Long-term Security and Universal Composability

Jörn Müller-Quade[1] and Dominique Unruh[2*]

[1] IAKS, Universität Karlsruhe (TH), Germany
[2] Saarland University, Saarbrücken, Germany

Abstract. Algorithmic progress and future technology threaten today's cryptographic protocols. Long-term secure protocols should not even in future reveal more information to a—then possibly unlimited—adversary.
In this work we initiate the study of protocols which are long-term secure *and* universally composable. We show that the usual set-up assumptions used for UC protocols (e.g., a common reference string) are not sufficient to achieve long-term secure *and* composable protocols for commitments or general zero knowledge arguments. Surprisingly, nontrivial zero knowledge protocols are possible based on a coin tossing functionality: We give a long-term secure composable zero knowledge protocol proving the knowledge of the factorisation of a Blum integer.
Furthermore we give practical alternatives (e.g., signature cards) to the usual setup-assumptions and show that these allow to implement the important primitives commitment and zero-knowledge argument.

Keywords: Universal Composability, long-term security, zero-knowledge, commitment.

Table of Contents

[*] Most of the work was done while the second author was at the IAKS, Universität Karlsruhe (TH)

1 Introduction

Computers and algorithms improve over time and so does the power of an adversary in cryptographic protocols. The VENONA project is an example where NSA and GCHQ stored Russian ciphertexts over years until they could eventually be cryptanalysed. Official key length recommendations, e.g. by the Federal Office for Information Security (BSI) in Germany, usually do not exceed six years and future technology like quantum computers could render even paranoid choices for the key length obsolete.

Everlasting security from assumptions which have to hold only during the protocol execution would be an ideal solution to this problem. In this work we combine the notions of universal composability and long-term security. For the first time we investigate protocols which are long-term secure *and* exhibit a composition theorem which allows a modular design of such protocols. In particular, we investigate commitment protocols and zero knowledge schemes which are composable and robust against future improvements of the adversary's computing technology.

To capture the threat of an adversary with increasing power we introduce the security notion of *long-term universal composability* (long-term-UC) with the intuition that the adversary becomes unlimited at some point of time after termination of the protocol. The protocols do not run after this point of time, but all information stored from past executions should not reveal any additional information to the then unlimited adversary. A surprising consequence of our work is that unconditionally hiding universally composable commitments [DN02] are not necessarily long-term-UC.

Long-term-UC is preserved under composition, i.e., idealised building blocks can be replaced by long-term-UC protocols while preserving the long-term security of the complete application. The security notion of long-term-UC lies strictly between information theoretical security, where the adversary is unlimited from the start, and computational security, where for a concrete security parameter the computational power of the adversary must be limited for all times to come.

The idea of everlasting security has been considered with respect to memory bounded adversaries. Key exchange protocols and protocols for oblivious transfer have been developed in the bounded storage model [CM97, CCM02]. These protocols can be broken by an adversary with more memory than assumed, however they cannot be broken in retrospect even by an unlimited adversary. A scheme using distributed servers of randomness (virtual satellites) to achieve everlasting security has been implemented [Rab03]. In this scheme the access of the adversary to the communication of the parties is limited during the key exchange. It was shown by [DM04] that in the bounded-storage model composability cannot be taken for granted. They gave a key-exchange protocol that is secure in the bounded-storage model even if the initial key leaks after protocol termination, and then showed that if the initial key was generated by a computationally secure key exchange protocol, the resulting protocol is insecure. However, theirs was a purely negative result in that they did not give any criteria under which composition would be possible.

Long-term security has been investigated in quantum cryptography. It is generally accepted (even though not formally proven) that an only computationally secure authentication of a quantum key exchange yields a long-term secure key. Bit commitment and oblivious transfer quantum protocols which become unconditionally secure, but rely on temporary computational assumptions have been searched, but are now known to be impossible[3] (see, e.g. [BCMS99]).

Zero knowledge proofs where the verifier cannot (ever) break the protocol and the prover can only on-line break the protocol where given in [BCC88]. In [MQ05] protocols achieving long-term security were stated, however, only secure function evaluation with constant input size was considered.

Another related topic is that of forward security, where it is demanded that past session keys remain computationally secure even if a long-term secret is given to the adversary. This notion is related to but less strict than long-term-UC as the session keys will not remain secure forever.

With exception of [DM04], previous work on long-term security did not take the problem of composability into account. When composability is required the situation changes drastically. E.g., an unconditionally hiding UC commitment is not long-term-UC and a straightforward adaption of e.g., the protocol of [BCC88] using an unconditionally hiding UC commitment does not yield long-term-UC zero knowledge arguments.

In this work we thoroughly investigate under which assumptions long-term-UC commitments and long-term-UC zero knowledge arguments exist. We prove that a common reference string or a coin toss functionality are not sufficient for realising long-term-UC commitments. To be more general we define a functionality $\mathcal{F}$ to be only temporarily secret for a party P if, roughly speaking, every secret known to P and $\mathcal{F}$ can in principle (but not necessarily efficiently) be computed from the communication of $\mathcal{F}$ with all the other parties. Coin tossing and a common reference string are only temporarily secret for all parties and we show that long-term-UC commitments are impossible given any functionality which is only temporarily secret for the committer.

In contrast to this impossibility of commitments there exist nontrivial languages for which zero knowledge protocols are possible even with an only temporarily secret functionality. More concrete we give a zero knowledge proof of knowledge of the factorisation of a Blum integer using a helping coin toss functionality. This is astonishing as such a proof is not possible using a common reference string instead of a coin toss (unless factoring of Blum integers is easy for nonuniform machines). More generally we prove that no nonuniformly nontrivial language has a zero knowledge argument with the help of any functionality which works “offline” in the sense that it needs, like a common reference string, only be invoked before the start of the protocol and which is only temporarily secret for both parties. Even though most PKI are of this form and hence do not allow any nontrivial long-term-UC zero knowledge or commitment protocols we give an interesting and not too academic example of a PKI which allows to implement a long-term-UC commitment.

Further we give two helping functionalities which are motivated from (temporarily) tamper proof hardware which allow to implement an unlimited number of long-term-UC

[3] Unless additional assumptions are made, such as bounded quantum storage or the availability of a piece of trusted hardware.

commitments and zero knowledge arguments for all in NP. One of these functionalities resembles a trusted device which is computationally indistinguishable from a random oracle and the other a smart card which can generate digital signatures, but from which the secret key cannot be extracted. Note however that in contrast to the classical (i.e., not long-term secure) UC definition, commitments and ZK are not sufficient to implement any functionality.

1.1 Preliminaries

Notation. We call a function f *negligible*, if for any polynomial p and sufficiently large k, $f(k) \leq 1/p(k)$. We call f *overwhelming*, when $1 - f$ is negligible.

A *PPT-algorithm (probabilistic polynomial time)* is a uniform probabilistic algorithm that runs in polynomial-time in the length of its inputs.

We call a relation R on $\{0,1\}^* \times \{0,1\}^*$ *poly-balanced* if there is a polynomial p, s.t. $|w| \leq p(|x|)$ for all x, w with xRw. We call R an *NP-relation* if it is poly-balanced and deciding $(x,w) \in R$ is in P. We call R an *MA-relation* if it is poly-balanced and deciding $(x,w) \in R$ is in BPP. The language L_R associated with R is $L_R := \{x \in \{0,1\}^* : \exists w : xRw\}$. We usually call x the *statement* and w with xRw the *witness* for x. We call a MA-relation R *(uniformly) trivial* if there is a PPT-algorithm that upon input $x \in L_R$ outputs a witness for x with overwhelming probability. We call R *nonuniformly deterministically trivial* there is a nonuniform deterministic polynomial-time algorithm that upon input $x \in L_R$ outputs a witness for x.

An integer $n > 0$ is called a *Blum-integer*, if $n = pq$ for two primes p, q with $p \equiv q \equiv 3 \bmod 4$.

Cryptographic tools. In [NOVY98], it is shown that assuming the existence of a one-way permutation, an unconditionally hiding commitment scheme exists. This scheme has the additional properties that the unveil-phase consists of only one message, and that given the message, the committed value v, and the transcript of the interaction in the commit phase, there is a deterministic polynomial-time algorithm that checks whether the verifier accepts the value v.

Using that commitment-scheme in the zero-knowledge proof-system for graph-3-colourability from [GMW91], we get a statistically witness indistinguishable argument of knowledge for any NP-relation given any one-way permutation.[4] Using a statistically witness indistinguishable argument of knowledge for any NP-relation and a unconditionally hiding commitment scheme, we can easily construct a statistically witness indistinguishable argument of knowledge for any MA-relation using any one-way permutation.[5]

[4] The resulting scheme is of course also zero-knowledge, but we do not need that property here.

[5] Let B be a PPT-algorithm s.t. $B(w,x) = 1$ with overwhelming probability for xRw and with negligible probability otherwise. Such an algorithm exists for any MA-relation R. To prove a statement $x \in L_R$, the prover first commits to the witness w, then commits to randomness r'. The verifier sends to the prover randomness r''. Then the prover proves using a statistically witness indistinguishable argument of knowledge that he knows a witness, s.t. $B(w,x) = 1$ with random-tape $r := r' \oplus r''$. Since the latter statement is in NP, this can be done given a one-way permutation.

2 Modelling long-term UC

We now present our modelling of universally composable long-term security (short long-term UC). We build on the Universal Composability framework [Can05]. In that modelling, a computationally limited entity called the environment has to distinguish between an execution of the protocol (with some adversary) and an execution of an ideal functionality (with some simulator). To define long-term security, we have to add the requirement that even if some entity gets unlimited computational power after the execution of the protocol, security is maintained. In the Universal Composability framework, this is quite easily done: We simply require that *after* the execution of the protocol (which is still performed against computationally limited adversaries) even an unlimited entity could not distinguish between an execution of the real protocol or of the functionality, i.e., we require that the output of the environment is *statistically* indistinguishable.[6]

Definition 1 (Long-term UC). *Let* $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ *denote the output of* $\mathcal{Z}$ *in an execution of the protocol* π *with adversary* $\mathcal{A}$ *and environment* $\mathcal{Z}$*, where* k *is the security parameter and* z *the auxiliary input of the environment* $\mathcal{Z}$*.* $\mathrm{EXEC}_{\mathcal{F},\mathcal{A},\mathcal{Z}}(k,z)$ *is defined analogously.*[7]

A protocol π long-term-UC realises *a functionality* $\mathcal{F}$*, if for any polynomial-time adversary* $\mathcal{A}$ *there exists a polynomial-time simulator* $\mathcal{S}$*, s.t. for any polynomial-time environment*[8] $\mathcal{Z}$ *the families of random variables* $\{\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}}$ *and* $\{\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}}$ *are* statistically *indistinguishable.*

Note that the Universal Composition Theorem from [Can05] applies with a virtually unmodified proof.

Conventions. In all our results we assume that secure channels are given for free (i.e., we are in the secure-channel network-model).[9] Further, security always denotes security with respect to static adversaries, i.e. parties are not corrupted *during* the protocol execution. However, we believe that our results can be adapted to adaptive adversaries.

We consider the case without an honest majority, since given an honest majority we could use information-theoretically secure protocols.

2.1 On the minimality of the security notion

At this point one might wonder whether this definition is possibly stricter than necessary, especially in view of the various impossibility results presented below. However,

[6] Note that we can w.l.o.g. assume that the output of the environment contains the whole view of that environment.

[7] See [Can05] for details.

[8] Not limited to environments with single bit output.

[9] This much simplifies the presentation. Since all our results concern the two-party case, it is easy to adapt our results to authenticated channels, if one adapts the definitions of the functionalities accordingly (e.g., the commitment functionality would then send the value of an unveil to the adversary as well as to the adversary). However, we cannot expect to use a key exchange protocol to make the authenticated channels secure, since such an approach would not be long-term secure.

if one is willing to accept stand-alone security (i.e., simulation-based security *without* an environment, see e.g. [Gol04]), with the extra requirement that the outputs of the parties and the adversary/simulator are *statistically* indistinguishable in real and ideal model (long-term stand-alone security), as a minimal security notion, we can argue as follows: If we want this minimal security *and* composability simultaneously, the proof from [Lin03][10] states that the minimal security notion satisfying these two requirements is a security notion similar to Definition 1, with the only difference that the simulator is allowed to depend on the environment (specialised-simulator long-term UC). Since all our impossibility results also apply for this weaker notion (we never use the fact that the simulator does not depend on the environment), we see that we cannot find an essentially more lenient security notion than Definition 1 if we accept long-term stand-alone security as a minimal security notion.

