Non-interactive and Reusable Non-malleable Commitment Schemes

Ivan Damgård[a] Jens Groth[b]

June 16, 2003

Abstract

We consider non-malleable (NM) and universally composable (UC) commitment schemes in the common reference string (CRS) model. We show how to construct non-interactive NM commitments that remain non-malleable even if the adversary has access to an arbitrary number of commitments from honest players - rather than one, as in several previous schemes. We show this is a strictly stronger security notion. Our construction is the first non-interactive scheme achieving this that can be based on the minimal assumption of existence of one-way functions. But it can also be instantiated in a very efficient version based on the strong RSA assumption. For UC commitments, we show that existence of a UC commitment scheme in the CRS model (interactive or not) implies key exchange and - for a uniform reference string - even implies oblivious transfer. This indicates that UC commitment is a strictly stronger primitive than NM. Finally, we show that our strong RSA based construction can be used to improve the most efficient known UC commitment scheme so it can work with a CRS of size independent of the number of players, without loss of efficiency.

Keywords: Commitment, non-malleability, universal composability, signature, one-way function.

---

[a]Ivan Damgård, BRICS, Department of Computer Science, Aarhus University, Denmark, e-mail: `ivan@brics.dk`
BRICS: Basic Research in Computer Science (`www.brics.dk`) funded by the Danish National Research Foundation.
This authors work was partially funded by FICS, Foundations in Cryptography and Security, funded by the Danish National Science Research Council.
[b]Jens Groth, Cryptomathic A/S, Denmark (`www.cryptomathic.com`) and BRICS, Department of Computer Science, Aarhus University, Denmark, e-mail: `jg@brics.dk`.

1 Introduction

The notion of commitment is at the heart of cryptographic protocol design. A commitment scheme allows a committer to release some string related to a message m, a *commitment* to m. Later the committer may choose to *open* the commitment thereby revealing m. The scheme must be *hiding*, i.e., the commitment does not help the receiver to compute anything about m. It must also be *binding*, i.e., the committer cannot produce a commitment that he can open successfully to reveal different messages m and m'.

For some applications, extra security properties are needed. One example of such a property is *non-malleability*, a concept introduced by Dolev, Dwork and Naor [9]. As a motivating example for NM commitments, consider the case of fair contract bidding, which can be implemented by having each participant commit to his bid first, after which commitments can be opened and the winner determined. The goal here is to prevent players from producing bids that are correlated to the bids of other players. However, even though the commitment scheme is hiding and binding, it may still be possible for an adversary to compute, from a commitment c to m from another player, a new commitment c' containing a *related* value m'. At this point, the adversary may not even know m or m', but when c is opened, he may be able to open c' to reveal m'. It is well known that many existing commitment schemes are malleable – in some cases, the adversary can always obtain $m' = m - 1$, thus allowing the adversary to consistently underbid the honest player.

Intuitively, a commitment scheme is non-malleable if such an attack is infeasible. Several NM commitment schemes have been suggested. While the first [9] was highly interactive, the scheme by Di Crescenzo, Ishai and Ostrovsky [6] is the only previous scheme that is non-interactive and can be based on the minimal assumption of existence of one-way functions. This scheme, and later more efficient ones based on special assumptions [7, 11] are all in the common reference string (CRS) model, where it is assumed that players all have access to a string that is guaranteed to be selected with a prescribed distribution. A special case of this is a uniformly random CRS; we call this the uniform reference string (URS) model. It is unknown if non-interactive NM commitment with minimal assumptions is possible without a CRS, but since part of our goal is to look at the relation to universally composable commitments where a CRS is necessary (see below for details) we only consider the CRS model here.

The constructions from [6, 7, 11] were all proven secure according to a definition where the adversary sees *one* commitment from an honest player and then tries to make his own (related) commitment. However, if we consider the motivating example, it is clearly more natural to require non-malleability, even if the adversary gets any polynomial number of commitments as input (the adversary's goal might be to underbid everyone else, for instance). We call this notion of security *reusability*, referring to the fact that in the CRS model, it means that the same CRS can be reused for several commitments from any number of players. We show that reusability is a strictly stronger notion, for both unconditionally hiding and unconditionally binding schemes. This may be slightly surprising since it was argued in [9] that the corresponding notions for NM *encryption* are equivalent. Unfortunately, the security proofs of [6, 7] break down in the more general setting, and so it might seem that to get reusability, one would have to either use several rounds of interaction [9] or use stronger, non-minimal, assumptions [4].

In this paper, we show a general technique for constructing reusable, non-interactive NM commitments in the CRS model. The main new technical idea we contribute is a way to use any digital signature scheme (even one with rather weak security properties) as a basis for NM commitments. Our construction can therefore be based on any one-way function, but can also be instantiated in a very efficient version, based on the strong RSA assumption. The version based on general one-way functions also extends to the URS model.

Universally composable (UC) commitment schemes is a notion introduced by Canetti in [2], with the first such scheme being suggested by Canetti and Fischlin in [3]. An efficient scheme based on a specialized assumption was suggested by Damgård and Nielsen in [8], while a scheme based on any trapdoor one-way permutation can be found in [4]. In a UC scheme, making a commitment is "equivalent" to giving in private the committed value to a trusted third party, who will then later reveal it on request from the committer. This is a very strong security notion: it implies (reusable) non-malleability, and also security against concurrent composition and adaptive adversaries. In fact, there exists no 2-party UC commitment scheme in the "bare" model where no extra resources are given a priori. However, 2-party schemes *are* possible in the CRS model, and all the schemes mentioned above work in this scenario.

It is easy to see that *non-interactive* UC commitment implies key exchange [4], but previously no consequences of UC commitments in general were known. In this paper, we show that any 2-party UC commitment scheme in the CRS model (interactive or not) implies key exchange. Furthermore, UC commitment in the URS model implies oblivious transfer. Key exchange and OT are generally regarded as a stronger primitive than one-way functions. For instance, Impagliazzo and Rudich [13] show a relativized separation between them. So, our results can be seen as an indication that UC commitment is a strictly stronger primitive than NM commitment – even if we require the NM scheme to be non-interactive and reusable.

Our last result is an application of our efficient NM scheme to improve the efficient UC scheme from [8], where the size of the CRS must grow linearly with the number of players involved. We show that by combining the two, we obtain an equally efficient UC scheme where the size of the CRS can be independent of the number of players. To prove this we rely on the underlying properties of the commitment scheme that were also used in the proof of NM.

1.1 Preliminaries

Unless otherwise stated we work with probabilistic polynomial time algorithms. Sometimes we talk about an adversary, typically denoted $\mathcal{A}$. The adversary is modelled by a probabilistic polynomial time interactive Turing machine, see [12].

Both the normal algorithms and the adversary algorithms may get some auxiliary input z. This advice is always polynomially bounded in k, the security parameter. This means that the algorithms are non-uniform.[1]

We say a function $f(k)$ is negligible if

$$\forall c > 0 \exists K \forall k > K : f(k) < k^{-c}.$$

We write $f(k) < negl(k)$ to indicate that $f(k)$ is bounded by some negligible function. We say a function is significant if it is not negligible.

We write $\mathrm{time}_A(k)$ to indicate an easily computable polynomial bounding the runtime of A.

2 Commitments

Following the previous literature on non-malleability, we do not assume that players have global unique usernames, are assigned certified public keys or

[1] Remove this auxiliary input to set the paper in a uniform setting.

are even aware of each other. Making such assumptions generally makes non-malleability problems easier to solve, but the solutions will of course be less generally applicable. We only assume that the players have access to a CRS.

2.1 Commitment

We explain the notation used in the paper for a standard commitment scheme in the CRS model. A commitment scheme has a probabilistic polynomial time *key generator* K, which on input 1^k outputs a public key pk, the CRS. Associated with this public key are a message space $\mathcal{M}_{pk}$, a commitment space $\mathcal{C}_{pk}$ and two polynomial time algorithms $commit_{pk}$ and $decommit_{pk}$.

To commit to a message $m \in \mathcal{M}_{pk}$ we choose at random a randomizer r. We give m, r as input to $commit_{pk}$. The resulting output is $(c, d) = commit_{pk}(m; r)$, where c belongs to $\mathcal{C}_{pk}$, while d is the decommitment information needed to open the commitment. Typically $d = (m, r)$.

To open a message the sender sends d to the receiver. The receiver computes $decommit_{pk}(c, d)$. When the commitment is constructed as above the output of this computation is m. If something is wrong, e.g., $c \notin \mathcal{C}_{pk}$ or d is not a valid opening of the commitment; the output of the decommitment algorithm is $\perp$.

