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

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, one based on the Strong RSA Assumption and the other on the Strong Diffie-Hellman Assumption.
The main application of our new notion 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 (in the common reference string model). When using our specific implementations, this compiler is very efficient (requires no more than four exponentiations) and maintains the round complexity of the original proof of knowledge.
The main practical applications of our results are concurrently secure identification protocols. For these applications our results are the first simple and efficient solutions based on the Strong RSA or 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.

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 on when messages are delivered to honest parties.

* Extended Abstract. The full version of the paper is available at http://eprint.iacr.org/2003/214/

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* (MTC) which we introduce in this paper. After defining the notion we show specific number-theoretic constructions based on the Strong RSA Assumption and the recently introduced Strong Diffie-Hellman Assumption. These constructions are very efficient, and when applied to the concurrent compiler described above, this is the whole overhead.

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 considering 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 positioned on the left of the picture). 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 *on-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 [16]. We use his techniques, so our protocols are fully concurrently zero-knowledge. Extraction is more complicated. Lindell in [30] shows how to do post-protocol extraction for the case of right concurrency. We can use his techniques as well. But for many applications what really matters is on-line extraction. We are able to do that only under left-concurrency[1]. This is however enough to build fully concurrently secure applications like identification protocols.

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 bounds on the round complexity of concurrent zero-knowledge in the black-box model [13, 34], 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 [16] 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

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

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 [29], 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 [30]). In this model, the first theoretical solution to our problem was presented in [17]. Following on the ideas presented in [17] more efficient solutions were presented in [27, 22, 31].

Our compiler uses ideas from both the works of Damgård [16] and Katz [27], with the only difference that it uses multi-trapdoor instead of regular trapdoor commitments in order to achieve concurrent non-malleability.

Simulation-Sound Trapdoor Commitments. The notion of Simulation-Sound Trapdoor Commitments (SSTC), introduced in [22] and later refined and improved in [31], is very related to our notion of MTC. The notion was introduced for analogue purposes: to compile (in a way similar to ours) any Σ-protocol into one which is left-concurrently non-malleable. They show generic constructions of SSTC and specific direct constructions based on the Strong RSA Assumption and the security of the DSA signature algorithm.

The concept of SSTC is related to ours, though we define a weaker notion of commitment (we elaborate on the difference in Section 3). The important contribution of our paper with respect to [22, 31] 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 more efficient number theoretic instantiations. Indeed our Strong RSA construction is about a factor of 2 faster than the one presented in [31]. 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[2].

[2] In [22, 31] Ω-protocols are introduced, 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). 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 know how to construct efficient direct constructions of Ω-protocols only for knowledge of discrete logarithms, and even that is not particularly efficient. Since for the

Remark. Because of space limitations, all the proofs of the Theorems, and various technical details are omitted and can be found in the full version of the paper.

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 $Prob[A_1; \ldots; A_k : B]$ we denote the probability of event B happening after $A_1, \ldots, A_k$.

2.1 One-time Signatures

Our construction requires a *strong* one-time signature scheme which is secure against chosen message attack. Informally this means that the adversary is given the public key and signatures on any messages of her choice (adaptively chosen after seeing the public key). Then it is infeasible for the adversary to compute a signature of a new message, or a different signature on a message already asked. The following definition is adapted from [25].

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

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

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, 28]). 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 [35] 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 $x^e = s \bmod N$.

applications we had in mind left-concurrency was sufficient, we did not follow this path in this paper.

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

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

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

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

is negligible in n.

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 $G, g, g^x, g^{x^2}, g^{x^3}, \ldots$, 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*

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

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 $c = ab \bmod q$ or not. The *gap-DDH* property will also be required by our construction of multi-trapdoor commitments that uses the SDH Assumption[3].

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

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.

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 (poly-time) 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. [36, 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, 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 (crsG,P,V), where crsG is a randomized algorithm that on input the security parameter 1^n outputs the common reference string crs. In our definition we limit the prover to be a probabilistic polynomial time machine, thus technically our protocols are *arguments* and not proofs. But for the rest of the paper we will refer to them as proofs.

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

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

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

Definition 2. *We say that* (crsG,P,V) *is a concurrently non-malleable proof of knowledge for a relationship* $(\mathcal{PTR}, \mathsf{IG})$ *if the following properties are satisfied:*

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

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

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

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

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

are indistinguishable.