2.2 Functionalities

In this section, we define some commonly used functionalities that we will investigate in the course of this paper.

We assume the following conventions in specifying functionalities:

We always assume that the adversary is informed of every invocation of the functionality, and the functionality only delivers its output when the adversary has triggered that delivery. So a phrase like "upon input x from P_1, $\mathcal{F}$ sends y to P_2" should be understood as "upon input y from P_1, $\mathcal{F}$ sends (*i-th input from* P_1) to the adversary, and upon a message (*deliver* i) from the adversary, $\mathcal{F}$ sends y to P_2". For better readability, we use the shorter formulation.

Most of the functionalities defined here are parametrised by a function m giving the length of their input and outputs. We will often omit explicitly stating this m if it is clear from the context.

When a functionality receives an invalid input from some party, it simply forwards that input to the adversary.

The first functionality used in this paper is the common reference string (CRS). Intuitively, the CRS denotes a random string that has been chosen by some trusted party or by some natural process, and that is known to all parties prior to the start of the protocol.

Definition 2 (Common Reference String (CRS)). *Let $\mathcal{D}_k$ ($k \in \mathbb{N}$) be an efficiently samplable distribution on $\{0,1\}^*$. At its first activation the functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ chooses a value r according to the distribution $\mathcal{D}_k$ (k being the security parameter). Upon any input from P_i, send r to the adversary and to P_i (in particular, all parties P_i get the same r).*

If $\mathcal{D}_k$ is the uniform distribution on $\{0,1\}^{m(k)}$ for any k, we speak of a uniform CRS *of length m. We then write $\mathcal{F}_{\mathrm{CRS}}^{m}$ instead of $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}_k}$.*

[10] With minor modifications: simply replace computational indistinguishability by statistical indistinguishability.

The second functionality is the coin toss. At a first glance, the coin toss looks very similar to the CRS, since also the coin toss consists of a random string that is given to both parties involved (and to the adversary). However, the coin toss guarantees that no party can learn the coin toss before *both* parties agree to toss the coin.[11] As we will see below, a coin toss is more powerful than a CRS in the context of long-term UC.[12]

Definition 3 (Coin Toss (CT)). *When* both P_1 *and* P_2 *have given some input, the functionality* $\mathcal{F}_{\mathrm{CT}}^m$ *chooses a uniformly distributed* $r \in \{0,1\}^{m(k)}$ *and sends* r *to the adversary, to* P_1*, and to* P_2*.*

The next functionality models the setup assumption, that there is a trusted (pre-distributed) public key infrastructure, which provides each party with a secret key and attests the corresponding public key to any interested party.

Definition 4 (Public Key Infrastructure (PKI)). *Let* G *be a PPT-algorithm that upon input* 1^k *outputs two string* sk *and* pk*.*[13] *When* $\mathcal{F}_{\mathrm{PKI}}^G$ *runs with parties* $P_1, \dots, P_n$*, upon its first activation it chooses independent key pairs* $(sk_i, pk_i) \leftarrow G(1^k)$ *for* $i = 1, \dots, n$ *and sends* $(pk_1, \dots, pk_n)$ *to the adversary. When receiving any input from* P_i*, send* $(sk_i, pk_1, \dots, pk_n)$ *to* P_i*.*

The next two functionalities are well-known cryptographic building blocks that find application in the construction of many protocols.

Definition 5 (Commitment (COM)). *Let* C *and* R *be two parties. The functionality* $\mathcal{F}_{\mathrm{COM}}^{C \to R, m}$ *behaves as follows: Upon (the first) input* $x \in \{0,1\}^{m(k)}$ *from* C *send* (*committed*) *to* R*. Upon input* (*unveil*) *from* C *send* x *to* R*.*

We call C *the sender and* R *the recipient.*

Definition 6 (Zero-Knowledge (ZK)). *Let* R *be a MA-relation, and let* P *and* V *be two parties. The functionality* $\mathcal{F}_{\mathrm{ZK}}^{R, P \to V, m}$ *behaves as follows: Upon the first input of* (x, w) *from* P *satisfying* xRw *and* $|x| \leq m(k)$*, send* x *to* V*.*[14]

We call P *the prover and* V *the verifier.*

[11] This can be illustrated by the following example: Alice and Bob want to know which of them pays the bill. So Alice and Bob agree: "We toss a coin, if the outcome is 1, Bob pays, otherwise Alice pays." Of course, if they were to use a CRS instead of a coin toss they could not use this simple protocol, because the outcome of the CRS is known before the start of the protocol.

[12] Although, in contrast, a UC secure (without long-term) coin toss can be realised using a CRS under reasonable complexity assumptions, see [CF01].

[13] I.e., G is a key generation algorithm.

[14] The resulting functionality $\mathcal{F}_{\mathrm{ZK}}$ is not polynomial-time if R is not an NP-relation. However, in that case $\mathcal{F}_{\mathrm{ZK}}$ can be replaced by an efficient implementation that uses a BPP-algorithm for checking xRw and errs only with negligible probability. The resulting functionality is then indistinguishable from $\mathcal{F}_{\mathrm{ZK}}$.

3 Commitment

In this section we will examine the possibility of long-term-UC realising commitments. It will turn out, that commitment cannot be long-term-UC realised using CRS or coin-toss, nor with an arbitrary PKI. In particular unconditionally hiding UC commitments, which are possible with a CRS [DN02], are not necessarily long-term UC.[15] Note that the incompleteness of the CRS stands in stark contrast to the situation of (non-long-term) UC. In [CLOS02] it was shown that given a CRS, any functionality has a UC secure realisation. Furthermore, in [BCNP04] it was shown that the same holds for a PKI.[16] However, given a ZK functionality, commitments can be realised even with respect to long-term UC.

To state the impossibility results in a more general fashion, we first need the following definition:

Definition 7 (Only temporarily secret). *We say a functionality $\mathcal{F}$ is* only temporarily secret (OTS) for party P*, if the following holds in any protocol: Let* $trans$ *denote the transcript of all communication between* $\mathcal{F}$ *and the other machines (including the adversary). Let* $trans \setminus P$ *denote the transcript of all communication between* $\mathcal{F}$ *and all machines except* P*. Then there is a deterministic function* f *(not necessarily efficiently computable) s.t. with overwhelming probability we have* $trans = f(k, trans \setminus P)$*.*

The intuition behind this definition is that if $\mathcal{F}$ is only temporarily secret (OTS) for P, then any secrets that P and $\mathcal{F}$ share may eventually become public. The following lemma gives some examples:

Lemma 8. *Coin toss ($\mathcal{F}_{\mathrm{CT}}$) and CRS ($\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ with any $\mathcal{D}$) are OTS for all parties. Commitment ($\mathcal{F}_{\mathrm{COM}}$) and ZK ($\mathcal{F}_{\mathrm{ZK}}$) are OTS for the recipient/verifier. If G is a key generation algorithm, s.t. the secret key depends deterministically on the public key (e.g., RSA, ElGamal[17]), the PKI $\mathcal{F}_{\mathrm{PKI}}^{G}$ is OTS for all parties.*

Proof. In the case of coin toss and CRS the adversary learns the random value r when if some party learns it, so all communication can be deduced from the communication with the adversary. In the case of commitment and ZK the communication with the recipient/verifier can be deduced from the communication with the sender. (In these cases, the function f is even efficiently computable.) All secret keys chosen by $\mathcal{F}_{\mathrm{PKI}}^{G}$ can be calculated from the public keys $pk_1, \dots, pk_n$ sent to the adversary. □

Using this definition, we can prove that using a CRS, coin-toss or other functionalities that are OTS for the sender, one cannot long-term-UC realise a commitment:

[15] The intuitive reason being that the simulator may choose a value for the CRS which is only *computationally* indistinguishable from the uniform distribution without loosing the unconditional hiding property.

[16] Their definition $\mathcal{F}_{krk}$ of a PKI is somewhat different to ours. However, their proof directly carries over to $\mathcal{F}_{\mathrm{PKI}}$.

[17] Under the condition, that in the secret key, group elements are always given using a *unique* representative (e.g., the secret exponent e in RSA is chosen smaller than $\varphi(n)$). See also Section 4.3.

Theorem 9 (Impossibility of commitment with OTS functionalities). *Let $\mathcal{F}$ be a functionality that is OTS for party C. Then there is no nontrivial protocol that long-term-UC realises commitment with sender C ($\mathcal{F}_{\mathrm{COM}}^{C\to R}$) in the $\mathcal{F}$-hybrid model.*

If one is willing to assume NP $\not\subseteq$ P/poly, this theorem is an immediate consequence of Lemma 18 stating that $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT},C\to R}$ (ZK for SAT with the sender C being the prover) is possible from $\mathcal{F}_{\mathrm{COM}}^{C\to R}$, and Corollary 15 stating that $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT},C\to R}$ cannot be realised using $\mathcal{F}$ (both shown in Section 4). However, in Appendix B.1 we give a direct proof (similar in spirit to that of Theorem 14) for this theorem that does not depend on NP $\not\subseteq$ P/poly.

An interesting corollary from this theorem is that long-term-UC commitments cannot be turned around, i.e. using one (or many) long-term-UC commitments from A to B, one cannot long-term-UC realise a commitment from B to A.

Corollary 10 (Commitments cannot be turned around). *There is no nontrivial protocol long-term-UC realising $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ using any number of instances of $\mathcal{F}_{\mathrm{COM}}^{B\to A}$.*

Proof. Immediate from Lemma 8 and Theorem 9. □

In contrast to the impossibility results above, it is possible to get long-term-UC secure commitments using a ZK functionality:

Lemma 11 (Commitment from ZK). *Assume that a one-way permutation exists. Then there is a nontrivial protocol π that long-term-UC realises $\mathcal{F}_{\mathrm{COM}}^{C\to R}$ (commitment with sender C) and that uses two instances of $\mathcal{F}_{\mathrm{ZK}}^{SAT,C\to R}$ (ZK for SAT with the sender C being the prover).*

The protocol π looks as follows:

- *To commit* to v, the sender C first commits to v using an unconditionally hiding commitment scheme.
- Then C proves (using the first instance of $\mathcal{F}_{\mathrm{ZK}}$) that he knows v and matching unveil information u.[18]
- *To unveil*, the sender C sends v to the recipient and proves (using the second instance of $\mathcal{F}_{\mathrm{ZK}}$) that he knows matching unveil information u.

The long-term-UC security of this protocol stems from the following two facts. Equivocability: the simulator can unveil to any value v' since he controls the second instance of $\mathcal{F}_{\mathrm{ZK}}$. Extractability: Since the sender cannot (efficiently) compute different unveil informations u and u', the message v given to the first instance of $\mathcal{F}_{\mathrm{ZK}}$ must be the same as that used in the unveil phase. Since the simulator controls the first instance of $\mathcal{F}_{\mathrm{ZK}}$, he learns that message v during the commit phase.

The actual proof is given in Appendix B.2.

4 Zero-Knowledge

In the present section we examine to what extend long-term-UC secure zero-knowledge proofs can be implemented using various functionalities. Besides several impossibility results, we also have a quite surprising possibility result (Theorem 16).

[18] I.e., unveil information that would convince the verifier.

4.1 Using OTS functionalities

First, analogous to our investigations concerning commitments in Section 3, we are now going to examine whether long-term-UC secure ZK can be realised using functionalities that are OTS for one of the parties.

Whether long-term-UC realising ZK for some relation R is possible strongly depends on the relation R under consideration. The following definition specifies a class of relations which is going to play an important role in our results:

Definition 12 (Essentially unique witnesses). *A MA-relation R has* essentially unique witnesses *if there is a PPT-algorithm U_R (the* witness unifier*), that has the following properties:*

- *If w is a witness for x, $U_R(1^k, x, w)$ outputs a witness for x with overwhelming probability, formally: for sequences w_k, x_k with $x_k R w_k$ the probability $P(x_k R U_R(1^k, x_k, w_k))$ is overwhelming in k.*
- *If w is a witness for x, the output of $U_R(1^k, x, w)$ is almost independent of w, formally: for sequences w_k^1, w_k^2, x_k with $x_k R w_k^1$ and $x_k R w_k^2$, the families of random variables $U_R(1^k, x_k, w_k^1)$ and $U_R(1^k, x_k, w_k^2)$ are statistically indistinguishable.*

A possible way to interpret the witness unifier is as a statistically witness indistinguishable proof, that simply sends a witness in the clear.

