Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor

Ivan Damgård Jesper B. Nielsen*

November 5, 2001

Abstract

Canetti and Fischlin have recently proposed the security notion *universal composability* for commitment schemes and provided two examples. This new notion is very strong. It guarantees that security is maintained even when an unbounded number of copies of the scheme are running concurrently, also it guarantees non-malleability, resilience to selective decommitment, and security against adaptive adversaries. Both of their schemes uses $\Theta(k)$ bits to commit to one bit and can be based on the existence of trapdoor commitments and non-malleable encryption.

We present new universally composable commitment schemes based on the Paillier cryptosystem and the Okamoto-Uchiyama cryptosystem. The schemes are efficient: to commit to k bits, they use a constant number of modular exponentiations and communicates $O(k)$ bits. Further more the scheme can be instantiated in either perfectly hiding or perfectly binding versions. These are the first schemes to show that constant expansion factor, perfect hiding, and perfect binding can be obtained for universally composable commitments.

We also show how the schemes can be applied to do efficient zero-knowledge proofs of knowledge that are universally composable.

* `{ivan,buus}@brics.dk`.

Contents

1 Introduction **3**

2 An Intuitive Explanation of some Main Ideas **4**

3 Mixed Commitments **5**
3.1 Σ-protocols . . . 6
3.2 Proof of Relation between Mixed Commitments . . . 7
3.3 Security under Lunchtime Opening . . . 8

4 Universally Composable Commitments **11**
4.1 The General Framework . . . 11
4.2 The Commitment Functionality . . . 13
4.3 The Common Reference String Model . . . 14

5 UCC with Constant Expansion Factor **14**
5.1 The Commitment Scheme . . . 14
5.2 The Simulator . . . 15
5.3 Analysis . . . 17
5.4 Perfect Hiding and Perfect Binding . . . 19

6 A Special Mixed Commitment Scheme based on the p-Subgroup Assumption **20**
6.1 Key Space, Message Space, and Committing . . . 20
6.2 Equivocability . . . 21
6.3 Extraction . . . 21
6.4 The Transformation . . . 21
6.5 Proof of Commitment to 0 (Base Scheme) . . . 21
6.6 Proof of Commitment to 1 (Base Scheme) . . . 22
6.7 Monotone Logical Combination of Non-Erasure Σ-Protocols . . . 22
6.8 Proof of Binary Boolean Relations . . . 23

7 A Special Mixed Commitment Scheme based on the DCRA **23**
7.1 Key Space, Message Space, and Committing . . . 24
7.2 Equivocability . . . 24
7.3 Extraction . . . 24
7.4 The Transformation . . . 24
7.5 Proof of Multiplicative Relation . . . 24
7.6 Proof of Identity and Proof of Additive Relation . . . 28
7.7 Proof of Binary Boolean Relations . . . 28

8 Efficient Universally Composable Zero-Knowledge Proofs **28**
8.1 Exploiting the Multi-Bit Commitment Property . . . 29
8.2 Exploiting Efficient Proofs of Relations . . . 31

1 Introduction

The notion of commitment is one of the most fundamental primitives in both theory and practice of modern cryptography. In a commitment scheme, a *committer* chooses an element m from some finite set M, and releases some information about m through a commit protocol to a *receiver*. Later, the committer may release more information to the receiver to open his commitment, so that the receiver learns m. Loosely speaking, the basic properties we want are first that the commitment scheme is *hiding:* a cheating receiver cannot learn m from the commitment protocol, and second that it is *binding:* a cheating committer cannot change his mind about m, the verifier can check in the opening that the value opened was what the committer had in mind originally. Each of the two properties can be satisfied unconditionally or relative to a complexity assumption.

A very large number of commitment schemes are known based on various notions of security and various complexity assumptions. Although commitment schemes can be implemented as a game between more than two players, we will concentrate here on the two-player case with standard digital communication. This immediately implies that we cannot build schemes where both the binding and the hiding properties are satisfied unconditionally.

In [CF01] Canetti and Fischlin proposed a new security measure for commitment schemes called universally composable commitments. This is a very strong notion: it guarantees that security is maintained even when an unbounded number of copies of the scheme are running concurrently in an asynchronous way. It also guarantees non-malleability and resilience to selective decommitment, and finally it maintains security even if an adversary can decide adaptively to corrupt some of the players and make them cheat. The new security notion is based on the framework for universally composable security in [Can01]. In this framework one specifies desired functionalities by specifying an idealized version of them. An idealized commitment scheme is modeled by assuming a trusted party to which both the committer and the receiver have a secure channel. To commit to m, the committer simply sends m to the trusted party who notifies the receiver that a commitment has been made. To open, the committer asks the trusted party to reveal m to the receiver. Security of a commitment scheme now means that the view of an adversary attacking the scheme can be simulated given access to just the idealized functionality.

It is clearly important for practical applications to have solutions where only the two main players need to be active. However, in [CF01] it is shown that universal composability is so strong a notion that no universally composable commitment scheme for only two players exist. However, if one assumes that a common reference string with a prescribed distribution is available to the players, then two-player solutions do exist and two examples are given in [CF01]. Note that common reference strings are often available in practice, for instance if a public key infrastructure is given.

The commitment scheme(s) from [CF01] uses $\Omega(k)$ bits to commit to one bit, where k is a security parameter, and it guarantees only computational hiding and binding. In fact, as detailed later, one might even get the impression from the construction that perfect hiding, respectively binding cannot be achieved. Here, by perfect, we mean that an unbounded receiver gets zero information about m, respectively an unbounded committer can change his mind about m with probability zero.

Our contribution is a new construction of universally composable commitment schemes, which uses $O(k)$ bits of communication to commit to k bits. The scheme can be set up such that it is perfectly binding, or perfectly hiding, without loosing efficiency[1]. The

[1] [CF01] also contains a scheme which is statistically binding and computationally hiding, the scheme

construction is based on a new primitive which we call a mixed commitment scheme. We give two efficient implementations of mixed commitments, one based on the Paillier cryptosystem and one based on the Okamoto-Uchiyama cryptosystem. Our commitment protocol has three moves, but the two first messages can be computed independently of the message committed to and thus the latency of a commitment is still one round as in [CF01].

As a final contribution we show that if a mixed commitment scheme comes with protocols in a standard 3-move form for proving in zero-knowledge relations among committed values, the resulting UCC commitment scheme inherits these protocols, such that usage of these is also universally composable. For our concrete example schemes, this results in efficient protocols for proving binary Boolean relations among committed values and also (for the version based on Paillier encryption) additive and multiplicative relations modulo N among committed values. We discuss how this can be used to construct efficient universally composable zero-knowledge proofs of knowledge for NP, improving the complexity of a corresponding protocol from [CF01].

2 An Intuitive Explanation of some Main Ideas

In the simplest type of commitment scheme, both committing and opening are non-interactive, so that committing just consists of running an algorithm commit_K, keyed by a public key K, taking as input the message m to be committed to and a uniformly random string r. The committer computes $c \leftarrow \text{commit}_K(m, r)$, and sends c to the receiver. To open, the committer sends m and r to the receiver, who checks that $c = \text{commit}_K(m, r)$. For this type of scheme, hiding means that given just c the receiver does not learn m and binding means that the committer cannot change his mind by computing m', r', where $c = \text{commit}(m', r')$ and $m' \neq m$.

In a *trapdoor scheme* however, to each public key K a piece of trapdoor information t_K is associated which, if known, allows the committer to change his mind. Note that the existence of such trapdoor information implies that the scheme can never be unconditionally binding. Most trapdoor schemes even have the property that from t_K, one can compute commitments that can be opened in any way desired. Such trapdoor schemes are called *equivocable*.

One may also construct schemes where a different type of trapdoor information d_K exists, such that given d_K, one can efficiently compute m from $\text{commit}(m, r)$. Such schemes are called *extractable* and clearly cannot be unconditionally hiding.

As mentioned, the scheme in [CF01] guarantees only computational binding and computational hiding. Actually this is important to the construction: recall that to prove security, we must simulate an adversary's view of the real scheme with access to the idealized model only. Now, if the committer is corrupted by the adversary and sends a commitment c, the simulator must find out which message was committed to, and send it to the trusted party in the ideal model. The universally composable framework makes very strict demands to the simulation implying that rewinding techniques cannot be used for extracting the message. A solution is to use an extractable scheme, have the public key K in the reference string, and set things up such that the simulator knows the trapdoor d_k. A similar consideration leads to the conclusion that if instead the receiver is corrupt, the scheme must be equivocable with trapdoor known to the simulator, because the simulator must generate a commitment on behalf of the honest committer before finding out from

however requires a new setup of the common reference string per commitment and is thus mostly interesting because it demonstrates that statistically binding can be obtained at all.

the trusted party which value was actually committed to. So to build universally composable commitments it seems we must have a scheme that is simultaneously extractable *and* equivocable — although such a scheme can of course only be computationally secure. This is precisely what Canetti's and Fischlin's ingenious construction provides.

In this paper, we propose a different technique for universally composable commitments based on what we call a mixed commitment scheme. A mixed commitment scheme is basically a commitment scheme which on some of the keys is perfectly hiding and equivocable, we call these keys the E-keys, and on some of the keys is perfectly binding and extractable, we call these keys the X-keys. Clearly, no key can be both an X- and an E-key, so if we were to put the entire key in the common reference string, either extractability or equivocability would fail and the simulation could not work. We remedy this by putting only a part of the key, the so-called system key, in the reference string. The rest of the key is set up once per commitment using a two-move protocol. This allows the simulator to force the key used for each commitment to be an E-key or an X-key depending on whether equivocability or extractability is needed. In other words, our observation is that successful simulation does not really require a scheme that is globally extractable and simulatable at the same time, it is enough if the simulator can decide between extractability and equivocability on a per commitment basis.

Our basic construction is neither perfectly binding nor perfectly hiding because the set-up of keys is randomized and is not guaranteed to lead to any particular type of key. However, one may add to the reference string an extra key that is guaranteed to be either an X- or an E-key. Using this in combination with the basic scheme, one can obtain either perfect hiding or perfect binding.

3 Mixed Commitments

We now give a more formal description of mixed commitment schemes. The most important difference to the intuitive discussion above is that the system key N comes with a trapdoor t_N that allows efficient extraction for all X-keys. The E-keys, however, each come with their own trapdoor for equivocability.

Definition 1 *By a* mixed commitment scheme *we mean a commitment scheme* commit_K *with some global system key N, which determines the message space $\mathcal{M}_N$ and the key space $\mathcal{K}_N$ of the commitments. The key space contains two sets, the E-keys and the X-keys, for which the following holds:*

Key generation *One can efficiently generate a system key N along with the so-called X-trapdoor t_N. One can, given the system key N, efficiently generate random commitment keys and random X-keys. Given the system key, one can efficiently generate an E-key K along with the so-called E-trapdoor t_K.*

Key indistinguishability *Random E-keys and random X-keys are both computationally indistinguishable from random keys as long as the X-trapdoor is not known.*

Equivocability *Given E-key K and E-trapdoor t_K one can generate fake commitments c, distributed exactly as real commitments, which can later be open arbitrarily, i.e. given a message m one can compute uniformly random r for which $c = \mathrm{commit}_K(m, r)$.*

Extraction *Given a commitment $c = \mathrm{commit}_K(m, r)$, where K is an X-key, one can given the X-trapdoor t_N efficiently compute m, where m is uniquely determined by the perfect binding.*

Note that the indistinguishability of random E-keys, random X-keys, and random keys implies that as long as the X-trapdoor is not known the scheme is computationally hiding for all keys and as long as the neither the X-trapdoor nor the E-trapdoor is known the scheme is computationally binding for all keys.

For the construction in the next section we will need a few special requirements on the mixed commitment scheme.

First of all we will assume that the message space $\mathcal{M}_N$ and the key space $\mathcal{K}_N$ are finite groups in which we can compute efficiently. We will denote the group operation by $+$. There are no special requirements on the group structure; If e.g. the key space is the set of all bit-strings of some fixed length l, the group operation could be the xor operation on strings or addition modulo 2^l. Second we need that the number of E-keys over the total number of keys is negligible and that the number of X-keys over the total number of keys is negligible close to 1. Note that this leaves only a negligible fraction which is neither X-keys nor E-keys. We call a mixed commitment scheme with these properties a special mixed commitment scheme.

The last requirement is that the scheme is on a particular form. We ensure this be a transformation. The keys for the transformed scheme will be of the form (K_1, K_2). We let the E-keys be the pairs of E-keys and let the X-keys be the pairs of X-keys. Note that this leaves a negligible fraction of the keys which is neither E-keys nor X-keys. The message space will be the same. Given a message m we commit as $(\text{commit}_{K_1}(\overline{m}_1), \text{commit}_{K_2}(\overline{m}_2))$, where $\overline{m}_1$ and $\overline{m}_2$ are uniformly random values for which $m = \overline{m}_1 + \overline{m}_2$. Note that if both keys are X-keys, then $\overline{m}_1$ and $\overline{m}_2$ and thus m can be computed by extraction. It is trivial to check that all other properties of a special mixed commitment scheme are also maintained under this transformation.

3.1 Σ-protocols

For the mixed commitment schemes we exhibit later, there are efficient protocols for proving in zero-knowledge relations among committed values. Depending on the mixed commitment scheme used, a number of relations between committed values could be considered, e.g equality, additive, or multiplicative relations between committed values. As we shall see, it is possible to have the derived universally composable commitment schemes inherit these protocols while maintaining universal composability. In order for this to work, we need the protocols to have a special form:

A non-erasure Σ-protocol for relation R is a protocol for two parties, called the prover P and the verifier V. The prover gets as input $(x, w) \in R$, the verifier gets as input x, and the goal is for the prover to convince the verifier that he knows w such that $(x, w) \in R$, without revealing information about w. We require that it is done using a protocol of the following form. The prover first computes a message $a \leftarrow A(x, w, r_a)$, where r_a is a uniformly random string, and sends a to V. Then V returns a random challenge e of length l. The prover then computes a responds to the challenge $z \leftarrow Z(x, w, r_a, e)$, and sends z to the verifier. The verifier then runs a program B on (x, a, e, z) which outputs $b \in \{0, 1\}$ indicating where to believe that the prover knows a valid witness w or not.

Besides the protocol being of this special three-move form we furthermore require that the following requirements hold, in order for the protocol to be called a non-erasure Σ-protocol for R.

Completeness If $(x, w) \in R$, then the verifier always accepts $(b = 1)$.