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

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

The main difference between our definition and the notion of SSTC [22, 31] 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 $C(M) = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R)$ 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 $T = \mathsf{TK}$ or $T = \mathsf{tk}$ then Open outputs R' such that $C(M) = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$.

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 $(\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$. 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 $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R), M, R, \mathsf{pk}$ and a message $M' \neq M$. If $\mathsf{pk} = \mathsf{pk}_i$ for some i, and is a valid public key, then $\mathcal{EQ}$ answers with R' such that $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$ otherwise it outputs *nil*. We say that $\mathcal{A}$ wins if it outputs $C, M, R, M', R', \mathsf{pk}$ such that $C = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M, R) = \mathsf{Com}(\mathsf{PK}, \mathsf{pk}, M', R')$, $M \neq M'$ and $\mathsf{pk} \neq \mathsf{pk}_i$ for all i. We require that for all efficient algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ wins is negligible in the security parameter.

We can define a stronger version of the **Secure Binding** property by requiring that the adversary $\mathcal{A}$ receives the trapdoors tk_i's matching the public keys pk_i's, instead of access to the equivocator oracle $\mathcal{EQ}$. In this case we say that the multi-trapdoor commitment family is *strong*[4].

3.1 A scheme based on the Strong RSA Assumption.

The starting point for the our construction of multi-trapdoor commitments based on the Strong RSA Assumption, is a commitment scheme based on the (regular) RSA Assumption, which has been widely used in the literature before (e.g. [14, 15]).

The master public key is a number N product of two large primes p, q, and s a random element of Z_N^*. The master trapdoor is the factorization of N, i.e. the integers p, q. The public key of a scheme in the family is an ℓ-bit prime number

[4] This was actually our original definition of multi-trapdoor commitments. Phil MacKenzie suggested the possibility of using the weaker approach of giving access to an equivocator oracle (as done in [31]) and we decided to modify our main definition to the weaker one, since it suffices for our application. However the strong definition may also have applications, so we decided to present it as well.

e such that $GCD(e, \phi(N)) = 1$. The specific trapdoor of the scheme with public key e is the e-root of s, i.e. a value $\sigma_e \in Z_N^*$ such that $\sigma_e^e = s \bmod N$.

To commit to $a \in [1..2^{\ell-1}]$ the sender chooses $r \in_R Z_N^*$ and computes $A = s^a \cdot r^e \bmod N$. To decommit the sender reveals a, r and the previous equation is verified by the receiver.

Proposition 1. *Under the Strong RSA Assumption the scheme described above is a multi-trapdoor commitment scheme.*

Sketch of Proof: Each scheme in the family 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 $A \cdot s^{a-a'}$. Observe, moreover that r' can be computed efficiently as $\sigma_e^{a-a'}$, thus knowledge of σ_e allows to open a commitment (for which we know an opening) in any desired way.

We now argue the **Secure Binding** property under the Strong RSA Assumption. Assume we are given a Strong RSA problem istance N, σ. Let's now run the **Secure Binding** game. The adversary is going to select k public keys which in this case are k primes, $e_1, \ldots, e_k$. We set $s = \sigma^E$ where $E = \prod_{i=1}^k e_i$ and return N, s as the public key of the multi-trapdoor commitment family. This will easily allow us to simulate the oracle $\mathcal{EQ}$, as we know the e_i-roots of s, i.e. the trapdoors of the schemes identified by e_i.

Assume now that the adversary equivocates a commitment scheme in the family identified by a prime $e \neq e_i$. The adversary returns a commitment A and two distinct openings of it (a, r) and (a', r'). Thus

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

Let $\delta = a - a'$. Since $a, a' < e$ and e and the e_i's are all distinct primes we have that $GCD(\delta E, e) = 1$. We can find integers α, β such that $\alpha\delta E + \beta e = 1$. Now we can compute (using Shamir's GCD trick [37] and Eq.(1))

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

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

Remark: 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 $[1..2^{\ell-1}]$. In this case the commitment is computed as $s^{H(a)} r^e$. 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}.

