness extraction.

We build a knowledge extractor to satisfy Definition 4. The extractor runs CKG for the multi-trapdoor commitment scheme to obtain PK, TK. It places PK and the hash function H in the common reference string. Notice that the extractor knows the master trapdoor TK. It then runs the adversary $\mathcal{A}$ on this common reference string. We call this process $\mathsf{Extraction}$.

For each execution that $\mathcal{A}$ starts as a verifier, on input y chosen by $\mathcal{A}$, the extractor does the following, acting as a prover (the label P1 on the step refers to the simulation of the first step by the prover):

P1a chooses a key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SG}(1^n)$ and computes $\mathsf{pk} = H(\mathsf{vk})$

P1b commits to a random value $\hat{a}$ by computing $A = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, \hat{a}, \hat{r})$ for a random $\hat{r}$.

P1c sends A and vk to the adversary.

CNM-POK

Common Reference String: PK the master public key for a multi-trapdoor commitment scheme. A collision resistant hash function H which maps inputs to public keys for the multi-trapdoor commitment scheme determined by PK.

Common Input: A string y.

Private Input for the Prover: a witness w for the statement y, i.e. $(y, w) \in \mathcal{R}$.

- The Prover computes $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SG}(1^n)$; $\mathsf{pk} = H(\mathsf{vk})$; $a \leftarrow \Sigma_1[y, w]$; $r \in_R Z_N^*$; $A = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, a, r)$

  The Prover sends A and vk to the Verifier.

$$\mathsf{P} \xrightarrow{\quad A, \mathsf{vk} \quad} \mathsf{V}$$

- The Verifier selects a random challenge $c \leftarrow \Sigma_2$ and sends it to the Prover.

$$\mathsf{P} \xleftarrow{\quad c \quad} \mathsf{V}$$

- The Prover computes $z \leftarrow \Sigma_3[y, w, a, c]$ and $\mathsf{sig} = \mathsf{Sig}_{\mathsf{sk}}(y, A, c, a, r, z)$. He sends a, r, z, sig to the Verifier.

$$\mathsf{P} \xrightarrow{\quad a, r, z, \mathsf{sig} \quad} \mathsf{V}$$

- The Verifier accepts iff $A = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, a, r)$; $\mathsf{Ver}_{\mathsf{vk}}(y, A, c, a, r, z) = 1$ and $\mathsf{Acc}(y, a, c, z) = 1$.

Figure 2: A Concurrently Non-malleable Proof of Knowledge

The adversary replies with a random challenge c. We show how the extractor answers for the prover:

P2a computes an accepting conversation (a, c, z) using the special HVZK property of Σ protocols.

P2b opens A as (a, r). He can do this since he knows TK, using algorithm Equiv.

P2c computes sig on the transcript (y, A, c, a, r, z).

P2d sends back (a, r, z) and the signature sig.

Consider now the point in which $\mathcal{A}$ acts as a prover again on input y' chosen by her. Let $\pi_{\mathcal{A}}$ be her probability of success. Notice that by definition of Σ-protocol, here the "knowledge error" function κ is $1/\ell$ where ℓ is the size of the challenge set (indeed $\mathcal{A}$ can win a Σ-protocol by guessing the challenge and run the special ZK simulator). We thus assume that $\pi_{\mathcal{A}} > \kappa$ and we need to show that we can extract the witness with probability at least $\pi_{\mathcal{A}} - \kappa - \epsilon$ where ϵ is negligible.

We act as an honest verifier, sending a random challenge c'. This will give us one accepting conversation (a', c', z') for y'. Notice that the view of the adversary during this process is identical to the view of a real execution of the protocol.

However now we rewind $\mathcal{A}$ to the point in which she sent the first message, i.e. the values A', vk'. We now ask a different challenge c''. Notice that the rewinding will not cause any problems in any concurrent session in which the simulator is acting as the prover. Indeed if in one of those sessions $\mathcal{A}$ changes the challenge, the simulator will still be able to proceed as described above (opening the commitment A in the appropriate way).

$\mathcal{A}$ will answer challenge c'' with probability $\pi_{\mathcal{A}} - \kappa$ and if she opens A' in the same way as in the first iteration we obtain another accepting conversation a', c'', z'' which by the special soundness of Σ protocols, will allow us to compute a witness w' for y' as required by the definition.

So the above extraction fails only if $\mathcal{A}$ is able to open a commitment in two ways. The following Proposition 3 shows that the probability ϵ of this event is negligible. Thus the proof of Theorem 2 is complete once Proposition 3 is proven. □

Proposition 3 *If multi-trapdoor commitments exist and if* $(\mathsf{SG}, \mathsf{Sig}, \mathsf{Ver})$ *is a strong one-time signature scheme, then the probability that $\mathcal{A}$ is able to open A' in two different ways during* $\mathsf{Extraction}$ *is negligible.*

Proof: Assume that $\mathcal{A}$ has a non-negligible probability of opening A' in two ways during $\mathsf{Extraction}$. We set up a new procedure (which we call $\mathsf{MTC\text{-}Breaker}$) which will be indistinguishable from $\mathsf{Extraction}$, and which will allow us to contradict the **Secure Binding** condition in the definition of multi-trapdoor commitment schemes.