Special Honest Verifier Zero-Knowledge There exists a PPT algorithm, the honest verifier simulator hvs, which given instance x (where there exists w such that

$(x,w) \in R$) and any challenge $e \in \{0,1\}^l$ generates $(a,z) \leftarrow \mathrm{hvs}(x,e,r)$, where r is a uniformly random string, such that (x,a,e,z) is distributed identically to a successful conversation, where e occurs as challenge.

State Construction Given (x,w,a,e,z,r), where $(a,z) = \mathrm{hvs}(x,e,r)$ and $(x,w) \in R$ it should be possible to compute uniformly random r_a for which $a = A(x,w,r_a)$ and $z = Z(x,w,r_a,e)$.

Special Soundness There exists a PPT algorithm extract, which given x, (a,e,z), and (a,e',z'), where $e \neq e'$, $B(x,a,e,z) = 1$, and $B(x,a,e',z') = 1$, outputs $w \leftarrow \mathrm{extract}(x,a,e,z,e',z')$ such that $(x,w) \in R$.

In [Dam00] it is shown how to use Σ-protocols in a concurrent setting. This is done by letting the first message be a commitment to a and then letting the third message be (a,r,z), where (a,r) is an opening of the commitment and z is computed as usual. If the commitment scheme used is a trapdoor commitment scheme this will allow for simulation using the honest verifier simulator. In an adaptive non-erasure setting, where an adversary can corrupt parties during the execution, it is also necessary with the State Construction property as the adversary is entitled to see the internal state of a corrupted party. In the following we call r_a the internal state of the Σ-protocol.

3.2 Proof of Relation between Mixed Commitments

In order to use non-erasure Σ-protocols in our context, it is convenient to specialize the above definition somewhat. We therefore now review the notion of a Σ-protocol in the context of mixed commitment schemes.

Let $\mathcal{M}_N$ be the message space for system key N and let $R^M \subset \mathcal{M}_N^a$ be a a-ary relation over M. We denote a commitment by (K_1,c_1,K_2,c_2), where $K_1,K_2 \in \mathcal{K}_N$ and c_1 and c_2 are commitments in the base scheme under K_1 respectively K_2. From a relation on $\mathcal{M}_N^a$ we can define a binary relation on commitments, where

$$(((K_1,c_1,K_2,c_2),\ldots,(K_{2a-1},c_{2a-1},K_{2a},d_{2a})),(m_1,r_1,\ldots,m_{2a},r_{2a})) \in R$$

iff

$$\left(\bigwedge_{i=1}^{2a} c_i = \mathrm{commit}_{K_i}(m_i,r_i)\right) \wedge (m_1+m_2,\ldots,m_{2a-1}+m_{2a}) \in R^M \ .$$

The instance is

$$x = (K_1,c_1,K_2,c_2,\ldots,K_{2a},c_{2a})$$

and the witness is

$$w = (m_1,r_1,m_2,r_2,\ldots,m_{2a},r_{2a}) \ .$$

Because of the particular context in which we will be using the proofs, it is enough that the special honest verifier zero-knowledge and the state construction holds for E-keys and fake commitments, and that the special soundness holds for X-keys. More precisely, we assume that

1. In the honest verifier simulator and the state construction

   (a) All the keys $K_1,K_2,\ldots,K_{2a}$ are E-keys and the E-trapdoors of the keys $K_b,K_{2+b},\ldots,K_{2(a-1)+b}$ are known, where b is either 1 and 2 and is not known before the simulation.

(b) The commitments $c_b, c_{2+b}, \ldots, c_{2(a-1)+b}$ are fake commitments (under the above keys) and the random bits used to construct them are known and the commitments $c_{2-b}, c_{4-b}, \ldots, c_{2a-b}$ are commitments to random values and their openings are known.

(c) The witnesses given in the state construction are consistent with the above information. One can think of the input to the state construction as a real opening of the fake commitments $c_b, c_{2+b}, \ldots, c_{2(a-1)+b}$, where for the given opening $(m_1 + m_2, \ldots, m_{2a-1} + m_{2a}) \in R^M$. The job of the state construction is then to come up with an internal state consistent with these openings.

2. In the special soundness, given two accepting conversations one must compute either a valid witness or a proof that one of the involved keys is not an X-key. The proof must point to the key which is not an X-key.

3.3 Security under Lunchtime Opening

In the following constructions, we will need to use a mixed commitment scheme in a coin-flip protocol for generating random keys, and also for committing to the first message in Σ-protocols. In order for this to work, we need that the scheme satisfies the following: an adversary who sees a number of fake commitments under E-keys and is allowed adaptively to specify how they should be opened, is nevertheless unable to produce such an arbitrary opening himself. It is advantageous to prove this as a lemma at the current abstraction level.

So let A be a PPT algorithm and consider the following game, which we call the lunchtime opening game. First we generate a system key N and hand N to A. Let E denote a subset of $\mathcal{M}_N$ for which $\frac{|E|}{|\mathcal{M}_N|}$ is negligible. Then during the game A can issue the following types of requests.

Generate Key If A requests a key generation we generate a random E-key $K = (K_1, K_2)$ along with the E-trapdoor t_{K_b} of either K_1 or K_2 and hand K to A. The same b is used for all keys.

Generate Commitment If A requests a commitment generation for a key $K = (K_1, K_2)$ earlier generated in a Generate Key request, we generate a random fake commitment $c = (c_1, c_2)$ under $K = (K_1, K_2)$ using t_{K_b} and hand c to A. The fake commitment is generated by committing honestly to a random message under K_{2-b}, $c_{2-b} = \text{commit}_{K_{2-b}}(m_{2-b}, r_{2-b})$, and faking under K_b.

Open c If A requests to open a commitment c to message m, where c was generated in a Generate Commitment request and has not earlier been requested opened, then we let $m_b = m - m_{2-b}$ and generate uniformly random bits r_b for which $c_b = \text{commit}_K(m_b, r_b)$ and hand (m_1, m_2, r_1, r_2) to A.

Prove Relation $R((K_1, c_1), \ldots, (K_a, c_a))$ **using** K If A requests receiving a proof of $R((K_1, c_1), \ldots, (K_a, c_a))$ using key $K = (K_1, K_2)$, where K was generated due to a Generate Key request and $c_1, \ldots, c_a$ was generated due to Generate Commitment requests, then using a non-erasure Σ-protocol for the relation we do as follows: Generate a fake commitment c using t_{K_b} and send it to A. Receive a challenge from A and using the honest verifier simulator compute (a, z) and using t_{K_b} compute uniformly random r for which $c = \text{commit}_K(a, r)$. Then send (a, r, z) to A.

It must be possible to open any unopened commitment among $(c_1, \ldots, c_a)$ in a way consistent with the relation R and A must never request opened any unopened commitment in a way inconsistent with the relation R.

If A later has requested all of $c_1, \ldots, c_a$ opened, then by the above restriction the openings are a witness for the relation and using the state construction property we can compute the internal state r_a of the Σ-protocol and hand it to A.[2]

Test on c using K If A requests a test on commitment c and key K, where c can either be a commitment generated for K due to a Generate Commitment request or by A alone, we hand A as challenge a uniformly random element m_1 from $\mathcal{M}_N$. Then A returns m_2, r_2. If $c = \text{commit}_K(m_2, r_2)$ and $m_1 + m_2 \in E$, then A scores one point.

Test on $R((K_1, c_1), \ldots, (K_a, c_a))$ using K On a proof of relation test A proves the relation as in the proof of relation test (with the roles changed). We require that K was generated due to a Generate Key request and that each of the K_i was either generated due to a Generate Key request or is an X-key.

If A succeeds in the proof and can ever come up with openings $c_j = \text{commit}_{K_j}(m_j, r_j)$ of the E-commitments such that $R(m_1, \ldots, m_a)$ does not hold, where the value m_i for the X-commitments are defined by extraction, then he scores one point.

We allow the interactions between A and the game to be scheduled arbitrarily by A, meaning that e.g. two proofs of relation can be run concurrently. However, we enforce a two phase structure on the game by requiring that after the first test request is issued by A, only test requests can be issued in the following. Further more, after the first test request is issued, all proof of relation requests that are not ended yet is terminated, meaning that if A returns the challenge after the first test request, the challenge will not be answered. Note that this means that after the first test request, the game does not need the trapdoors t_{K_b} anymore. This is essential in proving the following lemma.

Lemma 1 *Any special mixed commitment scheme has the property that the expected score of any PPT algorithm A in the lunchtime opening game against the scheme is negligible.*

Proof: Assume for the sake of contradiction that there exists some PPT algorithm A which has an expected score which is significant[3]. Now consider the following experiment. We run the game with A as specified above with two modifications.

First of all, each time A sends m_2, r_2, where $c = \text{commit}_K(m_2, r_2)$ in a Test on c using K, save the state of A and rewind it to the point, where m_1 was send. Then repeat the following until A returns m_2', r_2', where $c = \text{commit}_K(m_2', r_2')$: Generate a new random m_1' and give it to A. Then run A until either A returns m_2', r_2' or A stops. When the challenge has been answered correctly again, continue the game at the saved state.

Further more, each time A answers a challenge correctly in a proof of relation test save the state of A and rewind to the point where that challenge was send. Then repeat the following until a challenge for that particular proof of relation test is answered correctly again: Generate a new random challenge and give it to A and run A until either the challenge is answered correctly again or A stops. When the challenge has been answered correctly again continue the game at the saved state.

[2] Note that the honest verifier simulator and the state construction were used in a context meeting the relaxations we put on the protocols in the previous section.

[3] We are using significant to denote 'not negligible'.

We will argue that the expected running time of this experiment is polynomial in k. The experiment is of the form that each time certain challenges are answered correctly by A we replay until a challenge is answered correctly again. It is enough to prove that the expected running time of each of these replays is polynomial. For this purpose let E denote the event that challenge l is answered correctly and let $\Pr[E|s]$ denote the probability that challenge l is answered correctly given that the experiment is in state s at the time the challenge is given. Let $\Pr[s|E]$ denote the probability that the experiment was in state s at the time challenge l was given conditioned on the event that the challenge was answered correctly. Let $T(s)$ be the expected running time of the game given that it is in state s at the time challenge l is given, let $T_1(s)$ denote the expected running time of the game from challenge l is given to the end of the game given that the challenge is not answered correctly and let $T_2(s)$ denote the running time from challenge l is given until it is answered correctly (given that it is answered correctly). Then $T(s) \geq (1 - \Pr[E|s])T_1(s) + \Pr[E|s]T_2(s)$ and the time spend in loop l is

$$
\begin{aligned}
&\Pr[E]\sum_s \Pr[s|E]\left(\sum_{i=1}^{\infty}(1-\Pr[E|s])^{i-1}\Pr[E|s]((i-1)T_1(s)+T_2(s))\right)\\
&=\Pr[E]\sum_s \frac{\Pr[s|E]\Pr[E|s]}{(1-\Pr[E|s])}\sum_{i=1}^{\infty}(iT_1(s)+(T_2(s)-T_1(s)))(1-\Pr[E|s])^i\\
&=\Pr[E]\sum_s \Pr[s|E]\left(\frac{T_1(s)}{\Pr[E|s]}+(T_2(s)-T_1(s))\right)\\
&=\Pr[E]\sum_s \frac{\Pr[s|E]}{\Pr[E|s]}((1-\Pr[E|s])T_1(s)+\Pr[E|s]T_2(s))\\
&\leq \sum_s \Pr[s]T(s) = T\ ,
\end{aligned}
$$

which is polynomial.

By the linearity of expectation there must be an opening challenge l where the challenge is answered correctly and $m_1 + m_2 \in E$ with significant probability or there must be a successful proof of relation test for which A returns a contradictory opening with significant probability. We will argue that this means that the experiment generates a double opening with significant probability.

Consider the first case. Let p be a polynomial for which the probability that $m_1 + m_2 \in E$ is larger than $p^{-1}(k)$ for infinitely many k and consider a value of k for which this is the case. Assume that the challenge is answered correctly and $m_1 + m_2 \in E$. Let $\overline{E} = m_2 \oplus E$. Since $\frac{|\overline{E}|}{|M|} = \frac{|E|}{|M|}$ is negligible and the expected number of iterations in the loop is less than $p(k)$ we have that when a challenge m_1' is answered correctly again $\Pr[m_1' \notin \overline{E}] = 1 - \epsilon$, where ϵ is negligible. Further more $\Pr[m_1' \oplus m_2' \in E] \geq p^{-1}(k)$. Thus $\Pr[m_1' \oplus m_2' \in E \wedge m_1' \notin \overline{E}] \geq (1-\epsilon) + p^{-1}(k) - 1 = p^{-1}(k) - \epsilon$. Since $m_1' \oplus m_2' \in E \wedge m_1' \notin \overline{E}$ implies that $m_2 \neq m_2'$ we thus have a significant probability of obtaining a double opening.

In the second case we have that since the proof was successful we ran a loop after the proof until a challenge e' was answered successfully again. We can assume without lose of generality that the two challenges answered in the loop are different. If the commitment to a is opened differently in the two cases we are done. Assume therefore that it is opened to the same value a. This means that we have (a, e, e', z, z'), where (a, e, z) and (a, e', z') are both accepting conversations. This allows us to either extract witnesses or get (K, P), where K is not an X-key and P is a proof that K is not an X-key.[4] If no witness is obtained,

[4] See the relaxation of the extraction in the previous section.

let $b' \in \{1,2\}$ indicate whether K is a left or a right key and give up the experiment and output (b', K, P). If a witness is obtained, the witness contains an opening of each of the commitments to values in the relation. Thus one of the openings of the E-commitments must be different than the contradictory ones provided by A.

Assume that the game is never given up and consider the following experiment. We are given as input a system key N and an E-key K. Assume without loose of generality that we know the index l of a key for which A produces a double opening with a significant probability. We are going to embed K in the l'th key requested by A. To do this we pick a random bit b and let $K_b = K$ and generate K_{1-b} as a random E-key with known E-trapdoor. For all other key we learn both of the trapdoors.

We then do the above experiment with A using (K_0, K_1) as the l'th key. With a significant probability this produces a double opening for the key (K_0, K_1), which contains a double opening for K_0 or K_1. Since all values handed to A are independent of b, including those handed to A during the rewinding (because of the two phase structure of the game), this means that with at least half the probability of generating a double opening for (K_0, K_1) it will produce a double opening according to the key K_b, a contradiction to the computational binding.