It is most likely that relations without essentially unique witnesses exist:

Lemma 13. *If one-way-functions (secure against* uniform *adversaries) exist,* or *if $NP \not\subseteq P/poly$, then SAT does not have essentially unique witnesses.*

The proof is given in Appendix B.3.

We are now ready to present the first impossibility result concerning long-term-UC secure ZK:

Theorem 14 (Impossibility of ZK with OTS functionalities). *Let R be a MA-relation* without *essentially unique witnesses. Let $\mathcal{F}$ be a functionality that is OTS for party P. Then there is no nontrivial protocol that long-term-UC realises ZK for the relation R with prover P ($\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$) in the $\mathcal{F}$-hybrid model.*

The rough idea of the proof is as follows: Clearly, if π was to be long-term-UC secure, the interaction between prover P and verifier V must be (almost) statistically independent from the witness V received from the environment. Further, a simulator that is able to simulate convincingly in the case of a corrupted prover must be able to extract a witness $\tilde{w}$ from the communication with that prover, which is (almost) statistically independent from the witness w. So in particular, $\tilde{w}$ is (almost) statistically independent from w. Therefore, combining the prover and the simulator into one algorithm, we get an algorithm that given one witness w returns another almost independent one, in other words, a witness unifier in the sense of Definition 12. Therefore R must have essentially unique witnesses, which gives the desired contradiction.

The proof is given in Appendix B.4.

Note that we cannot expect an analogous result in the case that $\mathcal{F}$ is OTS for the verifier V, since commitments are OTS for the recipient and Lemma 18 show that $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$ can be long-term-UC implemented using commitments with the verifier V as recipient.

Combining the results in this section, we get the impossibility of long-term-UC secure ZK for SAT:

Corollary 15. *Let $\mathcal{F}$ be a functionality that is OTS for party P. If one-way-functions (secure against* uniform *adversaries) exist,* or *if $NP \nsubseteq P/poly$, there is no nontrivial long-term-UC secure protocol for ZK with prover P for SAT in the $\mathcal{F}$-hybrid model.*

Proof. Immediate from Lemma 13 and Theorem 14. □

At this point one might ask why our impossibility result needs the restriction to relations without essentially unique witnesses. Would not the following argumentation show that given a, say, coin-toss, there is no long-term-UC ZK protocol π for any nontrivial relation: The simulator is able to extract a witness w from the interaction with the prover. Therefore w must information-theoretically already be "contained" in the interaction. On the other hand, in an interaction between simulator and verifier, the witness w cannot be "contained" in the interaction, since the simulator does not know w. However, since the interaction in both cases must be statistically indistinguishable from the interaction in the uncorrupted case, that latter both "contains" and does not "contain" w, which gives a contradiction. Surprisingly, this intuition is not sound as shows the following possibility result:

Theorem 16 (ZK for Blum-Integers using coin toss). *Assume that a one-way permutation exists. Let $nR(p, q)$ if $n = pq$, p, q prime and $p \equiv q \equiv 3 \bmod 4$. There is a nontrivial protocol using two instances of $\mathcal{F}_{\mathrm{CT}}$ that long-term-UC realises $\mathcal{F}_{\mathrm{ZK}}^{R}$ in the coin toss hybrid model.*

To construct such a protocol, we have to achieve two seemingly contradictory goals simultaneously. If the prover or verifier is corrupted, the simulator may choose the value r the coin-toss functionality returns. First, since the simulator should be able to extract a witness (p, q) (i.e., a factorisation of n in this case) in case of the corrupted prover, the simulator should be able to choose r having a trapdoor X s.t. it is possible to extract (p, q) under knowledge of that trapdoor. However, in the case of long-term-UC the value r should be statistically indistinguishable from uniform randomness. So the trapdoor should be present (but possibly unknown) even if r is chosen randomly. Further, if the verifier is corrupted, the simulator should be able to simulate the proof without knowing a witness. However, since also in this case r is almost uniformly distributed, the trapdoor X is also present. So by finding that trapdoor X we could extract a witness from the proof although the simulator never used that witness in constructing the proof. This can only be realised, if finding the witness can be reduced to finding the trapdoor.

In the case of factoring n, an example for such a trapdoor is the knowledge of random square roots modulo n. Given an oracle that finds square roots modulo n, we can factor

n. So if the trapdoor X consists of the square roots of r (when we consider r as a sequence of integers modulo n) finding the trapdoor is as hard as factoring n, so there is no contradiction in the fact that by finding the trapdoor we can extract a witness (p, q) from an interaction that was produced without knowledge of (p, q).

This leads us to the following simplified version of our protocol:

- The prover sends n to the verifier.
- Prover and verifier invoke the coin-toss. The result r of that coin-toss is considered as a sequence $r_1, \dots, r_k$ of integers modulo n.
- For each i, the prover chooses a random s_i with $s_i^2 = r_i$. It sets $s_i := \perp$ if r_i does not have a square root.[19]
- The prover sends $s_1, \dots, s_k$ to the verifier.
- The verifier checks, whether $s_i^2 = r_i$ for all $s_i \neq \perp$, and whether at least $\frac{1}{5}$ of all $s_i \neq \perp$.

This protocol is not yet a long-term-UC realisation of $\mathcal{F}_{\mathrm{ZK}}^{R}$, since it fails if n is not a Blum-integer, but it will demonstrate the main point. So why is this protocol long-term-UC secure if we guarantee that n is a Blum-integer? First, we see that if prover and verifier are both honest, the verifier will always accept. This is due to the fact that for a Blum-integer n, a random residue is a square with probability at least $\frac{1}{4}$.

Now we consider the case that the *verifier is corrupted.* In this case, the simulator has to produce coin-toss values $r_1, \dots, r_n$ that are indistinguishable from the uniform distribution, and a proof that is statistically indistinguishable from the proof given by the prover. In other words, the simulator needs to simultaneously produce (almost) uniformly distributed $r_1, \dots, r_n$, and for each r_i a random square root s_i modulo n if such s_i exists. Fortunately, if n is a Blum-integer, there is an efficient algorithm Q for choosing such r_i and s_i (Lemma 32). So the simulator can successfully simulate by simply choosing the r_i and s_i using Q. Note that for this, it is vital that the simulator knows n before having to send the coin-toss result $r_1, \dots, r_n$ to the environment. This is why we let the prover send n to the verifier *before* they invoke the coin-toss. In particular, we could not use a CRS here, because then the simulator might have to choose the r_i before the environment sends n to the prover.

Now for the case that the *prover is corrupted.* In this case, the simulator needs to interact with the environment incorporating the prover and to extract the witness (p, q) if the prover's proof would convince the honest verifier. To do this, the simulator again chooses the coin-toss $r_1, \dots, r_n$ using the algorithm Q and therefore knows random square roots $\tilde{s}_i$ of all r_i that are quadratic residues. Now the environment sends s_i to the simulator. The uncorrupted verifier would only accept if at least $k/5$ of these s_i satisfy $s_i^2 = r_i$. Therefore after receiving the s_i from the environment, the simulator knows $k/5$ independently chosen pairs $(s_i, \tilde{s}_i)$ of square roots of r_i. For each such pair the probability of $s_i \not\equiv \tilde{s}_i \bmod n$ is $\frac{1}{2}$ (we ignore the finer detail of non-invertible r_i at this point), and in this case we get a factor of n by evaluating $\gcd(s_i \pm \tilde{s}_i, n)$. This happens with overwhelming probability, so the simulator is successful in extracting a factor and therefore the witness (p, q).

[19] This is feasible given the factorisation of n.

However, the protocol as described so far has a major flaw: If n is not a Blum-integer, the above security proof does not work. So we must ensure that n is in fact a Blum-integer. If the verifier is corrupted, the simulator gets n from the functionality $\mathcal{F}_{\mathrm{ZK}}^{R}$ which ensures (by definition of R) that n is a Blum-integer. So in this case there is no problem. However, if the prover is corrupted, the simulator will have to choose the coin-toss $r_1, \dots, r_n$. If n is not a Blum-integer, he might learn this later on (since he learns (p, q) in case of a successful proof), but then it might already be too late, because the simulator sends the r_i to the environment before the end of the proof (the algorithm Q does not guarantee $r_1, \dots, r_n$ to be (almost) uniformly distributed if n is not a Blum-integer). To overcome this difficulty, we add an additional step to the beginning of the protocol. *Before the coin-toss is invoked*, the prover proves that n is indeed a Blum-integer. If the prover succeeds in this proof, the simulator can use the algorithm Q without danger, otherwise the simulator may abort (since the verifier would have done so, too). However, this introduces the additional difficulty that in case of a corrupted verifier, the simulator has to perform that proof, too, and without knowledge of the witness. To achieve this, we make use of the FLS-technique [FLS99]: Prover and verifier first invoke another instance of the coin-toss functionality (in this case, a CRS would be sufficient, too) and then the prover proves using a statistically witness indistinguishable argument of knowledge to the verifier that either n is a Blum-integer or that he knows a the preimage of the coin-toss t under a one-way permutation f. Then the simulator can simulate this proof by simply choosing $t = f(u)$ for uniform u. Since $f(u)$ is uniformly distributed, this is indistinguishable from what an honest prover knowing the witness would produce. After having successfully performed this first step, prover and verifier proceed with the protocol as described above.

The actual proof for Theorem 16 is given in Appendix B.5.

Actually, we can somewhat strengthen this result and get a long-term-UC secure ZK that does not only show the existence of a factorisation of n, but can also show that the factorisation satisfies some predicate:

Corollary 17. *Assume that a one-way permutation exists. Let X be any predicate that is in BPP. Let R be as in Theorem 16. Let $(p, q)R'(n, x)$ if $(p, q)Rn$ and $X(p, q, n, x)$ evaluates to true. Then there is a protocol using two instanced of $\mathcal{F}_{\mathrm{CT}}$ that realises $\mathcal{F}_{\mathrm{ZK}}^{R'}$ in the coin toss hybrid model.*

Proof. This protocol construction is almost identical to that of Theorem 16. The only difference is the following: Instead of proving that n is a Blum-integer (using the statistically witness indistinguishable argument of knowledge), the prover proves that $n = pq$ is a Blum-integer and that $X(p, q, n, x)$ evaluates to true. The rest of the protocol is unmodified. The security proof is completely analogous. □

Furthermore, given a commitment, long-term-UC secure ZK for any NP-relation is (unsurprisingly) possible:

Lemma 18 (ZK from commitment). *Let R be a NP-relation. Then there is a long-term-UC secure protocol π for ZK with relation R (i.e., $\mathcal{F}_{\mathrm{ZK}}^{R,P \to V}$) using a polynomial number of commitments from prover P to verifier V (i.e., $\mathcal{F}_{\mathrm{COM}}^{P \to V}$).*

Proof. [CF01] gives a UC secure protocol that realises $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$ using $\mathcal{F}_{\mathrm{COM}}^{P\to V}$ where R is the relation for the Hamilton cycle problem. Their result even holds unconditionally (i.e., even when the environment is unlimited *during* the execution of the protocol) and therefore in particular with respect to long-term UC. Since the Hamilton cycle problem is NP-complete, the lemma follows. □

Note that we cannot expect a similar result using commitments from verifier to prover, since $\mathcal{F}_{\mathrm{COM}}$ is OTS for the recipient and thus Theorem 14 applies.

4.2 Using offline functionalities

In the preceding section, we saw that using a coin toss, long-term-UC secure ZK for the factorisation of Blum-integer can be realised. It is therefore a natural question to ask whether something similar is also possible using a CRS, which can be seen as the offline variant of a coin-toss. Unfortunately, the answer is no. To state this result in greater generality, let us first formalise what we mean by an offline functionality.

Definition 19 (Offline functionalities). *We call a functionality $\mathcal{F}$ offline, if it has the following form: When $\mathcal{F}$ runs with parties $P_1, \dots, P_n$, upon its first activation, it chooses values $(c, c_{P_1}, \dots, c_{P_n})$ according to a fixed distribution and sends c to the adversary. When receiving any input from P_i, send c_{P_i} to P_i.*

Lemma 20. *CRS and PKI are offline functionalities.*

Proof. For $\mathcal{F}_{\mathrm{CRS}}$, set $c := c_i := r$ (cf. Definition 2), and for $\mathcal{F}_{\mathrm{PKI}}$, set $c := (pk_1, \dots, pk_n)$ and $c_i := (sk_i, pk_1, \dots, pk_n)$ (cf. Definition 4). □

The following result shows that a CRS as well as a PKI where the secret key is information-theoretically determined by the public key (cf. Lemma 8) cannot be used for long-term-UC secure ZK for any relation R unless that relation is trivial for nonuniform algorithms anyway.