Let m be a bound on the number of sessions that the adversary will start as a verifier. The simulator will prepare all the one-time signature keys in advance $(\mathsf{sk}_1, \mathsf{vk}_1), \ldots, (\mathsf{sk}_m, \mathsf{vk}_m)$. Then he computes $\mathsf{pk}_i = H(\mathsf{vk}_i)$. It then runs the **Secure-Binding** game and obtains a public key PK and access to the equivocator oracle $\mathcal{EQ}$.

The adversary is run on the common reference string PK, H. The simulation of the prover's steps is identical to $\mathsf{Extraction}$. Here, for each new session, the prover uses a new one-time key $(\mathsf{sk}_i, \mathsf{vk}_i)$. Notice that in this session he can open the commitment A in two ways by querying the equivocator $\mathcal{EQ}$ appropriately.

Thus it should be clear that the views of the adversary during $\mathsf{Extraction}$ and $\mathsf{MTC\text{-}Breaker}$ are indeed indistinguishable. Thus $\mathcal{A}$ will open A' in two ways with the same probability during this process.

Assume for now that the key vk' is different from all the keys vk_i used by the prover. Then $\mathsf{pk}' \neq \mathsf{pk}_i$ for all $i = 1, \ldots, m$ (by the collision-resistance of H). Thus we have contradicted the **Secure Binding** condition in the definition of multi-trapdoor commitments.

So MTC-Breaker succeeds unless $\mathcal{A}$ uses a verification key vk' which is identical to one of the vk_i's and manages to have the verifier accept. The next Proposition 4 shows that this event also has negligible probability. Thus the proof of Proposition 3 is complete once Proposition 4 is proven.

□

Proposition 4 *If* (SG,Sig,Ver) *is a strong one-time signature scheme secure against chosen message-attack, then the probability that* $\mathcal{A}$ *uses* $\mathsf{vk}' = \mathsf{vk}_i$ *for some* i*, and the verifier accepts during* MTC-Breaker *is negligible.*

Proof: Assume that $\mathcal{A}$ succeeds with non-negligible probability in having the verifier accept during MTC-Breaker, while choosing her own verification key vk' equal to vk_i for some i.

We show how to break the strong one-time signature scheme. We run on input a verification key vk. We are given access to a signing oracle which will sign a single message for us using the matching secret key sk. Our goal is to come up with a valid signature for a different message *or* a different signature on the same message asked to the oracle[6].

We run a new procedure (called Forger), which is very similar to Extraction, except that we choose i at random between 1 and m, and we set the corresponding verification key chosen by the prover $\mathsf{vk}_i = \mathsf{vk}$. With probability $1/m$ then $\mathcal{A}$ will chose $\mathsf{vk}' = \mathsf{vk}$.

When the adversary ask the simulator for the third message in the i^{th} session, the simulator queries his signing oracle to compute $\mathsf{sig}_i = \mathsf{Sig}_{\mathsf{sk}}(y_i, A_i, c_i, a_i, r_i, z_i)$.

On the other hand, the adversary will produce a different transcript in her interaction with the verifier. Since $\mathsf{vk}' = \mathsf{vk}_i$, we must have that

$$(y_i, A_i, c_i, a_i, r_i, z_i, \mathsf{sig}_i) \neq (y', A', c', a', r', z', \mathsf{sig}')$$

and sig' is a valid signature on (y', A', c', a', r', z'). We now have two case:

1. $$(y_i, A_i, c_i, a_i, r_i, z_i) \neq (y', A', c', a', r', z')$$

   in which case we have a valid signature on a message different from the one asked to the oracle.

2. $$(y_i, A_i, c_i, a_i, r_i, z_i) = (y', A', c', a', r', z')$$

   but then we must have $\mathsf{sig}_i \neq \mathsf{sig}'$ which means that we found a signature different from the one the oracle returned on the message $(y_i, A_i, c_i, a_i, r_i, z_i)$.

□

[6] This is where the requirement for a *strong* signature scheme arises. This requirement was overlooked in [32, 27]

7 The Strong RSA Version

In this section we are going to add a few comments on the specific implementations of our protocol, when using the number-theoretic constructions described in Sections 4.1 and 4.2. The main technical question is how to implement the collision resistant hash function H which maps inputs to public keys for the multi-trapdoor commitment scheme.

The SDH implementation is basically ready to use "as is". Indeed the public keys pk of the multi-trapdoor commitment scheme are simply elements of Z_q, thus all is needed is a collision-resistant hash function with output in Z_q.

On the other hand, for the Strong RSA based multi-trapdoor commitment, the public keys are prime numbers of the appropriate length.

A prime-outputting collision-resistant hash function is described in [28]. However we can do better than that, by modifying slightly the whole protocol. We describe the modifications (inspired by [39, 16]) in this section.

7.1 Modifying the One-Time Signatures

First of all, we require the one-time signature scheme (SG,Sig,Ver) to have an extra property: i.e. that the distribution induced by SG over the verification keys vk is the uniform one[7]. Virtually all the efficient one-time signature schemes have this property.

Let $\mathcal{H} = \cup_{\{n,k:n>k\}}\mathcal{H}_{n,k}$ be a collection of families of hash functions. If $H \in \mathcal{H}_{n,k}$ then $H : \{0,1\}^n \rightarrow \{0,1\}^k$. We are going to require that $\mathcal{H}$ is both a collision-resistant collection and a collection of families of universal hash functions.

