

Multi-trapdoor Commitments and their Applications to Proofs of Knowledge Secure under Concurrent Man-in-the-middle Attacks

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

June 7, 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 application of our result is the construction of a *compiler* that takes *any* proof of knowledge and transforms it into one which is secure against a concurrent man-in-the-middle attack.

When using our Strong RSA construction of multi-trapdoor commitments, this compiler is very efficient (requires no more than four exponentiations) and 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.

Efficient solutions were known only for proofs of knowledge for some specific language, while our transformation works over *any* proof. Moreover, compared to previously known efficient proofs, our solution is a factor of eight more efficient in computation.

The main practical applications of our results are concurrently secure identification and deniable authentication protocols. For these applications our results are the first simple and efficient solutions based on the Strong RSA Assumption or the Strong Diffie-Hellman Assumption.

1 Introduction

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. By securely binding a message to such identification protocols, one also obtains interactive public-key message authentication schemes. An interesting feature of these schemes is that they are *deniable* i.e. while the receiver is guaranteed that the message comes from the sender, the sender can always deny to a third party to have sent it.

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 relays messages unchanged from an honest prover to an honest verifier).

OUR MAIN CONTRIBUTION. 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 to construct such compiler 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 3 in 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 (think of it as putting it inside a sealed envelope on the table) and a second one in which the sender reveals the committed message (opens the envelope).

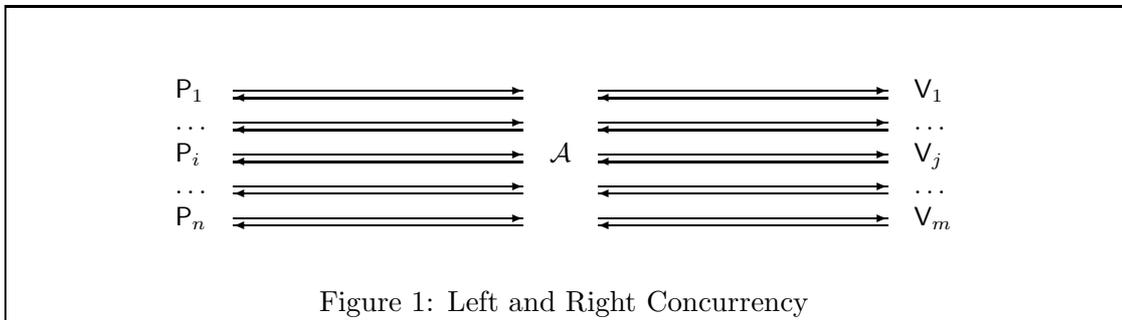
A trapdoor commitment scheme allows a sender to commit to a message with information-theoretic privacy. I.e., given the transcript of the commitment phase the receiver, even with infinite computing power, cannot guess the committed message better than at random. On the other hand when it comes to opening 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.



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 [27]. 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 [32] 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-concurrency¹. This is however enough to build fully concurrently secure applications like identification and deniable authentication protocols.

PRIOR WORK. Zero-knowledge protocols were introduced in [24]. The notion of proof of knowledge (already implicit in [24]) was formalized in [21, 6].

Concurrent zero-knowledge was introduced in [20]. 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

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

bounds on the round complexity of concurrent zero-knowledge in the black-box model [13, 37], 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. security against a man-in-the-middle who may start concurrent sessions.

The problem of malleability in cryptographic algorithms, and specifically in zero-knowledge proofs, was formalized by Dolev *et al.* in [19], where a non-malleable ZK proof with a polylogarithmic number of rounds is presented. 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 [31], 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 [32]). In this model, the first theoretical solution to our problem was presented in [18]. Following on the ideas presented in [18] more efficient solutions were presented in [27, 22, 33]. Katz [27] 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 [22, 33]. A detailed comparison between our work and [27, 22, 33] appears below.

COMPARISON WITH [27, 22]. Our result uses Katz’s protocol [27] as a starting point and modifies it to be left concurrently secure without timing assumptions. The most important change is to use a specific number-theoretic commitment scheme (based on RSA) which will allow us to defeat a concurrent man-in-the-middle. Some of the other modifications we make (which borrow Damgård’s techniques from [17]) result in a slightly more efficient protocol even in the non-concurrent case, in terms of communication complexity.

Garay *et al.* in [22] introduce the notion of *simulation-sound* trapdoor commitments (SSTC), which was later refined and improved in [33]. They show generic constructions of SSTC and specific direct constructions based on the Strong RSA Assumption and the security of the DSA signature algorithm. They use SSTC’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 SSTC is related to ours, though we define a somewhat weaker notion of commitment (we elaborate on the difference in Section 3). The important contribution of our paper with respect to [22, 33] is twofold: (i) we show that this weaker notion is sufficient to construct concurrently non-malleable proofs; (ii) because our notion is weaker, we are able to construct a more efficient number theoretic instantiation. Indeed our Strong RSA construction is about a factor of 2 faster than the one presented in [33]. 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 [22], 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.

Below we show all the applications of this result. For each application we highlight the prior work in that field, and the impact of our result.

1.1 Applications

CONCURRENTLY SECURE IDENTIFICATION SCHEMES. The main application of proofs of knowledge is in the design of secure identification schemes [21]. 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.

For example, by using Schnorr’s identification scheme [39], 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 [26], 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.

DENIABLE AUTHENTICATION. A deniable authentication protocol allows a Sender to authenticate a message M to a Receiver, in a way that while the Receiver is certain that the message M originated with the Sender, at the same time the Receiver cannot convince anybody else of this fact. We are interested in a public-key scenario, in which Sender and Receiver do not share any secret information², and authentication is achieved via a public key associated with the Sender.