Theorem 21 (Impossibility of ZK with OTS offline functionalities). *Let R be a* nonuniformly deterministically *nontrivial MA-relation.*[20] *Let $\mathcal{F}$ be an* offline *functionality that is OTS for party P and for party V. Then there is no nontrivial protocol that long-term-UC realises ZK for relation R with prover P and verifier V (i.e., $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$) in the $\mathcal{F}$-hybrid model.*

To understand the proof idea, assume that $\mathcal{F}$ is a CRS. Assume that there is a protocol π for $\mathcal{F}_{\mathrm{ZK}}^{R}$. Then there is a simulator $\mathcal{S}_1$ that is able to choose the CRS r_1 and calculate a corresponding trapdoor T_1, s.t. he can simulate the prover and convince the verifier using this trapdoor (without knowledge of a witness). Furthermore, there is another simulator $\mathcal{S}_2$ that is able to choose the CRS r_2 and calculate a corresponding trapdoor T_2, s.t. he can simulate the verifier and — if the verifier accepts — extract

[20] I.e., there is no nonuniform deterministic polynomial-time algorithm that finds witnesses for R.

a witness w. Since both r_1 and r_2 are statistically indistinguishable from an honestly chosen CRS, it follows that an honestly chosen CRS always already "contains" such trapdoors T_1 and T_2 (however, given a CRS it can be infeasible to find these trapdoors). Therefore, if we provide $\mathcal{S}_1$ and $\mathcal{S}_2$ with a CRS and with trapdoors T_1 and T_2, $\mathcal{S}_1$ will be able to produce a convincing proof (due to trapdoor T_1), and $\mathcal{S}_2$ will be able to extract a witness from this convincing proof. Since $\mathcal{S}_1$ and $\mathcal{S}_2$ are polynomial-time, and CRS and trapdoors can be given as an auxiliary input, it follows that a nonuniform polynomial-time algorithm can find witnesses for R in contradiction to the nontriviality of R. Functionalities other than a CRS are handled almost identically, see the full proof.

The full proof is given in Appendix B.6.

A natural question arising in this context is whether this impossibility result can be made stronger. In particular, one might ask whether such an impossibility result already holds if $\mathcal{F}$ is OTS for P *or* for V. This however is refuted by Corollary 24 below. Further one might ask, whether the theorem can be strengthened to state impossibility of ZK for *uniformly nontrivial* relations. The following gives strong evidence that this cannot be done without new results about integer-factorisation.

Corollary 22. *Let γ be an efficiently computable function from Σ^* to $\mathbb{N} \cup \{\bot\}$, s.t. $\gamma(x)$ depends only on the length of x, and $\gamma(x)$ is a Blum-Integer or $\bot$ for all x. Let R be as in Theorem 16. Let $(n,x)R_\gamma(p,q)$ iff $nR(p,q)$ and $\gamma(x) = n$. Then there is a protocol that long-term-UC realises $\mathcal{F}_{\mathrm{ZK}}^{R_\gamma}$ with prover P in the CRS-hybrid model.*

It is not an unreasonable (although strong) assumption that such an γ exists, s.t. R_γ is uniformly nontrivial. So to strengthen Theorem 21 one would have to disprove the existence of such a γ.

The rough proof idea for Corollary 22 is the following: Recall, why protocol π from Theorem 16 needs a coin-toss instead of a CRS. The simulator had to choose the value $r = (r_1, \dots, r_k)$ of the second invocation of the coin-toss functionality in a manner so that it new the square roots of r_i modulo n. Therefore, it was necessary for the simulator to know n before choosing r. In the case of R_γ however, there are only polynomially many $n = \gamma(x)$ since $\gamma(x)$ depends only on the length of x. So we can modify the protocol π as follows: Instead of using coin-toss, we use a different CRS $r^{(|x|)} = (r_1^{|x|}, \dots, r_k^{|x|})$ for each length $|x|$. Then the simulator can choose the CRS $r^{(|x|)}$ before the start of the protocol, since the n for which the CRS $r^{(|x|)}$ is to be used is already known ($n = \gamma(0^{|x|})$).

The proof is given in Appendix B.7.

4.3 Using a PKI

Lemma 8 tells us that at least for some commonly used encryption schemes, $\mathcal{F}_{\mathrm{PKI}}^{G}$ is OTS for all parties (here and in the following G denotes the key generation algorithm) and therefore cannot be used for long-term-UC realising commitment or zero-knowledge[21]. However, in general this is not necessarily the case. So the question arises

[21] Except for nonuniformly trivial relations, see Theorem 21.

whether there are encryption schemes so that $\mathcal{F}_{\mathrm{PKI}}^{G}$ can be used to realise, say, a commitment. In this section, we specify an encryption scheme (or more to the point, its key generator G) and give a protocol that using $\mathcal{F}_{\mathrm{PKI}}^{G}$ implements a commitment. Surprisingly, the encryption scheme we use is not a pathological construction, but a relatively natural variant of ElGamal in the RSA group. So we cannot expect a generalisation of Theorems 9 and 21 that covers all PKIs with "natural" encryption schemes.

The ElGamal-Variant we consider has the following key generation algorithm G_{amal}: Upon input 1^k, G_{amal} chooses a random n of length k as the product of two safe primes. Further it chooses a random $x \in \{0, \dots, 2^{2k}\}$,[22] and a random invertible $g \in \mathbb{Z}_n$ (note that then with overwhelming probability g has high order). Then it outputs the secret key (n, g, x) and the public key (n, g, g^x).

Lemma 23. *Assume that factoring the product of two random safe primes is hard (w.r.t. nonuniform adversaries). Then there is a protocol π using one instance of $\mathcal{F}_{\mathrm{PKI}}^{G_{amal}}$ that long-term-UC realises $\mathcal{F}_{\mathrm{COM}}^{C \to R, 1}$ (a 1-bit commitment from C to R).*

The protocol π is quite simple:

- Let (n, g, x) be C's secret key and (n, h, x) the corresponding public key (as provided by $\mathcal{F}_{\mathrm{PKI}}^{G_{amal}}$).
- To commit to a bit $b \in \{0, 1\}$, the sender C sends $c := x + b \bmod 3$ to the recipient R.
- To unveil b, the sender C sends (b, x) to the recipient R. The recipient R checks, that $x + b \equiv c \bmod 3$ and that $h \equiv g^x \bmod n$.

The rough intuition behind this protocol is the following: The protocol is binding, because it is hard to find an $x' \neq x$ satisfying $h = g^{x'} \bmod n$ without knowledge of the factorisation of n. The protocol is unconditionally hiding, because there are many different x' with length $2k$ satisfying $h = g^{x'} \bmod n$, and for a random such x', $x' \bmod 3$ is almost equally distributed on $\{0, 1, 2\}$ (note that this does not hold modulo 2, since $2 \mid \varphi(n)$). The scheme is equivocable (i.e., the simulator can choose b after committing), since the simulator knows the factorisation of n, and therefore can choose a random x' with $g^{x'} \equiv h \bmod n$ and $b + x' \equiv c \bmod 3$. The scheme is extractable (i.e., the simulator can learn the bit b before unveil), since the simulator knows the x that will be sent by C and thus calculates $b := c - x \bmod 3$.

That the protocol actually long-term-UC realises $\mathcal{F}_{\mathrm{COM}}^{1}$ is proven in Appendix B.8.

Note that the protocol given here only shows that we cannot expect a generalisation of Theorem 9 to general PKIs, it does not show that it is practicable to use PKIs for implementing long-term-UC secure commitments. The reason for this is that during the unveil phase, the secret key is transmitted and the PKI thus rendered useless for further use. In contrast, the next section presents functionalities that can be used for an arbitrary number of commitments/ZK-proofs.

We additionally get the following:

[22] This is probably the most uncanonical choice in our construction, since an x of length $|n|$ would be fully sufficient.

Corollary 24. *Assume that factoring the product of two random safe primes is hard (w.r.t. nonuniform adversaries), and let A and B be two parties. Then there is an offline functionality $\mathcal{F}$ that is OTS for B, s.t. there are long-term-UC protocols using $\mathcal{F}$ for: commitment with recipient B ($\mathcal{F}_{\mathrm{COM}}^{A\to B,m}$), zero-knowledge for any NP-relation R with prover A ($\mathcal{F}_{\mathrm{ZK}}^{A\to B}$), m-bit coin-toss ($\mathcal{F}_{\mathrm{CT}}^{m}$) and zero-knowledge for some nonuniformly nontrivial NP-relation R with prover B ($\mathcal{F}_{\mathrm{ZK}}^{R,B\to A}$).*

Proof. Let $\mathcal{F}$ consist of m copies of $\mathcal{F}_{\mathrm{PKI}}^{G_{amal}}$.[23] Since the protocol π from Lemma 23 does not uses the recipient's secret key, we can assume that $\mathcal{F}$ chooses a public/secret key pair only for A, so that $\mathcal{F}$ is OTS for B. Then using π we can implement m instances of $\mathcal{F}_{\mathrm{COM}}^{A\to B,1}$. From this, $\mathcal{F}_{\mathrm{COM}}^{A\to B,m}$ can be trivially realised. Further, by Lemma 18 we get $\mathcal{F}_{\mathrm{ZK}}^{A\to B}$ from sufficiently many instances of $\mathcal{F}_{\mathrm{COM}}^{A\to B,1}$. From $\mathcal{F}_{\mathrm{COM}}^{A\to B,m}$ we easily get $\mathcal{F}_{\mathrm{CT}}^{m}$.[24] By Theorem 16 we get $\mathcal{F}_{\mathrm{ZK}}^{R,B\to A}$ for the NP-relation R from Theorem 16. Since by assumption factoring the product of two random safe primes is hard, R is nonuniformly nontrivial. □

5 Other setup-assumptions

As the preceding sections have shown, trying to design long-term-UC secure protocols using a CRS, coin toss or PKI is a futile endeavour. Therefore, in the following sections we will investigate alternative setup-assumptions that are more fruitful in the context of long-term-UC.

5.1 Trusted devices implementing a random oracle

A very powerful assumption in the context of universally composable security is the random oracle. It may therefore seem worthwhile to investigate whether a random oracle can be used to realise long-term-UC secure commitment and ZK. However, a closer look shows that in the context of long-term-UC security the random oracle is a very unrealistic assumption due to the following fact: Real-life implementations of the random oracle have to be done via some efficiently computable function (e.g., using trusted hardware that calculates some pseudorandom function with a secret seed). In the context of long-term-UC, this function could be "broken" by an unlimited adversary after protocol execution. In contrast, a random oracle functionality ensures, that even for an unlimited adversary, the function looks completely random. Therefore, we advocate that in the context of long-term-UC, instead of a random oracle one should use a functionality that evaluates a pseudorandom function with a secret seed (representing e.g. a (temporarily) trusted device).

We now give a definition of such a functionality $\mathcal{F}_{\mathrm{TPF}}$. Note however, that all possibility results given in this section also hold (with identical proofs) when using a random oracle instead of $\mathcal{F}_{\mathrm{TPF}}$.

[23] I.e., m public/secret key pairs are generated for each party.

[24] A commits to a random string r' of length m. B sends a random string r'' to A. A unveils. $r' \oplus r''$ is the result of the coin-toss.

Definition 25 (Trusted pseudorandom function (TPF)). *Let f_s be an efficiently computable family of deterministic functions $f_s : \{0,1\}^{l(|s|)} \to \{0,1\}^{l(|s|)}$ with polynomially bounded l.*

Then, the functionality trusted pseudorandom function (TPF) $\mathcal{F}^f_{\mathrm{TPF}}$ *is defined as follows: Upon its first activation, it chooses a uniformly random $s \in \{0,1\}^k$. When receiving a message $x \in \{0,1\}^{l(k)}$ from a party P or the adversary, it sends $f_s(x)$ to P or the adversary, respectively.*

At this point, one should note that the UC definition (and therefore our variant, too) implicitly assumes that when using a TPF, that TPF is accessed only by the protocol (and the adversary), but that it cannot be directly accessed by the environment. This in particular rules out that different protocols share a single TPF. A more detailed analysis of the consequences of this assumption can be found in [HMQU05, CDPW07]. However, we show that using a single TPF we can perform an *arbitrary number* of zero knowledge arguments or commitments, so that at least we do not need a large number of TPFs when constructing a larger protocol that performs many ZK arguments or commitments.