2.2 Equivocable Commitment

Equivocable commitment schemes are a special type of commitment schemes where we can generate the public key in a special way getting some equivocation information about the public key. This extra information, the equivocation key, allows us to violate the binding property. With it we are able to generate commitments that we can open as containing any message we wish, without the adversary being able to notice the deceit.

We generate the public key and the equivocation key ek using a modified key generator $\widehat{K}$. We create an equivocable commitment by running an algorithm $\widehat{\mathrm{commit}}_{pk}$ on ek. This produces a commitment c and some associated equivocation information e. We open c as containing any message $m \in \mathcal{M}_{pk}$ by running $\mathrm{equiv}_{pk,ek}(c, e, m)$.

Definition 1 *We say the commitment scheme is* equivocable *if for any D and auxiliary input z we have*

$$\begin{aligned} &P[(pk, ek) \leftarrow \widehat{K}(1^k) : D^{\widehat{\mathcal{O}}}(pk, z) = 1] \\ &< P[pk \leftarrow K(1^k) : D^{\mathcal{O}}(pk, z) = 1] + negl(k), \end{aligned}$$

where

- $\widehat{\mathcal{O}}$ *on query* $m \in \mathcal{M}_{pk}$ *returns* (c, d)*, where* $(c, e) \leftarrow \widehat{\mathrm{commit}}_{pk}(ek)$ *and* $d \leftarrow \mathrm{equiv}_{pk,ek}(c, e, m)$.
- $\mathcal{O}$ *on query* $m \in \mathcal{M}_{pk}$ *returns* $(c, d) \leftarrow \mathrm{commit}_{pk}(m)$.

There are several ways to strengthen the notion of equivocability. We could for instance require that the distribution of public keys is identical not just indistinguishable. Another strengthening would be to require that we can equivocate using just the equivocation information produced by $\widehat{K}$, i.e., that we do not need e at all. We call a commitment scheme strengthened in this way for *strongly equivocable*.

2.3 Non-malleable Commitment

Non-malleability is a security notion concerned with man-in-the-middle attacks. With respect to commitments, the intuition is that the execution of some commitment protocols should not affect the execution of other commitment protocols. We capture this in a notion of non-malleability where the adversary does not get an advantage from having access to the execution of commitment protocols compared with the case where the adversary has no such access. In the latter case, we simply let the adversary specify the messages rather than first forming commitments and then opening them later on.

We consider two games. In the first game, we generate a tuple of messages according to some distribution. An adversary receives commitments to these messages and outputs a tuple of commitments himself. After receiving openings of the original commitments, the adversary then tries to open his own commitments in a way such that the contents are related to the original messages. It wins if indeed the messages are related.

In the second game, we generate a tuple of messages according to the same distribution as in the first game. However, this time we do not give the adversary the commitments to the messages. The adversary in the second game must try to output related messages without knowing anything about the original messages.

We wish that for any adversary playing the first game we can find one that fares (almost) as well in the second game. In this case, we will consider the commitment scheme non-malleable. In our case, the fact that we use the CRS model allows us to give some help to the adversary in the second game (without which we could not prove security). More specifically, the model allows the adversary in the second game to produce the public key for the commitment scheme himself by an algorithm that also provides some extra information that the adversary can use to its advantage (in this paper the extra information will enable it to equivocate commitments).

Let us describe the two games more accurately. In both games, there is a message generator, M. The message generator receives the public key, pk, for the commitment scheme as input and it also gets some auxiliary input z_M. M returns a value s and a vector of messages $\vec{m}$. In both games, the adversaries receive a description of M. We demand that the time complexity of M is bounded by some polynomial in k, and that the adversaries may depend on this polynomial. This ensures that they may have time to sample outputs from M.

In the first game we model the adversary $\mathcal{A}$ as a probabilistic polynomial time interactive Turing machine. It learns pk, M and z_M. In its firsts invocation $\mathcal{A}$ receives a tuple of commitments $\vec{c}$ to the messages in $\vec{m}$. It responds with $\vec{c}'$, a tuple of elements from $\mathcal{C}_{pk}$. We do not allow $\mathcal{A}$ to directly copy any of the commitments in $\vec{c}$ into $\vec{c}'$. Later $\mathcal{A}$ is activated again, this time receiving a tuple $\vec{d}$ of decommitments to the commitments in $\vec{c}$. It must now try to produce a tuple of decommitments $\vec{d}'$ to its own commitments.

In the second game, we run a *modified key generator* $\widehat{K}$ to generate the public key pk. This key must be indistinguishable from a real key. In addition to the key pk, the modified key generator also produces some extra information s_{pk} about the key.

The adversary in the second game $\mathcal{B}$ is only invoked once. Like $\mathcal{A}$, it learns pk, M, z_M. It also gets s_{pk} as input. However, it will not receive any information about the tuple of messages $\vec{m}$ except for the number of messages in the tuple, $t = |\vec{m}|$. It returns a tuple $\vec{m}'$ of messages from $\mathcal{M}_{pk} \cup \{\perp\}$.

We compare the outcome of the two games by running a polynomial time distinguisher D. This distinguisher gets as input $s, \vec{m}, \vec{m}', z_D$, where in the first game $\vec{m}'$ is the resulting vector of messages when running the decommitment algorithm on the tuples $\vec{c}', \vec{d}'$. Note, s contains information from M possibly including the public key pk and the auxiliary input. The distinguisher returns a single bit, where we interpret 1 as being a success. We demand that the probability of D outputting 1 cannot be increased by changing a message in $\vec{m}'$ to $\perp$. This way the adversary $\mathcal{A}$ cannot get an advantage by deliberately refusing to open its commitments.

Let us write down the probabilities of success in the two games. For the first game we define

$$\begin{aligned}\mathrm{Succ}_{\mathcal{A},M,D}(k, z_M, z_D) \quad = \quad & P[pk \leftarrow K(1^k); (s, \vec{m}) \leftarrow M(pk, z_M); \\ & (\vec{c}, \vec{d}) \leftarrow commit_{pk}(\vec{m}); \vec{c}' \leftarrow \mathcal{A}(pk, \vec{c}, M, z_M); \\ & \vec{d}' \leftarrow \mathcal{A}(\vec{d}); \vec{m}' \leftarrow decommit_{pk}(\vec{c}, \vec{d}) : \\ & D(s, \vec{m}, \vec{m}', z_D) = 1],\end{aligned}$$

where we demand that neither of the commitments in $\vec{c}$ is contained in $\vec{c}'$.

In the second game the success probability is given by

$$\begin{aligned}\widehat{\mathrm{Succ}}_{\mathcal{B},M,D}(k, z_M, z_D) \quad = \quad & P[(pk, s_{pk}) \leftarrow \widehat{K}(1^k); (s, \vec{m}) \leftarrow M(pk, z_M); \\ & \vec{m}' \leftarrow \mathcal{B}(pk, s_{pk}, t, M, z_M) : D(s, \vec{m}, \vec{m}', z_D) = 1],\end{aligned}$$

where $t = |\vec{m}|$, and where we interpret commitments and decommitments of vectors in the natural way.

Definition 2 *We say a commitment scheme is* non-malleable *if it has a modified key generator $\widehat{K}$ such that for all $\mathcal{A}$ there exists a $\mathcal{B}$, where for all M and D we have*

$$\mathrm{Succ}_{\mathcal{A},M,D}(k, z_M, z_D) - \widehat{\mathrm{Succ}}_{\mathcal{B},M,D}(k, z_M, z_D) < negl(k)$$

for all z_M, z_D with lengths bounded by some polynomial in k.

We say a commitment scheme is ϵ-non-malleable *if for every $\mathcal{A}$ and ϵ, there exists $\mathcal{B}$ running in time polynomial in k and ϵ^{-1}, such that the above difference is at most $\epsilon + negl(k)$.*

We will later construct two ϵ-non-malleable commitment schemes, where ϵ^{-1} may be any positive polynomial in k.

Remark Our definition does not allow the adversaries to receive any "history", i.e., side information about the messages they receive commitment to. Like [6] and [7] we do not know how to achieve such security. However, in all cases where messages consistent with the side information can be sampled efficiently, our security proofs remain valid.

2.4 Comparison with Previous Definitions of Non-malleability

As mentioned before non-malleability is a concept introduced in [9]. Their goal is to avoid man-in-the-middle attacks. Therefore, they define a protocol as being non-malleable if the adversary seeing a commitment cannot commit to a related value. [11] call this non-malleability with respect to commitment.

Unfortunately, this definition does not make much sense when considering unconditionally hiding commitments since we cannot speak about the content of such a commitment. Following the definition of [6, 7], we choose to consider the content of a commitment as what it is opened to. Such a definition is called non-malleability with respect to opening in [11]. We note that for unconditionally binding commitments non-malleability with respect to commitment is a stronger notion than non-malleability with respect to opening.