This means that the game must be given up with significant probability. Since the values send to A is independent of b the probability that $b = b'$ is $\frac{1}{2}$. Now consider the following experiment. We run the game as usual except that the sub keys K_{1-b} for which the trapdoor is not used is replaced by random X-keys. By the indistinguishability of E-keys and X-keys the game is still given up with significant probability, but as a proof of K not being an X-key is output along with K it must now be the case that the probability that $b' = b$ is 1 when the game is given up. This essentially makes the game a distinguisher of E-keys and X-keys, a contradiction. The reduction is left to the reader. □

In the next sections we will build universally composable commitments from mixed commitments. To the reader who likes to have a concrete example of a mixed commitment scheme in mind during the reading, we recommend reading Section 7.

4 Universally Composable Commitments

4.1 The General Framework

In the framework from [Can01] the security of a protocol is defined in three steps. First the real-life execution of the protocol is defined. Here the protocol π is modeled by n interactive Turing Machines $P_1, \dots, P_n$ called the parties of the protocols. Also present in the execution is an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ modeling the environment in which $\mathcal{A}$ is attacking the protocol. The environment gives inputs to honest parties, receives outputs from honest parties, and can communication with $\mathcal{A}$ at arbitrary points in the execution. Both $\mathcal{A}$ and $\mathcal{Z}$ are PPT interactive Turing Machines. Second an ideal evaluation is defined. In the ideal evaluation an ideal functionality $\mathcal{F}$ is present to which all the parties have a secure communication line. The ideal functionality is an interactive Turing Machine defining the desired input-output behavior of the protocol. Also present is an ideal adversary $\mathcal{S}$, the environment $\mathcal{Z}$, and n so-called dummy parties $\tilde{P}_1, \dots, \tilde{P}_n$ — all PPT interactive Turing Machines. The only job of the dummy parties is to take inputs from the environment and send them to the ideal functionality and take messages from the ideal functionality and output them to the environment. This basically makes the ideal process a trivially secure protocol with the same input-output behavior as the ideal functionality. The security of the protocol is then defined by requiring that the protocol emulates the ideal process.

The framework also defines the hybrid models, where the execution proceeds as in the real-life execution, but where the parties in addition have access to an ideal functionality. An important property of the framework is that these ideal functionalities can securely be replaced with sub-protocols securely realizing the ideal functionality. The real-life model including access to an ideal functionality $\mathcal{F}$ is called the $\mathcal{F}$-hybrid model.

Below we add a few more details. For a more elaborate treatment of the general framework, see [Can01].

The framework as we will be using it models asynchronous authenticated communication over point-to-point channels, erasure free computation, and an active adaptive adversary. In the real-life execution all parties are assumed to share an open point-to-point channel. In the ideal evaluation all parties are assumed to have a secure channel to the ideal functionality. These assumptions are modeled by the way the execution proceeds. The environment $\mathcal{Z}$ is the driver of the execution. It can either provide a honest party, P_i or $\tilde{P}_i$, with an input or send a message to the adversary. If a party is given an input, that party is then activated. The party can then, in the real-life execution, send a message to another party or give an output to the environment. In the ideal evaluation an activated party just copies its input to the ideal functionality and the ideal functionality is then activated, sending messages to the parties and the adversary according to it program. After the party and/or the ideal functionality stops, the environment is activated again. If the adversary, $\mathcal{A}$ or $\mathcal{S}$, is activated it can do several things. It can corrupt a honest party, send a message on behalf of a corrupt party, deliver any message send from one party to another, or communicate with the environment. On corrupting a party the adversary sees the entire communication history of that party including the random bits used in the execution. After the corruption the adversary sends and receives messages on behalf of the corrupted party. The adversary controls the scheduling of the message delivery. In the real-life execution the adversary $\mathcal{A}$ can see the contents of all message and may decide which messages should be delivered and when — it can however not change or add messages to a channel. In the ideal evaluation the adversary $\mathcal{S}$ cannot see the contents of the messages as the channels are assumed to be secure. It can only see that a message has been send and can then decide when the message should be delivered, if ever. If the adversary delivers a message to some party, then this party is activated and the environment resumes control when the party stops. At the beginning of the protocol all parties, the adversary, and the environment is given as input the security parameter k and random bits. Furthermore the environment is given an auxiliary input z. At some point the environment stops activating parties and outputs some bit. This bit is taken to be the output of the execution. We use $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ to denote the random variable describing the real-life execution and use $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)$ to denote the random variable describing the ideal evaluation.

We are now ready to state the definition of securely realizing an ideal functionality. For this purpose let $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble $\{\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$ and let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)$ denote the distribution ensemble $\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$. We recall the definition of computationally indistinguishable distribution ensembles over $\{0,1\}$.

Definition 2 (indistinquishable ensembles) *We say distribution ensembles $X = \{X(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$ and $Y = \{Y(k,z)\}_{k\in\boldsymbol{N},z\in\{0,1\}^*}$ over $\{0,1\}$ are indistinguishable (written $X \stackrel{c}{\approx} Y$) if for any $c \in \boldsymbol{N}$ there exists $k_0 \in \boldsymbol{N}$ such that $|\Pr[X(k,z) = 1] - \Pr[Y(k,z) = 1]| < k^{-c}$ for all $k > k_0$ and all z.*

Definition 3 ([Can01]) *We say that* π securely realizes $\mathcal{F}$ *if for all real-life adversaries* $\mathcal{A}$ *there exists an ideal-evaluation adversary* $\mathcal{S}$ *such that for all environments* $\mathcal{Z}$ *we have that* $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \stackrel{c}{\approx} \mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$.

An important fact about the above security notion is that it is maintained even if an unbounded number of copies of the protocol (and other protocols) are carried out concurrently — see [Can01] for a formal statement and proof. In proving the composition theorem it is used essentially that the environment and the adversary can communicate at any point in an execution. The price for this strong security notion, which is called universally composability in [Can01], is that rewinding cannot be used in the simulation.

4.2 The Commitment Functionality

We now specify the task that we want to implement as an ideal functionality. We look at a slightly different version of the commitment functionality than the one in [CF01]. The functionality in [CF01] is only for committing to one bit. Here we generalize. The domain of our commitments will be the domain of the special mixed commitment used in the implementation. Therefore the ideal functionality must specify the domain by giving a system key N. An important point related to this is that the X-trapdoor of N is revealed to the adversary in the ideal evaluation. This is to model the fact that the job of the commitment functionality is to hide the contents the commitments, not to hide the X-trapdoor of N; the value N so to say only has the job of specifying the domain of the commitment scheme. An implementation is therefore entitled to reveal the X-trapdoor of the N used to specify the domain of the commitment scheme — as long as commitments hides the values committed to. That the implementation which we are going to give actually keeps the X-trapdoor of N hidden relies only on the fact that this X-trapdoor is actually a trapdoor of the computational assumption on which the security of the implementation is based. The ideal functionality for homomorphic commitments is named $\mathcal{F}_{\mathrm{HCOM}}$ and is as follows.

0. Generate a uniformly random system key N along with the X-trapdoor t_N. Send N to all parties and send (N, t_N) to the ideal adversary $\mathcal{S}$.
1. Upon receiving $(\text{commit}, sid, cid, P_i, P_j, m)$ from $\tilde{P}_i$, where m is in the domain of system key N, record (cid, P_i, P_j, m) and send the message $(\texttt{receipt}, sid, cid, P_i, P_j)$ to $\tilde{P}_j$ and $\mathcal{S}$. Ignore subsequent $(\text{commit}, sid, cid, \ldots)$ messages.
2. Upon receiving the message $(\texttt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ from $\tilde{P}_i$, where $(cid_1, P_i, P_j, m_1), \ldots, (cid_a, P_i, P_j, m_a)$ has been recorded, R is an a-ary relation, and $(m_1, m_2, \ldots, m_a) \in R$, send the message $(\texttt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ to $\tilde{P}_j$ and $\mathcal{S}$.
3. Upon receiving a message $(\texttt{open}, sid, cid, P_i, P_j)$ from $\tilde{P}_i$, where (cid, P_i, P_j, m) has been recorded, send the message $(\texttt{open}, sid, cid, P_i, P_j, m)$ to $\tilde{P}_j$ and $\mathcal{S}$.

It should be noted that a version of the functionality where N and t_N are not specified by the ideal functionality could be used. We could then let the domain of the commitments be a domain contained in the domain of all the system keys.

4.3 The Common Reference String Model

As mentioned in the introduction we cannot hope to construct two-party UCC in the plain real-life model. We need a that a common reference string (CRS) with a prescribed distribution is available to the players. In [CF01] the CRS is modeled by the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model, where the ideal functionality $\mathcal{F}_{\mathrm{CRS}}^{D}$, parameterized by distribution ensemble $D = \{D(k)\}_{k \in \boldsymbol{N}}$, proceeds as follows.

1. When initialized, choose a value d from the distribution $D(k)$.
2. When activated on input $(\mathtt{value}, sid)$, send d back to the activating party and the adversary.

This functionality is a slightly modified version of the one from [CF01]. The difference is that we send d to the adversary. This only makes a difference if all parties are honest, but in that case it makes an important difference. The implication of this formulation is that a protocol implementing the $\mathcal{F}_{\mathrm{CRS}}$ functionality (e.g. by a multi-party protocol) does not have to guarantee privacy.

5 UCC with Constant Expansion Factor

We now describe how to construct universally composable commitments from special mixed commitments.

5.1 The Commitment Scheme

Given a special mixed commitment scheme $\mathtt{com}$ we construct the following protocol $\mathrm{UCC}_{\mathtt{com}}$.

The CRS The CRS is $(N, \overline{K}_1, \dots, \overline{K}_n)$, where N is a random system key and $\overline{K}_1, \dots, \overline{K}_n$ are n random E-keys for the system key N, $\overline{K}_i$ for P_i.

Committing

C.1 On input $(\mathrm{commit}, sid, cid, P_i, P_j, m)$ party P_i generates a random commitment key K_1 for system key N and commits to it as $c_1 = \mathrm{commit}_{\overline{K}_i}(K_1, r_1)$, and sends $(\mathtt{com}_1, sid, cid, c)$ to P_j.[5]

R.1 The party P_j replies with $(\mathtt{com}_2, sid, cid, K_2)$ for random commitment key K_2.

C.2 On a message $(\mathtt{com}_2, sid, cid, K_2)$ from P_j the party P_i computes $K = K_1 + K_2$ and $c_2 = \mathrm{commit}_K(m, r_2)$ for random r_2. Then P_i records $(sid, cid, P_j, K, m, r_2)$ and sends the message $(\mathtt{com}_3, sid, cid, K_1, r_1, c_2)$ to P_j.

R.2 On a message $(\mathtt{com}_3, sid, cid, K_1, r_1, c_2)$ from P_i, where $c_1 = \mathrm{commit}_{\overline{K}_i}(K_1, r_1)$, the party P_j computes $K = K_1 + K_2$, records (sid, cid, P_j, K, c_2), and outputs $(\mathtt{receipt}, sid, cid, P_i, P_j)$.

Opening

C.3 On input $(\mathtt{open}, sid, cid, P_i, P_j)$, P_i sends $(\mathtt{open}, sid, cid, m, r_2)$ to P_j.

[5]We assume that the key space is a subset of the message space. If this is not the case the message space can be extended to a large enough $\mathcal{M}_N^l$ by committing to l values in the original scheme.

R.3 On message $(\texttt{open}, sid, cid, m, r_2)$ from P_i party P_j checks if $c_2 = \text{commit}_K(m, r_2)$. If so party P_j outputs $(\texttt{open}, sid, cid, P_i,\ P_j, m)$.

Proving Relation

C.4 On input $(\texttt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$, where $(sid, cid_1, P_j, K_1, m_1, r_1)$, ..., $(sid, cid_a, P_j, K_a, m_a, r_a)$ are recorded commitments, compute a from the recorded witnesses and compute $c_3 = \text{commit}_{\overline{K}_i}(a, r_3)$ for random r_3 and send $(\texttt{prv}_1, sid,\ cid, R, cid_1, \ldots, cid_a, c_3)$ to P_j.

R.4 Generate a random challenge e and send $(\texttt{prv}_2, sid, cid, P_j, e)$ to P_i.

C.5 Compute the answer z and send $(\texttt{prv}_3, sid, cid, a, r_3, z)$ to P_j.

R.5 Check that $c_3 = \text{commit}_{\overline{K}_i}(a, r_3)$ and that (a, e, z) is an accepting conversation. If so output $(\texttt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots,\ cid_a)$.

5.2 The Simulator

Let $\mathcal{A}$ be an adversary attacking $\text{UCC}_{\texttt{com}}$ in the $\mathcal{F}_{\text{com}}$-hybrid model. We construct a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between $\mathcal{A}$ attacking $\text{UCC}_{\texttt{com}}$ in the CRS-hybrid model and $\mathcal{S}$ attacking the ideal process for $\mathcal{F}_{\text{HCOM}}$.

The CRS The CRS is $(N, \overline{K}_1, \ldots, \overline{K}_n)$, where N is the system key obtained (along with its X-trapdoor) from the ideal functionality $\mathcal{F}_{\text{HCOM}}$ and $\overline{K}_1, \ldots, \overline{K}_n$ are n random E-keys for the system key N. The keys are set up such that $\mathcal{S}$ knows the E-trapdoors of $\overline{K}_1, \ldots, \overline{K}_n$.

Relaying All message from $\mathcal{Z}$ to $\mathcal{S}$ are relayed to $\mathcal{A}$ and all message from $\mathcal{A}$ intended for the environment are relayed to $\mathcal{Z}$.

Committing On input $(\texttt{receipt}, sid, cid, P_i, P_j)$ from the ideal functionality, where P_i is honest, we know that $\mathcal{Z}$ has given $\tilde{P}_i$ input $(\text{commit}, sid, cid, P_i, P_j, m)$ for some $m \in \boldsymbol{Z}_{n^s}$. We have to simulate P_i's behavior on the input $(\text{commit}, sid, cid, P_i, P_j, m)$ without knowing m. We do this as follows.

C.1 Using the E-trapdoor of $\overline{K}_i$ generate a fake commitment c_1 and proceed as in the protocol.

If P_i is corrupted before the next step, then corrupt P_i in the ideal evaluation. Then generate random K_1 and compute r_1 such that $c_1 = \text{commit}_{\overline{K}_i}(K_1, r_1)$.

C.2 On a message $(\texttt{com}_2, sid, cid, K_2)$ from P_j generate a random E-key K for system key N with known E-trapdoor and let $K_1 = K - K_2$. Then compute r_1 such that $c_1 = \text{commit}_{\overline{K}_i}(K_1, r_1)$. Finally generate a fake commitment c_2 using the E-trapdoor of K and send $(\texttt{com}_3, sid, cid, K_1, r_1, c_2)$ to P_j.