We show how our concurrently non-malleable proofs of knowledge can be turned into concurrently secure deniable authentication protocols, by binding the message M to the proof of knowledge in a secure way.

Concurrently secure deniable authentication protocols (without timing assumptions) were presented in [20, 28] using CCA2-secure encryption (either interactive or not). Our technique shows another (somewhat more natural) paradigm for deniable authentication (transforming any proof

²In the shared key model, deniable authentication is easily achieved by using Message Authentication Codes

of knowledge into a deniable authentication protocol). Using this approach we obtain new protocols which are very efficient under new assumption. For example we obtain the first efficient concurrently secure deniable authentication protocol based on the Strong RSA Assumption.

We stress that our solutions for these applications are fully concurrently secure: i.e. tolerate an adversary that starts any concurrent interleaving of protocols execution either as the prover/sender or the verifier/receiver. Indeed it is sufficient for these applications that the proof of knowledge satisfies on-line extraction against left concurrency, which is what our protocols achieves³. The applications are discussed in Section 6.

1.2 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 construction.

In section 3 we introduce the notion of multi-trapdoor commitments and present a specific number-theoretic construction based on the Strong RSA Assumption.

Section 4 describes our main protocol and its proof of security. In Section 5 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 6 we present the main applications 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 $n > n_Q$, $f(n) \leq 1/Q(n)$.

Also if $A(\cdot)$ is a randomized algorithm, with $a \leftarrow A(\cdot)$ we denote the event that A outputs the string a . With $Prob[A_1; \dots; A_k : B]$ we denote the probability of event B happening after A_1, \dots, 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 [25].

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

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

is negligible in n .

³An example of an application which requires on-line extraction against both types of concurrency is interactive chosen-ciphertext security, and thus we are not able to use our results in that case.

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 already⁴. This is an issue if the signature is randomized, and a message may have many valid signatures.

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

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

is negligible in n .

One-time signatures can be constructed more efficiently than general signatures since they do not require public key operations (see [7, 8, 30]). 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 [38] states that it is 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 $x^e = s \pmod N$.

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 $x^e = s \pmod N$. 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 $\text{RSA}(n)$ 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*

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

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

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

is negligible in n .

⁴The non-malleable protocols in [27, 22] also require one-time strong signatures, but the strong requirement was actually overlooked.

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 briefly 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 g . 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 $G, g, g^x, g^{x^2}, g^{x^3}, \dots$, for some random $x \in Z_q$, should be able to come up with a pair (e, h) such that $h^{x+e} = g$.

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

$$\text{Prob}[x \leftarrow Z_q : \mathcal{A}(g, g^x, g^{x^2}, \dots, g^{x^\ell}) = (e \in Z_q, h \in G) \text{ s.t. } h^{x+e} = g]$$

is negligible in $n = |q|$.

Notice that, depending on the group G , there may not be an efficient way to determine if \mathcal{A} succeeded in outputting (e, h) as above. Indeed in order to check if $h^{x+e} = g$ when all we have is g^{x^i} , we need to solve the Decisional Diffie-Hellman (DDH) problem on the triple $(g^x g^e, h, g)$. 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 (g^a, g^b, g^c) if $c = ab \pmod q$ or not. The *gap-DDH* property will also be required by our construction of multi-trapdoor commitments that uses the SDH Assumption⁵.

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 .

⁵Gap-DDH groups where Assumption 2 is believed to hold can be constructed using bilinear maps introduced in the cryptographic literature by [10].

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

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

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. [39, 26]).

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 (see for example [27]) that she can actually prove even a different statement to V .

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 $(\text{crsG}, \text{P}, \text{V})$, 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

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

where the statements y, y_1, \dots, y_k are adaptively chosen by \mathcal{A} . Also with $\text{View}[\mathcal{A}, \text{P}, \text{V}]_{\text{crs}}$ 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 $(\text{crsG}, \text{P}, \text{V})$ is a concurrently non-malleable proof of knowledge for a relationship $(\mathcal{PTR}, \text{IG})$ if the following properties are satisfied:

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

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 $\text{SIM} = (\text{SIM}_1, \text{SIM}_{\text{P}}, \text{SIM}_{\text{V}})$, such that the two random variables

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

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

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 Multi-trapdoor Commitment Schemes

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 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. 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 select these schemes, *before* seeing the definition of the whole family. It should remain infeasible for the adversary to equivocate any other scheme in the family.

The main difference between our definition and the notion of SSTC [22, 33] is that SSTC allow the adversary to choose the schemes in which it wants to equivocate even *after* seeing the definition of the family. Clearly SSTC are a stronger requirement, which is probably why we are able to obtain more efficient constructions.

We now give a formal definition. A (non-interactive) multi-trapdoor commitment scheme consists of five algorithms: CKG, Sel, Tkg, Com, Open with the following properties.

CKG is 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*.

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

Tkg is 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 outputs $C(M) = \text{Com}(\text{PK}, \text{pk}, M, R)$ where R are the coin tosses. To open a commitment the sender reveals M, R and the receiver recomputes C .

Open is 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 M, R , a message $M' \neq M$ and a string T . If $T = \text{TK}$ or $T = \text{tk}$ then Open outputs R' such that $C(M) = \text{Com}(\text{PK}, \text{pk}, M', R')$.