Theorem 26 (ZK from TPF). *Assume that a one-way permutation exists. Let f_s be a pseudorandom function (as in [Gol01]), and R an NP-relation. Then there is a nontrivial protocol π using one instance of $\mathcal{F}^f_{\mathrm{TPF}}$ that long-term-UC realises unlimited number of instances of $\mathcal{F}^R_{\mathrm{ZK}}$ (i.e., ZK for the relation R).*

We give the proof idea here. First a commitment scheme is constructed which is computationally binding, unconditionally hiding and extractable (however, this commitment is not necessarily UC). The extractable commitment is constructed from a given commitment which is unconditionally hiding. To commit to a value v one first commits to $v, f_s(v)$. Then one commits to $u, f_s(u)$ where u is the unveil information for the first commitment. As the function $f_s(.)$ can only be evaluated by using the functionality $\mathcal{F}_{\mathrm{TPF}}$ a simulator can extract the committed value v from the calls which are placed to $\mathcal{F}_{\mathrm{TPF}}$.

Using this extractable commitment we modify the zero knowledge protocol for graph-3-colourability of [GMW91]. Instead of letting the prover commit to a colouring and then let the verifier choose a random edge e for which the colours are unveiled and checked we let the verifier commit to e before the prover commits to the colouring.

In this protocol the simulator can, if the prover is corrupted, extract a witness from the commitments of the simulated real adversary or the protocol will fail and is then easily simulated. In the case of a corrupted verifier the simulator can extract the edge which will later be investigated before committing to the colouring. So the simulator can easily commit to a fake colouring and still pass the test at the edge in question.

In both cases the communication between the parties, the adversary and the environment are statistically indistinguishable in the real protocol and in this simulation and we achieve a long-term-UC zero knowledge argument for graph-3-colouring and hence for all NP-statements. The complete proof can be found in Appendix B.9.

According to Lemma 11 one commitment can be obtained from two invocations of a zero knowledge scheme and we can hence conclude:

Corollary 27 (Commitments from TPF). *Assume that a one-way permutation exists. Let f_s be a pseudorandom function. Then there is a nontrivial protocol π using one instance of $\mathcal{F}_{\mathrm{TPF}}^{f}$ that long-term-UC realises an unlimited number of instances of $\mathcal{F}_{\mathrm{COM}}$ (i.e., commitments).*

Proof. Immediate from Lemma 11 and Theorem 26. □

5.2 Signature cards

One disadvantage of the TPF-assumption from the foregoing section is that trusted hardware implementing a pseudorandom function are unlikely to be available for practical use.[25] However, another kind of trusted device is already available commercially today: the signature card. A signature card is a tamperproof device with an built-in secret key. Upon request, this card signs an arbitrary document, but *never* reveals the secret key. The corresponding public key can be obtained from some certification authority. These properties are required e.g. from the German signature law [Sig01].

These properties are captured by the following ideal functionality (based on [HMQU05]):

Definition 28 (Signature Card (SC)). *Let $\mathfrak{S} = (\mathit{KeyGen}, \mathit{Sign}, \mathit{Verify})$ be a signature scheme. Let H be a party. Then the functionality $\mathcal{F}_{\mathrm{SC}}^{H,\mathfrak{S}}$ (signature card for scheme $\mathfrak{S}$ with holder H) behaves as follows: Upon the first activation, $\mathcal{F}_{\mathrm{SC}}^{H,\mathfrak{S}}$ chooses a public/secret key pair (pk, sk) using the key generation algorithm $\mathit{KeyGen}(1^k)$. Upon a message (pk) from a party P or the adversary, send pk to that party or the adversary, resp. Upon a message (sign, m)* from the holder *H, produce a signature σ for m using the secret key sk and send σ to H.*[26]

As was the case with TPFs, our definition implicitly assumes that the environment has no direct access to the signature card. See the discussion after Definition 25. However, in [HMQU05] techniques where introduced that allow to share a single signature card in different protocols. It would be interesting to explore whether their approach can also be applied to our scenario.

It was shown in [HMQU05] that signature cards are powerful assumptions in the context of universal composability. Using an adaption of their technique, we can show that these signature cards are also very useful for long-term-UC security:

Theorem 29 (ZK from a signature card). *Assume that a one-way permutation exists. Let $\mathfrak{S}$ be an EF-CMA secure signature scheme. Let R be any MA-relation. Then there is a nontrivial protocol π that long-term-UC realises an unbounded number of instances of $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$ (i.e., ZK for the relation R with prover P) using a single instance of $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ (i.e., a signature card for $\mathfrak{S}$ with P as the holder).*

[25] Not because of technical difficulties, but simply and plainly due to the forces of supply and demand.

[26] The definition from [HMQU05] additionally provides the possibility of locking the card (called *seize* and *release* there). These however are not needed in our protocols, so we omit them.

The idea of the proof is as follows: To prove the existence of a witness w for some statement x, the prover P signs x using his signature card (resulting in a signature σ) and then performs a statistically witness indistinguishable argument of knowledge that one of the following holds: (i) he knows a w and a σ, so that xRw and σ is a valid signature for w, or (ii) he knows a secret key sk' matching the public key pk provided by the signature card functionality.

Consider the case of a corrupted prover. Since $\mathfrak{S}$ is EF-CMA secure, it is infeasible to get a secret key sk' matching the public key pk chosen by the signature card (since the signature card allows only black-box access to the signing algorithm). So the prover has to show the knowledge of a signature σ of the witness w. The only way to obtain such a signature σ is to sign the witness w using the signature card. Since in the ideal model, the signature card $\mathcal{F}_{\mathrm{SC}}$ is simulated by the simulator, the simulator learns that witness w. So the simulator is able to extract w while honestly simulating verifier and $\mathcal{F}_{\mathrm{SC}}$.

In case the verifier is corrupted, the simulator knows the secret key sk matching the public key pk. So the simulator can prove (ii) instead of (i). Since the proof system we use is statistically witness indistinguishable, the resulting interaction is statistically indistinguishable.

The full proof is given in Appendix B.10.

Corollary 30 (Commitments from a signature card). *Assume that a one-way permutation exists. Let $\mathfrak{S}$ be an EF-CMA secure signature scheme. Then there is a nontrivial protocol π that long-term-UC realises an unbounded number of instances of $\mathcal{F}_{\mathrm{COM}}^{C\to R}$ (i.e., commitment with sender C) using a single instance of $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ (i.e., a signature card for $\mathfrak{S}$ with P as the holder).*

Proof. This is an immediate consequence of Theorem 29 and Lemma 11. □

6 Conclusions

We have examined the notion of long-term UC which allows to combine the advantages of long-term security (i.e., security that allow for unlimited adversaries after protocol end) and Universal Composability. We saw that the usual set-up assumptions used for UC protocols (e.g., CRS) are not sufficient any more in the case of long-term UC. However, we could show that there are other practical alternatives to these setup-assumptions (e.g., signature cards) that allow to implement the important primitives commitments and zero-knowledge proofs.

Further research in this directions might include the following:

- Which protocol tasks can or cannot be long-term-UC realised using commitments and zero-knowledge proofs.
- What other setup-assumptions might be useful in the context of long-term UC. In particular, under which assumptions can OT (and therefore any functionality) be realised?
- Our investigations were in the secure-channels communication-model. If only authenticated channels are present, the important issue of key exchange occurs. What setup-assumptions are necessary to implement the latter?

- The protocols presented here were not optimised for efficiency. To what extend can efficient protocols be found for the tasks discussed in this work?
- In [HMQU05] techniques were presented that allow to share a single signature card between different protocols. Can these techniques be applied to our setting, too?
- Much work on unconditional and long-term security has been done in the field of quantum cryptography. How does long-term UC behave in the presence of quantum communication. Can some of the impossibility results given in this work be avoided? In particular, quantum communication could solve the problem of key exchange mentioned above.

Acknowledgements. We thank the anonymous referees for many helpful suggestions.

A Auxiliary lemmas

Lemma 31. *Let R be a MA-relation. Let A be an PPT-algorithm, and $\mathcal{D}_k$ a family of distributions over strings of polynomial length (not necessarily an efficiently samplable one). Let $P > 0$. Assume that for sufficiently large $x \in L_R$, $A(x, \mathcal{D}_{|x|})$ outputs some witness w with xRw with probability at least P.*

Then there is a nonuniform deterministic polynomial-time algorithm $\tilde{A}$ that upon input x outputs a witness w with xRw.

Proof. W.l.o.g. we assume that all $x \in \{0,1\}^*$.

From A, we can construct a deterministic polynomial-time algorithm that takes its random tape as input, i.e., for sufficiently long $x \in L_R$, $A(x, \mathcal{D}_{|x|}, \mathcal{T})$ outputs a witness with probability at least P if $\mathcal{T}$ is the uniform distribution on strings polynomial in $|x|$. Further, since there are only finitely many x that are not solved with probability at least P, we can assume that A' solves these by table-lookup.

We can amplify the probability of yielding a witness by repeating A', so there is an deterministic polynomial-time algorithm $\tilde{A}$ and a family of distributions $\mathcal{E}_k$ of strings of polynomial length $p(k)$ (constructed as sufficiently many copies of $\mathcal{D}_k, \mathcal{T}$, padded to length $p(k)$), s.t. $\tilde{A}(x, \mathcal{E}_{|x|})$ outputs a witness w for x with probability greater than $1 - 2^{-|x|}$.

Let $G(x,e) := 1$ iff $xR\tilde{A}(x,e)$. Let $L_k := L_R \cap \{0,1\}^k$. For each k, let $e_k \in \{0,1\}^{p(k)}$ be the string maximising $P(G(x, e_k) = 1)$ for randomly chosen $x \in L_k$. For contradiction, we assume that there is an $x_k \in L_k$, s.t. $G(x_k, e_k) \neq 1$. Then for random $x \in L_k$ and $e \leftarrow \mathcal{E}_k$ we would have

$$\begin{aligned}
2^{-n} &> P(G(x,e) \neq 1) \\
&= \sum_{e' \in \{0,1\}^{p(k)}} P(e = e') P(G(x,e') \neq 1) \\
&\geq \sum_{e' \in \{0,1\}^{p(k)}} P(e = e') P(G(x,e_k) \neq 1) \\
&\geq \sum_{e' \in \{0,1\}^{p(k)}} P(e = e') P(x = x_k) P(G(x_k,e_k) \neq 1) \\
&\geq \sum_{e' \in \{0,1\}^{p(k)}} P(e = e') \cdot 2^{-\#L_k} \cdot 1 \\
&= 2^{-\#L_k} \geq 2^{-n}.
\end{aligned}$$

So for all $x \in L_k$, $G(x, e_k) = 1$, i.e., for all $x \in L_k$, $\tilde{A}(x, e_k)$ gives a witness for x. But $\tilde{A}(\cdot, e_k)$ is a deterministic nonuniform polynomial-time algorithm with auxiliary input e_k, which concludes the proof. □

Lemma 32. *There is a PPT-algorithm Q s.t. $U(1^k, n)$ outputs two values $r \in \{0, \dots, 2^{k|n|} - 1\}$, $s \in \{0, \dots, n-1\} \cup \{\perp\}$, s.t. the following holds if n is a Blum-integer*

- *The distribution of r is almost uniformly distributed on $\{0, \dots, 2^{k|n|} - 1\}$.*[27]
- *If r is a quadratic residue mod n, then s is a almost uniformly distributed root of r modulo n (and $s = \perp$ otherwise).*

Proof (of Lemma 32). First, we remember some facts: The Legendre-symbol $\left(\frac{a}{p}\right)$ for prime p is defined as $\left(\frac{a}{p}\right) = +1$ if a is a quadratic residue modulo p, $\left(\frac{a}{p}\right) = -1$ if a is a quadratic non-residue modulo p, and $\left(\frac{a}{p}\right) = 0$ if $a \equiv 0 \bmod p$. The Jacobi-symbol $\left(\frac{a}{pq}\right)$ for different primes p, q is defined as $\left(\frac{a}{pq}\right) := \left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$. If $n = pq$ is a Blum-integer and not a square, -1 is a quadratic non-residue modulo p, modulo q and modulo n. There is an efficient algorithm R, so that $R(p, q, r)$ returns a random square root of r modulo pq (or $\perp$ if r is not a square) if p, q are primes.