If P_i is corrupted by $\mathcal{A}$ after this step and before Step C.3, then corrupt P_i in the ideal evaluation and learn m. Then construct consistent random bits by generating r_2 such that $c_2 = \text{commit}_K(m, r_2)$.

Opening (C.3) On input $(\texttt{open}, sid, cid, P_i, P_j, m)$ from the ideal functionality, where P_i is honest, we know that $\mathcal{Z}$ has given $\tilde{P}_i$ input $(\texttt{open}, sid, cid, P_i, P_j)$. To simulate this construct r_2 as specified in step C.2 and send $(\texttt{open}, sid, cid, m, r_2)$ to P_j.

Receiving a commitment This is how to simulate a honest receiver P_j receiving a commitment.

R.1 Generate K_2 as in the protocol.

R.2 On receiving $(\texttt{com}_3, sid, cid, K_1, r_1, c_2)$ from P_i, where $c_1 = \text{commit}_{\overline{K}_i}(K_1, r_1)$, we have to make $\tilde{P}_j$ output the value $(\texttt{receipt}, sid, cid, P_i, P_j)$ in the ideal evaluation.

(a) If the $\texttt{com}_3$-message was send by $\mathcal{S}$, then it was send because $\mathcal{S}$ received a $(\texttt{receipt}, sid, cid, P_i, P_j)$ message from the ideal functionality and thus $\tilde{P}_j$ has also been send a $(\texttt{receipt}, sid, cid, P_i, P_j)$ message from the ideal functionality, so simply deliver the receipt to $\tilde{P}_j$ which makes it output the receipt.

(b) If the $\texttt{com}_3$-message was send by $\mathcal{A}$ and K is an X-key, then use the X-trapdoor of N to decrypt c_2 and let m' denote the obtained value. If K is an E-key, then let $m' = 0$. Then input $(\text{commit}, sid, cid, P_i, P_j, m')$ to the ideal functionality on behalf of $\tilde{P}_i$. Then deliver the receipt to $\tilde{P}_j$ which makes it output the receipt.

Receiving an opening (R.3)

This is how to simulate a honest receiver P_j receiving an opening. On receiving a message $(\texttt{open}, sid, cid, m', r_2')$ from P_i check whether $c_2 = \text{commit}_K(m', r_2')$. If so we must make $\tilde{P}_j$ output $(\texttt{open}, sid, cid, P_i, P_j, m')$ in the ideal evaluation.

(a) If the $\texttt{open}$-message was send by $\mathcal{S}$, then $(\texttt{open}, sid, cid, P_i, P_j, m')$ has already been send to $\tilde{P}_j$ by the ideal functionality, so simply deliver that message and activate $\tilde{P}_j$ to make it output the message.

(b) If the $\texttt{open}$-message was send by the adversary, then $\tilde{P}_i$ is corrupt and we can input $(\texttt{open}, sid, cid, P_i, P_j)$ on behalf of $\tilde{P}_i$ to the ideal functionality. As response $\mathcal{S}$ receives a message $(\texttt{open}, sid, cid, P_i, P_j, m)$ from the ideal functionality. If $m = m'$, then deliver the $\texttt{open}$-message send to $\tilde{P}_j$ which makes it output the message.

(c) If in Case (b) $m \neq m'$, then give up the simulation.

Proving Relation

C.4 On input $(\texttt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ from the ideal functionality we know that the ideal functionality has recorded $(sid, cid_1, P_i, P_j, m_1), \ldots, (sid, cid_a, P_i, P_j, m_a)$, where $(m_1, \ldots, m_a) \in R$. Since we do not know the messages $m_1, \ldots, m_a$, we send a fake commitment c_3 under key $\overline{K}_i$. If P_i is corrupted before Step C.5, then corrupt $\tilde{P}_i$ in the ideal evaluation and learn the messages $m_1, m_2, \ldots, m_a$ and generate a as in the protocol and compute r_3 such that $c_2 = \text{commit}_{\overline{K}_i}(a, r_3)$.

C.5 Receive a challenge e. Then compute (a, z) using the honest verifier simulator and compute r_3 such that $c_3 = \text{commit}_{\overline{K}_i}(a, r_3)$. Then send (a, z, r_3). If P_i is corrupted later, then corrupt $\tilde{P}_i$ in the ideal evaluation and learn the message $m_1, m_2, \ldots, m_a$ and use the state construction property of the Σ-protocol to construct an internal state consistent with the conversation (a, e, z).

Receiving a Proof of Relation

R.4 Send a random challenge e.

R.5 Do the check as in the protocol. If the proof is accepted, then we need to make $\tilde{P}_j$ output $(\mathtt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$. If the proof was initiated by the simulator on receiving the value $(\mathtt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$, then $\tilde{P}_i$ has received $(\mathtt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ as input from $\mathcal{Z}$ and delivered it to the ideal functionality, which send $(\mathtt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ to $\tilde{P}_j$ and $\mathcal{S}$. We then simply deliver $(\mathtt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ to $\tilde{P}_j$, which makes $\tilde{P}_j$ output the desired value. If the proof was initiated by the adversary, then $\tilde{P}_i$ is corrupt and we can input $(\mathtt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ to the ideal functionality on behalf of it. If the ideal functionality has recorded $(cid_1, P_i, P_j, m_1), \ldots, (cid_a, P_i, P_j, m_a)$, where $(m_1, m_2, \ldots, m_a) \in R$, then it sends $(\mathtt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ to $\tilde{P}_j$ and $\mathcal{S}$. If $\mathcal{S}$ receives this message deliver the one send to $\tilde{P}_j$ as above. If $\mathcal{S}$ does not receive this message, then give up the simulation.

5.3 Analysis

Let $\mathrm{REAL}(\mathcal{A}, \mathcal{Z})$ denote the distribution of an execution of the protocol and let $\mathrm{IDEAL}(\mathcal{S}, \mathcal{Z})$ denote the distribution of the simulation. In order to prove $\mathrm{REAL}(\mathcal{A}, \mathcal{Z}) \stackrel{c}{\approx} \mathrm{IDEAL}(\mathcal{S}, \mathcal{Z})$ we are going to define two hybrid distributions $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ and $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ and prove the following relations: $\mathrm{REAL}(\mathcal{A}, \mathcal{Z}) \stackrel{c}{\approx} \mathrm{HYB}_1(\mathcal{A}, \mathcal{Z}) \stackrel{c}{\approx} \mathrm{HYB}_2(\mathcal{A}, \mathcal{Z}) \stackrel{c}{\approx} \mathrm{IDEAL}(\mathcal{S}, \mathcal{Z})$.

To produce $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ we execute the protocol with adversary $\mathcal{A}$ and environment $\mathcal{Z}$ with one difference: When generating the keys $\overline{K}_1, \ldots, \overline{K}_n$ we learn the E-trapdoors and then generate the values K_1 in step C.2 as in the simulator, i.e. generate them as $K_1 = K - K_2$ for random E-key K and then do a fake opening of c_1 to make a consistent state. Further more we do the proofs of relations as in the simulator.

To produce $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ we execute the simulator, but without the knowledge of the X-trapdoor of N. As we cannot do the decryption in R.2.b, we then always use $m = 0$. Then in R.3.b instead of delivering $(\mathtt{open}, sid, cid, P_i, P_j, 0)$ to $\tilde{P}_j$, we patch the simulation and deliver $(\mathtt{open}, sid, cid, P_i, P_j, m'')$, where m'' is computed as follows. If the $\mathtt{com}_3$-message was send by the simulator, then since P_i is now corrupt it was corrupted after Step C.2 and $\mathcal{S}$ has constructed an opening $c_2 = \mathrm{commit}_K(m, r_2)$ as specified in Step C.2. We then take $m'' = m$. If the $\mathtt{com}_3$-message was send by $\mathcal{A}$, then we take $m'' = m'$, where m' is the value send by $\mathcal{A}$ in the opening. Further more, each time $\mathcal{S}$ is about to give up the simulation in Step R.5, we patch the simulation and deliver $(\mathtt{prove}, sid, cid, P_i, P_j, R, cid_1, \ldots, cid_a)$ anyway.

5.3.1 $\mathrm{REAL}(\mathcal{A}, \mathcal{Z})$ vs. $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$

The difference between $\mathrm{REAL}(\mathcal{A}, \mathcal{Z})$ and $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ is that in $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ the keys K used by honest parties in Step C.2 are random E-keys and not random keys. However random E-keys and random keys are assumed to be indistinguishable and thus this difference should not be detectable by the adversary. To prove this rigorously consider the following distinguisher.

The distinguisher is given as input keys K which are either all random keys or random E-keys. It then generates the keys $\overline{K}_1, \ldots, \overline{K}_n$ with known E-trapdoors. Then using these trapdoors it runs a real execution except that for honest parties the value of K in step C.2 is replaced by a random key from the distinguisher game and K_1 and r_1 are computed as in $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$. Further more, the proofs of relations are done as in the simulator.

Then note that if the keys K from the distinguisher game are all random keys, then all values are distributed exactly as in a real execution. If on the other hand the keys K

are random E-keys, then all values are distributed exactly as in $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$. This proves that $\mathrm{REAL}(\mathcal{A}, \mathcal{Z})$ and $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ are computationally indistinguishable.

5.3.2 $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ vs. $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$

The only difference between $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ and $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ is the following. In $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ the $(\mathtt{open}, sid, cid, P_i, P_j, m)$ value output by honest P_j when receiving correct opening message $(\mathtt{open}, sid, cid, m', r_2')$ in R.3 always has $m = m'$. In $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ this is also the case if the $\mathtt{com}_3$-message was send by $\mathcal{S}$. However, if the $\mathtt{com}_3$-message was send by the adversary, then the m output is the one for which an opening $c_2 = \mathrm{commit}_K(m, r_2)$ was computed in Step C.2 when P_i was corrupted. Note however, that if $m \neq m'$, then $\mathcal{S}$ knows its own opening $c_2 = \mathrm{commit}_K(m, r_2)$ and the new opening $c_2 = \mathrm{commit}_K(m', r_2')$ send by the adversary. Since the X-key of N is not used when producing $\mathrm{HYB}_1(\mathcal{A}, \mathcal{Z})$ and $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ one can easily prove that $m \neq m'$ therefore only occurs with negligible probability using Lemma 1. This reduction is an easier case of the one done in the next section and we leave the details to the reader.

5.3.3 $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ vs. $\mathrm{IDEAL}(\mathcal{S}, \mathcal{Z})$

There are two differences between $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ and $\mathrm{IDEAL}(\mathcal{S}, \mathcal{Z})$. First of all in $\mathrm{IDEAL}(\mathcal{S}, \mathcal{Z})$ the simulator $\mathcal{S}$ might give up in R.5, whereas in $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ the simulation is patched. To prove that this difference is negligible it is enough to prove that the simulator gives up in $\mathrm{IDEAL}(\mathcal{S}, \mathcal{Z})$ only with negligible probability. For the second difference, in $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$, when a honest P_j receives a correct opening message $(\mathtt{open}, sid, cid, m', r_2')$ in R.3 and when the $\mathtt{com}_3$-message was send by $\mathcal{A}$, then the $(\mathtt{open}, sid, cid, P_i, P_j, m)$ value output has $m = m'$, where m' is the value send by $\mathcal{A}$ in the opening. In $\mathrm{IDEAL}(\mathcal{S}, \mathcal{Z})$ the value of m is the one computed in Step R.2.b. To prove that this difference is negligible it is enough to prove that these two values are different only with negligible probability. Observe that to be different it must be the case that the key K used is an E-key as X-keys are perfect binding. All in all, it is therefore enough to prove that when the $\mathtt{com}_3$-message is send by $\mathcal{A}$, then the key K is an X-key except with negligible probability and that the simulation is only given up in R.5 with negligible probability.

For sake of contradiction assume that an E-key is used with probability p_1 and that the probability that no E-key is used, but the simulation is given up in R.5 is p_2, where $p_1 + p_2$ is significant. Then consider the following lunchtime opening game, where we set the set E to be the set of E-keys. We get a random system key N. Then request a key generation and get E-key $\overline{K}$. Then uniformly pick one of the parties P_s and use the key received as the key $\overline{K}_s$ for P_s. For the remaining parties we generate keys and learn the corresponding E-trapdoors. We then generate $\mathrm{HYB}_2(\mathcal{A}, \mathcal{Z})$ with the following differences: First of all we request a trapdoor commitment for $\overline{K}_s$ from the game each time we would have used the E-trapdoor of $\overline{K}_s$ in Step C.1 and we ask for an opening each time such is needed in Step C.1 or C.2. Second, the proofs of relation are done with the game for key $\overline{K}_s$. Finally, the random E-key K generated in C.2 is generated by requesting a key generation in the game and the fake commitment under K in C.2 is generated by requesting a commitment generation, and when a fake opening is needed for K in C.2 or C.3 we ask for this opening in the game. Finally all the proofs of relations (C.4 and C.5) are done by having the adversary doing them with the game.

If P_s is ever corrupted we go to the test phase in the lunchtime opening game and each time a $\mathtt{com}_1$-message is received from the now corrupt P_s we give the commitment c from the message to the lunchtime opening game and receive as a challenge a uniformly random K_2 and send K_2 to P_s, and if c is ever opened, then if the value K_1 opened to is such that $K_1 + K_2$ is an E-key, we score one point in the game. Also, all the proofs of relation done by the adversary (R.4 and R.5) are done with the game as proof of relation tests (this requires that the keys in the test are either generated due to key generation requests or are X-key, which is proven to be the case except with negligible probability in the following paragraph). Each time the simulation is given up in R.5, then if an opening is known for all the commitments and these openings are in the relation, then give them to the game and score one point.

That p_1 is negligible follows directly from the assumption that the score in the lunchtime opening game is negligible. We prove that p_2 is negligible. Assume for the sake of contradiction that p_2 is significant. This means that with a significant probability the simulation is given up *and* an opening is known for all E-commitments in the relation given up on. This is so as for the E-commitments generated by $\mathcal{S}$, $\mathcal{S}$ generated an opening, and a commitment generated by the adversary is never an E-commitment. Further more the value of the openings of the E-commitments together with the values extractable[6] from the X-commitments are exactly those recorded in the ideal simulator, and since the simulation is given up we thus conclude that the openings are not in the relation. Since we only give up if the proof of relation succeeded we can hand the contradictory openings of the E-commitments to the game and scored one point. This implies that the expected score is significant in contradiction to Lemma 1.

We have proven the following theorem.