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

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

Secure Binding Consider the following game. The adversary \mathcal{A} selects k strings $(\text{pk}_1, \dots, \text{pk}_k)$. It is then given a public key PK for a multi-trapdoor commitment family, generated with the same distribution as the ones generated by CKG. Also, \mathcal{A} is given access to an oracle \mathcal{EQ} (for Equivocator), which is queried on the following string $C = \text{Com}(\text{PK}, \text{pk}, M, R), M, R, \text{pk}$ and a message $M' \neq M$. If $\text{pk} = \text{pk}_i$ for some i , and is a valid public key, then \mathcal{EQ} answers with R' such that $C = \text{Com}(\text{PK}, \text{pk}, M', R')$ otherwise it outputs *nil*. We say that \mathcal{A} wins if it outputs $C, M, R, M', R', \text{pk}$ such that $C = \text{Com}(\text{PK}, \text{pk}, M, R) = \text{Com}(\text{PK}, \text{pk}, M', R')$, $M \neq M'$ and $\text{pk} \neq \text{pk}_i$ for all i . We require that for all efficient algorithms \mathcal{A} , the probability that \mathcal{A} wins is negligible in the security parameter.

Again the main difference between our notion and SSTCs [22, 33] is that in their case the pk_i were selected by the adversary *after* seeing PK.

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*⁶.

3.1 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 $\text{GCD}(e, \phi(N)) = 1$, 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 $\text{Com}_{N,s,e}(\cdot, \cdot)$ 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 $A = \text{Com}(a, r) = s^a \cdot r^e \bmod N$. 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 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 [40]. 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

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

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

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

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

⁶This was actually our original definition of multi-trapdoor commitments. After reading [33], which follows the weaker approach of giving access to an equivocator oracle, 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.

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 $\text{Com}(a, r) = s^a r^e$, 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

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

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

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 $\text{Com}(a, r) = s^{H(a)} r^e$. 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 $s^{1/e} \pmod N$.

We only need to show that the **Secure Binding** condition holds, under the Strong RSA Assumption. Assume we are given a Strong RSA problem instance 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, \dots, e_k . We set $s = \sigma^{\prod_{i=1}^k e_i} \pmod N$ 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 instance N, σ by computing an e -root of σ as follows. Let $E = \prod_{i=1}^k e_i$. Then $\text{GCD}(e, E) = 1$ which means that we can find integers α, β such that $\alpha e + \beta E = 1$. Then

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

Jumping ahead, this is what will allow us to use multi-trapdoor commitment schemes in several concurrent executions of our protocols, each one identified by a different e_i . Indeed the simulator will set up a public parameter s so that it can always compute the e_i -root of s (this is done by simply setting $s = \sigma^{\prod_i e_i} \pmod N$ for $\sigma \in_R Z_N^*$). On the other hand the adversary will be forced to open a commitment in two ways for a *new* e , different from all the previous e_i 's. This will allow the simulator to compute the e -root of s (and thus of σ via the above GCD trick).

As a final observation, 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 $s_i = \sqrt[e_i]{\sigma} \pmod N$ of a bunch of commitments in the family.

3.2 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 **DDH-Test** which on input a triplet (g^a, g^b, g^c) of elements in G outputs 1 if and only if, $c = ab \pmod q$. We also assume that the 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 \mathbf{pk} which is simply an element $e \in Z_q$. The specific trapdoor \mathbf{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 $\mathbf{pk} = e$, the sender runs Pedersen's commitment [36] 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 $\text{DDH-Test}(F, h \cdot g^e, A \cdot g^{-a}) = 1$.

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

Sketch of Proof: Each scheme in the family is easily seen to be unconditionally secret. 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 $f_{e_1}, \dots, f_{e_\ell}$ for values e_1, \dots, e_ℓ 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 $e \neq e_i$.

The proof is then completed if we can show that opening a commitment in two different ways for a specific e is equivalent to finding f_e .

Assume we can open a commitment $A = g^\alpha$ in two ways $a, F = g^\beta$ and $a', F' = g^{\beta'}$ with $a \neq a'$. The DDH-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 $F' = F \cdot f_e^{a-a'}$. \square