Our definition is stronger than the one in [6, 7]. Any commitment scheme that is non-malleable according to our definition is also non-malleable according to their definition. The main reason for our modification, reusability, was mentioned in the introduction: rather than just mauling one commitment the adversary may very well be attempting to maul many commitments at the same time. We show later that our definition is strictly stronger, there are schemes secure according to the [6, 7] definitions that are not secure according to our definition.

Increasing the number of commitments the adversary may see and may produce is not the only modification we have made. We give the message generator access to the public key for the commitment scheme. In our opinion, this is reasonable since in real life messages may indeed depend on this key. It does turn out that we pay a price for this though, since it forces us to give $\mathcal{B}$ access to some side information about the public key. In [6, 7] this was not needed since they could simply let $\mathcal{B}$ choose a completely new public key and still work within the same message space since it was independent of the key.

We have also changed the notation. [6, 7] speak of a distribution D instead of a messages generator M, and a relation approximator R instead of a distinguisher D. This change is purely cosmetic, but our notation seems to be more in line with other cryptographic literature.

Other definitions [11, 1] deal with interactive commitments. Barak [1] deals with a general method of setting up a URS interactively in a way such that a subsequent execution of a non-malleable commitment scheme based on a URS will lead to a non-malleable interactive commitment. One variation of our scheme is set in the URS model and can therefore be used with Barak's compiler.

We note that the most secure version of commitment is universally composable commitments [3, 4, 8]. The scheme presented in [4] is both non-interactive and universally composable. However, it is based on the assumption that trapdoor permutations exist. We provide some evidence in Section 5 that UC commitment must be based on stronger assumptions than NM commitment.

2.5 Comparison with Non-reusable NM

In this section we will be informal and call a commitment scheme (t, u)-NM if it is non-malleable (or ϵ-non-malleable for small ϵ's) in the case where the message generator produces a message vector of length t and the adversary produces a message of length u.

We will argue that $(1, 1)$-NM implies neither $(t > 1, 1)$-NM nor $(1, u > 1)$-NM. We give counterexamples both in the unconditionally hiding case and the unconditionally binding case.

Consider the unconditionally hiding commitment scheme from [7], which was proven $(1, 1)$-NM. This scheme is actually $(1, u)$-NM for any $u > 1$. However, for $(t, 1)$-NM the security proof fails for $t > 1$. It is not known whether the scheme is even $(2, 1)$-NM.

We can construct a variant of the scheme that is provably not $(2, 1)$-NM. We simply select two keys pk_1, pk_2 for the scheme, and a commitment to a message m now consists of the pair $(commit_{pk_1}(m), commit_{pk_2}(m))$. By running the proof from [7] "twice in parallel", we can prove that this commitment scheme is unconditionally hiding and $(1, 1)$-NM. However, consider a message generator M that simply outputs a message vector (m, m). Given commitments (c_1, c_2) and (c'_1, c'_2) to this message vector we may form the commitment (c_1, c'_2). After seeing openings to the original commitments, it is easy to open this commitment to the message m. In the first game, the adversary may therefore have success probability 1 in trying to commit to the same message. In the second game, however, we do not get any help in producing the message, and we can only try to guess m. The scheme is therefore not $(2, 1)$-NM.

All $(1, 1)$-NM commitment schemes we know of in the literature are in fact $(1, u)$-NM. However, we can also construct an example showing that this need not be the case. Again, we pick two public keys pk_1, pk_2 for the commitment scheme in [7]. A commitment to message m is formed as $(commit_{pk_1}(r), commit_{pk_2}(m \oplus r))$, where r is chosen at random. Again, it can be shown that this scheme is an unconditionally hiding $(1, 1)$-NM commitment scheme. Consider, however, an adversary in the first game getting such a commitment (c_1, c_2). He form the commitments $(commit_{pk_1}(0), c_2)$ and $(c_1, commit_{pk_2}(0))$. Later when receiving an opening of (c_1, c_2), he can open his commitments to $m_1 = r, m_2 = r \oplus m$. The exclusive-OR of these messages is m. On the other hand, any adversary in the second game has to guess m to form a pair of messages m_1, m_2 with $m = m_1 \oplus m_2$.

To show that for unconditionally binding commitment schemes $(1, 1)$-NM does not imply $(t > 1, 1)$-NM we use a semantically secure encryption scheme with errorless decryption. We construct a $(1, 1)$-NM commitment scheme the following way. As public key, we generate $2k$ public keys $pk_{1,0}, pk_{1,1}, \ldots, pk_{k,0}, pk_{k,1}$. When having to commit to a message m the sender selects a k-bit public key vk for a one-time signature scheme. For each $i = 1, \ldots, k$ he encrypts m using pk_{i,vk_i}, where vk_i is the i'th bit of vk. The commitment now consists of the k ciphertexts, the public verification key for the signature scheme, and a signature on all of it.

To see that this scheme is $(1, 1)$-NM for any u, consider an adversary $\mathcal{A}$ that upon seeing a commitment to some message m generates a commitment to a related message m'. Certainly, he cannot use the same public verification key since he does not know how to make signatures with this key. Therefore, he must produce at least one ciphertext with a previously unused public key.

We may now use $\mathcal{A}$ to break the semantic security of the cryptosystem. Given k ciphertexts under keys $pk_1, \ldots, pk_k$ of a message m (this is still semantically secure) we may select at random public keys $pk'_1, \ldots, pk'_k$, where we know the corresponding secret keys. We then generate a key vk for the signature scheme and arrange the keys $pk_1, \ldots, pk_k, pk'_1, \ldots, pk'_k$ in a pattern so that $pk_{i,vk_i} = pk_i$, while $pk_{i,1-vk_i} = pk'_i$. This way we can transform the ciphertexts into what looks like a commitment. Giving the adversary the commitment and the public key just constructed we let him form a commitment. Since his commitment uses one of the keys we formed ourselves, we may decrypt his commitment. If this commitment has any relation to the original ciphertexts, this means that we have broken the semantic security of the cryptosystem. Therefore, the commitment scheme must be non-malleable.

The commitment scheme is not $(k, 1)$-NM though. If the message generator outputs a message vector with k identical messages, then with overwhelming probability we will know encryptions of the message under all $2k$ keys, and therefore we may easily pick a new one-time signature key ourselves and form a commitment to the message. An adversary without access to commitments does not have the same easy time creating a commitment to the message in the message vector from the message generator.

Finally, we may argue as in [9] that $(1, 1)$-NM does not imply $(1, 2)$-NM for unconditionally hiding commitment schemes. Here we simply take a non-malleable cryptosystem with errorless encryption, pick two public keys pk_1 and pk_2, and form a commitment to m as $(E_{pk_1}(r), E_{pk_2}(m \oplus r))$. It can be proved that this commitment scheme is $(1, 1)$-NM. However, by the same argument as in the unconditionally binding case it is not $(1, 2)$-NM.

3 A Framework for Constructing Non-malleable Commitment

We present four tools that we are going to use in constructing non-malleable commitment schemes. After that, we present the construction based on these tools and prove that it yields an ϵ-non-malleable commitment scheme, for ϵ^{-1} being any positive polynomial in k. We leave constructions of the tools to Section 4.

3.1 Honest Sender Commitment

We introduce the notion of *honest sender* commitments. This covers a special type of commitment scheme with a weak binding property, it is not a real commitment scheme. When speaking of an honest sender commitment scheme, we do not require the commitment scheme to be binding. Instead, we only demand that it is binding whenever a commitment has been formed by an honest sender, i.e., when it has been formed according to the protocol.

Honest Sender Binding: For all adversaries $\mathcal{A}$:

$$P[pk \leftarrow K(1^k); m \leftarrow \mathcal{A}(pk); (c, d) \leftarrow HScommit_{pk}(m); d' \leftarrow \mathcal{A}(c, d);$$
$$m' \leftarrow HSdecommit_{pk}(c, d') : m' \neq \perp \wedge m' \neq m] < negl(k).$$

3.2 Known Message Attack Secure Signature

A signature scheme consists of a key generation algorithm K outputting a verification key vk and a signature key sk, together with algorithms $\text{sign}_{vk,sk}$ and verify_{vk}. We let A_{vk} denote the message space for the signature scheme. We allow some state to be kept with the signature key such that it has to be updated after each signature. Furthermore, we demand that signatures are bounded in length by some polynomial in k.

We will need a signature scheme that is existential forgery secure against known message attacks. This means that to break the scheme the adversary has to forge a signature on a message that has not been signed before. As help, the adversary has access to a message-signature oracle $\mathcal{O}$ that works like this. When queried $\mathcal{O}$ chooses a message m and some side information s about this message. The adversary receives (m, s) as well as a signature on m. In reality when speaking about known message attacks, we therefore have to specify the distribution with which the messages and the side information are chosen. Note that security against known message attack is weaker than attacks where the adversary is allowed to choose the messages.