Theorem 1 *If* $\mathtt{com}$ *is a special mixed commitment scheme, then the protocol* $\mathrm{UCC}_{\mathtt{com}}$ *securely realizes* $\mathcal{F}_{\mathrm{HCOM}}$ *in the* CRS*-hybrid model.*

5.4 Perfect Hiding and Perfect Binding

The scheme described above has neither perfect binding nor perfect hiding. Here we construct a version of the commitment scheme with both perfect hiding and perfect binding. The individual commitments are obviously not simultaneously perfect hiding and perfect binding, but it can be chosen at the time of commitment whether a commitment should be perfect binding or perfect hiding and proofs of relations can include both types of commitments. We sketch the scheme and the proof of its security. The details are left to the reader.

In the extended scheme we add to the CRS a random E-key K_E and a random X-key K_X (both for system key N). Then to do a perfect binding commitment to m the committer will in Step C.2 compute $c_2 = \mathrm{commit}_K(m, r_2)$ as before, but will in addition compute $c_3 = \mathrm{commit}_{K_X}(m, r_3)$. To open the commitment the committer will then have to send both a correct opening (m, r_2) of c_2 and a correct opening (m, r_3) of c_3. This is perfect binding as the X-key commitment is perfect binding.

To do a perfect hiding commitment the committer computes a uniformly random message $\overline{m}$ and commits with $c_2 = \mathrm{commit}_K(\overline{m} + m, r_2)$ and $c_3 = \mathrm{commit}_{K_E}(\overline{m}, r_3)$. To open to m the committer must then send a correct opening (m_2, r_2) of c_2 and a correct opening (m_3, r_3) of c_3 for which $m_2 = m_3 + m$. This is perfect hiding because c_3 hides $\overline{m}$ perfectly and $\overline{m} + m$ thus hides m perfectly.

To do the simulation simply let the simulator make the excusable mistake of letting K_E be a random X-key and letting K_X be a random E-key. This mistake will allow to

[6] Observe that we need not actually extract the values, as the lunchtime opening game only requires to see openings of the E-commitments. This is an important detail as we do not know the X-trapdoor.

simulate and cannot be detected by the assumption that both E-keys and X-keys are indistinguishable from random keys, the switch can be handled between $\mathrm{REAL}(\mathcal{A},\mathcal{Z})$ and $\mathrm{HYB}_1(\mathcal{A},\mathcal{Z})$. In the perfect binding version both K and K_X will then be E-keys when the simulator does a commitment, which allows to fake. When the adversary does a commitment K will (except with negligible probability p_1) be an X-key and the simulator can extract m from $\mathrm{commit}_K(m)$. In the perfect hiding version both K and K_E will (except with negligible probability p_1) be X-keys when the adversary does a commitment, which allows to extract. When the simulator commits, K will be an E-key, which allows to fake an opening by faking $\mathrm{commit}_K(m)$.

In the version with perfect binding the proofs of relation can be used directly for the modified version by doing the proof on the $\mathrm{commit}_K(m)$ values. In the version with perfect hiding there is no general transformation that will carry proofs of relations over to the modified system. If however there is a proof of additive relation, then one can publish $\mathrm{commit}_K(m)$ and prove that the sum of the values committed to by $\mathrm{commit}_K(m)$ and $\mathrm{commit}_{K_E}(\overline{m})$ is committed to by $\mathrm{commit}_K(\overline{m}+m)$, and then use the commitment $\mathrm{commit}_K(m)$ when doing the proofs of relation.

To see that this is secure assume that the adversary succeeds in proving a relation which is not meet by the values input to the ideal functionality by the simulator. We can safely exclude the case where an E-key has been generated by the adversary as it occurs with negligible probability p_1. This means that the simulator knows the opening of all the $\mathrm{commit}_K(m)$ commitments and the opening of the $\mathrm{commit}_{K_E}(\overline{m})$ and $\mathrm{commit}_K(m+\overline{m})$ commitments for the perfect hiding commitments, and that the value of these openings were those used to compute the input to the ideal functionality. This means that either 1) the adversary has proven a relation for the $\mathrm{commit}_K(m)$ commitments and the simulator knows contradictory openings or 2) the adversary has proven an additive relation among $\mathrm{commit}_K(m)$, $\mathrm{commit}_{K_E}(\overline{m})$, and $\mathrm{commit}_K(\overline{m}+m)$ and the simulator knows a contradictory opening. In both cases the simulator scores one point in the lunchtime opening game and both cases must thus occur with negligible probability.

6 A Special Mixed Commitment Scheme based on the p-Subgroup Assumption

In this section we provide our first construction of a special mixed commitment. It is build on the encryption scheme from [OU98].

6.1 Key Space, Message Space, and Committing

Let p and q be random k-bit primes, where $p \neq q$ and $\gcd(p-1,q) = \gcd(q-1,p) = 1$. Let $n = pq$ and let $N = pn$. The system key will be (N, g), where $g = (1+n)^{m_g} {r_g}^p \bmod N$ for random $r_g \in \boldsymbol{Z}_N^*$. The X-trapdoor will be (p, q). The key space will be $\boldsymbol{Z}_N^*$ and the message space will be $\{0,1\}^{k-1}$, both have the necessary group structure. The E-keys will be $P = \{r^p \bmod N | r \in \boldsymbol{Z}_N^*\}$ and the X-keys will be $\boldsymbol{Z}_N^* \setminus P$. The density of both E-keys and X-keys are obviously as required.

To sample an E-key simply let $K = r^N \bmod N$ for random $r \in \boldsymbol{Z}_N^*$. It is clear that P is a subgroup of $\boldsymbol{Z}_N^*$ of size $\phi(n) = (p-1)(q-1)$. Since $r \mapsto r^p \bmod N$ is a homomorphism from $\boldsymbol{Z}_N^*$ to P it is p to 1, and since $\gcd(p-1,q) = \gcd(q-1,q) = 1$, the map $r \mapsto r^{pq} \bmod N$ is an automorphism on P. Therefore the map $r \mapsto r^N = (r^p)^{pq}$ is p to 1 in $\boldsymbol{Z}_N^*$. Thus $r^N \bmod N$ is a random E-key for random $r \in \boldsymbol{Z}_N^*$.

To sample a random X-key let m be a random element from $\boldsymbol{Z}_p^*$, let r be a random element for $\boldsymbol{Z}_N^*$, and let $K = g^m r^N \bmod N$. It is easy to see that $(1+n)^i \equiv 1 + in \pmod N$. Thus $(n+1)$ has order p in $\boldsymbol{Z}_N^*$ and $\boldsymbol{Z}_N^*/P = \{(1+n)^i P | i \in \boldsymbol{Z}_p^*\}$. Thus $g^m r^N = (1+n)^{m_g m}({r_g}^{mp} r^N)$ is a random element from $\boldsymbol{Z}_N^* \setminus P$.

The key spaces are indistinguishable relative to the p-subgroup assumption, that uniformly random elements from P are indistinguishable from uniformly random elements from $\boldsymbol{Z}_N^*$. For details, see [OU98].

We commit as $c = \mathrm{commit}_K(m, r) = K^m r^N \bmod N$.

6.2 Equivocability

Assume that K is an E-key $K = r_K^N \bmod N$. Then $c = (r_K^m r)^N$. Assume that we are given any message $\overline{m} \in \boldsymbol{Z}_{N^s}$ and let $\overline{r} = r_K^{m-\overline{m}} r \bmod N$.

Then $\mathrm{commit}_K(\overline{m}, \overline{r}) \equiv K^{\overline{m}} \overline{r}^N \equiv K^m r^N \pmod N$. Further more it is easy to see that if $c = \mathrm{commit}_K(\overline{m}, \tilde{r})$, then $\tilde{r} = \overline{r}$. This proves that r_K gives equivocability.

6.3 Extraction

Let $K = (n+1)^{m_g m_K}(r_g^{m_K} r_K)^N \bmod N$ be an X-key, i.e. $m_K \in \boldsymbol{Z}_p^*$. Assume that c can be opened to $m \in \{0,1\}^{k-1}$. I.e. there exists $r \in \boldsymbol{Z}_N^*$ such that $c \equiv K^m r^N \equiv (n+1)^{m_g m_K m}(r_g^{m_K m} r_K^m r)^N \pmod N$. Let $\lambda = \mathrm{lcm}(p-1, q-1)$ and let

$$
\begin{aligned}
d &= c^\lambda \bmod N \\
&= (n+1)^{\lambda m_g m_K m}((r_g^{m_K m} r_K^m r)^{p\lambda(p-1,q-1)})^{pq} \bmod N \\
&= 1 + (\lambda m_g m_K m)n \bmod N \ .
\end{aligned}
$$

Since $\lambda m_g m_K \in \boldsymbol{Z}_p^*$ we can efficiently compute $\alpha, \beta \in \boldsymbol{N}$ such that

$$
\begin{aligned}
\alpha(d-1) \bmod N &= (1+\beta p)mn \bmod N \\
&= mn \bmod N \\
&= mn
\end{aligned}
$$

and from this number we can compute m. This proves that (p,q) allows extraction.

6.4 The Transformation

The transformed scheme has keys which are pairs of keys and commits as $(C_1, C_2) = (K_1^{m_1} r_1^N \bmod N, K_2^{m_2} r_2^N \bmod N)$, where r_1, r_2 are random in $\boldsymbol{Z}_N^*$, m_1 is random in $\{0,1\}^{k-1}$, and $m = m_1 \oplus m_2$, where $\oplus$ denotes bit-wise xor.

In the following we are going to provide proofs of relations for the transformed scheme with message space $\{0,1\}$.

6.5 Proof of Commitment to 0 (Base Scheme)

To prove that a commitment C in the base scheme commits to 0 (i.e. $C \in P$) one has to prove knowledge of r such that $C = r^N \bmod N$. Here is a protocol for doing that.

1. The prover is given $r \in \boldsymbol{Z}_N^*$, computes $D = s^N \bmod N$ for random $s \in \boldsymbol{Z}_N^*$, and sends D to the verifier.

2. The verifier sends random $e < 2^{k-1}$.

3. The prover sends $t = sr^e \bmod N$.

4. The verifier checks that $t^N \equiv DC^e \pmod{N}$.

The correctness is obvious. For the special soundness note that if two different challenges e_1, e_2 are answered correctly, we have

$$t_1^N \equiv DC^{e_1} \pmod{N}, \quad t_2^N \equiv DC^{e_2} \pmod{N} ,$$

which allows us to compute $r \in \boldsymbol{Z}_N^*$ such that

$$C = r^N \bmod N .$$

For the special honest verifier zero-knowledge, given a challenge e pick $t \in \boldsymbol{Z}_N^*$ at random and let $D = t^N C^{-e} \bmod N$. When given r such that $C = r^N \bmod N$ in the state construction, let $s = tr^{-1} \bmod N$. Then s is uniform in $\boldsymbol{Z}_N^*$ and $s^N \equiv t^N(r^N)^{-1} \equiv D \pmod{N}$ as desired.

6.6 Proof of Commitment to 1 (Base Scheme)

To prove that a commitment C commits to 1 under the key K (i.e. $C \in KP$) simply prove that $C/K \in P$ using the above proof.

6.7 Monotone Logical Combination of Non-Erasure Σ-Protocols

Using techniques from [CDS94] we can now combine the proofs of committing to 0 and the proof of committing to 1 using monotone logical circuits to obtain proof of relations between the transformed commitments. We review the construction for disjunction here and verify that it preserves the non-erasure property.

Assume that we are given non-erasure Σ-protocols for relations R_0 and R_1 and that we want a non-erasure Σ-protocol for the relation $((x_0, x_1), (b, w)) \in R \Leftrightarrow (x_b, w) \in R_b$.

1. The input of the prover is (x_0, x_1, b, w), where $b \in \{0, 1\}$ and $(x_b, w) \in R_b$. Compute a random challenge $e_{1-b} < 2^{k-1}$ and using the honest verifier simulator compute a_{1-b}, z_{1-b} such that $(x_{1-b}, a_{1-b}, e_{1-b}, z_{1-b})$ is an accepting conversation. Then using (x_b, w) compute the first message a_b in the protocol for R_b and send (a_0, a_1) to the verifier.

2. The verifier sends random $e < 2^{k-1}$.

3. The prover sets $e_b = e - e_{1-b} \bmod 2^{k-1}$ and computes z_b such that (x_b, a_b, e_b, z_b) is an accepting conversation, and sends (e_0, e_1, z_0, z_1) to the verifier.

4. The verifier checks that $e_0 + e_1 = e \bmod 2^{k-1}$ and (x_0, a_0, e_0, z_0) and (x_1, a_1, e_1, z_1) are accepting conversations.

The correctness is obvious. For the special soundness assume that two challenges e, e' are answered correctly, with $e = e_0 + e_1 \bmod 2^{k-1}$ and $e' = e'_0 + e'_1 \bmod 2^{k-1}$. This means that either $e_0 \neq e'_0$ or $e_1 \neq e'_1$, which allows to extract a witness.

For the special honest verifier zero-knowledge assume that we are given challenge e. Then pick e_0 and e_1 as random values for which $e = e_0 + e_1 \bmod 2^{k-1}$ and using the honest verifier simulator compute accepting conversations (x_0, a_0, e_0, z_0) and (x_1, a_1, e_1, z_1). When given (b, w) in the state construction e_{1-b} is uniformly random and $e_b = e - e_{1-b} \bmod 2^{k-1}$ and (a_{1-b}, z_{1-b}) was generated using the honest verifier simulator as required. To construct an internal state consistent with (a_b, z_b) being generated as in the protocol with challenge e_b, use the state construction property of R_b.