4 The Protocol

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

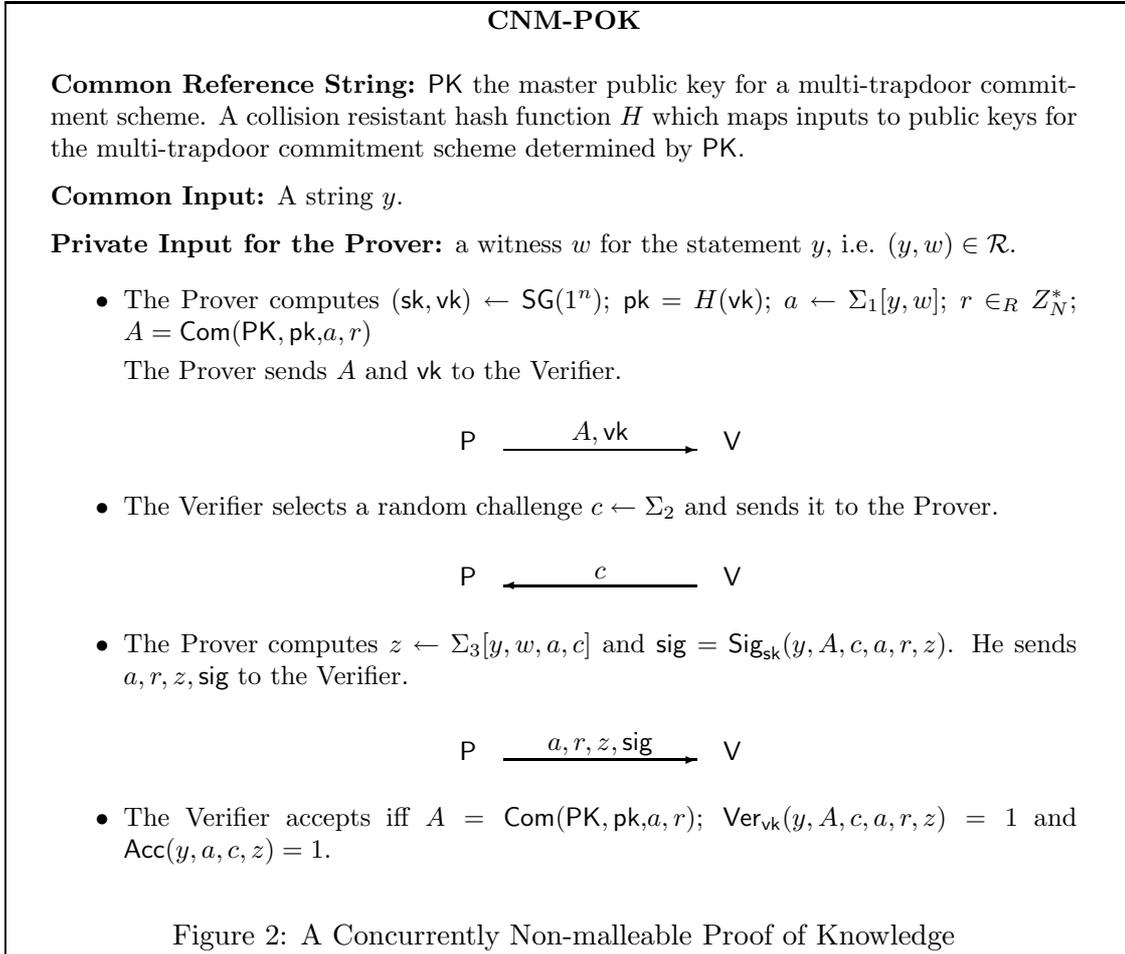
INFORMAL DESCRIPTION. We start from a Σ -protocol as described in Section 2. That is the prover \mathbf{P} wants to prove to a verifier \mathbf{V} that he knows a witness w for some statement y . The prover sends a first message a . The verifier challenges the prover with a random value c and the prover 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 \mathbf{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 \mathbf{vk} of the one-time signature scheme, to the set of public keys \mathbf{pk} in the multi-trapdoor commitment scheme determined by the master public key \mathbf{PK} .

The prover chooses a key pair $(\mathbf{sk}, \mathbf{vk})$ for a one-time strong signature scheme. The prover computes $\mathbf{pk} = H(\mathbf{vk})$ and $A = \text{Com}(\mathbf{PK}, \mathbf{pk}, a, r)$ 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 \mathbf{vk}, A to the verifier. The crucial trick is that we use the verification key \mathbf{vk} to determine the value \mathbf{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 \mathbf{sig} a signature over the whole transcript, computed using \mathbf{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 1 *If multi-trapdoor commitments exist, if H is a collision-resistant hash function, and if $(\text{SG}, \text{Sig}, \text{Ver})$ is a strong one-time signature scheme, then CNM-POK is a concurrently non-malleable proof of knowledge (see Definition 4).*

Sketch of 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 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 $(\text{sk}, \text{vk}) \leftarrow \text{SG}(1^n)$ and computes $\text{pk} = H(\text{vk})$

P1b commits to a random value \hat{a} by computing $A = \text{Com}(\text{PK}, \text{pk}, \hat{a}, \hat{r})$ 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 TK , using algorithm `Open`.

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 3 shows that the probability of this event is negligible. Thus the proof of Theorem 1 is complete once Proposition 3 is proven. \square

Proposition 3 *If multi-trapdoor commitments exist and if $(\text{SG}, \text{Sig}, \text{Ver})$ 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.*

Sketch of 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 `MTC-Breaker`) which will be indistinguishable from 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 $(\text{sk}_1, \text{vk}_1), \dots, (\text{sk}_m, \text{vk}_m)$. Then he computes $\text{pk}_i = H(\text{vk}_i)$. 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 Extraction. Here, for each new session, the prover uses a new one-time key $(\text{sk}_i, \text{vk}_i)$. 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 Extraction and MTC-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 $pk' \neq pk_i$ for all $i = 1, \dots, m$ (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. \square

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

Sketch of 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 oracle⁷.

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 $vk_i = vk$. With probability $1/m$ then \mathcal{A} will chose $vk' = vk$.

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

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

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

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 $\text{sig}_i \neq \text{sig}'$ 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

⁷This is where the requirement for a *strong* signature scheme arises. This requirement was overlooked in [27, 22]

5 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 3.1 and 3.2. 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 [23]. However we can do better than that, by modifying slightly the whole protocol. We describe the modifications (inspired by [34, 16]) in this section.

5.1 Modifying the One-Time Signatures

First of all, we require the one-time signature scheme ($\text{SG}, \text{Sig}, \text{Ver}$) to have an extra property: i.e. that the distribution induced by SG over the verification keys vk is the uniform one⁸. 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*

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

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 $y_1, y_2 \in \{0,1\}^k$*

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

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 $e = 2P \cdot H(\text{vk}) + 1$ 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:

⁸This requirement can be relaxed to asking that the distribution has enough min-entropy.

- $H(\text{vk})$ 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 [34]).

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 $\text{GCD}(e, \phi(N)) = 1$ 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.

5.2 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 .