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 $\mathsf{DDH\text{-}Test}$ which on input a triplet (g^a, g^b, g^c) of elements in G outputs 1 if and only if, $c = ab \bmod q$. We also assume that 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 pk which is simply an element $e \in Z_q$. The specific trapdoor tk of this scheme is the value f_e in G, such that $f_e^{x+e} = g$.

To commit to a message $a \in Z_q$ with public key $\mathsf{pk} = e$, the sender chooses at random $\phi \in Z_q$ and computes $h_e = (h \cdot g^e)^\phi$. It then runs Pedersen's commitment [33] with bases g, h_e, 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 $F = g^{\phi \cdot r}$. The receiver accepts the opening if $\mathsf{DDH\text{-}Test}(F, h \cdot g^e, A \cdot g^{-a}) = 1$.

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

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 $f_{e_1}, \ldots, f_{e_\ell}$ for values $e_1, \ldots, e_\ell$ chosen before seeing the public key g, h, and still will not be able (under the $(\ell+1)$-SDH) to compute f_e for a new $e \neq e_i$.

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 committment $A = g^\alpha$ in two ways $a, F = g^\beta$ and $a', F' = g^{\beta'}$ with $a \neq a'$. The $\mathsf{DDH\text{-}Test}$ tells us that $\alpha - a = \beta(x+e)$ and $\alpha - a' = \beta'(x+e)$, thus $a - a' = (\beta' - \beta)(x+e)$ or

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

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

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 P wants to prove to a verifier V that he knows a witness w for some statement y. The prover sends a first message a. The verifier challenges the prover with a random value c and the prover answers with his response z.

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

CNM-POK

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

Common Input: A string y.

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

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

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

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

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

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

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

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

Fig. 1. A Concurrently Non-malleable Proof of Knowledge

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

The verifier sends the challenge c. The prover sends back a, r as an opening of A and the answer z of the Σ-protocol. It also sends sig a signature over the whole transcript, computed using sk. The verifier checks that a, r is a correct opening of A, that sig is a valid signature over the transcript using vk and also that (a, c, z) is an accepting conversation for the Σ-protocol. The protocol is described in Figure 1.

Theorem 1. *If multi-trapdoor commitments exist, if H is a collision-resistant hash function, and if* (SG,Sig,Ver) *is a strong one-time signature scheme, then* CNM-POK *is a concurrently non-malleable proof of knowledge (see Definition 2).*

4.1 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 [32, 15]) in this section.

Modifying the One-Time Signatures. First of all, we require the one-time signature scheme (SG,Sig,Ver) to have an extra property: i.e. that the distribution induced by SG over the verification keys vk is the uniform one[5]. Virtually all the known efficient one-time signature schemes have this property.

Then we assume that the collision resistant hash function used in the protocol is drawn from a family which is *both* a collision-resistant collection and a collection of families of universal hash functions [6].

Assume that we have a randomly chosen hash function H from such a collection mapping n-bit strings (the verification keys) into k-bit strings and a prime $P > 2^{k/2}$.

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

We note the following facts:

- $H(\mathsf{vk})$ follows a distribution over k-bit strings which is statistically close to uniform; thus using results on the density of primes in arithmetic progressions (see [1], the results hold under the Generalized Riemann Hypothesis)

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

[6] This is a reasonable assumption that can be made on families built out of a collision-resistant hash function (such as SHA-1). See also [18] for analysis of this type of function families.

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 [32]).

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

Notice that we are not compromising the security of the modified signature scheme. Indeed the keys of the modified scheme are a polynomially large fraction of the original universe of keys. Thus if a forger could forge signature on this modified scheme, then the original scheme is not secure as well.

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

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

4.2 Identification Protocols

The main application of our result is the construction of concurrently secure identification protocols. In an identification protocol, a prover, associated with a public key pk, communicates with a verifier and tries to convince her to be the legitimate prover (i.e. the person knowing the matching secret key sk.) An adversary tries to mount an impersonation attack, i.e. tries to make the verifier accept without knowing the secret key sk.

The adversary could be limited to interact with the real prover only before mounting the actual impersonation attack [21]. On the other hand a more realistic approach is to consider the adversary a "man-in-the-middle" possibly in a concurrent fashion [5]. Clearly such an attacker can always relays messages unchanged between the prover and the verifier. In order to make a security definition meaningful, one defines a successful impersonation attack as one with a transcript different from the ones between the attacker and the real prover[7].