The scheme is secure against known message attacks with distribution specified by M if for all $\mathcal{A}$ and auxiliary inputs z_M and $z_{\mathcal{A}}$ we have:

$$P[(vk, sk) \leftarrow K(1^k); (m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(vk, z_{\mathcal{A}}) : \\ \text{verify}_{vk}(m, \sigma) = 1] < negl(k),$$

where m has not been output by $\mathcal{O}$, and where $\mathcal{O}$ outputs triplets (m, s, σ), where (m, s) are distributed as the output of $M(pk, z_M)$, and σ is a signature on m.

3.3 Σ-protocol for the Signature Scheme

A *Σ-protocol for relation R* is a special type of a 3-move honest verifier zero-knowledge proof. It is a protocol for two parties, the prover P and the verifier V. P gets as input $(x, w) \in R$, V gets as input x, and the goal is for P to convince V that he knows w such that $(x, w) \in R$, without revealing information about w. We require that it be done using a protocol of the following form: P first computes and sends a message a to V. Then V returns a random challenge m of length k bits and P sends a response z to V. Finally, V outputs accept or reject. Besides the protocol being of this form, we require the following properties, which correspond to basic properties of well-known protocols of this form, e.g., Schnorr or Guillou-Quisquater:

Completeness: If $(x, w) \in R$, then the verifier accepts with overwhelming probability.

Special soundness: There exists an algorithm, which given x, and two accepting conversations (a, m, z) and (a, m', z'), where $m \neq m'$, outputs w such that $(x, w) \in R$.

Special honest verifier zero-knowledge: There exists an algorithm, the honest verifier simulator S, which given instance x (where there exists w such that $(x, w) \in R$) and any challenge m generates $(a, m, z) \leftarrow S(x, m)$ such that (x, a, m, z) is computationally indistinguishable from a successful conversation where m occurs as challenge.

In our case, what we need is a Σ-protocol for proving that we know a signature on an element $\alpha \in A_{vk}$, where the signature scheme is the one presented before. In other words, the relation R consists of elements $((vk, \alpha), w)$, where vk is a public key for the signature scheme, α is an element in A_{vk}, and w is a signature on α.

3.4 Message Authentication Code

A message authentication code is used for checking the integrity of messages. It consists of an algorithm MAC that takes as input an authentication key ak and some message. On basis of this one can compute a message authentication code mac. Given a message and a mac for this message, anybody who knows ak can check whether they fit together. However, without ak it is hard to compute correct message authentication codes.

In our case, we need to authenticate the initial message of the Σ-protocol for the signature scheme. If we assume such an initial message has at most length k', then we can do all operations in $\mathrm{GF}(2^{k'})$. We pick at random two elements $r_1, r_2 \in \mathrm{GF}(2^{k'})$. The authentication key will be $ak = (r_1, r_2)$. When given a we then compute $mac = r_1 a + r_2$. Clearly, anybody knowing ak can verify this. However, without ak, one only has a negligible chance of computing such a message authentication code correctly.

3.5 A Non-malleable Commitment Scheme

We are now ready to construct an NM commitment scheme.

Key Generation: We choose a key pk for a statistically hiding honest sender commitment scheme. Then we generate a public key vk for the signature scheme. We select a universal one-way hash function $h : \mathcal{C}_{pk} \to A_{vk}$.

The signature scheme must be known message attack secure with the message distribution given as hashes of honest sender commitments with side information that is openings of the honest sender commitments.

The public key is $PK = (pk, vk, h)$.

Commitment: To commit to a k-bit message m we choose a key ak for the message authentication scheme. We set $(c, d) = HScommit_{pk}(ak)$, and compute $\alpha = h(c)$. Now we simulate a proof of knowledge of a signature on α with challenge m. We set $(a, m, z) = S((vk, \alpha), m)$. Finally, we compute $mac = MAC_{ak}(a)$.

The commitment to m is $C = (c, a, mac)$, while the decommitment information is $D = (m, d, z)$.

Decommitment: We open the honest sender commitment to get the authentication key. We compute α as above. We verify the message authentication code. Finally, we verify the proof. If everything goes well we output m. Otherwise, we output $\perp$.

This commitment scheme is binding. If we could open a commitment in two different ways, then we could by the special soundness of the Σ-protocol find a signature on α, which is infeasible since we do not know the signature key. The commitment scheme is also hiding. The only possible link to information about m is a. Since the Σ-protocol is special honest verifier zero-knowledge this a is indistinguishable from the corresponding value occurring in a real conversation – but in real conversations, a is independent of m.

Remark The only part of the commitment scheme that can contain any information about m is the initial message a of the simulated proof. Therefore, if the proof system is statistically or perfectly special honest verifier zero-knowledge then the commitment is unconditionally hiding.

Theorem 3 *The commitment scheme is equivocable.*

Proof. We let the modified key generator $\widehat{K}$ be one that runs K but as side information outputs the signature key sk associated with the verification key. This means that we can sign any message.

With this information, we can of course sign any $\alpha \in A_{vk}$. This means, no matter the α we can, in a real proof, answer the challenge m in the Σ-protocol. Therefore, we simply modify the commitment procedure to pick a according to the Σ-protocol and to return also the equivocation information σ, a signature on α.

When having to produce the decommitment information we then open as usual, except we compute the answer z to the challenge m with our knowledge of the signature on α.

To see that this gives us the algorithms needed for the commitment scheme note first that the public keys generated by K and $\widehat{K}$ are identically distributed.

Imagine now that we could distinguish real commitments and equivocated commitments. By a hybrid argument, we can reduce this to distinguishing two scenarios that only differ in one commitment being equivocated instead of being formed and opened according to the commitment scheme.

But this means that we can distinguish simulated and real proofs using the Σ-protocol, since this is the only difference between equivocable and real commitments. We thus arrive at a contradiction with the special honest verifier zero-knowledge property of the Σ-protocol. $\square$

Remark If the initial message a in the Σ-protocol can be computed without knowledge of the witness then the commitment scheme becomes strongly equivocable because the witness, the signature, can just be generated when we open the commitment.

3.6 Non-malleability

We claim that the commitment scheme in the previous section is non-malleable.

Theorem 4 *The commitment scheme is ϵ-non-malleable for any ϵ^{-1} polynomial in k.*

Proof. We let the modified key generator $\widehat{K}$ be the same as the modified key generator used for equivocation. In other words along with the public key

PK it also provides as side information the signature key sk. According to Lemma 5 below, the adversary $\mathcal{A}$ cannot open commitments in two different ways, even when seeing equivocations of other commitments. We use this fact when constructing $\mathcal{B}$ and later proving that $\mathcal{B}$ fares as well as $\mathcal{A}$.

First let us look at the real execution of the commitment scheme. We modify this experiment by letting the public key be generated by $\widehat{K}$ and forming equivocable commitments instead of real commitments. We then use the equivocation information to open the commitments to the messages chosen by M. According to Theorem 3 these two experiments are indistinguishable. Our task is therefore reduced to find an algorithm $\mathcal{B}$ that can make $\widehat{\mathrm{Succ}}_{\mathcal{B},M,D}$ at least as high as the success probability in the modified experiment minus ϵ.

We now describe the algorithm $\mathcal{B}$. It runs a copy of $\mathcal{A}$ and starts by giving t equivocable commitments to $\mathcal{A}$. As a response, $\mathcal{A}$ produces a vector of commitments. As mentioned. the intuition is that $\mathcal{A}$ cannot open the commitments he produced in more than one possible way. Our goal is therefore to find out the contents of as many of $\mathcal{A}$'s commitments as possible, and we will then submit this information to D.

To this end $\mathcal{B}$ runs $M(pk, z_M)$ $4k\mathrm{time}_M(k)\mathrm{time}_{\mathcal{A}}(k)/\epsilon^2$ times. It picks out the first $2k\mathrm{time}_{\mathcal{A}}(k)/\epsilon$ of the message vectors that have length t. We note that in case there is more than $\epsilon/(2 \cdot \mathrm{time}_M(k))$ chance that M will output a vector of length t there is also overwhelming chance that $\mathcal{B}$ samples enough vectors of length t. The probability of being in a situation where $\mathcal{B}$ cannot sample enough vectors of length t is therefore less than $\epsilon/2 + negl(k)$ since $\mathrm{time}_M(k)$ is an upper bound on t. In the following, we only investigate the experiment conditioned on having sampled enough message vectors.