Definition 5 *A collection of families of hash functions $\mathcal{H}$ is a* UCR-collection *if*

1. *H is a collision intractable collection i.e. for every probabilistic polynomial time adversary A the following*

$$Prob[\ H \leftarrow \mathcal{H}_{n,k}\ ;\ A(H) = (x_1, x_2)\ s.t.\ x_1 \neq x_2\ and\ H(x_1) = H(x_2)\ ]$$

   is negligible in k.

2. *for every n, k, we have that $\mathcal{H}_{n,k}$ is a family of universal hash functions, i.e. for all $x_1, x_2 \in \{0,1\}^n$ with $x_1 \neq x_2$, and for all $y_1, y_2 \in \{0,1\}^k$*

$$Prob[\ H \leftarrow \mathcal{H}_{n,k}\ :\ H(x_1) = y_1\ and\ H(x_2) = y_2\ ] = 2^{-2k}$$

Starting from any collision-resistant hash function (such as SHA-1), efficient collections of family can be easily constructed that can be reasonably *conjectured* to be UCR-collections.

Assume that we have a randomly chosen hash function in the UCR-collection $H \in_R \mathcal{H}_{n,k}$ (where n is the length of the verification keys) and a prime $P > 2^{k/2}$.

We modify the key generation of our signature scheme as follows. We run SG repeatedly until we get a verification key vk such that $e = 2P \cdot H(\mathsf{vk}) + 1$ is a prime. Notice that $\ell = |e| > \frac{3}{2}k$. Let us denote with SG' this modified key generation algorithm.

We note the following facts:

[7] This requirement can be relaxed to asking that the distribution has enough min-entropy.

- $H(\mathsf{vk})$ follows a distribution over k-bit strings which is statistically close to uniform; thus using results on the density of primes in arithmetic progressions (see [1], the results hold under the Generalized Riemann Hypothesis) we know that this process will stop in polynomial time, i.e. after an expected ℓ iterations.

- Since e is of the form $2PR+1$, and $P > e^{1/3}$, primality testing of all the e candidates can be done deterministically and very efficiently (see Lemma 2 in [39]).

Thus this is quite an efficient way to associate primes to the verification keys.

What about the security of this modified signature scheme? We are selecting only a subset of all the possible keys of the original signature scheme. Does this make it easier to forge signatures? The answer is no, and it can be easily argued as follows.

If a forger could forge signature on this modified scheme, then the original scheme is not secure as well, since the subset of keys of the modified scheme is a polynomially large fraction of the original universe of keys.

On the length of the primes. In our application we need the prime e to be relatively prime to $\phi(N)$ where N is the RSA modulus used in the protocol. This can be achieved by setting $\ell > n/2$ (i.e. $e > \sqrt{N}$). In typical applications (i.e. $|N| = 1024$) this is about 512 bits (we can obtain this by setting $|P| = 352$ and k, the length of the hash function output, to 160). Since the number of iterations to choose vk depends on the length of e, it would be nice to find a way to shorten it.

If we use safe RSA moduli, then we can enforce that $GCD(e, \phi(N)) = 1$ by choosing e small enough (for 1024-bit safe moduli we need them to be smaller than 500 bits). In this case the collision-resistant property will become the limiting factor in choosing the length. By today's standards we need k to be at least 160. So the resulting primes will be ≈ 240 bits long.

7.2 The full scheme

Informal Description. We start from a Σ-protocol as described in Section 2. That is the prover P wants to prove to a verifier V that he knows a witness w for some statement y. The prover sends a first message a. The verifier challenges the prover with a random value c and the prover answers with his response z.

We modify this Σ-protocol in the following way. We assume that the parties share a common reference string that contains the following:

- an RSA modulus N and $s \in_R Z_N^*$ (which will be used for the commitment scheme described in Section 4.1 (modified following Remark 2);

- H a randomly chosen hash function in a UCR-collection $\mathcal{H}$, with output length k;

- a prime P of the appropriate length: if N is a strong RSA modulus then $|P| > k/2$ otherwise $|P| > |N|/2 - k$.

The prover chooses a key pair $(\mathsf{sk}, \mathsf{vk})$ for a modified one-time strong signature scheme. Recall that this means that the prover chooses $(\mathsf{sk}, \mathsf{vk})$ until $e = 2PH(\mathsf{vk}) + 1$ is a prime. The prover computes $A = \mathsf{Com}_{N,s,e}(a, r)$ where a is the first message of the Σ-protocol and r is chosen at random (as prescribed by the definition of Com). The prover sends vk, A to the verifier. The crucial trick is that we use the verification key vk to determine the value e used in the commitment scheme.

The verifier sends the challenge c. The prover sends back a, r as an opening of A and the answer z of the Σ-protocol. It also sends sig a signature over the whole transcript, computed using sk. The verifier checks that a, r is a correct opening of A, that sig is a valid signature over the transcript using vk and also that (a, c, z) is an accepting conversation for the Σ-protocol. Notice that the verifier does *not* need to check that e is a prime.

The protocol appears in detail in Figure 3.

SRSA-CNM-POK

Common Reference String: An RSA modulus N, and $s \in_R Z_N^*$. A function H from a UCR-collection. A prime P.

Common Input: A string y.

Private Input for the Prover: a witness w for the statement y, i.e. $(y, w) \in \mathcal{R}$.

- The Prover computes $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SG}'(1^n)$; $e = 2P \cdot H(\mathsf{vk}) + 1$; $a \leftarrow \Sigma_1[y, w]$; $r \in_R Z_N^*$; $A = \mathsf{Com}_{N,s,e}(a, r) = s^a r^e \bmod N$.