6.8 Proof of Binary Boolean Relations

We show how to combine the proof of committing to 0 and the proof of committing to 1 using a monotone logical circuit to obtain proofs of binary Boolean relations by using the conjunctive relation as an example. We are given commitments $(C_1, C_2), (C_3, C_4), (C_5, C_6)$, where

$$\begin{gathered} C_1 = K_1^{m_1} r_1^{N^s} \bmod N^{s+1}, \quad C_2 = K_2^{m_2} r_2^{N^s} \bmod N^{s+1} \\ C_3 = K_3^{m_3} r_3^{N^s} \bmod N^{s+1}, \quad C_4 = K_4^{m_4} r_4^{N^s} \bmod N^{s+1} \\ C_5 = K_5^{m_5} r_5^{N^s} \bmod N^{s+1}, \quad C_6 = K_6^{m_6} r_6^{N^s} \bmod N^{s+1} \\ (m_1 \oplus m_2) \wedge (m_3 \oplus m_4) = (m_5 \oplus m_6) \ . \end{gathered}$$

To prove knowledge, simply prove

$$\begin{gathered} [(m_1 \in P \wedge m_2 \in P \vee m_1 \in K_1 P \wedge m_2 \in K_2 P) \\ \vee \\ (m_3 \in P \wedge m_4 \in P \vee m_3 \in K_3 P \wedge m_4 \in K_4 P] \\ \wedge \\ (m_5 \in P \wedge m_6 \in P \vee m_5 \in K_5 P \wedge m_6 \in K_6 P) \\ \bigvee \\ (m_1 \in P \wedge m_2 \in K_2 P \vee m_1 \in K_1 P \wedge m_2 \in P) \\ \wedge \\ (m_3 \in P \wedge m_4 \in K_4 P \vee m_3 \in K_3 P \wedge m_4 \in P) \\ \wedge \\ (m_5 \in P \wedge m_6 \in K_6 P \vee m_5 \in K_5 P \wedge m_6 \in P) \ . \end{gathered}$$

As a final remark on verifying Boolean relations, note that if (C_1, C_2) is a commitment to m, then $(K_1 C_1^{-1}, C_2)$ is a commitment to $1 - m$, so this implements Boolean negation assuming m is a $0/1$ value. This, together with the monotone logical combinations of proofs we have seen above is clearly enough to verify any relation defined by a binary Boolean function f, i.e., we obtain a proof that committed values m_1, m_2, m_3 are $0/1$ values and satisfy $f(m_1, m_2) = m_3$. Summarizing, we have:

Theorem 2 *Relative to the p-subgroup assumption, the above is a special mixed commitment scheme. When restricted to message space $\{0, 1\}$ the scheme has proofs of relations between committed values for all relations of the form $m_3 = f(m_1, m_2)$, where f is a binary Boolean function.*

7 A Special Mixed Commitment Scheme based on the DCRA

In this section we describe another special mixed commitment scheme. It is based on the decisional composite residuosity assumption (DCRA) introduced in [Pai99]. The scheme has a constant expansion factor and allows efficient proofs of additive and multiplicative relations.

7.1 Key Space, Message Space, and Committing

Let $N = pq$ be an RSA modulus and let $s \geq 0$ be some fixed constant. Consider the map $\psi : \boldsymbol{Z}_{N^s} \times \boldsymbol{Z}_N^* \to \boldsymbol{Z}_{N^{s+1}}^*$ given by $(m, r) \mapsto (N+1)^m r^{N^s} \bmod N^{s+1}$. In [DJN01] it is proven that ψ is an isomorphism, where $\psi(m_1, r_1)^e \psi(m_2, r_2) \bmod N^{s+1} = \psi(em_1 + m_2 \bmod N^s, r_1^e r_2 \bmod N)$, and that it can be efficiently inverted given the factorization of N. Further more it is proven that if the DCRA holds, then elements of the form $\psi(0, r)$ cannot be distinguished from elements of the form $\psi(m, r)$, where r is uniformly random from $\boldsymbol{Z}_N^*$ and m is any fixed element $m \in \boldsymbol{Z}_{N^s}$. We will use this to construct a special mixed commitment scheme.

The system key will be a random RSA-modulus $N = pq$, where p and q are k-bit primes, and some fixed s and the X-trapdoor will be (p, q). The key space will be $\boldsymbol{Z}_{N^{s+1}}^*$ and the message space will be $\boldsymbol{Z}_{N^s}$, both have the necessary group structure. The E-keys will be elements of the form $\psi(0, r)$ and the E-trapdoor will be r. The X-keys will be the elements of the form $\psi(m, r)$, where $m \in \boldsymbol{Z}_{N^s}^*$. The density in the key space of both the E-keys and the X-keys are obviously as required and the keys can be sampled as required. Further more random E-keys, random X-keys, and random keys are computationally indistinguishable by the DCRA as discussed above.

Given a key $K \in \boldsymbol{Z}_{N^{s+1}}^*$ and a message $m \in \boldsymbol{Z}_{N^s}$, let r be a uniformly random element from $\boldsymbol{Z}_N^*$ and commit as $c = \text{commit}_K(m, r) = K^m r^{N^s} \bmod N^{s+1}$.

7.2 Equivocability

Assume that K is an E-key $K = \psi(0, r_K)$. Then $c = \psi(0, r_K)^m \psi(0, r) = \psi(0, r_K^m r \bmod N)$. Assume that we are given any message $\overline{m} \in \boldsymbol{Z}_{N^s}$ and let $\overline{r} = r_K^{m - \overline{m}} r \bmod N$. Then $\text{commit}_K(\overline{m}, \overline{r}) = \psi(0, r_K^{\overline{m}} \overline{r} \bmod N) = \psi(0, r_K^m r \bmod N) = c$, and if $\text{commit}_K(\overline{m}, \tilde{r}) = c$, then $\tilde{r} = \overline{r}$. This proves that r_K gives equivocability.

7.3 Extraction

Assume that K is an X-key $K = \psi(m_K, r_K)$. Assume that c can be opened to m, i.e. there exists r such that $c = \psi(m_K, r_K)^m \psi(0, r) = \psi(m m_K \bmod N^s, r_K^m r \bmod N)$. Given the X-trapdoor (p, q) we can by [DJN01] invert ψ on K and c and compute m_K and $m m_K \bmod N^s$. Since $m_K \in \boldsymbol{Z}_{N^s}^*$ this allows us to compute m. This proves that (p, q) allows extraction.

7.4 The Transformation

The transformed scheme has keys (K_1, K_2), which are pairs of keys, and commits as

$$\text{commit}_{K_1, K_2}(m, (r_1, r_2, m_1)) = (K_1^{m_1} r_1^{N^s} \bmod N^{s+1}, K_2^{m_2} r_2^{N^s} \bmod N^{s+1})$$

where r_1, r_2 are random in $\boldsymbol{Z}_N^*$, m_1 is random in $\boldsymbol{Z}_{N^s}$, and $m_2 = m - m_1 \bmod N^s$.

7.5 Proof of Multiplicative Relation

We are given commitments $(C_1, C_2), (C_3, C_4), (C_5, C_6)$, where

$$\begin{aligned} C_1 &= K_1^{m_1} r_1^{N^s} \bmod N^{s+1}, \quad & C_2 &= K_2^{m_2} r_2^{N^s} \bmod N^{s+1} \\ C_3 &= K_3^{m_3} r_3^{N^s} \bmod N^{s+1}, \quad & C_4 &= K_4^{m_4} r_4^{N^s} \bmod N^{s+1} \\ C_5 &= K_5^{m_5} r_5^{N^s} \bmod N^{s+1}, \quad & C_6 &= K_6^{m_6} r_6^{N^s} \bmod N^{s+1} \end{aligned}$$

$$(m_1 + m_2)(m_3 + m_4) \equiv (m_5 + m_6) \pmod{N^s} .$$

To prove knowledge we send

$$\begin{array}{ll} C_7 = K_5^{m_3} r_7^{N^s} \bmod N^{s+1}, & C_8 = K_6^{m_4} r_8^{N^s} \bmod N^{s+1} \\ \overline{C}_1 = K_1^{\overline{m}_1} \overline{r}_1^{N^s} \bmod N^{s+1}, & \overline{C}_2 = K_2^{\overline{m}_2} \overline{r}_2^{N^s} \bmod N^{s+1} \\ \overline{C}_3 = K_5^{\overline{m}_3} \overline{r}_3^{N^s} \bmod N^{s+1}, & \overline{C}_4 = K_6^{\overline{m}_4} \overline{r}_4^{N^s} \bmod N^{s+1} \\ \overline{C}_7 = K_5^{\overline{m}_3} \overline{r}_7^{N^s} \bmod N^{s+1}, & \overline{C}_8 = K_6^{\overline{m}_4} \overline{r}_8^{N^s} \bmod N^{s+1} \end{array}$$

$$\overline{C}_5 = K_5^{m_3(\overline{m}_1+\overline{m}_2)+\overline{m}} \overline{r}_5^{N^s} \bmod N^{s+1}$$

$$\overline{C}_6 = K_6^{m_4(\overline{m}_1+\overline{m}_2)-\overline{m}} \overline{r}_6^{N^s} \bmod N^{s+1} ,$$

where the values $r_7, r_8, \overline{m}_i, \overline{r}_i, \overline{m}$ are chosen at random in the respective domains. After receiving $e < 2^{k-1}$ we let $l = \frac{m_6+m_5-(m_1+m_2)(m_3+m_4)}{N^s}$ and send the following values (except $\tilde{m}_1, \tilde{m}_2, \tilde{m}_3, \tilde{m}_4, \tilde{m}$):

$$\tilde{m}_1 = em_1 + \overline{m}_1, \quad \tilde{m}_1^{(r)} = \tilde{m}_1 \bmod N^s, \quad \tilde{r}_1 = K_1^{\tilde{m}_1 \operatorname{div} N^s} r_1^e \overline{r}_1 \bmod N$$

$$\tilde{m}_2 = em_2 + \overline{m}_2, \quad \tilde{m}_2^{(r)} = \tilde{m}_2 \bmod N^s, \quad \tilde{r}_2 = K_2^{\tilde{m}_2 \operatorname{div} N^s} r_2^e \overline{r}_2 \bmod N$$

$$\tilde{m}_3 = em_3 + \overline{m}_3, \quad \tilde{m}_3^{(r)} = \tilde{m}_3 \bmod N^s$$

$$\tilde{r}_3 = K_3^{\tilde{m}_3 \operatorname{div} N^s} r_3^e \overline{r}_3 \bmod N, \quad \tilde{r}_7 = K_5^{\tilde{m}_3 \operatorname{div} N^s} r_7^e \overline{r}_7 \bmod N$$

$$\tilde{m}_4 = em_4 + \overline{m}_4, \quad \tilde{m}_4^{(r)} = \tilde{m}_4 \bmod N^s$$

$$\tilde{r}_4 = K_4^{\tilde{m}_4 \operatorname{div} N^s} r_4^e \overline{r}_4 \bmod N, \quad \tilde{r}_8 = K_6^{\tilde{m}_4 \operatorname{div} N^s} r_8^e \overline{r}_8 \bmod N$$

$$\tilde{m} = em_5 + m_3(\overline{m}_1 + \overline{m}_2 - \tilde{m}_1 - \tilde{m}_2) + \overline{m}, \quad \tilde{m}^{(r)} = \tilde{m} \bmod N^s$$

$$\tilde{r}_5 = K_5^{\tilde{m} \operatorname{div} N^s} \overline{r}_5 r_5^e r_7^{-(\tilde{m}_1+\tilde{m}_2)} \bmod N$$

$$\tilde{r}_6 = K_6^{-(\tilde{m} \operatorname{div} N^s)+[el+(m_3+m_4)((\tilde{m}_1 \operatorname{div} N^s)+(\tilde{m}_2 \operatorname{div} N^s))]} \overline{r}_6 r_6^e r_8^{-(\tilde{m}_1+\tilde{m}_2)} \bmod N .$$

The verifier checks that

$$C_1^e \overline{C}_1 \equiv K_1^{\tilde{m}_1^{(r)}} \tilde{r}_1^{N^s}, \quad C_2^e \overline{C}_2 \equiv K_2^{\tilde{m}_2^{(r)}} \tilde{r}_2^{N^s} \pmod{N^{s+1}}$$

$$C_3^e \overline{C}_3 \equiv K_3^{\tilde{m}_3^{(r)}} \tilde{r}_3^{N^s}, \quad C_4^e \overline{C}_4 \equiv K_4^{\tilde{m}_4^{(r)}} \tilde{r}_4^{N^s} \pmod{N^{s+1}}$$

$$C_7^e \overline{C}_7 \equiv K_5^{\tilde{m}_3^{(r)}} \tilde{r}_7^{N^s}, \quad C_8^e \overline{C}_8 \equiv K_6^{\tilde{m}_4^{(r)}} \tilde{r}_8^{N^s} \pmod{N^{s+1}}$$

$$C_5^e \overline{C}_5 \equiv C_7^{\tilde{m}_1^{(r)}+\tilde{m}_2^{(r)}} K_5^{\tilde{m}^{(r)}} \tilde{r}_5^{N^s} \pmod{N^s}$$

$$C_6^e \overline{C}_6 \equiv C_8^{\tilde{m}_1^{(r)}+\tilde{m}_2^{(r)}} K_6^{-\tilde{m}^{(r)}} \tilde{r}_6^{N^s} \pmod{N^s} .$$

7.5.1 Correctness

We check the last equivalence. The other equivalences are checked using simpler computations. To check the equivalence it is enough to prove that

$$C_6^e \overline{C}_6 K_6^{\tilde{m}^{(r)}} \tilde{r}_6^{-N^s} C_8^{-(\tilde{m}_1^{(r)}+\tilde{m}_2^{(r)})} \equiv$$
$$K_6^{em_6} r_6^{eN^s} K_6^{m_4(\overline{m}_1+\overline{m}_2)-\overline{m}} \overline{r}_6^{N^s} K_6^{\tilde{m}^{(r)}} \tilde{r}_6^{-N^s} K_6^{-m_4(\tilde{m}_1^{(r)}+\tilde{m}_2^{(r)})} (r_8^{\tilde{m}_1^{(r)}+\tilde{m}_2^{(r)}})^{-N^s} \equiv$$
$$1 \pmod{N^{s+1}} .$$

By collecting terms in the exponent of K_6, this follows from the following computation

$$\begin{aligned}
& m_4(\overline{m}_1+\overline{m}_2)+em_6-\overline{m}-m_4(\tilde{m}_1^{(r)}+\tilde{m}_2^{(r)}) \\
&\quad + N^s(\tilde{m}\operatorname{div}N^s-[el+(m_3+m_4)((\tilde{m}_1\operatorname{div}N^s) \\
&\quad + (\tilde{m}_2\operatorname{div}N^s))]) + \tilde{m}^{(r)} \\
\equiv & m_4(\overline{m}_1+\overline{m}_2)+em_6-\overline{m}+\tilde{m}-m_4(\tilde{m}_1^{(r)}+\tilde{m}_2^{(r)}) \\
&\quad - N^s[el+(m_3+m_4)((\tilde{m}_1\operatorname{div}N^s)+(\tilde{m}_2\operatorname{div}N^s))] \\
\equiv & m_4(\overline{m}_1+\overline{m}_2)+e(m_6+m_5) \\
&\quad + m_3(\overline{m}_1+\overline{m}_2-\tilde{m}_1-\tilde{m}_2)-m_4(\tilde{m}_1^{(r)}+\tilde{m}_2^{(r)}) \\
&\quad - N^sel-(m_3+m_4)(N^s(\tilde{m}_1\operatorname{div}N^s)+N^s(\tilde{m}_2\operatorname{div}N^s)) \\
\equiv & m_4(\overline{m}_1+\overline{m}_2)+e(m_6+m_5)+m_3(\overline{m}_1+\overline{m}_2) \\
&\quad - N^sel-(m_3+m_4)(\tilde{m}_1+\tilde{m}_2) \\
\equiv & (m_3+m_4)(-em_1+-em_2)+e(m_6+m_5)-N^sel \\
\equiv & e[-(m_3+m_4)(m_1+m_2)+(m_6+m_5)-lN^s]\equiv 0 \pmod{N^s}\ .
\end{aligned}$$