For each message tuple $\vec{m}$ sampled from M, $\mathcal{B}$ now equivocates the commitments it gave to (its copy of) $\mathcal{A}$, such that $\vec{m}$ is opened. For each of its own commitments $\mathcal{A}$ may or may not in each run produce an opening to a message $m' \neq \perp$. For each commitment, $\mathcal{B}$ takes the first opening different from $\perp$. Finally, $\mathcal{B}$ hands the message vector $\vec{m}^*$ found to the distinguisher (putting $\perp$ in unopened positions).

This ends the description of $\mathcal{B}$. We observe that $\mathcal{A}$ simulated by $\mathcal{B}$ sees exactly the same distribution as $\mathcal{A}$ does in the modified real life experiment, where we condition both experiments on M outputting a message vector with a length t (and where t is such that the probability of getting length t is at least $\epsilon/2\mathrm{time}_M(k)$). More precisely, the two games behave in exactly the same way up to the point where $\mathcal{A}$ is about to receive the decommitment of $\vec{m}$. At this point, we can either play the modified real-life experiment, i.e., give $\mathcal{A}$ the "right" $\vec{d}$ and have him produce $\vec{m}'$. Or we can execute $\mathcal{B}$'s algorithm producing $\vec{m}^*$.

Now fix a "snapshot" of all variables known to $\mathcal{A}$ just after it has formed its commitments. Consider any single commitment $\mathcal{A}$ made. If the probability (taken over the choice of $\vec{m}$) that $\mathcal{A}$ will open it is smaller than $\epsilon/(2 \cdot \text{time}_A(k))$, we say the commitment is *bad*, otherwise it is *good*. Since $\text{time}_A(k)$ is an upper bound on the number of commitments $\mathcal{A}$ can produce, the probability that $\vec{m}'$ contains non-$\perp$ values for any of the bad commitments is at most $\epsilon/2$. On the other hand, since $\mathcal{B}$ uses a total of $2k\text{time}_{\mathcal{A}}(k)/\epsilon$ samples of $\vec{m}$-values, the probability that $\vec{m}^*$ contains non-$\perp$ values for all good commitments is overwhelming (it may contain more non-$\perp$ values, but this does not matter since the probability of the distinguisher outputting success cannot be decreased by replacing $\perp$'s with messages).

Summarizing what we have so far, except with probability $\epsilon + negl(k)$, $\mathcal{B}$ manages to sample enough $\vec{m}$-values, $\mathcal{A}$ does not open any of the bad commitments in the modified real life game, and $\mathcal{B}$ learns a way to open all good commitments. Hence, the only way in which $\mathcal{B}$ could do worse than $\mathcal{A}$ is if, even assuming all the above, D outputs 1 with significantly larger probability when seeing $\vec{m}'$, than when seeing $\vec{m}^*$. But this in particular means that for at least one good commitment $\vec{m}'$ contains m', and $\vec{m}^*$ contains m^* such that $m' \neq m^*$ and $m', m^* \neq \perp$. By Lemma 5 below, this occurs with negligible probability, assuming security of the one-way hash function, the honest sender commitments, the mac scheme and the signature scheme. This is so since $\mathcal{B}$ does not use the signature key beyond what it could do with access to an oracle as described in Lemma 5. Overall, $\mathcal{B}$ does worse than $\mathcal{A}$ in the real experiment only in case of events that occur with total probability at most $\epsilon + negl(k)$. $\square$

Lemma 5 *For any adversary $\mathcal{A}$ and any auxiliary input z:*

$$\begin{aligned} P[(PK, sk) \leftarrow \widehat{K}(1^k); C \leftarrow \mathcal{A}^{\mathcal{O}_1}(PK, z); (D_1, D_2) \leftarrow \mathcal{A}^{\mathcal{O}_2};\\ m_1 \leftarrow \text{decommit}_{PK}(C, D_1); m_2 \leftarrow \text{decommit}_{PK}(C, D_2) :\\ m_1 \neq \perp \wedge m_2 \neq \perp \wedge m_1 \neq m_2] < negl(k). \end{aligned}$$

Here $\mathcal{O}_1, \mathcal{O}_2$ are oracles that acts like this:

- $\mathcal{O}_1$ *on query* (Commit) *sets* $(C, \sigma) \leftarrow \widehat{\text{commit}}_{PK}(sk)$. *It stores* (C, σ) *and answers the query with* C.

- $\mathcal{O}_2$ *on query* (Equiv, C, m) *where* (C, σ) *is stored returns* $\mathrm{equiv}_{PK,sk}(C, \sigma, m)$.

The adversary $\mathcal{A}$ is restricted in the sense that its output C must not have been produced by $\mathcal{O}_1$.

Remark. Lemma 5 has been modified slightly in comparison with the paper published at STOC 2003. We thank Phil MacKenzie and Ke Yang for pointing out a flaw in the lemma as originally stated.

Proof. Let us look at a commitment $C = (c, a, mac)$ produced by $\mathcal{A}$, which is different from any commitments produced by $\mathcal{O}_1$. We first argue that $\alpha = h(c)$ must be different from the α's of the equivocable commitments the adversary has seen. Imagine contrarily that the adversary had seen an equivocable commitment $C' = (c', a', mac')$ with $h(c') = \alpha$. When forming c' in an equivocable commitment it is generated independently of h. This means that $c = c'$ or the adversary would have broken the universal one-wayness of the hash. If $c = c'$ we recall that c' has been computed by an honest sender. In other words, the adversary can only open this commitment in the same way as when equivocations are made. Since the honest sender commitment is unconditionally hiding the adversary has no clue about the authentication key ak. For this reason the adversary has negligible chance of producing a', mac' that will be accepted with authentication key ak, unless the adversary recycles also a and mac. But in the latter case the adversary has just made a copy of the equivocable commitment. Left is to consider the case where α has not been used in any of the equivocable commitments.

We are now in the case where the adversary has a different α than the ones in the equivocable commitments. Notice that an adversary seeing many openings of these commitments may learn signatures on the α's used in the equivocable commitments. However, the equivocation information is used for nothing else so the adversary cannot learn more than these signatures.

This means the adversary has access to a known message attack. The oracle $\mathcal{O}_2$ only uses the signature key in a way corresponding to a known message attack, i.e., $\mathcal{O}_2$ could be implemented without the signature key but with access to a known message attack signature oracle. If the signature scheme is secure against known message attacks where the messages to be signed are distributed in the way the α's in the equivocable commitments are, and the side information to go with these messages are openings of the corresponding honest sender commitments, then the adversary is unable to sign his own α. By the special soundness of the Σ-protocol this means that the adversary cannot open his commitment to two different messages, because this would imply answering two different challenges and thus also the ability to extract a signature on α. $\square$

4 Constructions of Non-Malleable Commitment Schemes

4.1 An Implementation Based on the Strong RSA Assumption

Let n be an RSA modulus. The strong RSA assumption says that given a random number $y \in Z_n^*$ it is infeasible to find $e > 1, x \in Z_n^*$ so that $y = x^e \bmod n$.

Based on the RSA assumption we can construct an unconditionally hiding commitment scheme (and thereby honest sender binding) in the following way. We select n as a k-bit RSA modulus, q as a $2k + 1$-bit prime, and y at random from Z_n^*. A commitment to an element $x \in Z_q$ is formed as $commit_{(n,q,y)}(x; r) = y^x r^q \bmod n$, where r is chosen at random from Z_n^*. The decommitment information is (x, r).

We may set up a signature scheme with the same public key. Here the public key simply consists of (n, y). The signature scheme's message space A_n consists of primes of bit-length within $k+1$ and $2k$. We define a signature on $\alpha \in A_n$ to be w, where $y = w^\alpha \bmod n$. The function h is defined to be a function that on $c \in Z_n^*$ outputs the smallest prime larger than $2^k c$. Under well-established number theoretic conjectures [5], this prime is $2^k c + O(k^2)$. This makes h injective and efficiently computable.

We define the relation R as $\{((n, y, \alpha), w) | y = w^\alpha \bmod n\}$. Now, the required Σ protocol is just the Guillou-Quisquater protocol:

1. The prover sends $a = r^\alpha \bmod n$ for random $r \in Z_n^*$.

2. The verifier sends a random k-bit number m.

3. The prover sends $z = rw^m \bmod n$, and the verifier checks that $z^\alpha = ay^m \bmod n$.

It is easy to check (and well-known) that this protocol has the required properties. Note that by construction of A_n, it is guaranteed that all values of the challenge m are less than any $\alpha \in A_n$.

Finally, there is the message authentication scheme. The authentication key is $ak = (r_1, r_2)$, where r_1, r_2 are picked at random from Z_n^*. The message authentication code on an element $a \in Z_n^*$ is $mac = r_1 a + r_2 \bmod n$.