It is not hard to see that CNM-POK is indeed a concurrently secure identification protocol. It is important to notice that we achieve full concurrency here,

[7] 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].

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.

Acknowledgments. I would like to thank Hugo Krawczyk for conversations that started the research on this paper. Thanks also to Dario Catalano, Shai Halevi, Jonathan Katz, Dah-Yoh Lim, Yehuda Lindell and especially Phil MacKenzie for helpful conversations and advice.

References

1. E. Bach and J. Shallit. *Algorithmic Number Theory - Volume 1.* MIT Press, 1996.
2. B. Barak. *How to go beyond the black-box simulation barrier.* Proc. of 42^{nd} IEEE Symp. on Foundations of Computer Science (FOCS'01), pp.106–115, 2001.
3. B. Barak. *Constant-round Coin Tossing with a Man in the Middle or Realizing the Shared Random String Model.* Proc. of 43^{rd} IEEE Symp. on Foundations of Computer Science (FOCS'02), pp.345–355, 2001.
4. N. Barić, and B. Pfitzmann. *Collision-free accumulators and Fail-stop signature schemes without trees.* Proc. of EUROCRYPT'97 (LNCS 1233), pp.480–494, Springer 1997.
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 and U. Maurer. Optimal Tree-Based One-time Digital Signature Schemes. *STACS'96*, LNCS, Vol. 1046, pp.363–374, Springer-Verlag.
8. D. Bleichenbacher and U. Maurer. *On the efficiency of one-time digital signatures.* Proc. of ASIACRYPT'96 (LNCS 1163), pp.145–158, Springer 1996.
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 42^{nd} IEEE Symp. on Foundations of Computer Science (FOCS'01), pp.136–145, 2001.
12. R. Canetti and M. Fischlin. *Universally Composable Commitments.* Proc. of CRYPTO'01 (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 33^{rd} 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. Signature schemes based on the Strong RSA assumption. *Proc. of* 6^{th} *ACM Conference on Computer and Communication Security 1999.*
16. I. Damgård. *Efficient Concurrent Zero-Knowledge in the Auxiliary String Model.* Proc. of EUROCYPT'00 (LNCS 1807), pp.174–187, Springer 2000.

17. 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.
18. Y. Dodis, R. Gennaro, J. Håstad, H. krawczyk and T. Rabin. *Randomness Extraction and Key Derivation using the CBC, Cascade and HMAC Modes.* This proceedings.
19. D. Dolev, C. Dwork and M. Naor. *Non-malleable Cryptography.* SIAM J. Comp. 30(2):391–437, 2000.
20. C. Dwork, M. Naor and A. Sahai. *Concurrent Zero-Knowledge.* Proc. of 30^{th} ACM Symp. on Theory of Computing (STOC'98), pp.409–418, 1998.
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. L. Lamport. Constructing Digital Signatures from a One-Way Function. *Technical Report SRI Intl.* CSL 98, 1979.
29. Y. Lindell. *Composition of Secure Multi-Party Protocols.* Lecture Notes in Computer Science vol.2815, Springer 2003.
30. Y. Lindell. *Lower Bounds for Concurrent Self Composition.* Proc of the 1st Theory of Cryptography Conference (TCC'04), LNCS 2951, pp.203–222, Springer 2004.
31. P. MacKenzie and K. Yang. *On Simulation-Sound Trapdoor Commitments.* Proc. of EUROCRYPT'04 (LNCS 3027), pp.382–400, Springer 2004.
32. U. Maurer. *Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters.* J. of Crypt. 8(3):123–156, Springer 1995.
33. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Crypto '91*, pages 129–140, 1991. LNCS No. 576.
34. M. Prabhakaran, A. Rosen and A. Sahai. *Concurrent Zero-Knowledge with logarithmic round complexity.* Proc. of 43^{rd} IEEE Symp. on Foundations of Computer Science (FOCS'02), pp.366–375, 2002.
35. 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
36. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.
37. A. Shamir. On the generation of cryptographically strong pseudorandom sequences. ACM Trans. on Computer Systems, 1(1), 1983, pages 38-44.