  The Prover sends A and vk to the Verifier.

  $$\mathsf{P} \xrightarrow{\quad A, \mathsf{vk} \quad} \mathsf{V}$$

- The Verifier selects a random challenge $c \leftarrow \Sigma_2$ and sends it to the Prover.

  $$\mathsf{P} \xleftarrow{\quad c \quad} \mathsf{V}$$

- The Prover computes $z \leftarrow \Sigma_3[y, w, a, c]$ and $\mathsf{sig} = \mathsf{Sig}_{\mathsf{sk}}(y, A, c, a, r, z)$. He sends a, r, z, sig to the Verifier.

  $$\mathsf{P} \xrightarrow{\quad a, r, z, \mathsf{sig} \quad} \mathsf{V}$$

- The Verifier accepts iff $A = \mathsf{Com}_{N,s,e}(a, r)$; $\mathsf{Ver}_{\mathsf{vk}}(y, A, c, a, r, z) = 1$ and $\mathsf{Acc}(y, a, c, z) = 1$.

Figure 3: A Concurrently Non-Malleable Proof of Knowledge based on the Strong RSA Assumption

Theorem 3 *If the Strong RSA Assumption holds, and if* $(\mathsf{SG}, \mathsf{Sig}, \mathsf{Ver})$ *is a strong one-time signature scheme, then* SRSA-CNM-POK *is a concurrently non-malleable proof of knowledge (see Definition 4).*

The proof of Theorem 3 is similar to the proof of Theorem 2, but for completeness it appears in Appendix B.

In Appendix A we present an example of the power and simplicity of our techniques. We show how our transformation can be applied to the Schnorr's proof of knowledge for discrete logarithm.

8 Applications

In this section we describe the main application of our results: concurrently secure identification protocols.

In an identification protocol, a prover, associated with a public key idpk, communicates with a verifier and tries to convince her to be the legitimate prover (i.e. the person knowing the matching secret key idsk.)

An adversary tries to mount an impersonation attack, i.e. tries to make the verifier accept without knowing the secret key sk.

The first definition of security limited the adversary to interact with the real prover only before mounting the actual impersonation attack [25]. Recently a definition against for the more realistic case in which the adversary acts as a "man-in-the-middle" has been proposed [5]. Clearly such an attacker can always relays messages between the prover and the verifier, without changing them. In order to make the definition meaningful, one defines a successful impersonation attack as one with a transcript different from the ones between the attacker and the real prover.

In a concurrent scenario, the prover may engage in several identification sessions, arbitrarily interleaved. In [5] an even more powerful adversary is considered, one that can even *reset* the internal state of the prover. The resulting notion of security implies security in the concurrent model. We do not consider the resettable scenario, but our protocols are more efficient than the ones proposed in [5].

Formally an identification protocol consists of a triple $(\mathsf{IdG}, \mathsf{P}, \mathsf{V})$. IdG is a randomized key generation algorithm: on input the security parameter 1^n it returns a pair of keys $(\mathsf{isk}, \mathsf{ipk})$, respectively the secret and public key. P and V are interactive Turing Machines. The prover runs on input the key pair $(\mathsf{isk}, \mathsf{ipk})$ while the verifier only knows the public key ipk. With $[\mathsf{P}(\mathsf{isk}, \mathsf{ipk}), \mathsf{V}(\mathsf{ipk})]$ we denote the output of the protocol (i.e. 1 iff the verifier accepts).

Definition 6 *We say that* $(\mathsf{IdG}, \mathsf{P}, \mathsf{V})$ *is an* identification scheme secure against concurrent man-in-the-middle attacks *if the following conditions are satisfied:*

Correctness *For all* $(\mathsf{isk}, \mathsf{ipk})$ *output by* $\mathsf{IdG}(1^n)$ *we have that* $[\mathsf{P}(\mathsf{isk}, \mathsf{ipk}), \mathsf{V}(\mathsf{ipk})] = 1$.

Security *For every probabilistic polynomial time adversary* $\mathcal{A}$*, with oracle access to* $\mathsf{P}(\mathsf{isk},\mathsf{ipk})$*, we have that the following probability is negligible in* n

$$Prob[\ (\mathsf{isk}, \mathsf{ipk}) \leftarrow \mathsf{IdG}(1^n)\ :\ [\mathcal{A}^{\mathsf{P}(\mathsf{isk},\mathsf{ipk})}, \mathsf{V}(\mathsf{ipk})] = 1 \ \ and \ \ \pi \notin \Pi\ ]$$

where π *is the transcript of the interaction between* $\mathcal{A}$ *and* V*, and* Π *is the set of transcripts of the interactions between* P *and* $\mathcal{A}$*.*

The basic idea is to use $\mathsf{CNM\text{-}POK}$ with a hard relationship $\mathcal{R}$ as the identification protocol.

Theorem 4 *If the relationship* $\mathcal{R}$ *is hard, multi-trapdoor commitments exist,* $(\mathsf{SG}, \mathsf{Sig}, \mathsf{Ver})$ *is a strong one-time signature scheme, then* $\mathsf{CNM\text{-}POK}$ *is an identification scheme secure against concurrent man-in-the-middle attacks.*

The proof is identical to the proof of Theorem 2. Notice however that we achieve full concurrency here, indeed the extraction procedure in the proof of Theorem 2 does not "care" if there are many other executions in which the adversary is acting as a prover. Indeed we do not need to rewind *all* executions, but only one in order to extract the one witness we need. Thus if there are other such executions "nested" inside the one we are rewinding, we just run them as the honest verifier.