We now define an auxiliary algorithm Q' as follows:

- Input: a Blum-integer $n = pq$ with $p \neq q$, and an $r_0 \in \mathbb{Z}_n$.
- Calculate the Jacobi symbol $J := \left(\frac{r_0}{n}\right)$.
- If $J = -1$, output $(r_0, \perp)$.
- If $J = 0$ and $r_0 \neq 0$, factor n.[28] Output $(r_0, R(p, q, r_0))$.
- If $r_0 = 0$, return $(0, 0)$.
- If $J = +1$, choose a uniformly random invertible $s \in \mathbb{Z}_n$,[29] and with probability $\frac{1}{2}$, output (s^2, s), otherwise $(-s^2, \perp)$.

Let $n = pq$ with $p \neq q$.

Let $S_1 \subseteq \mathbb{Z}_n$ be the set of all r_0 with $\left(\frac{r_0}{n}\right) \in \{-1, 0\}$. Then given a Blum-integer n, for uniformly chosen $r_0 \in S_1$ and $(r, s) \leftarrow Q'(n, r_0)$, we have that r is uniformly distributed on S_1 (since $r = r_0$ for $r_0 \in S_1$). If r_0 is not a square, $s = \perp$ (since 0 is a square, and by the definition of R). If r_0 is a square, s is a uniformly distributed root of r (since 0 has only one root, and by the definition of R).

Let $S_2 \subseteq \mathbb{Z}_n$ be the set of all r_0 with $\left(\frac{r_0}{n}\right) = +1$. All elements of S_2 are invertible. Let further Q be the set of all invertible squares in $\mathbb{Z}_n$. Then $Q \subseteq S_2$, and $S_2 \setminus Q$ is the set of all invertible elements that are neither quadratic residues modulo p nor modulo q. For a uniformly random invertible $s \in \mathbb{Z}_n$, s^2 is uniformly distributed over Q, and s is a uniformly random root of s^2. Since also -1 has that property (see above), multiplying an element of Q with -1 gives an element of $S_2 \setminus Q$ and vice versa. So $\#Q = \#(S_2 \setminus Q)$ and $-s^2$ is uniformly distributed on $S_2 \setminus Q$.

Therefore for any $r_0 \in S_2$, $Q'(n, r_0)$ outputs (r, s), s.t. r is uniformly distributed on S_2, and if r is a square, s is a uniformly chosen root (and $s = \perp$ otherwise).

It follows that for uniformly chosen $r_0 \in \mathbb{Z}_n$, $Q'(n, r_0)$ outputs (r, s), s.t. r is uniformly distributed on $\mathbb{Z}_n$, and if r is a square, s is a uniformly chosen root (and $s = \perp$ otherwise).

Now we define algorithm Q'':

[27] I.e., the distribution of r is statistically indistinguishable (in k) from the uniform distribution on $\{0, \dots, 2^{k|n|} - 1\}$.

[28] This can be done efficiently, since if $\left(\frac{r_0}{n}\right) = 0$, $\gcd(r_0, n)$ is a factor of n.

[29] How to do this is discussed later.

- Input: a Blum-integer n.
- Check whether n is a square. If so, let $p, q := \sqrt{n}$, choose a random $r \in \mathbb{Z}_n$ and output $(r, R(p, q, r))$.
- Otherwise, choose a random $r_0 \in \mathbb{Z}_n$, let $(r, s) \leftarrow Q'(n, r_0)$ and output (r, s).

Obviously, if n is a Blum-integer (possibly with identical prime factors), $Q''(n)$ outputs (r, s), s.t. r is uniformly distributed on $\mathbb{Z}_n$, and if r is a square, s is a uniformly chosen root (and $s = \bot$ otherwise).

Now, consider the following algorithm Q:

- Input: a parameter 1^k and a Blum-integer n.
- Let $(r, s) \leftarrow Q''(n)$.
- Let $\bar{r} \in \{0, \dots, n-1\}$ be a representative of $r \in \mathbb{Z}_n$.
- Let $d := \lfloor 2^{k|n|}/n \rfloor$, and choose a uniformly random $e \in \{0, \dots, d-1\}$.
- Return $(\bar{r} + en, s)$.

Obviously, for uniform $r \in \mathbb{Z}_n$, $\bar{r}+en$ is almost uniformly distributed on $\{0, \dots, 2^{k|n|} - 1\}$. So if n is a Blum-integer, $Q(k, n)$ outputs (r, s), s.t. r is almost uniformly distributed on $\{0, \dots, 2^{k|n|} - 1\}$, and if r is a square, s is a uniformly chosen root (and $s = \bot$ otherwise), so Q has the properties stated in the lemma.

However, algorithm Q' (which again is called by Q) contains the instruction "choose a uniformly random invertible $s \in \mathbb{Z}_n$". We have to check whether we can do this efficiently (with some error probability negligible in k). If n is a Blum-integer with different prime factors p, q, a random element s is invertible if it is nonzero modulo p and modulo q. Since $p, q \geq 3$, the probability for this is at least $(\frac{2}{3})^2$. So we can choose an invertible $s \in \mathbb{Z}_n$ by choosing random $s \in \mathbb{Z}_n$ and check whether it is invertible (e.g., using Euclid's algorithm). If we repeat this up to k times, the probability of failure is negligible. □

Lemma 33. *Let U, $\tilde{U}$, L, $\tilde{B}$ be interactive machines that send only a polynomially-bounded number of messages. Let $\langle U, L\rangle^{k,z}$ denote the transcript of the communication in interaction of U and L where both machines get input k, z. Assume*

$$\langle U, L\rangle^{k,z} \approx \langle \tilde{U}, L\rangle^{k,z} \approx \langle U, \tilde{L}\rangle^{k,z}$$

where $\approx$ denotes statistical indistinguishability (in k). Then

$$\langle U, L\rangle^{k,z} \approx \langle \tilde{U}, \tilde{L}\rangle^{k,z}.$$

Proof. In the following, we omit k, z for readability. W.l.o.g. we can assume, that in a run of $\langle U, L\rangle$ the machines alternatingly send messages to each other, with the first message sent by U. Analogously for the other networks. Let $U(v_i)$ denote the distribution of message sent by machine U under the condition that the communication has been v_i so far. Note that this distribution does not depend on which other machine U is communicating with. Define $L, \tilde{U}, \tilde{L}$ analogously. Let $\langle U, L\rangle_i$ denote the communication of U and L up to the i-th message. Then if e.g., i is odd, $U(\langle U, L\rangle_{i-1})$ has the same distribution as $\langle U, L\rangle_i$. If i is even, the same holds for L (since U sends the odd and L the even messages). Let $p(k)$ be the polynomial upper bound on the number of messages sent by the machines.

Let $\varDelta$ denote the statistical distance of random variables, and let

$$u_i := \varDelta(\langle U,L\rangle_i, \langle \tilde U, L\rangle_i), \qquad l_i := \varDelta(\langle U,L\rangle_i, \langle U, \tilde L\rangle_i) \qquad \text{and} \qquad d_i := \varDelta(\langle U,L\rangle_i, \langle \tilde U, \tilde L\rangle_i)$$

Further, $\delta := \max\{u_{p(k)}, l_{p(k)}\}$. By assumption, δ is negligible in k. To show the lemma, it is sufficient to show that $d_{p(k)}$ is negligible, too. Assume that i is odd (i.e. it is the U's or $\tilde U$'s turn to send a message).

$$\begin{aligned}
d_i &= \varDelta\big(U(\langle U,L\rangle_{i-1}), \tilde U(\langle \tilde U, \tilde L\rangle_{i-1})\big)\\
&\le \varDelta\big(U(\langle U,L\rangle_{i-1}), \tilde U(\langle \tilde U, L\rangle_{i-1})\big) + \varDelta\big(\tilde U(\langle \tilde U,L\rangle_{i-1}), \tilde U(\langle U, L\rangle_{i-1})\big)\\
&\qquad\qquad + \varDelta\big(\tilde U(\langle U,L\rangle_{i-1}), \tilde U(\langle \tilde U, \tilde L\rangle_{i-1})\big)\\
&\le \varDelta\big(\langle U,L\rangle_i), \langle \tilde U, L\rangle_i\big) + \varDelta\big(\langle \tilde U,L\rangle_{i-1}, \langle U, L\rangle_{i-1}\big) + \varDelta\big(\langle U,L\rangle_{i-1}, \langle \tilde U, \tilde L\rangle_{i-1}\big)\\
&= u_i + u_{i-1} + d_{i-1} \le 2\delta + d_{i-1}.
\end{aligned}$$

An analogous calculation (with $L(\dots)$ and $\tilde L(\dots)$ instead of $U(\dots)$ and $\tilde U(\dots)$, and with $\langle U, \tilde L\rangle$ instead of $\langle \tilde U, L\rangle$) gives $d_i \le l_i + l_{i-1} + d_{i-1} \le 2\delta + d_{i-1}$. Since obviously $d_0 = 0$, we have $d_{p(k)} \le 2p(k)\delta$ which is negligible, since δ is negligible and p polynomial. □

B Postponed proofs

B.1 Proof of Theorem 9

Proof (of Theorem 9). For this proof, let us first introduce some notation. If $A_{k,z}$ and $B_{k,z}$ are families of random variables, we write $A \lhd B$, if there is some probabilistic function G (not necessarily an efficiently computable one) s.t. $A_{k,z}$ and $G(k, B_{k,z})$ are statistically indistinguishable. Note that G knows k, but does not have direct access to z. (Intuitively $A \lhd B$ means, that A does not contain (noticeably) more information about z than B). Obviously, $\lhd$ is transitive. We will investigate different networks of machines (cf. Figure 1). To facilitate calculation, we use the following notation: $com^{k,z}_X(AB, CD, \dots)$ denotes the transcript of the communication between machines A and B, between machines C and D etc. in a run of the network X on security parameter k when the environment gets auxiliary input z. E.g., $com^{k,z}_{\text{II}}(R\mathcal{Z}_C, R\tilde{A}_C, R\mathcal{F})$ denotes all communication of party R in network II.

To produce a contradiction, we assume that there is a nontrivial protocol π that long-term-UC realises $\mathcal{F}^{C\to R,1}_{\text{COM}}$ (i.e., one-bit commitment with sender C and recipient R). First, consider the following network I (depicted in Figure 1, the adversary $\tilde{\mathcal{A}}$ has been omitted for simplicity): The uncorrupted sender C and recipient R run together with the environment $\mathcal{Z}_0$ and the dummy-adversary $\tilde{\mathcal{A}}$.[30] The environment $\mathcal{Z}_0$ behaves as follows: It takes an auxiliary input of the form b or (b, unveil) where $b \in \{0,1\}$. Then it sends b to the sender C (i.e., instructs C to commit to b) and waits for the $(\mathit{committed})$-message from the recipient R. If the auxiliary input was of the form (b, unveil), it then

[30] A dummy-adversary is an adversary, that forwards all messages to the environment, and follows all instructions given by that environment.

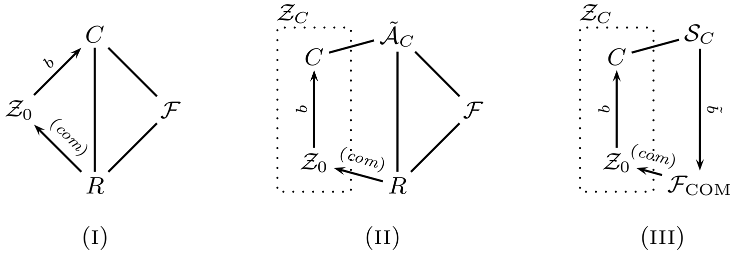


Fig. 1. Networks from the proof of Theorem 9.

sends (*unveil*) to C and waits for the bit $\tilde{b}$ sent by the recipient. During the protocol run, it instructs the dummy-adversary $\tilde{\mathcal{A}}$ to deliver all messages. We assume that all environments constructed in this proof simply output their view (i.e., the transcript of all messages they sent or got and of all their internal states).