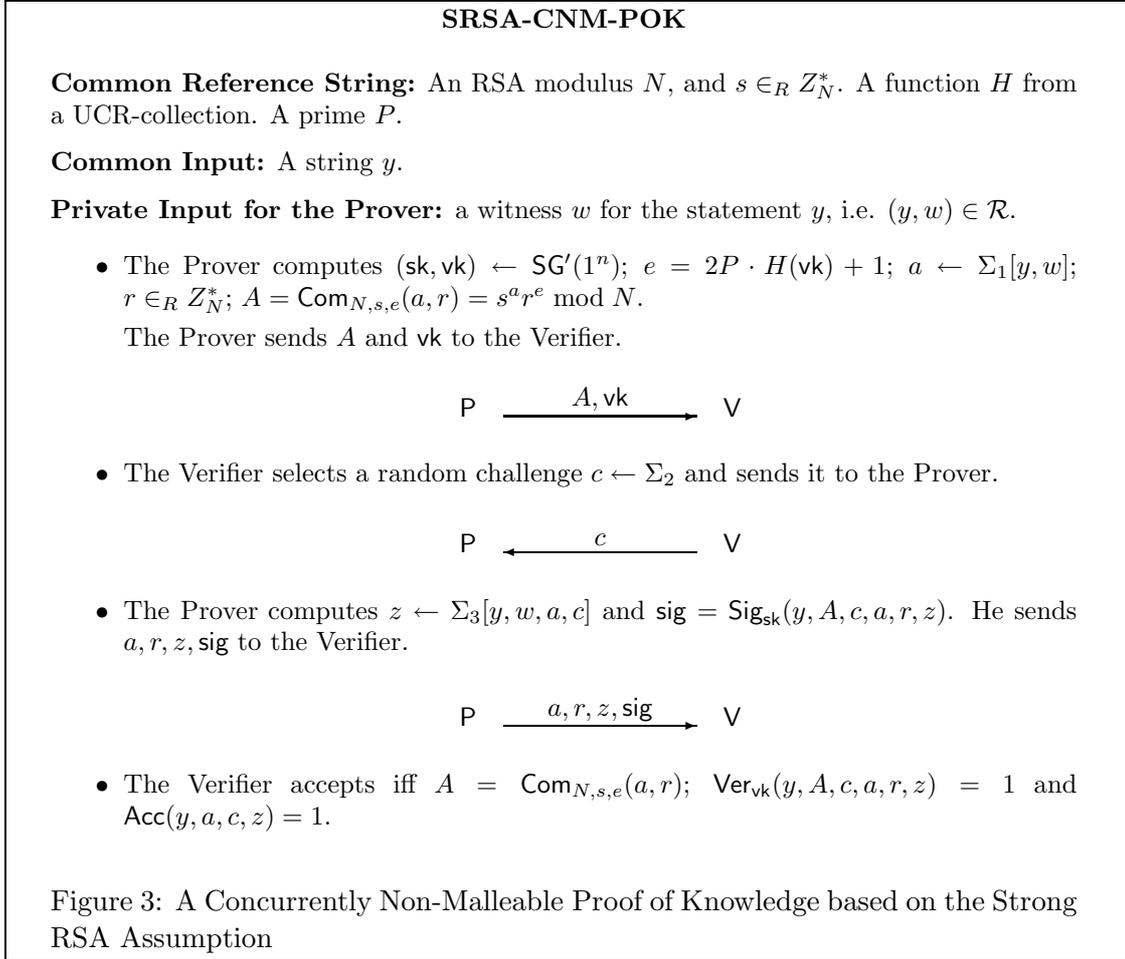
We modify this Σ -protocol in the following way. We assume that the parties share a common reference string that contains the following:

- an RSA modulus N and $s \in_R Z_N^*$ (which will be used for the commitment scheme described in Section 3.1 (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 (sk, vk) for a modified one-time strong signature scheme. Recall that this means that the prover chooses (sk, vk) until $e = 2PH(\text{vk}) + 1$ is a prime. The prover computes $A = \text{Com}_{N,s,e}(a, r)$ 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 \mathbf{sig} a signature over the whole transcript, computed using \mathbf{sk} . The verifier checks that a, r is a correct opening of A , that \mathbf{sig} is a valid signature over the transcript using \mathbf{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.



Theorem 2 *If the Strong RSA Assumption holds, and if $(\mathbf{SG}, \mathbf{Sig}, \mathbf{Ver})$ 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 2 is similar to the proof of Theorem 1, 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.

6 Applications

In this section we describe the main applications of our protocol for concurrently secure non-malleable proof of knowledge: concurrently secure identification and deniable authentication protocols.

6.1 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 [21]. 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 $(\text{IdG}, \text{P}, \text{V})$. IdG is a randomized key generation algorithm: on input the security parameter 1^n it returns a pair of keys (isk, ipk) , respectively the secret and public key. P and V are interactive Turing Machines. The prover runs on input the key pair (isk, ipk) while the verifier only knows the public key ipk . With $[\text{P}(\text{isk}, \text{ipk}), \text{V}(\text{ipk})]$ we denote the output of the protocol (i.e. 1 iff the verifier accepts).

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

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

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

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

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 CNM-POK with a hard relationship \mathcal{R} as the identification protocol.

Theorem 3 *If the relationship \mathcal{R} is hard, multi-trapdoor commitments exist, $(\text{SG}, \text{Sig}, \text{Ver})$ is a strong one-time signature scheme, then CNM-POK is an identification scheme secure against concurrent man-in-the-middle attacks.*

The proof is identical to the proof of Theorem 1. Notice however that we achieve full concurrency here, indeed the extraction procedure in the proof of Theorem 1 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 SRSA-CNM-POK.