A similar statement holds for $\mathsf{SRSA\text{-}CNM\text{-}POK}$.

9 Conclusions

We have presented a generic transformation that takes any proof of knowledge and makes it secure against a concurrent man-in-the-middle attack.

In its most efficient implementation, the transformation adds only two exponentiations and a very efficient generation of a prime number of a special form, to the computation of the original proof. The transformation is *oblivious* to the structure of original proof of knowledge, i.e. it is the same no matter what proof of knowledge is being performed.

The main tool in the construction is the definition of a new primitive that we call multi-trapdoor commitments and that can have a more general applicability.

Using our results we obtain new efficient concurrently-secure solutions for several interesting problems like identification schemes and deniable authentication. For some applications these are the first known efficient solutions under the Strong RSA Assumption.

Acknowledgments. I would like to thank Hugo Krawczyk since this research originated from several conversations with him about concurrently non-malleable proofs of knowledge.

Although the original work on this paper was done independently from [38], my subsequent reading of that paper, and conversations with Phil MacKenzie about it, improved this paper and in particular the current definition of multi-trapdoor commitments.

Thanks also to Shai Halevi, Jonathan Katz and Yehuda Lindell for helpful conversations and advice and to Dah-Yoh Lim for a careful reading of the paper and catching many typos.

References

[1] E. Bach and J. Shallit. *Algorithmic Number Theory - Volume 1.* MIT Press, 1996.

[2] B. Barak. *How to go beyond the black-box simulation barrier.* Proc. of 42^{nd} IEEE Symp. on Foundations of Computer Science (FOCS'01), pp.106–115, 2001.

[3] B. Barak. *Constant-round Coin Tossing with a Man in the Middle or Realizing the Shared Random String Model.* Proc. of 43^{rd} IEEE Symp. on Foundations of Computer Science (FOCS'02), pp.345–355, 2001.

[4] N. Barić, and B. Pfitzmann. *Collision-free accumulators and Fail-stop signature schemes without trees.* Proc. of EUROCRYPT '97, Springer LNCS 1233, pp.480-494.

[5] M. Bellare, M. Fischlin, S. Goldwasser and S. Micali. *Identification Protocols Secure against Reset Attacks.* Proc. of EUROCYPT'01, Springer LNCS 2045, pp.495–511.

[6] M. Bellare and O. Goldreich. *On defining proofs of knowledge.* Proc. of CRYPTO'92, Springer LNCS 740.

[7] D. Bleichenbacher, U. Maurer. *Optimal Tree-Based One-time Digital Signature Schemes. STACS'96*, Springer LNCS 1046, pp.363–374.

[8] D. Bleichenbacher, U. Maurer. *On the efficiency of one-time digital signatures.* Proc. of ASYACRYPT'96, Springer LNCS 1163, pp.145–158.

[9] D. Boneh and X. Boyen. *Short Signatures without Random Oracles.* Proc. of EUROCRYPT'04, Springer LNCS 3027, pp.382–400.

[10] D. Boneh and M. Franklin. *Identity-Based Encryption from the Weill Pairing.* SIAM J. Comp. 32(3):586–615, 2003.

[11] R. Canetti. *Universally Composable Security: A new paradigm for cryptographic protocols.* Proc. of 42^{nd} IEEE Symp. on Foundations of Computer Science (FOCS'01), pp.136–145, 2001.

[12] R. Canetti and M. Fischlin. *Universally Composable Commitments.* Proc. of CRYPTO'01, Springer LNCS 2139, pp.19–40.

[13] R. Canetti, J. Kilian, E. Petrank and A. Rosen. *Concurrent Zero-Knowledge requires $\tilde{\Omega}(\log n)$ rounds.* Proc. of 33^{rd} ACM Symp. on Theory of Computing (STOC'01), pp.570–579, 2001.

[14] R. Cramer and I. Damgard. *New Generation of Secure and Practical RSA-based signatures.* Proc. of CRYPTO'96, Springer LNCS 1109, pp.173-185.

[15] R. Cramer and V.Shoup. *Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack.* SIAM Journal of Computing 33:167-226, 2003.

[16] R. Cramer and V. Shoup. *Signature schemes based on the Strong RSA assumption.* ACM Transactions on Information and System Security (ACM TISSEC) 3(3):161-185, 2000.

[17] I. Damgård. *Efficient Concurrent Zero-Knowledge in the Auxiliary String Model.* Proc. of EUROCYPT'00, Springer LNCS 1807, pp.174–187.

[18] I. Damgård, J. Groth. *Non-interactive and reusable non-malleable commitment schemes.* Proc. of 35^{th} ACM Symp. on Theory of Computing (STOC'03), pp.426-437, 2003.

[19] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. *Robust Non-Interactive Zero Knowledge.* Proc. of CRYPTO'01, Springer LNCS 2139, pp.566-598.

[20] G. Di Crescenzo, Y. Ishai, R. Ostrovsky. Non-Interactive and Non-Malleable Commitment. Proc. of 30^{th} ACM Symp. on Theory of Computing (STOC'98), pp.141–150, 1998.