Since π is long-term-UC secure, for auxiliary input b (i.e., in the case that $\mathcal{Z}_0$ does not instruct C to unveil) the communication observed by the adversary $\tilde{\mathcal{A}}$ and the recipient R is *statistically* indistinguishable in the cases $b = 0$ and $b = 1$, i.e.,

$$com_{\mathrm{I}}^{k,0}(\mathcal{Z}_0\tilde{\mathcal{A}}, CR, R\mathcal{F}) \approx com_{\mathrm{I}}^{k,1}(\mathcal{Z}_0\tilde{\mathcal{A}}, CR, R\mathcal{F}) \tag{1}$$

where $\approx$ means statistical indistinguishability. (To see this, let the environment corrupt and honestly simulate the recipient R. Then all communication of R and $\tilde{\mathcal{A}}$ is known to the environment.)

We now make use of the fact that $\mathcal{F}$ is OTS for C. So, by Definition 7, the communication of $\mathcal{F}$ with C can be (inefficiently) calculated from the communication of $\mathcal{F}$ with R and with the dummy-adversary $\tilde{\mathcal{A}}$. The communication of $\mathcal{F}$ with $\tilde{\mathcal{A}}$ again can be calculated from the communication between $\tilde{\mathcal{A}}$ and $\mathcal{Z}_0$ (since $\tilde{\mathcal{A}}$ simply forwards messages for $\mathcal{Z}_0$). Summarising these facts, we have

$$com_{\mathrm{I}}^{k,z}(\mathcal{Z}_0\tilde{\mathcal{A}}, CR, C\mathcal{F}) \lhd com_{\mathrm{I}}^{k,z}(\mathcal{Z}_0\tilde{\mathcal{A}}, CR, R\mathcal{F}).$$

Now we corrupt C and simulate it honestly, i.e., we construct an environment $\mathcal{Z}_C$ that simulates $\mathcal{Z}_0$ and C, and forwards all messages C generates through the dummy-adversary $\tilde{\mathcal{A}}_C$. The resulting network II is given in Figure 1. Then the communication between $\mathcal{Z}_C$ and $\mathcal{A}_C$ consists of the following: (i) the communication of the simulated C with V and $\mathcal{F}$ and (ii) the communication of the simulated $\mathcal{Z}_0$ with the adversary. Therefore

$$com_{\mathrm{II}}^{k,z}(\mathcal{Z}_C\tilde{\mathcal{A}}_C) \lhd com_{\mathrm{I}}^{k,z}(\mathcal{Z}_0\tilde{\mathcal{A}}, CR, C\mathcal{F}).$$

Now, since π is long-term-UC secure, there is a simulator $\mathcal{S}_C$, s.t. in the network III depicted in Figure 1 the environment $\mathcal{Z}_C$ has a statistically indistinguishable output from $\mathcal{Z}_C$ in network II. Since the communication between $\mathcal{Z}_C$ and the adversary/simulator is output by $\mathcal{Z}_C$, we get

$$com_{\mathrm{III}}^{k,z}(\mathcal{Z}_C\mathcal{S}_C) \lhd com_{\mathrm{II}}^{k,z}(\mathcal{Z}_C\tilde{\mathcal{A}}_C).$$

Since $\mathcal{Z}_0$ gets the (*committed*) from R in network I, it also gets that message from $\mathcal{F}_{\mathrm{COM}}$ in network III (since the view of $\mathcal{Z}_0$ is indistinguishable in all three networks). Furthermore, if the auxiliary input is (b, unveil), $\mathcal{Z}_0$ receives b with overwhelming probability from the $\mathcal{F}_{\mathrm{COM}}$ after having sent (*unveil*) to C. So the bit $\tilde{b}$ that $\mathcal{S}_C$ sends to $\mathcal{F}_{\mathrm{COM}}$ in network III fulfils $\tilde{b} = b$ with overwhelming probability. This even holds if the auxiliary input had the form b (not (b, unveil)), since $\mathcal{S}_C$ cannot learn whether (*unveil*) is going to be sent until after it has sent $\tilde{b}$ (since $\mathcal{Z}_0$ waits until it receives (*committed*) before sending (*unveil*)). Therefore, if $B^{k,z}$ denotes the bit $\tilde{b}$ the simulator $\mathcal{S}_C$ sends in a run of network III with security parameter k and auxiliary input z, we have $B^{k,b} = b$ with overwhelming probability (for $b \in \{0,1\}$).

Note however, that in network III, the bit $\tilde{b}$ sent from $\mathcal{S}_C$ to $\mathcal{F}_{\mathrm{COM}}$ depends on b only through the communication between $\mathcal{Z}_C$ and $\mathcal{S}_C$. So

$$B^{k,z} \lhd \mathit{com}_{\mathrm{III}}^{k,z}(\mathcal{Z}_C\mathcal{S}_C).$$

Combining all $\lhd$-inequalities above, we get

$$B^{k,b} \lhd \mathit{com}_{\mathrm{I}}^{k,z}(\mathcal{Z}_0\tilde{\mathcal{A}}, CR, R\mathcal{F}).$$

By definition of $\lhd$ and (1), there is a probabilistic function G s.t.

$$B^{k,0} \approx G\big(\mathit{com}_{\mathrm{I}}^{k,0}(\mathcal{Z}_0\tilde{\mathcal{A}}, CR, R\mathcal{F})\big) \approx G\big(\mathit{com}_{\mathrm{I}}^{k,1}(\mathcal{Z}_0\tilde{\mathcal{A}}, CR, R\mathcal{F})\big) \approx B^{k,1},$$

which is a contradiction to $B^{k,b} = b$. □

B.2 Proof of Lemma 11

Proof (of Lemma 11).

Given one-way permutations exist there is a computationally binding and unconditionally hiding bit commitment for which the unveil information can deterministically be verified(see Section 1.1). We will use the $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$ functionality to turn this commitment into a commitment protocol π which is long-term-UC.

The protocol π looks as follows:

- *To commit* to v, the sender C first commits to v using the unconditionally hiding commitment scheme.
- Then C proves (using the first instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$) that he knows v and matching unveil information u.[31]
- *To unveil*, the sender C sends v to the recipient and proves (using the second instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$) that he knows matching unveil information u.

To use the $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$ functionality the statement to be proven must be an NP statement. This is the case for both usages of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$ in the protocol as the unveil information can deterministically be verified.

Next we prove that π is a long-term-UC commitment. To do this we have to look at three cases:

[31] I.e., unveil information that would convince the verifier.

No party is corrupted: It is easy to see that given all messages are delivered then commit and unveil will be successful with overwhelming probability so the protocol is nontrivial.

The sender C is corrupted: We construct a simulator $\mathcal{S}$ as follows:

- The simulator $\mathcal{S}$ runs a simulated copy of the real adversary $\mathcal{A}$ (playing the role of the corrupted sender C) which he connects
  - to the environment $\mathcal{Z}$,
  - to a simulated copy of an honest real recipient R, and
  - to two simulated instances of the functionality $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$ (of which all witnesses are stored).
- IF the recipient R accepts the commit phase THEN extract the witness u, v from the the call $\mathcal{A}$ places to $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$.
- $\mathcal{S}$ sends (commit, v) to the ideal functionality $\mathcal{F}_{\text{COM}}$.
- IF the recipient R accepts the unveil THEN $\mathcal{S}$ sends (unveil) to the ideal functionality $\mathcal{F}_{\text{COM}}$.

The interaction of the simulator with the environment is statistically indistinguishable from the interaction $\mathcal{Z}$ has with the real adversary in the real model as $\mathcal{S}$ runs a faithful simulation of the real adversary $\mathcal{A}$. It remains to be proven that the interaction of (a computationally limited) $\mathcal{Z}$ with the adversary *and* the recipient is statistically indistinguishable in the real and in the ideal model. The ideal recipient accepts a commitment iff the simulated real recipient accepts it, which itself is a faithful simulation of the real recipient, hence we have indistinguishability in the commit phase. To prove indistinguishability also for the unveil phase we additionally have to show that a successfully unveiled value in the real model equals the witness used in the first instance of the $\mathcal{F}_{\text{ZK}}$ functionality. As the bit commitment is binding the real adversary can know only one value v with corresponding unveil information v so the values v and u in the first instance of $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$ must equal the v and u used in the unveil. This concludes the proof of long-term-UC in the case of a corrupted sender.

The recipient R is corrupted: We will see that the protocol is statistically secure in this case and hence especially long-term-UC for a corrupted recipient.

We construct a simulator $\mathcal{S}$ as follows:

- The simulator $\mathcal{S}$ runs a simulated copy of the real adversary $\mathcal{A}$ which he connects
  - to the environment $\mathcal{Z}$,
  - to a simulated copy of an honest real sender C, and
  - to two simulated instances of the functionality $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$.
- IF $\mathcal{S}$ receives a value (commit) from $\mathcal{F}_{\text{COM}}$ THEN $\mathcal{S}$ lets C commit to 0.
- IF $\mathcal{S}$ receives a value (unveil, v) from $\mathcal{F}_{\text{COM}}$ THEN
  - $\mathcal{S}$ sends v in the name of C to $\mathcal{A}$ and then
  - $\mathcal{S}$ sends a fake message in the name of the simulated 2nd instance of $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$ to $\mathcal{A}$ indicating that the unveil information u corresponding to v is known to C.

As the commitment scheme used in the protocol is statistically hiding the communication of $\mathcal{Z}$ with the protocol in the commit phase is statistically indistinguishable in the real and in the ideal model.

In the unveil phase the communication the simulated $\mathcal{A}$ receives is statistically indistinguishable from an honest unveil to the value v. Hence the communication of the environment with the protocol in the unveil phase is also indistinguishable in the real and in the ideal model. This implies long-term-UC for the case of a corrupted recipient. $\square$

B.3 Proof of Lemma 13

Proof (of Lemma 13). Assume that SAT has essentially unique witnesses. Let R be the following relation: For two circuits f_1, f_2, it is $(f_1, f_2)Rw$ iff $f_1(w) = 1$ or $f_2(w) = 1$. Since SAT has essentially unique witnesses, so has R. Then let U_R be as in Definition 12.

We first assume that there is a one-way-function h (secure against *uniform* adversaries). Consider the following algorithm A that, upon input $(1^n, y)$, behaves as follows:

- Choose a random $w' \in \{0,1\}^n$ and let $y' := h(w')$.
- Let f be the circuit that upon input w outputs 1 iff $h(w) = y$.
- Let f' be the circuit that upon input w outputs 1 iff $h(w) = y'$.
- Let w be the result of evaluating $U_R(1^n, \mathbf{f}, w')$ where $\mathbf{f}$ is (f, f') or (f', f) (randomly chosen).
- If $h(w) = y$, output w.

By the properties of U_R, w is a witness for $\mathbf{f}$ with overwhelming probability (in n). This w is a witness of f or of f'. Further, when the input of A is $h(\tilde{w})$ for a uniformly chosen $\tilde{w} \in \{0,1\}^n$, f and f' will have the same distribution. Therefore, again by the properties of U_R, the probability that w is indeed a witness for f is negligibly far from $\frac{1}{2}$. So $A(1^n, h(\tilde{w}))$ returns a preimage of $h(\tilde{w})$ for random $\tilde{w} \in \{0,1\}^n$ with noticeable probability, in contradiction to the fact that h is a one-way-function.

We come to the second part of the statement and assume that NP $\not\subseteq$ P/poly. Let R and U_R be as above. Let L_k be the set of all satisfiable circuits of length k and L the set of all satisfiable circuits. For any $M \subseteq L_k$ let $\bar{\mathcal{U}}(M)$ be a distribution, that returns a pair (f, w), s.t. f is uniformly chosen from M and $f(w) = 1$. Note that these distributions are not necessarily efficiently samplable.

Consider the (non-efficient) algorithm A that upon input of a circuit f and a set M behaves as follows:

- Choose $(f', w') \leftarrow \bar{\mathcal{U}}(M)$.
- Let w be the result of evaluating $U_R(|f|, \mathbf{f}, w')$ where $\mathbf{f}$ is (f, f') or (f', f) (randomly chosen).
- If $f(w) = 1$, output w.

Analogously to the reasoning in the case of one-way-functions, we see that for any $M \in L_k$, the probability that $A(f, M)$ outputs a w with $f(w) = 1$ for w uniformly chosen from M is negligibly close to $\frac{1}{2}$ (in the length of f). In particular, for sufficiently large f that probability is greater than $\frac{7}{16}$. Then, for at least $\frac{1}{4}$ of all $f \in M$ the output $A(f, M)$ satisfies f with probability at least $\frac{1}{4}$, since otherwise the probability for a random $x \in M$ to be solved would be bounded by $\frac{1}{4} \cdot 1 + \frac{3}{4} \cdot \frac{1}{4} = \frac{7}{16}$.