7.5.2 Special Soundness

Assume that two different challenges $\tilde{e},\underline{e}$ are answered correctly. This gives us

$$C_1^{\tilde{e}}\overline{C}_1 \equiv K_1^{\tilde{m}_1}\tilde{r}_1^{N^s}, \qquad C_1^{\underline{e}}\overline{C}_1 \equiv K_1^{\underline{m}_1}\underline{r}_1^{N^s}$$

and thus

$$C_1^{\tilde{e}-\underline{e}} \equiv K_1^{\tilde{m}_1-\underline{m}_1}(\tilde{r}_1\underline{r}_1^{-1})^{N^s}\ .$$

Since $\tilde{e}-\underline{e}<p,q$ we can using the extended Euclid's algorithm compute α and β s.t.

$$1=\alpha N^s+\beta(\tilde{e}-\underline{e})\ .$$

Now let

$$\begin{gathered}
m_1=\beta(\tilde{m}_1-\underline{m}_1)\bmod N^s, \qquad u_1=\beta(\tilde{m}_1-\underline{m}_1)\operatorname{div}N^s \\
r_1=(\tilde{r}_1\underline{r}_1^{-1})^{\beta}C_1^{\alpha}K_1^{u_1}\bmod N\ .
\end{gathered}$$

Then it is easy to see that $C_1=K_1^{m_1}r_1^{N^s} \bmod N^{s+1}$. In the same way we can write

$$\begin{aligned}
C_1&=K_1^{m_1}r_1^{N^s}\bmod N^{s+1}, & C_2&=K_2^{m_2}r_2^{N^s}\bmod N^{s+1} \\
C_3&=K_3^{m_3}r_3^{N^s}\bmod N^{s+1}, & C_4&=K_4^{m_4}r_4^{N^s}\bmod N^{s+1} \\
C_7&=K_5^{m_3}r_7^{N^s}\bmod N^{s+1}, & C_8&=K_6^{m_4}r_8^{N^s}\bmod N^{s+1} \\
C_5&=K_5^{m_5}r_5^{N^s}\bmod B^{s+1}, & C_6&=K_6^{m_6}r_6^{N^s}\bmod B^{s+1}
\end{aligned}$$

where

$$\begin{gathered}
m_1=\beta(\tilde{m}_1-\underline{m}_1)\bmod N^s, \qquad m_2=\beta(\tilde{m}_2-\underline{m}_2)\bmod N^s \\
m_3=\beta(\tilde{m}_3-\underline{m}_3)\bmod N^s, \qquad m_4=\beta(\tilde{m}_4-\underline{m}_4)\bmod N^s \\
m_5=\beta(m_3((\tilde{m}_1+\tilde{m}_2)-(\underline{m}_1+\underline{m}_2)+\tilde{m}-\underline{m}))\bmod N^s \\
m_6=\beta(m_4((\tilde{m}_1+\tilde{m}_2)-(\underline{m}_1+\underline{m}_2)+\underline{m}-\tilde{m}))\bmod N^s\ .
\end{gathered}$$

This is exactly what we want as

$$\begin{aligned}
m_6+m_5 &\equiv (m_3+m_4)(\beta(\tilde{m}_1-\underline{m}_1)+\beta(\tilde{m}_2-\underline{m}_2)) \\
&\equiv (m_3+m_4)(m_1+m_3) \pmod{N^s}\ .
\end{aligned}$$

7.5.3 Special Honest Verifier Zero-Knowledge

We can assume without lose of generality that we know m_3 and that we know t_6 such that $K_6 = t_6^{N^s} \bmod N^{s+1}$. Now pick $r_7, v_8, \tilde{m}_1^{(r)}, \tilde{r}_1, \tilde{m}_2^{(r)}, \tilde{r}_2, \tilde{m}_3^{(r)}, \tilde{r}_3, \tilde{m}_4^{(r)}, \tilde{r}_4, \tilde{r}_7, \tilde{r}_8, \tilde{m}^{(r)}, \tilde{r}_5, \tilde{r}_6$ at random, and then let

$$
\begin{aligned}
C_7 &= K_5^{m_3} r_7^{N^s}, \quad & C_8 &= v_8^{N^s} \\
\overline{C}_1 &= K_1^{\tilde{m}_1^{(r)}} \tilde{r}_1^{N^s} C_1^{-e}, \quad & \overline{C}_2 &= K_2^{\tilde{m}_2^{(r)}} \tilde{r}_2^{N^s} C_2^{-e} \\
\overline{C}_3 &= K_3^{\tilde{m}_3^{(r)}} \tilde{r}_3^{N^s} C_3^{-e}, \quad & \overline{C}_4 &= K_4^{\tilde{m}_4^{(r)}} \tilde{r}_4^{N^s} C_4^{-e} \\
\overline{C}_7 &= K_5^{\tilde{m}_3^{(r)}} \tilde{r}_7^{N^s} C_7^{-e}, \quad & \overline{C}_8 &= K_6^{\tilde{m}_4^{(r)}} \tilde{r}_8^{N^s} C_8^{-e} \\
\overline{C}_5 &= C_7^{\tilde{m}_1^{(r)} + \tilde{m}_2^{(r)}} K_5^{\tilde{m}^{(r)}} \tilde{r}_5^{N^s} C_5^{-e}, \quad & \overline{C}_6 &= C_8^{\tilde{m}_1^{(r)} + \tilde{m}_2^{(r)}} K_6^{-\tilde{m}^{(r)}} \tilde{r}_6^{N^s} C_6^{-e}
\end{aligned}
$$

7.5.4 State Construction

After the special honest verifier simulation we are given

$$
\begin{aligned}
C_1 &= K_1^{m_1} r_1^{N^s}, \quad & C_2 &= K_2^{m_2} r_2^{N^s} \\
C_3 &= K_1^{m_3} r_3^{N^s}, \quad & C_4 &= K_2^{m_4} r_4^{N^s} \\
C_5 &= K_1^{m_5} r_5^{N^s}, \quad & C_6 &= K_2^{m_6} r_6^{N^s}
\end{aligned}
$$

where

$$(m_1 + m_2)(m_3 + m_4) \equiv (m_5 + m_6) \pmod{N^s} .$$

Let $l = \frac{m_6 + m_5 - (m_1 + m_2)(m_3 + m_4)}{N^s}$ and let $r_8 = v_8 t_6^{-m_4} \bmod N$ such that

$$C_7 = K_5^{m_3} r_7^{N^s}, \quad C_8 = K_6^{m_4} r_8^{N^s} .$$

Then let

$$
\begin{aligned}
\overline{m}_1 = \tilde{m}_1^{(r)} - e m_1 \bmod N^s, &\quad \tilde{m}_1 = e m_1 + \overline{m}_1 \\
\overline{r}_1 = \tilde{r}_1 r_1^{-e} K_1^{-(\tilde{m}_1 \operatorname{div} N^s)} &\bmod N \\
\overline{m}_2 = \tilde{m}_2^{(r)} - e m_2 \bmod N^s, &\quad \tilde{m}_2 = e m_2 + \overline{m}_2 \\
\overline{r}_2 = \tilde{r}_2 r_2^{-e} K_2^{-(\tilde{m}_2 \operatorname{div} N^s)} &\bmod N \\
\overline{m}_3 = \tilde{m}_3^{(r)} - e m_3 \bmod N^s, &\quad \tilde{m}_3 = e m_3 + \overline{m}_3 \\
\overline{r}_3 = \tilde{r}_3 r_3^{-e} K_3^{-(\tilde{m}_3 \operatorname{div} N^s)} &\bmod N \\
\overline{m}_4 = \tilde{m}_4^{(r)} - e m_4 \bmod N^s, &\quad \tilde{m}_4 = e m_4 + \overline{m}_4 \\
\overline{r}_4 = \tilde{r}_4 r_4^{-e} K_4^{-(\tilde{m}_4 \operatorname{div} N^s)} &\bmod N \\
\overline{r}_7 = \tilde{r}_7 r_7^{-e} K_5^{-(\tilde{m}_3 \operatorname{div} N^s)}& \\
\overline{r}_8 = \tilde{r}_8 r_8^{-e} K_6^{-(\tilde{m}_4 \operatorname{div} N^s)}& \\
\overline{m} = \tilde{m}^{(r)} - e m_5 - m_3(\overline{m}_1 + \overline{m}_2 - \tilde{m}_1 - \tilde{m}_2) &\bmod N^s \\
\tilde{m} = e m_5 + m_3(\overline{m}_1 + \overline{m}_2 - \tilde{m}_1 - \tilde{m}_2) + \overline{m}& \\
\overline{r}_5 = \tilde{r}_5 r_5^{-e} r_7^{\tilde{m}_1 + \tilde{m}_2} K_5^{-(\tilde{m} \operatorname{div} N^s)}& \\
\overline{r}_6 = \tilde{r}_6 r_6^{-e} r_8^{\tilde{m}_1 + \tilde{m}_2} K_6^{(\tilde{m} \operatorname{div} N^s) - [el + (m_3 + m_4)((\tilde{m}_1 \operatorname{div} N^s) + (\tilde{m}_2 \operatorname{div} N^s))]}& .
\end{aligned}
$$

It is straightforward to verify that these values all have the required distribution.

7.6 Proof of Identity and Proof of Additive Relation

A proof of identity between committed values can be extracted from the above proof of multiplicative relation, by setting $m_3 = 1$ and $m_4 = 0$. The proof can be somewhat simplified by eliminating the unnecessary values $\overline{C}_3, \overline{C}_4, C_7, C_8$.

Consider three commitments under the same key (K_1, K_2): $(K_1^{m_1} r_1^{N^s},\ K_2^{m_2} r_2^{N^s})$, $(K_1^{m_3} r_3^{N^s}, K_2^{m_4} r_4^{N^s})$, and $(K_1^{m_5} r_1^{N^s}, K_2^{m_6} r_2^{N^s})$, where $(m_1 + m_2) + (m_3 + m_4) = m_5 + m_6$. A proof of additive relation between such commitments can be constructed by proving that the base commitments $C_1 = K_1^{m_1+m_3-m_5}(r_1 r_3 r_5^{-1})^{N^s}$ and $C_2 = K_2^{m_6-m_2-m_4}(r_6 r_2^{-1} r_4^{-1})^{N^s}$ commit to the same value. Arbitrary commitments can be brought on this restricted form by recommitting in some common key and then using the proof of identity to check that this was done correctly. A proof that C_1 and C_2 commit to the same value can be constructed in line with the proof of identity of the transformed scheme.

7.7 Proof of Binary Boolean Relations

The proof that two committed values are identical implies a proof that a committed value equals a public constant, and hence, using the same techniques as in Section 6.7, a proof that a committed value equals 0 or 1. It is well known that multiplication and addition in any ring can be used to simulate Boolean operations efficiently, when the input is constrained to 0/1 values, so we obtain therefore immediately proofs for any relation defined by a binary Boolean function f, i.e., a proof that committed values m_1, m_2, m_3 are 0/1 values and satisfy $f(m_1, m_2) = m_3$.

Theorem 3 *Relative to the DCRA, the above commitment scheme is a special mixed commitment scheme with proofs of relations between committed values for identity, additive relation, multiplicative relation* ($\bmod N^s$), *and for Boolean relations defined by any binary Boolean function.*

8 Efficient Universally Composable Zero-Knowledge Proofs

In [CF01] Canetti and Fischlin showed how universally composable commitments can be used to construct simple Zero-Knowledge (ZK) protocols which are universally composable. This is a strong security property, which implies concurrent and non-malleable ZK proof of knowledge.

The functionality $\mathcal{F}_{\mathrm{ZK}}^{R}$ for universally composable zero-knowledge (for binary relation R) is as follows.

1. Wait to receive a value $(\texttt{verifier}, id, P_i, P_j, x)$ from some party P_i. Once such a value is received, send $(\texttt{verifier}, id, P_i, P_j, x)$ to $\mathcal{S}$, and ignore all subsequent $(\texttt{verifier}, \ldots)$ values.

2. Upon receipt of a value $(\texttt{prover}, id, P_j, P_i, x', w)$ from P_j, let $v = 1$ if $x = x'$ and $R(x, w)$ holds, and $v = 0$ otherwise. Send (id, v) to P_i and $\mathcal{S}$, and halt.

In [CF01] a protocol for Hamiltonian-Cycle (HC) is given and proven to securely realize $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{HC}}$.

8.1 Exploiting the Multi-Bit Commitment Property

First we show how the fact that we can commit to k bits using $O(k)$ bits of communication can be used to reduce the communication complexity of known solutions.

Before describing this optimization we demonstrate the technique from [CF01] to get a similar protocol for satisfiability of Boolean circuits (SAT). Parts of the following treatment will be very close in structure and text to the one in [CF01], but we include it for completeness.

A Boolean circuit C consists of l inputs and m binary gates. (For notational reason we only consider binary gates.) The binary gates are named $G_{l+1}, \ldots, G_{l+m}$. The binary gate G_i is a tuple $(i_1, i_2, G_{i,0,0}, G_{i,0,1}, G_{i,1,0}, G_{i,1,1})$, where $i_1, i_2 < i$ and $G_{i,0,0}, G_{i,0,1}, G_{i,1,0}, G_{i,1,1} \in \{0,1\}$. An evaluation $\text{eval}(C, x) = (G_1(x), \ldots, G_{l+m}(x)) \in \{0,1\}^{l+m}$ of C on input $x \in \{0,1\}^l$ is given by $G_i(x) = x_i$ for $i = 1, \ldots, l$, $G_i(x) = G_{i,G_{i_1}(x),G_{i_2}(x)}$ for $i = l+1, \ldots, l+m$. We let $C(x)$ denote $G_{l+m}(x)$.