[21] G.Di Crescenzo, J. Katz, R. Ostrovsky, A. Smith. *Efficient and Non-interactive Non-malleable Commitment.* Proc. of EUROCRYPT 2001, Springer LNCS 2045, pp.40-59.

[22] M. Di Raimondo and R. Gennaro. *New Approaches to Deniable Authentication.* Manuscript: `http://www.marioland.it/papers/auth.pdf`.

[23] D. Dolev, C. Dwork and M. Naor. *Non-malleable Cryptography.* SIAM J. Comp. 30(2):391–437, 200.

[24] C. Dwork, M. Naor and A. Sahai. *Concurrent Zero-Knowledge.* Proc. of 30^{th} ACM Symp. on Theory of Computing (STOC'98), pp.409–418, 1998.

[25] U. Feige, A. Fiat and A. Shamir. *Zero-Knowledge Proofs of Identity.* J. of Crypt. 1(2):77–94, Springer 1988.

[26] M. Fischlin, R. Fischlin. Efficient *Non-malleable Commitment Schemes.* Proc. of CRYPTO 2000, Springer LNCS 1880, pp.413-431.

[27] J. Garay, P. MacKenzie and K. Yang. *Strengthening Zero-Knowledge Protocols Using Signatures.* Proc. of EUROCRYPT'03, Springer LNCS 2656, pp.177–194. Final version at eprint.iacr.org

[28] R. Gennaro, S. Halevi and T. Rabin. *Secure Hash-and-Sign Signatures Without the Random Oracle.* Proc. of EUROCRYPT'99, Springer LNCS 1592, pp.123–139.

[29] S. Goldwasser, S. Micali, and C. Rackoff. *The knowledge complexity of interactive proof-systems.* SIAM. J. Computing, 18(1):186–208, February 1989.

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

[31] L.C. Guillou and J.J. Quisquater. *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing both Transmission and Memory.* Proc. of EUROCRYPT'88, Springer LNCS 330, pp.123–128.

[32] J. Katz. *Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications.* Proc. of EUROCRYPT'03, Springer LNCS 2656, pp.211-228.

[33] J. Katz. *Efficient Cryptographic Protocols Preventing Man-in-the-Middle Attacks.* Ph.D. Thesis, Columbia University, 2002.

[34] H. Krawczyk and T. Rabin. *Chameleon Hashing and Signatures.* Proceedings of NDSS 2000, pp.143-154.

[35] L. Lamport. *Constructing Digital Signatures from a One-Way Function.* Technical Report SRI Intl. CSL 98, 1979.

[36] Y. Lindell. *Composition of Secure Multi-Party Protocols.* Springer LNCS 2815, 2003.

[37] Y. Lindell. *Lower Bounds for Concurrent Self-Composition.* Proc. of TCC'04, Springer LNCS 2951, pp.203-222.

[38] P. MacKenzie and K. Yang. *On Simulation-Sound Commitments.* Proc. of EUROCRYPT'04, Springer LNCS 3027, pp.382-400.

[39] U. Maurer. *Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters.* J. of Crypt. 8(3):123–156, Springer 1995.

[40] P. Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes.* Proc. of EUROCRYPT'99, Springer LNCS 1592, pp.223-238.

[41] T. Pedersen. *Non-interactive and information-theoretic secure verifiable secret sharing.* Proc of CRYPTO'91, Springer LNCS 576, pp.129–140.

[42] M. Prabhakaran, A. Rosen and A. Sahai. *Concurrent Zero-Knowledge with logarithmic round complexity.* Proc. of 43^{rd} IEEE Symp. on Foundations of Computer Science (FOCS'02), pp.366–375, 2002.

[43] R. Rivest, A. Shamir and L. Adelman. *A Method for Obtaining Digital Signature and Public Key Cryptosystems.* Comm. of ACM, 21 (1978), pp. 120–126

[44] C. P. Schnorr. *Efficient signature generation by smart cards.* J. of Cryptology, 4:161–174, 1991.

[45] A. Shamir. *On the generation of cryptographically strong pseudorandom sequences.* ACM Trans. on Computer Systems, 1(1), 1983, pages 38-44.

A Proof of Knowledge of a discrete logarithm

Recall Schnorr's proof of knowledge for discrete logarithm [44]. On common input p, q, g, y, P wants to convince V that he knows $w : y = g^w \bmod p$. The prover chooses $r \in_R Z_q$ and sends $a = g^r \bmod p$, the verifier sends back a random $c \in_R Z_q$ and the prover answers with $z = r + cw \bmod q$. The verifier accepts iff $g^z = ay^c \bmod p$ (and if $y^q = a^q = 1 \bmod p$).

A MAN-IN-THE-MIDDLE ATTACK: While the honest prover P proves to $\mathcal{A}$ that he knows w, such that $y = g^w \bmod p$, the adversary $\mathcal{A}$ can succesfully prove to V that she knows the discrete log of $y' = yg^{w'}$. Notice that $\mathcal{A}$ does not know such discrete log. The attack is described in Figure 4.

Common Input: $y = g^w \bmod p$ between P and $\mathcal{A}$. $y' = yg^{w'} \bmod p$ between $\mathcal{A}$ and V.

$$\mathsf{P} \xrightarrow{a = g^r} \mathcal{A} \xrightarrow{a' = a} \mathsf{V}$$

$$\mathsf{P} \xleftarrow{c' = c} \mathcal{A} \xleftarrow{c \in_R Z_q} \mathsf{V}$$

$$\mathsf{P} \xrightarrow{z = r + cw} \mathcal{A} \xrightarrow{z' = z + cw'} \mathsf{V}$$

Notice that $g^{z'} = a'(y')^c$, thus V accepts.