A commitment on a k-bit message m is now $C = (c, a, mac)$ and the decommitment information is $D = (m, d, z)$, where $(c, d) = commit_{(n,q,y)}(r_1, r_2)$, $(a, m, z) = S((n, y, \alpha), m)$ and $mac = r_1 a + r_2 \bmod n$.

Theorem 6 *Under the strong RSA assumption and the number theoretic conjecture the commitment scheme is unconditionally hiding, strongly equivocable and ϵ-non-malleable for any ϵ^{-1} polynomial in k.*

Sketch of proof. We claim without proof that we have presented above the ingredients of the construction in Section 3.5. We presented a commitment scheme, a notion stronger than honest sender commitment schemes. We presented an injective function from such commitments, and thereby a universal one-way hash function. We presented a signature scheme that is existential forgery secure against known message attacks with respect to the distribution of the commitments. To see this observe that the commitments are formed independently of w. If an adversary $\mathcal{A}$ could break the signature scheme we could therefore set up $\mathrm{time}_{\mathcal{A}}(k)$ commitments in advance and choose $y = w^{\alpha_1 \cdots \alpha_{\mathrm{time}_{\mathcal{A}}(k)}} \bmod n$. $\mathcal{A}$ being able to find a new signature implies our ability to find a non-trivial root of w; thereby violating the strong RSA assumption. We presented a Σ-protocol for knowledge of a signature. Finally, we presented a message authentication scheme for the initial message of the Σ-protocol.

It follows from Theorem 4 that we are dealing with a commitment scheme, and the commitment scheme is ϵ-non-malleable for any ϵ^{-1} polynomial in k.

The Σ-protocol is actually perfect honest verifier zero-knowledge. This implies that from the initial message no information is released about the message. This shows that the commitment scheme is unconditionally hiding.

It follows from Theorem 3 that the commitment scheme is equivocable. Since the initial message a in the Σ-protocol is created independently of the signature the commitment scheme is strongly equivocable. $\square$

4.2 An Implementation Based on any One-Way Function

Pick any universal one-way hash function h that compresses strings to length k. Such universal one-way hash functions exist under the assumption of one-way functions [15].

Signatures can also be based on one-way functions. For instance, we may modify the Merkle one-time signature scheme the following way. We form a balanced binary tree of height k. Our public key consists of a key for a one-time signature on the root of the tree. A message to be signed is placed at a previously unused leaf, and the signature consists of one-time signatures and public keys on the path up to the root. At all internal nodes, we sign the public one-time signature keys of the children. Finally, for any internal node we have used we store the public keys chosen for the two children and always use these keys when passing by a node again. This signature scheme is existential forgery secure against adaptive chosen message attack, more than enough for our purpose.

A Σ-protocol exists for this signature scheme, simply because the known zero-knowledge protocols for NP-complete problems have the right form: since deciding validity of a signature is in NP, we can transform our problem to, say, a Hamiltonian path problem and use Blum's zero-knowledge interactive proof system for this language. As a building block, we need bit-commitment, which we can get from any one-way function [14]. Although this commitment scheme is originally interactive, it can be made non-interactive in the CRS model, following [6]. In its basic form, this is a 3-move protocol with soundness error 1/2, and we repeat this in parallel k times. It is easy to see (and well-known) that this will satisfy our conditions for the Σ-protocol.

With respect to the MAC, we may use the message authentication code mentioned in the framework. This leaves the question of how to make an unconditionally hiding honest sender commitment scheme. We choose a universal$_2$-function $f : \{0,1\}^{3k+1} \to \{0,1\}^{3k+1}$. We also choose a universal one-way hash function $g : \{0,1\}^{3k+1} \to \{0,1\}^k$. A commitment to a bit b is now formed by picking at random a $3k$-bit string r and letting $\mathit{commit}(b;r) = g(f(b \,\|\, r))$. This bit-commitment scheme is unconditionally hiding since most elements in the commitment space have a roughly equal number of $0 \,\|\, r$- and $1 \,\|\, r$-strings as preimage.

Theorem 7 *The commitment scheme is ϵ-non-malleable for any ϵ^{-1} being* *a positive polynomial in k. The commitment scheme is also equivocable.*

Proof. We have supplied the building blocks for the construction in Section 3.5. The theorem follows from Theorem 3 and Theorem 4. □

Remark Most cryptographic tools based on simpler primitives are black-box constructions. It is therefore worth noting that our construction is not a black-box construction. The reason for this is that we use general reduction techniques to form an NP problem and then create a Σ-protocol for this. This reduction depends on the one-way function that we use.

4.3 Unconditional Hiding, Unconditional Binding and Uniform Random String

It is not known whether unconditionally hiding commitment schemes can be constructed from one-way functions. Marc Fischlin has recently shown that there is no proof of one-way functions implying unconditionally hiding commitments that relativizes [10]. However, assuming the minimal possible, namely that we have an unconditionally hiding bit-commitment scheme, we note that it can be transformed into an unconditionally hiding non-malleable and equivocable bit-commitment scheme. This is done by using the unconditionally hiding commitment scheme to make all the commitments in the Σ-protocol in the previous section. This way, even a computationally unbounded adversary seeing a, has no idea about how the sender is capable of opening the commitments, and thus no idea about the bits of m.

Since we can make unconditionally binding commitment schemes from one-way functions, an obvious question is whether we can use our construction to get an unconditionally binding non-malleable commitment scheme. The answer to this question is yes.[2] The idea is to modify the commitment scheme we constructed before so the public key now also includes an unconditionally binding commitment to a bit $b = 0$. We may from one-way functions construct a Σ-protocol for proving that the commitment contains $b = 1$ with challenge m in addition to the proof of knowledge of a signature. Obviously, this is a false statement, so it is impossible for even a computationally unbounded sender to find a witness for this. Therefore, by the

[2]Of course, the non-malleability cannot hold against a computationally unbounded adversary. Only the binding property holds against a computationally unbounded adversary.

special soundness of the Σ-protocol he cannot open the commitment to reveal two different messages. However, since the commitment to b is hiding, there is no problem in simulating a proof for $b = 1$, nobody will notice that anything is wrong. For non-malleability we let the modified key-generator output a public key with a commitment to $b = 1$. Now we may equivocate commitments and therefore the simulation proof where we use rewinding to make the adversary open his commitments goes through.

The CRS used in the one-way function based commitment scheme is not uniformly random in its basic form, but we can modify the scheme so it can use the URS model instead. By backtracking through the papers constructing the tools we use we note that all of them can be built from a one-way function and uniformly random bits. The only exception is the signature scheme, where we do not know whether the verification key can be chosen uniformly at random. Nevertheless, we can get by with interpreting some of the uniformly random bits as an unconditionally binding commitment to a verification key. We can modify the Σ-protocol to the case where we prove that we know an opening of the bits to a public verification key and can sign the α with this key. In reality we do not have such a key, however, since the commitment scheme is hiding the security proof goes through.

5 UC Commitment implies Key Exchange or Oblivious Transfer

We briefly recall how the UC framework [2] is put together: The framework allows us to say that a protocol π securely implements an ideal functionality $\mathcal{F}$. In the case of UC commitment, $\mathcal{F}$ will be a trusted party that receives in private the committed value m, and later reveals it to the receiver on request from the sender. The real-life protocol π is attacked by an adversary $\mathcal{A}$ who schedules communication and adaptively corrupts players and controls their actions, while an attack in the ideal setting (where $\mathcal{F}$ is present) is limited to corrupting players, changing the inputs they send to $\mathcal{F}$ and block/observe the results coming back. Security of π now means that for every efficient real-life adversary $\mathcal{A}$, there is an efficient ideal model adversary $\mathcal{S}$ who can obtain "the same" as $\mathcal{A}$. To define what this means, an environment machine $\mathcal{Z}$ is used. $\mathcal{Z}$ may communicate with the adversary ($\mathcal{A}$ or $\mathcal{S}$) at any point during the attack, and it may specify inputs for the honest players and see the results they obtain. Security more precisely means that it is infeasible for $\mathcal{Z}$ to tell if it is talking to $\mathcal{A}$ in a real-life attack, or to $\mathcal{S}$ in an ideal model attack. So to prove UC security, one must construct, given $\mathcal{A}$, a suitable $\mathcal{S}$. It is important to note that, in order to ensure robustness against concurrent composition and adaptive security, the model explicitly forbids rewinding of $\mathcal{Z}$, so $\mathcal{S}$ is forced to go through the entire game in the same time sequence as in real life.