A scrambler is a value $S \in \{0,1\}^{l+m}$, where $S_{l+m} = 0$. An S-scrambling of circuit C is a circuit $S(C)$, where for $i = l+1, \ldots, l+m$ the gate G'_i is given by $G'_i = (i_1, i_2, G'_{i,0,0}, G'_{i,0,1} G'_{i,1,0} G'_{i,1,1})$, where $G_{i,b_1,b_2} = G_{i,b_1 \oplus S_{i_1}, b_2 \oplus S_{i_2}} \oplus S_i$. An S-scrambling of input x is $S(x) = (x_1 \oplus S_1, \ldots, x_l \oplus S_l)$. Note that $\text{eval}(S(G), S(x)) = \text{eval}(G, x) \oplus S$.

The protocol, which we name SAT, proceeds as follows.

1. Given input $(\texttt{Prover}, id, P, V, C, x)$, where C is a circuit and x is an input for the circuit the prover proceeds as follows. If $C(x) = 0$, then P sends a message `reject` to V. Otherwise, P proceeds as follows for $j = 1, \ldots, t$.

   (a) Pick a uniformly random scrambler S^j and compute $x^j = S^j(x)$ and $C^j = S^j(C)$.

   (b) Using $\mathcal{F}_{\text{com}}$ commit to the bits of the scrambler.

   (c) Using $\mathcal{F}_{\text{com}}$ commit to x^j and under commitment id (j, i, b_1, b_2) commit to G^j_{i,b_1,b_2} for $i = l+1, \ldots, l+m$, $b_1 \in \{0,1\}$, and $b_2 \in \{0,1\}$.

2. Given input $(\texttt{Verifier}, id, V, P, C)$, the verifier waits to receive either `reject` from P, or receipts from $\mathcal{F}_{\text{com}}$. If `reject` is received, then V outputs $(id, 0)$ and halts. Otherwise, once all the receipts are received V randomly chooses t bits $c_1, \ldots, c_t$ and sends these to P.

3. Upon receiving $c_1, \ldots, c_t$ from V, P proceeds as follows for $j = 1, \ldots, t$.

   (a) If $c_j = 0$, then reveal S^j and C^j.

   (b) If $c_j = 1$, then reveal x^j and for each gate $G^j_i = (i_1, i_2, G^j_{i,0,0}, G^j_{i,0,1}, G^j_{i,1,0}, G^j_{i,1,1})$ open the commitment with id $(j, i, G^j_{i_1}(x^j), G^j_{i_1}(x^j))$.

4. Upon receiving the appropriate $(\texttt{Open}, \ldots)$ messages from $\mathcal{F}_{\text{com}}$ the verifier V verifies the following for all $j = 1, \ldots, t$. If $c_j = 0$, then it holds that $C^j = S^j(C)$. If $c_j = 1$, then it holds that the revealed $x^j, v_{l+1}, \ldots, v_{l+m}$ are consistent with a acceptance, i.e. if gate i_1 was open to v_1 and gate i_2 was open to v_2, then for gate i the commitment with id (j, i, v_1, v_2) was opened, and further more gate $l+m$ was opened to 1.

Theorem 4 *The protocol* SAT *securely realizes* $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$ *in the* $\mathcal{F}_{\text{com}}$*-hybrid model.*

Proof (sketch): Let $\mathcal{A}$ be an adversary attacking SAT in the $\mathcal{F}_{\text{com}}$-hybrid model. We construct a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish between $\mathcal{A}$ attacking SAT in the $\mathcal{F}_{\text{com}}$-hybrid model and $\mathcal{S}$ attacking the ideal process for $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$.

The simulator $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$. Messages from $\mathcal{Z}$ are relayed to $\mathcal{A}$ and messages from $\mathcal{A}$ to its environment are relayed to $\mathcal{Z}$. Other types of messages are handled as follows.

1. If $\mathcal{A}$, controlling a corrupted party P, starts an interaction as a prover with an uncorrupted party V, then $\mathcal{S}$ records the values that $\mathcal{A}$ sends to $\mathcal{F}_{\text{com}}$, sends challenges $c_1, \ldots, c_t$ as in the protocol, and records $\mathcal{A}$'s responses. Now $\mathcal{S}$ simulates V's decision and if V accepts, then $\mathcal{S}$ tries to find a witness x and hands it to the ideal functionality. If such a witness cannot be found the simulator aborts.

   To find the witness $\mathcal{S}$ finds a j, where $c_j = 1$ and $C^j = S^j(C)$. It is proven below that the probability of not finding such j is less than $2^{-t/2}$. Let x^j and $v_{l+1}, \ldots, v_{l+m}$ be the values revealed by $\mathcal{A}$ and let x be given by $x_i = x_i^j \oplus S_i^j$. Then $\text{eval}(C, x) = (x_1^j, \ldots, x_l^j, v_{l+1}, \ldots, v_{l+m}) \oplus S^j$. To see this observe that for $i = 1, \ldots, l$ we have that $G_i(x) = x_i = x_i^j \oplus S_i^j$ and for $i = l+1, \ldots, l+m$ we inductively have that $G_i(x) = G_{i, G_{i_1}(x), G_{i_2}(x)} = G_{i, v_{i_1} \oplus S_{i_1}^j, v_{i_2} \oplus S_{i_2}^j} = G_{i, v_{i_1}, v_{i_2}} \oplus S_i^j = v_i \oplus S_i^j$. Especially we have that $C(x) = v_{l+m} \oplus S_{l+m}^j = 1 \oplus 0 = 1$ and thus x is a witness.

2. If an uncorrupted party P starts an interaction with a corrupted party V, then $\mathcal{S}$ learns from the ideal functionality $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$ whether V should accept or reject, and simulates the view of $\mathcal{A}$ accordingly. Note that $\mathcal{S}$ has no problem carrying out the simulation since it simulates for $\mathcal{A}$ an interaction with $\mathcal{F}_{\text{com}}$ where $\mathcal{F}_{\text{com}}$ is played by $\mathcal{S}$. Thus, $\mathcal{S}$ is not bound by the "commitments" and can "open" them in whichever way it pleases.

3. If two uncorrupted parties P and V interact, then $\mathcal{S}$ simulates for $\mathcal{A}$ the appropriate protocol messages. This is very similar to the case of corrupted verifier, since the protocol is a Σ-protocol.

4. Party corruptions are dealt with in a straightforward way. Since the protocol is a Σ-protocol, corrupting the verifier provides the adversary with no extra information. When the prover is corrupted $\mathcal{S}$ corrupts the prover in the ideal process, obtains x, and generates an internal state of the prover that matches the protocol stage. Generating such a state is not problematic since $\mathcal{S}$ is not bound by any "commitments", and it can freely choose the remaining values to match the (simulated) conversation up to the point of corruption.

Given that $\mathcal{S}$ does not abort in Step 1, the validity of the simulation is straightforward. We prove that the probability that a proof is accepted and there does not exist j, where $c_j = 1$ and $C^j = S^j(C)$, is less than $2^{-t/2}$.

Assume that the expected number of indices j where $C^j = S^j(C)$ is at least $t/2$. Then the probability that $c_j = 0$ for all these indices is less than $2^{-t/2}$ and thus the probability that there does not exist j, where $c_j = 1$ and $C^j = S^j(C)$, is less than $2^{-t/2}$. Assume on the contrary that the expected number of indices j, where $C^j = S^j(C)$, is less than $t/2$. Then with probability less than $2^{-t/2}$ we have $c_j = 1$ on all the indices where $C^j \neq S^j(C)$ and thus the probability of acceptance is less than $2^{-t/2}$. $\square$

We now show how to use our new universally composable commitments to make the protocol more efficient.

For each value of j the protocol commits to $2l + 5m - 1$ bits. Of these, the $2l + m - 1$ bits of x^j and S^j are always either revealed or not. We can therefore efficiently commit to them using the multi-bit commitments.

We cannot however not commit to the remaining $4m$ commitments using multi-bit commitments. The reason for this is that if $c_j = 0$ all the bits should be revealed and if $c_j = 1$, then only a subset S of them should be revealed, and more importantly, revealing the subset S in case $c_j = 0$ would effectively reveal x.

In [KMO89] Kilian, Micali, and Ostrovski presented a general technique for transforming a multi-bit commitment scheme into a multi-bit commitment scheme with the property that individual bits can be open independently. Unfortunately their technique adds one round of interaction. However, we do not need the full generality of their result. This allows us to modify the technique to avoid the extra round of interaction. Let $m \in \{0,1\}^{4m}$ denote the bits to be committed to and let $S \in \{0,1\}^{4m}$ denote the subset $\{i|S_i = 1\}$ of the entries of m which should be revealed if $c_j = 1$. Then m should be revealed if $c_j = 0$ and $(S, m \wedge S)$ should be revealed if $c_j = 1$. Then do the commitment by generating a uniformly random pad $m' \in \{0,1\}^{4m}$ and committing to the four values m', $m \oplus m'$, S, and $m' \wedge S$ individually using multi-bit commitments. The verifier then challenges uniformly with $d_j \in \{0, 1, 2\}$. If $d_j = 0$, then reveal m and $m \oplus m'$. If $d_j = 1$, then reveal $m \oplus m'$, S and $m' \wedge S$ and thereby $m \wedge S = (m \oplus m') \wedge S \oplus (m' \wedge S)$. If $d_j = 2$, then reveal m', S, and $m' \wedge S$ and let the verifier check the consistency of these values.

The extension of the above analysis to consider the modified protocol is straightforward. In Step 1 find a j, where $d_j = 1$ and $C^j = S^j(C)$ and the committed values are consistent. Then proceed as before. It is easy to see that the probability that such a j does not exist is less than $2^{-t/3}$.

It is clear that we can achieve the same error probability as in the original protocol by increasing t by a constant factor of 3/2. Furthermore, the number of bits committed to for each scrambling is the same up to a constant factor. Therefore, if we implement the modified protocol using our commitment scheme, we get communication complexity $O((l + m + k)t)$, where k is the security parameter of the commitment scheme and t is the number of iterations in the zero-knowledge proof. This follows because we can commit to $O(l + m)$ bits by sending $O(l + m + k)$ bits. This can be compared to the $O((l + m)kt)$ bits we would need for SAT using the commitments from [CF01]; this is an improvement by a factor $\theta(\frac{(l+m)k}{(l+m)+k})$.

8.2 Exploiting Efficient Proofs of Relations

Here we show how we can use the homomorphic properties and/or efficient proofs of relations on committed values to reduce the communication complexity and the round complexity in a different way. This is done by "evaluating" the circuit using the proofs of relations.

1. Given input $(\texttt{Prover}, id, P, V, C, x)$, where C is a circuit and x is an input for the circuit, the prover proceeds as follows. If $C(x) = 0$, then P sends a message `reject` to V. Otherwise, compute $\text{eval}(G, x) = (v_1, \ldots, v_{l+m})$ and using $\mathcal{F}_{\text{HCOM}}$ commit to the bits of $\text{eval}(G, x)$, except v_{l+m}.

   Then for each gate $G_i = (i_1, i_2, G_{i,0,0}, G_{i,1,0}, G_{i,0,1}, G_{i,1,1})$, $i = l + 1, \ldots, l + m - 1$, P proves (using $\mathcal{F}_{\text{HCOM}}$) that the commitments to v_i, v_{i_1}, and v_{i_2} are in the binary Boolean relation specified by $(G_{i,0,0}, G_{i,1,0}, G_{i,0,1}, G_{i,1,1})$. And for $G_{l+m} = (i_1, i_2, G_{l+m,0,0}, G_{l+m,1,0}, G_{l+m,0,1}, G_{l+m,1,1})$, proves that v_{i_1} and v_{i_2} are in the relation $G_{l+m,v_{i_1},v_{i_2}} = 1$. This can be done by committing to 1 (using $\mathcal{F}_{\text{HCOM}}$) and then doing a proof on the commitments.

2. Given input (`Verifier`, id, V, P, C), the verifier waits to receive either `reject` from P, or receipts and proofs from $\mathcal{F}_{\text{HCOM}}$. If `reject` is received, then V outputs $(id, 0)$ and halts. Otherwise, once all the appropriate receipts and proofs are received V outputs $(id, 1)$.

The following theorem is straightforward.

Theorem 5 *The protocol* SAT_2 *securely realizes* $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$ *in the* $\mathcal{F}_{\text{HCOM}}$*-hybrid model.*

This protocol has no messages of its own. All interaction is done through $\mathcal{F}_{\text{HCOM}}$. The security notion guarantees that the security is preserved under any scheduling of the messages of the underlying $\mathcal{F}_{\text{HCOM}}$ protocol. This especially allows us to run the overall zero-knowledge proof as a Σ-protocol by bundling the messages when our implementation of the commitment protocol is used. This protocol requires $O(l + m)$ commitments to single bits, each of which require $O(k)$ bits of communication. Then we need to do $O(l + m)$ proofs of relations, each of which require $O(k + t)$ bits of communication using our implementation and assuming that we want an overall error probability of $2^{-\Omega(t)}$. This amounts to $O((l + m)(k + t))$ bits of communication, and is an improvement over the $O((l + m)kt)$ bits for the original protocol, namely by a factor $O(\frac{kt}{k+t})$.

This is incomparable to the optimization using multi-bit commitments. But note that in the theory of zero-knowledge protocols, sometimes one only considers the case where all parameters are linear in the size m of the common input. In this case, both optimizations improve from $O(m^3)$ to $O(m^2)$ bits.

References

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science*. IEEE, 2001.

[CDS94] R. Cramer, I. B. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology - Crypto '94*, pages 174–187, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science Volume 839.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In J. Kilian, editor, *Advances in Cryptology - Crypto 2001*, pages 19–40, Berlin, 2001. Springer-Verlag. Lecture Notes in Computer Science Volume 2139.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology - EuroCrypt 2000*, pages 418–430, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science Volume 1807.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 110–136, Berlin, 2001. Springer. Lecture Notes in Computer Science Volume 1992.

[DJN01] Ivan B. Damgård, Mads J. Jurik, and Jesper B. Nielsen. A generalization, a simplification and some applications of Paillier's probabilistic public-key system. Manuscript, obtainable from authors. Submitted to journal. Preliminary version occurs in [DJ01], 2001.

[KMO89] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 474–479, Research Triangle Park, North Carolina, 30 October–1 November 1989. IEEE.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Advances in Cryptology - EuroCrypt '98*, pages 308–318, Berlin, 1998. Springer-Verlag. Lecture Notes in Computer Science Volume 1403.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residue classes. In Jacques Stern, editor, *Advances in Cryptology - EuroCrypt '99*, pages 223–238, Berlin, 1999. Springer-Verlag. Lecture Notes in Computer Science Volume 1592.