Figure 4: A Man-in-the-Middle Attack on Schnorr's protocol

As an example of the power and simplicity of our technique in Figure 5 we show our transformation applied to Schnorr's protocol.
In our analysis we are going to assume that $|q| = 160$, $|p| = |N| = 1024$, that N is a safe RSA moduli so that we can choose $|e| = 240$, that H, h output 160-bit strings. We also estimate that the key generation for the one-time signature scheme (recall that we have to iterate until e is prime) takes pretty much the same amount of time as 1.5 exponentiations with 160-bit integers (this is consistent with experimental results in [16] where generating 160-bit primes in a similar fashion took roughly the same amount of time as one exponentiation – we increase this because we are computing 240-bit primes).

So an easy check shows that our scheme requires only *five* exponentiations with 160-bit exponents, on 1024-bit integers for both prover and verifier (we consider one exponentiation with a 240-bit exponent as 1.5 exponentiations with a 160-bit exponent).

B Proof of Theorem 3

Proof: Completeness is obvious. Zero-knowledge follows pretty easily from the results in [17]. We focus on witness extraction.

We build a knowledge extractor to satisfy Definition 4. The extractor chooses N as the product of two large primes, the value $s \in_R Z_N^*$, the hash function H in the UCR-collection at random, and a prime P of the appropriate length. Notice that the extractor knows the factorization of N. It then runs the adversary $\mathcal{A}$ on the common reference string (N, s, H, P). We call this process $\mathsf{Extraction}$.

CNM-POK-DLOG

Common Reference String: An RSA modulus N, and $s \in_R Z_N^*$. A function H from a UCR-collection. A prime P. A collision-resistant hash h.

Common Input: Two primes p, q such that $q|(p-1)$, an element g of order q in Z_p^*, an element y in the subgroup generated by g.

Private Input for the Prover: an integer $w \in Z_q$ such that $y = g^w \bmod p$.

- The Prover computes $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SG}(1^n)$; $e = 2P \cdot H(\mathsf{vk}) + 1$; $a \leftarrow g^\rho \bmod p$ for $\rho \in_R Z_q$; $r \in_R Z_N^*$; $A = \mathsf{Com}_{N,s,e}(a, r) = s^{h(a)} r^e \bmod N$.

  The Prover sends A and VK to the Verifier.

$$\mathsf{P} \xrightarrow{\quad A, \mathsf{vk} \quad} \mathsf{V}$$

- The Verifier selects a random challenge $c \leftarrow Z_q$ and sends it to the Prover.

$$\mathsf{P} \xleftarrow{\quad c \quad} \mathsf{V}$$

- The Prover computes $z \leftarrow \rho + cw \bmod q$ and $\mathsf{sig} = \mathsf{Sig}_{\mathsf{sk}}(y, A, c, a, r, z)$. He sends a, r, z, sig to the Verifier.

$$\mathsf{P} \xrightarrow{\quad a, r, z, \mathsf{sig} \quad} \mathsf{V}$$

- The Verifier accepts iff $A = \mathsf{Com}_{N,s,e}(a, r) = s^{h(a)} r^e$; $\mathsf{Ver}_{\mathsf{vk}}(y, A, c, a, r, z) = 1$ and $g^z = ay^c \bmod p$.

Figure 5: A Concurrently Non-Malleable Proof of Knowledge for Discrete Logarithms

For each execution that $\mathcal{A}$ starts as a verifier, on input y chosen by $\mathcal{A}$, the extractor does the following, acting as a prover (the label P1 on the step refers to the simulation of the first step by the prover):

P1a chooses a key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SG}'(1^n)$ and computes $e = 2P \cdot H(\mathsf{vk}) + 1$

P1b commits to a random value $\hat{a}$ by computing $A = s^{\hat{a}} \cdot \hat{r}^m$ for a random $\hat{r}$.

P1c sends A and vk to the adversary.

The adversary replies with a random challenge c. We show how the extractor answers for the prover:

P2a computes an accepting conversation (a, c, z) using the special HVZK property of Σ protocols.

P2b opens A as (a, r). He can do this since he knows the factorization of N.

P2c computes sig on the transcript (y, A, c, a, r, z).

P2d sends back (a, r, z) and the signature sig.

When $\mathcal{A}$ decides to act as a prover (again on input y' chosen by her), we just act as an honest verifier, sending a random challenge c'. This will give us one accepting conversation (a', c', z') for y'. Notice that the view of the adversary during this process is identical to the view of a real execution of the protocol.

However now we rewind $\mathcal{A}$ to the point in which she sent the first message, i.e. the values A', vk'. We now ask a different challenge c''. Notice that the rewinding will not cause any problems in any concurrent session in which the simulator is acting as the prover. Indeed if in one of those sessions $\mathcal{A}$ changes the challenge, the simulator will still be able to proceed as described above (opening the commitment A in the appropriate way).

If $\mathcal{A}$ opens A' in the same way as in the first iteration we obtain another accepting conversation a', c'', z'' which by the special soundness of Σ protocols, will allow us to compute a witness w' for y' as required by the definition.