Theorem 8 *If there exists a non-adaptive UC commitment scheme in the CRS model, then there exists a key exchange protocol (secure against passive attacks). Furthermore, if there exists a non-adaptive UC commitment scheme in the URS model, then there exists an oblivious transfer protocol (secure against passive attacks).*

Remark Even if we start from a UC commitment scheme, the key exchange or OT protocol we obtain are only secure against a static attack where one of the players is corrupt from the beginning. They are not necessarily secure against an adaptive attack and so are not necessarily universally composable. We do not know if the construction can be improved to produce UC protocols, but it would be interesting if the answer were affirmative. For instance, this would mean that the implementation of any UC functionality from [4] could be based on UC commitment only.

Sketch of proof. Assume we have a UC commitment scheme for sender C and receiver R, and let π be the protocol used when C commits to, say, a bit b. Let $\mathcal{A}$ be the adversary that first corrupts C, it then sends the CRS to $\mathcal{Z}$, and now (through its communication with $\mathcal{Z}$) lets $\mathcal{Z}$ decide the actions of C. Let $\mathcal{Z}$ be the environment that expects the behaviour we just specified for the adversary (this means that $\mathcal{Z}$ gets to play C's part of the protocol). $\mathcal{Z}$ follows the honest protocol for C, in order to commit to some bit b. Once this is over, it asks to have the commitment opened, again $\mathcal{Z}$ plays honestly C's part of the opening, and it receives as a result of this the bit that R gets as output (which should normally be equal to b). Finally, $\mathcal{Z}$ tries to distinguish whether it is in the real-life model or the ideal process by outputting a bit.

UC security guarantees that there exists a good ideal model adversary $\mathcal{S}$ for this $\mathcal{A}$. Of course, $\mathcal{S}$ must send a CRS to $\mathcal{Z}$ and then play in an indistinguishable way R's part of π against $\mathcal{Z}$ – otherwise $\mathcal{Z}$ could immediately distinguish. Moreover, after completing π, $\mathcal{S}$ must be able to compute the bit b that $\mathcal{Z}$ "committed to", since this bit must be given to $\mathcal{F}$ before opening can take place in the ideal model. If this bit is not correct, R at opening time receive a bit different from b in the ideal model. This would allow $\mathcal{Z}$ to distinguish.

This observation immediately implies a secure key exchange protocol for parties A, B: B starts an instance of $\mathcal{S}$ and sends the CRS it produces to A, who then chooses a random bit b and runs the commitment protocol acting as the sender. B lets $\mathcal{S}$ play the receivers part, and can now extract b by the observation above. An eavesdropper is clearly in a situation that is no better than that of an honest receiver in the non-adaptive UC commitment scheme, and so he cannot distinguish $b = 0$ from $b = 1$.

If we are in the URS[3] model, we can build an OT protocol for sender A with input bits b_0, b_1 and receiver B with selection bit c. The goal is for B to learn b_c and nothing else, whereas A should learn nothing new. We assume both A and B are "honest-but-curious". The OT protocol goes a follows

1. B computes and sends to A strings CRS_0, CRS_1, where he runs $\mathcal{S}$ to get CRS_c and chooses CRS_{1-c} at random.

2. A executes the commitment protocol π twice, playing the role of C using CRS_0, respectively CRS_1 as reference string and b_0 respectively b_1 as the bit to commit to. B follows the protocol for R in the instance that uses CRS_{1-c}, and lets $\mathcal{S}$ conduct the protocol in the other case, i.e., he simply relays messages between A and $\mathcal{S}$.

3. When $\mathcal{S}$ outputs a bit (to give to $\mathcal{F}$), B uses this as his output.

From A's point of view, the two instances of π are indistinguishable, since otherwise $\mathcal{S}$ would not satisfy the conditions in the UC definition, as we discussed. Hence, A learns nothing new. On the other hand, we also argued that $\mathcal{S}$ must output the bit committed to, so B does indeed learn b_c. With respect to b_{1-c}, B is in exactly the same position as party R when receiving a commitment. Since UC commitments are hiding, B does not learn b_{1-c}. □

[3]We need to ensure that B can generate the CRS without getting any information from the key generation that allows it to break the OT protocol. Since it is easy to generate a URS at random, we certainly are guaranteed this in the URS model.

6 Application to UC Commitment

6.1 Damgård-Nielsen UC Commitment

Let us briefly sketch why the scheme of [8] get a CRS that grows linearly with the number of players.

The UC commitment scheme is based on a *mixed commitment scheme*. This is a commitment scheme where we first generate a master key N that defines the message space $\mathcal{M}_N$ of the commitment scheme. Knowing N we may now generate a public key K for the commitment scheme. This key will belong to a group $\mathcal{K}_N$. If chosen at random, this key with overwhelming probability will be an extraction key X-key. However, we may also select the key as an equivocation key E-key. X-keys and E-keys are indistinguishable. Knowing a trapdoor t_N associated to the master key, we may extract the contents of commitments formed under X-keys. On the other hand, if we use an E-key K to commit, with some associated trapdoor information t_K we may equivocate the commitment to anything.

The UC commitment scheme is interactive. The sender and receiver run a two round coin-flip protocol to determine a public key K for the mixed commitment scheme. For this purpose, they have an equivocable commitment scheme. The sender commits to $K_1 \in_R \mathcal{K}_N$, the receiver sends back $K_2 \in_R \mathcal{K}_N$, and the sender in the third round opens his initial commitment so both the sender and receiver now knows the key $K = K_1 + K_2$. In the third round, the sender also sends a commitment to the message m under key K. To open the commitment the sender just has to open the mixed commitment to m.

In the UC commitment scheme the ideal process adversary $\mathcal{S}$ runs a simulated real life execution where it uses $\mathcal{A}$ as a black-box to produce communication with $\mathcal{Z}$. It faces two problems: When a message is submitted to $\mathcal{F}$ it must simulate this without knowing the actual message. And when $\mathcal{A}$ sends a UC commitment on behalf of a corrupted party, then it must figure out which message to give to $\mathcal{F}$. These problems can be solved if we commit to the message using an E-key whenever sending a UC commitment on behalf of an honest party, and force $\mathcal{A}$ to use an X-key whenever it makes a UC commitment. We enable $\mathcal{S}$ to put itself in this situation by giving it equivocation information for the initial commitment scheme used in the coin-flip protocols. This way it may bias the coin-flip protocols and thus always end up with the right type of key K.

The problem in the commitment scheme above is that in [8] each player must have an individual key for the initial commitment scheme. Otherwise, $\mathcal{A}$ might be able to use the fact that $\mathcal{S}$ is equivocating commitments to equivocate commitments on his own. He could possibly even select E-keys for his own commitments and then ruin the entire simulation since $\mathcal{S}$ could no longer extract messages.

This is exactly the kind of problem that the commitment schemes in this paper can solve: they allow us to create a situation where for a fixed size public key, a simulator can make any number of commitments and equivocate them, while the adversary cannot do so. This is the essential property that implies non-malleability and is also useful here. So instead of letting each participant have his own commitment key on the CRS, we let a public key for the strong RSA based commitment scheme be on the CRS.

6.2 Improving the Damgård-Nielsen UC Commitment Scheme

In the following we describe the ideal functionality $\mathcal{F}$, the UC commitment scheme and the ideal process adversary $\mathcal{S}$.

6.2.1 The Ideal Functionality

We label the dummy parties in the ideal process $\widetilde{P}_1, \ldots, \widetilde{P}_n$ to distinguish them from the parties $P_1, \ldots, P_n$ in the real-life model.

Generating the key: Generate a public key (pk, N) for the commitment scheme. Send (sk, t_N) to $\mathcal{S}$.

Committing: On $(\text{commit}, sid, cid, P_i, P_j, m)$ from $\widetilde{P}_i$ send $(\text{receipt}, sid, cid, P_i, P_j)$ to $\mathcal{S}$ and $\widetilde{P}_j$. Ignore all subsequent $(\text{commit}, sid, cid, \ldots)$ messages.

Opening: On $(\text{open}, sid, cid, P_i, P_j)$ from $\widetilde{P}_i$ where $(\text{commit}, sid, cid, P_i, P_j, m)$ has been recorded, send $(\text{verify}, sid, cid, P_i, P_j, m)$ to $\mathcal{S}$ and $\widetilde{P}_j$.

6.2.2 The Commitment Scheme[4]

We call the public key and equivocation key for the strong RSA based scheme pk and sk. Like above we use N, t_N, K, t_K for the keys and trapdoors in the mixed commitment scheme. The CRS for the commitment scheme is (pk, N).

Commitment: Player P_i makes a commitment to a message m and sends it to P_j by doing the following:

Initial Commitment: P_i selects $K_1 \in \mathcal{K}_N$ randomly and sets $(c, d) = com_{pk}(sid, cid, P_i, P_j, m, K_1; r)$. He sends $(\text{init}, sid, cid, P_i, P_j, c)$ to P_j.