6.2 Deniable Authentication

A deniable authentication protocol allows a Sender to authenticate a message M to a Receiver, in a way that while the Receiver is certain that the message M originated with the Sender, at the same time the Receiver cannot convince anybody else of this fact. We are interested in a public-key scenario, in which Sender and Receiver do not share any secret information⁹, and authentication is achieved via a public key associated with the Sender. A formal definition appears below.

Deniable authentication must be secure against a man-in-the-middle attacker \mathcal{A} , who tries to convince the Receiver that a certain message M' originated with the sender, while \mathcal{A} interacts with the Sender on a different message M . We distinguish between the *sequential* case in which \mathcal{A} can only conduct one interaction at a time with the real Sender, and the *concurrent* case in which \mathcal{A} can have several sessions, arbitrarily interleaved, with the real Sender.

FORMAL DEFINITION OF DENIABLE AUTHENTICATION. A deniable authentication protocol consists of a tuple $(\text{crsDG}, \text{DG}, \text{Send}, \text{Rec})$. crsDG is the generator for the common reference string. DG is a randomized key generation algorithm: on input the security parameter 1^n it returns a pair of keys (ask, apk) , respectively the secret and public key. Send and Rec are interactive Turing Machines. The sender Send runs on input a message M and the key pair (ask, apk) . On the other hand the receiver Rec runs only on the public key apk .

With $[\text{Send}(M, \text{ask}, \text{apk}), \text{Rec}(\text{apk})]_{\text{crs}}$ we denote the output of Rec at the end of the interaction: this is either a message or the value nil denoting that Rec rejected the execution.

On the other hand with $\text{View}[\text{Send}(M, \text{ask}, \text{apk}), \text{Rec}(\text{apk})]$ we denote the *view* of the receiver at the end of the execution, i.e. the random variable defined by its internal coin tosses and all the messages exchanged during the execution.

Definition 7 *We say that $(\text{crsDG}, \text{DG}, \text{Send}, \text{Rec})$ is a strongly deniable authentication protocol, over a message space \mathcal{M} if the following properties are satisfied:*

Completeness *For all crs output by $\text{crsDG}(1^n)$ and (ask, apk) output by $\text{DG}(1^n)$ we have that*

$$[\text{Send}(M, \text{ask}, \text{apk}), \text{Rec}(\text{apk})]_{\text{crs}} = M;$$

Soundness *For every probabilistic polynomial time adversary \mathcal{A} , with oracle access to Send running on inputs M_i adaptively chosen by \mathcal{A} we have that*

$$\text{Prob}[\text{crs} \leftarrow \text{crsDG}(1^n); (\text{ask}, \text{apk}) \leftarrow \text{DG}(1^n); M \leftarrow \text{Output}[\mathcal{A}^{\text{Send}(M_i, \text{ask}, \text{apk})}, \text{Rec}(\text{apk})] : \\ M \neq M_i \forall i \text{ and } M \neq \text{nil}]$$

is negligible in n .

Deniability *For every probabilistic polynomial time receiver \mathcal{A} , interacting with the prover on inputs M_i adaptively chosen by \mathcal{A} , there exists a simulator SIM with oracle access to \mathcal{A}*

⁹In the shared key model, deniable authentication is easily achieved by using Message Authentication Codes

such that except with negligible probability on the output of DG we have that the following distributions

$$\text{View}[\text{Send}(M_i, \text{ask}, \text{apk}), \mathcal{A}(\text{apk})] \quad \text{and} \quad \text{View}[\text{SIM}(M_i, \text{apk}), \mathcal{A}(\text{apk})]$$

are indistinguishable for all i .

We say that $(\text{crsDG}, \text{DG}, \text{Send}, \text{Rec})$ is a concurrently secure strongly deniable authentication protocol if in the above conditions the adversarial executions over messages M_i can be arbitrarily interleaved.

Notice that the simulator of the deniability property is not allowed to choose crs (differently from the simulator in the ZK property for proofs of knowledge). This is because we are trying to capture deniability by “real” parties in the “real” world.

A weaker notion of deniability can be defined. In this notion the Simulator SIM is given access to the real sender, but only to authenticate a fixed sequence of messages, independent from the ones selected by \mathcal{A} during the simulation. In a weakly deniable authentication protocol, the sender will not be able to deny that he authenticated something to a receiver, though the receiver will not be able to prove to a third party what exactly the sender authenticated.

OUR SOLUTION. We follow a paradigm similar to [27] to build a deniable authentication protocol. The idea is to use a Σ -protocol for a specific language: knowledge of a pre-image for a trapdoor function.

The key generation algorithm outputs a trapdoor function f and its inverse f^{-1} . We assume that f admits an efficient Σ -protocol to prove knowledge of $f^{-1}(y)$. The public key is set to f and the sender keeps f^{-1} private.

To authenticate a message M sender and receiver run a modified version of CNM-POK. The receiver first generates $y = f(w)$ for a random w . It sends y and the message M to the sender who then runs CNM-POK as the prover to prove knowledge of w , using knowledge of f^{-1} . The receiver, acting as the verifier, sends a random challenge c . The sender answers as prescribed by the third round of CNM-POK, but it omits the signature. The receiver at this point reveals w . And finally the sender signs the whole transcript, including M and w .

The protocol is described in detail in Figure 4.

Theorem 4 *If f is a trapdoor function, multi-trapdoor commitments exist and $(\text{SG}, \text{Sig}, \text{Ver})$ is a secure one-time signature scheme, then Den-Auth is a concurrently secure strongly deniable authentication protocol.*

Sketch of Proof: Completeness is obvious by inspection.