So the above extraction fails only if $\mathcal{A}$ is able to open a commitment in two ways. The following Proposition 5 shows that the probability of this event is negligible. Thus the proof of Theorem 3 is complete once Proposition 5 is proven. □

Proposition 5 *If the Strong RSA Assumption holds and if* $(\mathsf{SG}, \mathsf{Sig}, \mathsf{Ver})$ *is a strong one-time signature scheme, then the probability that* $\mathcal{A}$ *is able to open* A' *in two different ways during* Extraction *is negligible.*

Proof: Assume that $\mathcal{A}$ has a non-negligible probability of opening A' in two ways during Extraction. We set up a new procedure (which we call SRSA-Inverter) which will be indistinguishable from Extraction, and which will allow us to contradict the Strong RSA Assumption.

The inverter runs on input an RSA modulus N and $\sigma \in_R Z_N^*$. His goal is to compute $x, e \neq 1$ such that $x^e = \sigma \bmod N$.

Let m be a bound on the number of sessions that the adversary will start as a verifier. The simulator will prepare all the one-time keys in advance $(\mathsf{sk}_1, \mathsf{vk}_1), \ldots, (\mathsf{sk}_m, \mathsf{vk}_m)$. Then he computes $e_i = 2P \cdot H(\mathsf{vk}_i) + 1$. It then sets $s = \sigma^E \bmod N$ where $E = \prod_{i=1}^k e_i$. Notice that all the e_i's are ℓ-bit primes.

The adversary is run on input N, s. The simulation of the prover's steps is identical to Extraction. Here, for each new session, the prover uses a new one-time key $(\mathsf{sk}_i, \mathsf{vk}_i)$. Notice that in this session he can open the commitment A in two ways as he knows the e_i-root of s.

Thus it should be clear that the view of the adversary during Extraction and SRSA-Inverter are indeed indistinguishable. Thus $\mathcal{A}$ will open A' in two ways with the same probability during this process. Using Proposition 1 we can then compute the e-root of s, where $e = 2P \cdot H(\mathsf{vk}') + 1$, the value associated with the verification key chosen by $\mathcal{A}$. Notice that e might *not* be necessarily a prime. Thus we have a value x' such that $(x')^e = s = \sigma^E$.

Assume for now that the key vk' is different from all the keys vk_i used by the prover. Then $e \neq e_i$ for all $i = 1, \ldots, m$. This implies that $GCD(e, E) = 1$, since $|e| = |e_i|$ for all i and the e_i's are primes. Thus we can find α, β such that $\alpha e + \beta E = 1$. Thus

$$\sigma = \sigma^{\alpha e + \beta E} = \sigma^{\alpha e} \cdot (\sigma^E)^\beta = [\sigma^\alpha \cdot (x')^\beta]^e$$

So we can return $x = \sigma^\alpha \cdot (x')^\beta$ and e as a solution to the Strong RSA challenge N, σ.

So SRSA-Inverter succeeds unless $\mathcal{A}$ uses a verification key vk' which is identical to one of the vk_i's and manages to have the verifier accept. The next Proposition 6 shows that this event also has negligible probability. Thus the proof of Proposition 5 is complete once Proposition 6 is proven. □

Proposition 6 *If* (SG,Sig,Ver) *is a strong one-time signature scheme secure against chosen message-attack, then the probability that $\mathcal{A}$ uses $\mathsf{vk}' = \mathsf{vk}_i$ for some i, and the verifier accepts during* SRSA-Inverter *is negligible.*

Proof: Assume that $\mathcal{A}$ succeeds with non-negligible probability in having the verifier accept during SRSA-Inverter, while choosing her own verification key vk' equal to vk_i for some i.

We show how to break the strong one-time signature scheme. We run on input a verification key vk. We are given access to a signing oracle which will sign a single message for us using the matching secret key sk. Our goal is to come up with a valid signature for a different message *or* a different signature on the same message asked to the oracle[8].

We run a new procedure (called Forger), which is very similar to Simulation, except that we choose i at random between 1 and m, and we set the corresponding verification key chosen by the prover $\mathsf{vk}_i = \mathsf{vk}$. With probability $1/m$ then $\mathcal{A}$ will chose $\mathsf{vk}' = \mathsf{vk}$.

When the adversary ask the simulator for the third message in the i^{th} session, the simulator queries his signing oracle to compute $\mathsf{sig}_i = \mathsf{Sig}_{\mathsf{sk}}(y_i, A_i, c_i, a_i, r_i, z_i)$.

On the other hand, the adversary will produce a different transcript in her interaction with the verifier. Since $\mathsf{vk}' = \mathsf{vk}_i$, we must have that

$$(y_i, A_i, c_i, a_i, r_i, z_i, \mathsf{sig}_i) \neq (y', A', c', a', r', z', \mathsf{sig}')$$

and sig' is a valid signature on (y', A', c', a', r', z'). We now have two case:

1. $$(y_i, A_i, c_i, a_i, r_i, z_i) \neq (y', A', c', a', r', z')$$

   in which case we have a valid signature on a message different from the one asked to the oracle.

[8]This is where the requirement for a *strong* signature scheme arises. This requirement was overlooked in [32, 27]

2.
$$(y_i, A_i, c_i, a_i, r_i, z_i) = (y', A', c', a', r', z')$$
but then we must have $\mathsf{sig}_i \neq \mathsf{sig}'$ which means that we found a signature different from the one the oracle returned on the message $(y_i, A_i, c_i, a_i, r_i, z_i)$.

$\square$