Commitment Response: P_j on incoming message $(\text{init}, sid, cid, P_i, P_j, c)$ selects $K_2 \in \mathcal{K}_N$ at random. He sends $(\text{response}, sid, cid, P_i, P_j, K_2)$ to P_i and ignores subsequent $(\text{init}, sid, cid, \ldots)$ messages.

Final Commitment: P_i on incoming message $(\text{response}, sid, cid, P_i, P_j, c)$ from P_j checks that he has sent an initial commitment and not sent a final commitment in this run of the protocol. In that case he sets $K = K_1 + K_2$ and $(C, D) = commit_K(m)$. He sends $(\text{final}, sid, cid, P_i, P_j, K, C)$ to P_j.

Receival: P_j on $(\text{final}, sid, cid, P_i, P_j, K, C)$ from P_i checks that he has earlier sent out a $(\text{response}, sid, cid, \ldots)$ message to P_i. He outputs $(\text{receipt}, sid, cid, P_i, P_j)$. He ignores subsequent $(\text{final}, sid, cid, \ldots)$ messages from P_i.

Opening: To open the commitment P_i sends $(\text{open}, sid, cid, P_i, P_j, d, D)$ to P_j.

Verification: P_j on $(\text{open}, sid, cid, P_i, P_j, d, D)$ from P_i checks that he has sent out a receipt $(\text{receipt}, sid, cid, \ldots)$. He sets $K_1 = K - K_2$ and checks that $(sid, cid, P_i, P_j, m, K_1) = decom_{pk}(c, d)$ and $m = decommit_K(C, D)$. In that case, he outputs $(\text{verify}, sid, cid, P_i, P_j, m)$.

[4]This protocol is modified slightly compared with [8], since in their proof they allowed $\mathcal{S}$ to block commitments from being submitted to $\mathcal{F}$, something which is not possible under the standard UC model [2]. The idea and the proof of security is the same as in [8] though.

6.2.3 The Ideal Process Adversary

$\mathcal{S}$ runs a copy of $\mathcal{A}$ trying to simulate everything $\mathcal{A}$ would see in a real-life execution of the commitment protocol. Messages between $\mathcal{A}$ and $\mathcal{Z}$ are simply forwarded.

Whenever commitments are made using $\mathcal{F}$ $\mathcal{S}$ will not know the content but knowing the equivocation keys it can select the commitments so that they are equivocable. This way it can make the simulation for honest parties work out by making appropriate equivocations.

If $\mathcal{A}$ decides to corrupt a (simulated) party, then $\mathcal{S}$ corrupts the corresponding ideal-process party and learns all messages it has committed to. It can then perform suitable equivocations and give $\mathcal{A}$ those equivocations. Here it is important that both the strong RSA based commitment scheme and the mixed-commitment scheme from [8] have decommitment information d, D that contain all randomness r, R used in forming the commitments, so $\mathcal{A}$ can see all the randomness it would when corrupting a party in the real-life model. In case $\mathcal{A}$ corrupts a party after sending the initial commitment but before sending the final commitment we equivocate to a randomly chosen K_1.

6.3 Security Proof of the UC Commitment Scheme

Theorem 9 *The commitment scheme described above is universally composable.*

Sketch of proof. We show that $\mathcal{Z}$ cannot distinguish the real-life protocol and the ideal process model. For this purpose we look at the following distributions of $\mathcal{Z}$'s (binary) output:

- $REAL$ is the real-life execution.
- HYB_1 is a modified real-life experiment: Instead of the usual CRS, we select the CRS with equivocable keys. We then equivocate both initial and final commitments. In particular this means that for honest parties about to send the final commitment we equivocate the initial commitment so K becomes an E-key.
- HYB_2 is a modified ideal process. When $\mathcal{A}$ makes a commitment where he is involved in both the initial phase and the final phase of

committing we do not extract the message m. Instead, we submit 0 to $\mathcal{F}$ if $\mathcal{A}$ later opens this commitment to contain m, then we patch $\mathcal{F}$ by inserting m in place of 0.[5]

- $IDEAL$ is the ideal process.

$REAL$ and HYB_1 are indistinguishable. To see this let us modify the real-life model step by step into the hybrid model. First, we select the CRS with equivocable key pk, ek and trapdoor t_N associated with N. Second we form the initial commitments in an equivocable manner and equivocate when we have to open. Whenever an honest party has to make the final commitment, we let it select K_1 at random at this point and equivocate the initial commitment to K_1. By definition of equivocability, this change cannot be noticed by $\mathcal{Z}$. However, now we may instead simply choose K as a random E-key and set $K_1 = K - K_2$. Since E-keys are indistinguishable from random keys this change cannot be noticed. Finally, this enables us to equivocate when the party's commitment has to be opened without $\mathcal{Z}$ being able to notice it.

HYB_1 and HYB_2 are the same experiment phrased in two different ways.

HYB_2 and $IDEAL$ are indistinguishable. The difference between them consists in the opening from the adversary that is patched. There are three possibilities for the message that is patched:

1. The final commitment uses X-key K. Then $\mathcal{S}$ extracts the correct message m.

2. The final commitment uses E-key K made by $\mathcal{S}$. The initial (equivocated) commitment has been opened to contain some m. Due to the binding property $\mathcal{A}$ cannot open the commitment otherwise so the patching is correct.

3. The final commitment does not use an X-key K and K has not been selected by $\mathcal{S}$.

   If the initial commitment is created by $\mathcal{A}$ it is binding and thus the probability of K not being an X-key is negligible.

   If $\mathcal{A}$ copies the initial commitment formed by $\mathcal{S}$ on behalf of an honest player then it is bound by any opening produced by $\mathcal{S}$. Such an opening

[5] Of course modifying $\mathcal{F}$ like this is illegal in the ideal process but those restrictions do not apply in a hybrid model.

would contain the wrong string $(sid, cid, P_i, P_j, \ldots)$ and therefore leave $\mathcal{A}$ unable to use it, so we do not have to worry about this case. Of course not seeing an opening leaves $\mathcal{A}$ in just as bad a situation in terms of opening it to something else than the supposed $(sid, cid, P_i, P_j, \ldots)$ contents.

All in all, we have negligible probability of ending up in this third situation. □

7 Acknowledgements

We would like to thank the anonymous referees for helpful and insightful comments.

References

[1] Boaz Barak: *Constant Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model*, pp. 245-255, FOCS 2002.

[2] Ran Canetti: *Universally Composable Security: A New Paradigm for Cryptographic Protocols*; pp. 136-145, FOCS 2001.

[3] Ran Canetti and Marc Fischlin: *Universally Composable Commitments*; pp. 19-40, Proc. of Crypto 2001, Springer Verlag LNCS series 2139.

[4] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky and Amit Sahai: *Universally Composable Two-Party and Multi-Party Secure Computation*; pp. 494-503, STOC 2002.

[5] Richard Crandall and Carl Pomerance: *Prime Numbers: A Computational Perspective*; pp. 34-35, Springer Verlag, 2001.

[6] Giovanni Di Crescenzo, Yuval Ishai and Rafail Ostrovsky: *Non-interactive and Non-malleable Commitment*; pp. 141-150, STOC 1998.

[7] Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky and Adam Smith: *Efficient and Non-interactive Non-malleable Commitments*; pp. 40-59, Proc. of EuroCrypt 2001, Springer Verlag LNCS series 2045.

[8] Ivan Damgård and Jesper Buus Nielsen: *Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor*; pp. 581-596, Proc. of Crypto 2002, Springer Verlag LNCS series 2442.

[9] Danny Dolev, Cynthia Dwork and Moni Naor: *Non-malleable Cryptography*, SIAM J.Computing, vol. 30, pp. 391-437, 2000. (Earlier version STOC 1991.)

[10] Marc Fischlin: *On the Impossibility of Constructing Non-interactive Statistically-Secret Protocols from Any Trapdoor One-Way Function*, CT-RSA 2002, pp. 79-95, 2002.

[11] Marc Fischlin and Roger Fischlin: *Efficient Non-malleable Commitment Schemes*; pp. 413-431, Proc. of Crypto 2000, Springer Verlag LNCS series 1880.

[12] Shafi Goldwasser, Silvio Micali and Charles Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, SIAM J.Computing, vol. 18, pp. 186-208, 1989.

[13] Russell Impagliazzo and Steve Rudich: *Limits on the Provable Consequences of One-Way Permutations*; pp. 44-61, STOC 1989.

[14] Moni Naor: *Bit Commitment Using Pseudorandomness*; pp. 151-158, Journal of Cryptology, vol. 4(2), 1991.

[15] John Rompel: *One-Way Functions are Necessary and Sufficient for Secure Signatures*; pp. 387-394, STOC 1990.