To argue **soundness** we follow the proof of Theorem 1. If there is an adversary \mathcal{A} that contradicts the soundness condition, we run it in a simulation similar to **Extraction**. In this simulation we run on input $y = f(w)$ and we want to compute w , thus contradicting the one-wayness of f .

As in the proof of Theorem 1 the simulator sets up the common reference string in a way that he knows TK the master trapdoor for the multi-commitment family. This will allow him to simulate (in any concurrent interleaving) the behavior of the honest sender (by opening the commitment A in any desired way).

We place y in the execution in which \mathcal{A} is trying to forge a message M .

As in the proof of Theorem 1 this simulator will successfully extract w as long as \mathcal{A} does not open its own commitment A' in two different ways. But then we can use Proposition 3 to show

Den-Auth

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.

Public Key of the Sender: A trapdoor function f . We assume that f admits a Σ -protocol Σ to prove knowledge of pre-images of f .

Private Input for the Sender: the inverse f^{-1} .

- The Receiver chooses a random w in the domain of f and computes $y = f(w)$ and sends it to the Sender together with the message M .

Sender $\xleftarrow{y, M}$ Receiver

- The Sender computes $w = f^{-1}(y)$; $(\text{sk}, \text{vk}) \leftarrow \text{SG}(1^n)$; $\text{pk} = H(\text{vk})$; $a \leftarrow \Sigma_1[y, w]$; $A = \text{Com}(\text{PK}, \text{pk}, a, r)$ for a random r .

The Sender sends A and vk to the Verifier.

Sender $\xrightarrow{A, \text{vk}}$ Receiver

- The Receiver sends a random challenge c .

Sender \xleftarrow{c} Receiver

- The Sender computes $z \leftarrow \Sigma_3[y, w, a, c]$ and sends a, r, z to the Receiver.

Sender $\xrightarrow{a, r, z}$ Receiver

- The Receiver checks if $A = \text{Com}(\text{PK}, \text{pk}, a, r)$ and $\text{Acc}(y, a, c, z) = 1$. If so, reveals w .

Sender \xleftarrow{w} Receiver

- The Sender checks that $y = f(w)$ and if so, signs the whole transcript $\text{sig} = \text{Sig}_{\text{sk}}(y, M, A, c, a, r, z, w)$.

Sender $\xrightarrow{\text{sig}}$ Receiver

- The Receiver accepts M iff the signature is correct: $\text{Ver}_{\text{vk}}(y, M, A, c, a, r, z, w) = 1$.

Figure 4: A Strongly Deniable Authentication Protocol secure under Concurrent Composition

that under the security of the multi-trapdoor commitment scheme used, and the security of the one-time signature scheme this event happens only with negligible probability.

Again (as in the proof of the identification scheme) we achieve full concurrency, since any nested execution in which the adversary acts as the sender needs not to be rewinded.

Notice however that in this proof we need not the “strong” property on the one-time signature scheme. Indeed, the message M that the adversary \mathcal{A} is trying to authenticate, is always different from the ones authenticated by the real prover, and thus the signature produced by the adversary will always be on a different transcript. We do not have the case that the transcripts are the same, but the signatures are different (case 2 in the proof of Proposition 4).

We now prove **deniability**. Notice that since w is part of the transcript, then a receiver can compute the whole transcript on his own, by performing the Σ -protocol as the prover/sender as well. Notice that this simulator does not need to know the trapdoor of PK since it does not need to open A in two ways. \square

The last observation in the proof of the soundness condition, points out to a potential improvement in the design of the protocol. It is possible to completely remove the one-time signature scheme, and use the fact that $M \neq M_i$ to enforce the non-malleability of the proof of knowledge. Details will appear in a longer version of this paper.

An efficient instantiation of Den-Auth can be obtained by using SRS-CNM-POK and RSA as the one-way function f (together with the Guillou-Quisquater identification scheme [26] to prove knowledge of e -roots). The resulting scheme is secure under the Strong RSA assumption. This is the first efficient scheme known under this assumption.

7 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 using deniable authentication instead of signatures in key exchange protocols. These discussions helped me focus on the problem of concurrently non-malleable proofs of knowledge.

Although the original work on this paper was done independently from [33], 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 42nd 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 43rd 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. In *Advances in Cryptology - Eurocrypt '97*, LNCS vol. 1233, Springer, 1997, pages 480-494.
- [5] M. Bellare, M. Fischlin, S. Goldwasser and S. Micali. *Identification Protocols Secure against Reset Attacks*. Proc. of EUROCRYPT'01 (LNCS 2045), pp.495–511, Springer 2001.
- [6] M. Bellare and O. Goldreich. *On defining proofs of knowledge*. Proc. of CRYPTO'92 (LNCS 740), Springer 1993.
- [7] D. Bleichenbacher, U. Maurer. Optimal Tree-Based One-time Digital Signature Schemes. *STACS'96*, LNCS, Vol. 1046, pp.363–374, Springer-Verlag.
- [8] D. Bleichenbacher, U. Maurer. On the efficiency of one-time digital signatures. *Advances in Cryptology-ASYACRYPT'96*, to appear.
- [9] D. Boneh and X. Boyen. *Short Signatures without Random Oracles*. Proc. of EUROCRYPT'04 (LNCS 3027), pp.382–400, Springer 2004.
- [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 42nd 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 (LNCS 2139), pp.19–40, Springer 2001.
- [13] R. Canetti, J. Kilian, E. Petrank and A. Rosen. *Concurrent Zero-Knowledge requires $\tilde{\Omega}(\log n)$ rounds*. Proc. of 33rd ACM Symp. on Theory of Computing (STOC'01), pp.570–579, 2001.
- [14] R. Cramer and I. Damgård. New Generation of Secure and Practical RSA-based signatures. *Proc. of Crypto '96* LNCS no. 1109, pages 173-185.
- [15] R. Cramer and V. Shoup. A practical public-key cryptosystem secure against adaptive chosen ciphertext attacks. In *CRYPTO'98*, Springer-Verlag (LNCS 1462), pages 13–25, 1998.
- [16] R. Cramer and V. Shoup. Signature schemes based on the Strong RSA assumption. *Proc. of 6th ACM Conference on Computer and Communication Security 1999*.
- [17] I. Damgård. *Efficient Concurrent Zero-Knowledge in the Auxiliary String Model*. Proc. of EUROCRYPT'00 (LNCS 1807), pp.174–187, Springer 2000.

- [18] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. *Robust Non-Interactive Zero Knowledge*. Proc. of CRYPTO'01), (LNCS 2139), pp.566-598, Springer 2001.
- [19] D. Dolev, C. Dwork and M. Naor. *Non-malleable Cryptography*. SIAM J. Comp. 30(2):391–437, 200.
- [20] C. Dwork, M. Naor and A. Sahai. *Concurrent Zero-Knowledge*. Proc. of 30th ACM Symp. on Theory of Computing (STOC'98), pp.409–418, 1998.
- [21] U. Feige, A. Fiat and A. Shamir. *Zero-Knowledge Proofs of Identity*. J. of Crypt. 1(2):77–94, Springer 1988.
- [22] J. Garay, P. MacKenzie and K. Yang. *Strengthening Zero-Knowledge Protocols Using Signatures*. Proc. of EUROCRYPT'03 (LNCS 2656), pp.177–194, Springer 2003. Final version at eprint.iacr.org
- [23] R. Gennaro, S. Halevi and T. Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. *Proc. of Eurocrypt '99* LNCS no. 1592, pages 123-139.
- [24] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM J. Computing*, 18(1):186–208, February 1989.
- [25] 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.
- [26] 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 (LNCS 330), pp.123–128, Springer 1989.
- [27] J. Katz. *Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications*. Proc. of EUROCRYPT'03 (LNCS 2656), pp.211-228, Springer 2003.
- [28] J. Katz. *Efficient Cryptographic Protocols Preventing Man-in-the-Middle Attacks*. Ph.D. Thesis, Columbia University, 2002.
- [29] H. Krawczyk and T. Rabin. *Chameleon Hashing and Signatures*. Proceedings of NDSS 2000, pp.143-154.
- [30] L. Lamport. Constructing Digital Signatures from a One-Way Function. *Technical Report SRI Intl*. CSL 98, 1979.
- [31] Y. Lindell. *Composition of Secure Multi-Party Protocols*. Lecture Notes in Computer Science vol.2815, Springer 2003.
- [32] Y. Lindell. *Lower Bounds for Concurrent Self-Composition*. Manuscript.
- [33] P. MacKenzie and K. Yang. *On Simulation-Sound Commitments*. To appear in EuroCrypt'04.
- [34] U. Maurer. *Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters*. J. of Crypt. 8(3):123–156, Springer 1995.
- [35] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - Eurocrypt '99*, LNCS vol. 1592, Springer, 1997, pages 223-238.

- [36] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Crypto '91*, pages 129–140, 1991. LNCS No. 576.
- [37] M. Prabhakaran, A. Rosen and A. Sahai. *Concurrent Zero-Knowledge with logarithmic round complexity*. Proc. of 43rd IEEE Symp. on Foundations of Computer Science (FOCS'02), pp.366–375, 2002.
- [38] 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
- [39] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.
- [40] 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

As an example of the power and simplicity of our technique in Figure 5 we show our transformation applied to the Schnorr’s proof of knowledge for discrete logarithm.

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 2

Sketch of 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 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 $(\text{sk}, \text{vk}) \leftarrow \text{SG}'(1^n)$ and computes $e = 2P \cdot H(\text{vk}) + 1$

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.

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 $y = g^w \pmod p$.

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

The Prover sends A and VK to the Verifier.

$$\text{P} \xrightarrow{A, \text{vk}} \text{V}$$

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

$$\text{P} \xleftarrow{c} \text{V}$$

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

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

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

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

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 2 is complete once Proposition 5 is proven. \square

Proposition 5 *If the Strong RSA Assumption holds and if $(\text{SG}, \text{Sig}, \text{Ver})$ 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.*

Sketch of 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 $(\text{sk}_1, \text{vk}_1), \dots, (\text{sk}_m, \text{vk}_m)$. Then he computes $e_i = 2P \cdot H(\text{vk}_i) + 1$. It then sets $s = \sigma^E \bmod N$ where $E = \prod_{i=1}^m e_i$. 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 $(\text{sk}_i, \text{vk}_i)$. 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 $e = 2P \cdot H(\text{vk}') + 1$, 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 $i = 1, \dots, m$. 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. \square

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 $vk' = vk_i$ for some i , and the verifier accepts during **SRSA-Inverter** is negligible.*

Sketch of 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 oracle¹⁰.

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 $vk_i = vk$. With probability $1/m$ then \mathcal{A} will chose $vk' = vk$.

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

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

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

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 $sig_i \neq sig'$ 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

¹⁰This is where the requirement for a *strong* signature scheme arises. This requirement was overlooked in [27, 22]