Let $S(M)$ be the set of the $f \in M$, s.t. $f(A(f, M)) = 1$ with probability less than $\frac{1}{4}$. By the above, $\#S(M) \leq \frac{3}{4}\#M$. We then define inductively: $M_k^0 := L_k$, $M_k^{i+1} := S(M_k^i)$. Then $\#M_{3k} \leq (\frac{3}{4})^{3k}\#L_n \leq (\frac{3}{4})^{3k}2^k < 1$, so $M_n^{3k} = \emptyset$.

Consider the (inefficient) algorithm A^* that upon input of a circuit f of length k behaves as follows:

- For each $i = 0, \dots, 3k - 1$, let $w_i \leftarrow A(M_k, f)$.
- If one of the w_i fulfils $f(w_i) = 1$, output $w := w_i$.

Since any f lies in some $M_k^i \setminus S(M_k^i)$ with $i < 3k$, this algorithm outputs a satisfying w with probability at least $\frac{1}{4}$.

Let now $\bar{\mathcal{U}}_k^*$ be the distribution $\bar{\mathcal{U}}(M_k^0) \times \cdots \times \bar{\mathcal{U}}(M_k^{3k-1})$. Then A^* can be rewritten as (with $k := |f|$):

- Let $(f'_0, w'_0, \dots, f'_{3k-1}, w'_{3k-1}) \leftarrow \bar{\mathcal{U}}_k^*$.
- For each $i = 0, \dots, 3k-1$, let $w \leftarrow U_R(k, \mathbf{f}, w'_i)$ where $\mathbf{f}$ is (f, f'_i) or (f'_i, f) (randomly chosen).
- If $f(w_i) = 1$ for some i, output $w := w_i$.

Since the only inefficient step of that algorithm is sampling $\bar{\mathcal{U}}_k^*$, there is a PPT-algorithm A^{**} s.t. for sufficiently long $f \in L_k$, $A(f, \bar{\mathcal{U}}_k^*)$ outputs some w satisfying f with probability at least $\frac{1}{4}$. Then, by Lemma 31 there is a nonuniform deterministic polynomial-time algorithm $\tilde{A}$ that finds witnesses for SAT, so SAT $\in$ P/poly and therefore NP $\subseteq$ P/poly, which stands in contradiction to our assumption. □

B.4 Proof of Theorem 14

Proof. In this proof, we again use the $\lhd$-notation and the $com_X^{k,z}(\dots)$-notation presented in the proof of Theorem 9 in Appendix B.1: If $A_{k,z}$ and $B_{k,z}$ are families of random variables, we write $A \lhd B$, if there is some probabilistic function G (not necessarily an efficiently computable one) s.t. $A_{k,z}$ and $G(k, B_{k,z})$ are statistically indistinguishable. Note that G knows k, but does not have direct access to z. (Intuitively $A \lhd B$ means, that A does not contain (noticeably) more information about z than B). Obviously, $\lhd$ is transitive. We will investigate different networks of machines (cf. Figure 2). To facilitate calculation, we use the following notation: $com_X^{k,z}(AB, CD, \dots)$ denotes the transcript of the communication between machines A and B, between machines C and D etc. in a run of the network X on security parameter k when the environment gets auxiliary input z. E.g., $com_{\text{II}}^{k,z}(R\mathcal{Z}_C, R\tilde{A}_C, R\mathcal{F})$ denotes all communication of party R in network II.

To produce a contradiction, we assume that there is a nontrivial protocol π that long-term-UC realises $\mathcal{F}_{\text{ZK}}^{R,P\to V,m}$ for some polynomially-bounded $m(k) \geq k$ (i.e. ZK for the relation R with prover P and Verifier V and with support for statements of length $\leq m(k)$). First consider the following network I (depicted in Figure 2, the adversary $\tilde{\mathcal{A}}$ has been omitted for simplicity): The uncorrupted prover P and verifier V run together with the environment $\mathcal{Z}_0$ and the dummy-adversary $\tilde{\mathcal{A}}$.[32] The environment $\mathcal{Z}_0$ behaves

[32] A dummy-adversary is an adversary, that forwards all messages to the environment, and follows all instructions given by that environment.

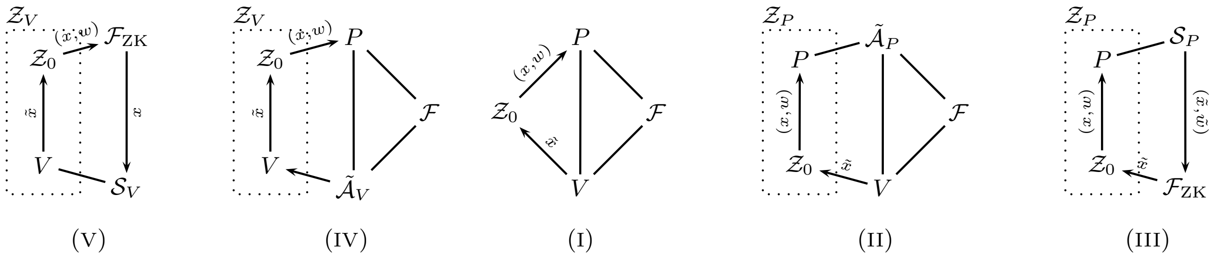


Fig. 2. Networks from the proof of Theorem 14.

as follows: It takes its auxiliary input (x, w) an sends that auxiliary input to P. Then it instructs the dummy-adversary $\tilde{\mathcal{A}}$ to deliver all messages. A message x from V is simply recorded. We assume that all environments constructed in this proof simply output their view (i.e., the transcript of all messages it sent or got and of all its internal states).

We will from now on assume that the auxiliary input of the environment is always of the form (x, w) with xRw and $|x| \le m(k)$. Then, since the protocol π is nontrivial, V will eventually send some $\tilde{x}$ to $\mathcal{Z}_0$ with overwhelming probability.

We now corrupt P and simulate it honestly. That is, we consider an environment $\mathcal{Z}_P$ that simulates $\mathcal{Z}_0$ and P, and forwards all messages P generates through the dummy-adversary $\tilde{\mathcal{A}}_P$. The resulting network II is shown in Figure 2. Then the communication between $\mathcal{Z}_P$ and $\mathcal{A}_P$ consists of the following: (i) the communication of the simulated P with V and $\mathcal{F}$ and (ii) the communication of the simulated $\mathcal{Z}_0$ with the adversary. Therefore

$$com_{\mathrm{II}}^{k,x,w}(\mathcal{Z}_P\tilde{\mathcal{A}}_P) \lhd com_{\mathrm{I}}^{k,x,w}(\mathcal{Z}_0\tilde{\mathcal{A}}, PV, P\mathcal{F}).$$

Now, since π is long-term-UC secure, there is a simulator $\mathcal{S}_P$, s.t. in the network III depicted in Figure 2 the environment $\mathcal{Z}_P$ has a statistically indistinguishable output from $\mathcal{Z}_P$ in network II. Since the communication between $\mathcal{Z}_P$ and the adversary/simulator is output by $\mathcal{Z}_P$, we get

$$com_{\mathrm{III}}^{k,x,w}(\mathcal{Z}_P\mathcal{S}_P) \lhd com_{\mathrm{II}}^{k,x,w}(\mathcal{Z}_P\tilde{\mathcal{A}}_P).$$

Note that the following fact hold with overwhelming probability in network III (since otherwise $\mathcal{Z}_0$ would not have indistinguishable view in networks I, II and III): A statement $\tilde{x}$ is sent from $\mathcal{F}_{\mathrm{ZK}}$ to $\mathcal{Z}_P$ that is equal to the x from $\mathcal{Z}_P$'s auxiliary input. Therefore, by definition of $\mathcal{F}_{\mathrm{ZK}}$, the $\tilde{w}$ sent from $\mathcal{S}_P$ to $\mathcal{F}_{\mathrm{ZK}}$ is a witness for x (but not necessarily $w = \tilde{w}$).

Let $\tilde{W}^{k,x,w}$ be the random variable denoting the distribution of $\tilde{w}$ in a run of network III. Since all machines in network III are polynomially-bounded, there is a PPT-algorithm $\bar{U}$, so that $\bar{U}(k, x, w)$ has the same distribution as $\tilde{W}^{k,x,w}$. That algorithm has the property, that for xRw and $|x| \le m(k)$, its output is a witness for x with overwhelming probability. To show that R has essentially unique witnesses, we have to show further that $\bar{U}$'s output is almost independent of w.

Note that in network III, the witness $\tilde{w}$ sent from $\mathcal{S}_P$ to $\mathcal{F}_{\mathrm{ZK}}$ depends on w only through the communication between $\mathcal{Z}_P$ and $\mathcal{S}_P$. In other words,

$$W^{k,x,w} \lhd com_{\mathrm{III}}^{k,x,w}(\mathcal{Z}_P\mathcal{S}_P).$$

To show that $W^{k,x,w}$ is almost independent of w, we have to got back to network I and make use of the fact that $\mathcal{F}$ is OTS for P. Then, by Definition 7, the communication of $\mathcal{F}$ with P can be (inefficiently) calculated from the communication of $\mathcal{F}$ with V and with the dummy-adversary $\tilde{\mathcal{A}}$. The communication of $\mathcal{F}$ with $\tilde{\mathcal{A}}$ again can be calculated from the communication between $\tilde{\mathcal{A}}$ and $\mathcal{Z}_0$ (since $\tilde{\mathcal{A}}$ simply forwards messages for $\mathcal{Z}_0$). Summarising these facts, we have

$$com_{\mathrm{I}}^{k,x,w}(\mathcal{Z}_0\tilde{\mathcal{A}}, PV, P\mathcal{F}) \lhd com_{\mathrm{I}}^{k,x,w}(\mathcal{Z}_0\tilde{\mathcal{A}}, PV, V\mathcal{F}).$$

Now, let us consider yet another network. Assume that V is corrupted and simulated honestly by the environment $\mathcal{Z}_V$, i.e. $\mathcal{Z}_V$ simulates both $\mathcal{Z}_0$ and V. V's communication is routed through $\tilde{\mathcal{A}}_V$. The resulting network IV is depicted in Figure 2. Since the communication of the simulated V with P and $\mathcal{F}$ is routed through $\tilde{\mathcal{A}}_V$ and therefore part of the latter's communication, we get

$$com_{\mathrm{I}}^{k,x,w}(\mathcal{Z}_0\tilde{\mathcal{A}}, PV, V\mathcal{F}) \lhd com_{\mathrm{IV}}^{k,x,w}(\mathcal{Z}_V\tilde{\mathcal{A}}_V).$$

Finally, since π is long-term-UC secure, there is a simulator $\mathcal{S}_V$, s.t. the output of $\mathcal{Z}_V$ in networks IVand V(cf. Figure 2) are statistically indistinguishable. It follows

$$com_{\mathrm{IV}}^{k,x,w}(\mathcal{Z}_V\tilde{\mathcal{A}}_V) \lhd com_{\mathrm{V}}^{k,x,w}(\mathcal{Z}_V\mathcal{S}_V).$$

Combining all $\lhd$-inequalities so far, we get

$$W^{k,x,w} \lhd com_{\mathrm{V}}^{k,x,w}(\mathcal{Z}_V\mathcal{S}_V). \tag{2}$$

Let now x_k, w_k^1, w_k^2 be sequences with $x_kRw_k^1$ and $x_kRw_k^2$. Assume further that $|x_k| \leq m(k)$. Since in network V for such x, w the functionality $\mathcal{F}_{\mathrm{ZK}}$ behaves independently of w (it only checks, whether w is indeed a witness), the communication between $\mathcal{Z}_V$ and $\mathcal{S}_V$ is independent of w. More formally,

$$com_V^{k,x_k,w_k^1}(\mathcal{Z}_V\mathcal{S}_V) \qquad \text{and} \qquad com_V^{k,x_k,w_k^2}(\mathcal{Z}_V\mathcal{S}_V)$$

are identically distributed. By definition of $\lhd$ and (2), there is a probabilistic function G s.t.

$$W^{k,x_k,w_k^1} \approx G\big(com_V^{k,x_k,w_k^1}(\mathcal{Z}_V\mathcal{S}_V)\big) \approx G\big(com_V^{k,x_k,w_k^2}(\mathcal{Z}_V\mathcal{S}_V)\big) \approx W^{k,x_k,w_k^2},$$

where $\approx$ denotes statistical indistinguishability. So $W^{k,x,w}$ and therefore also $\bar{U}(k,x,w)$ is independent of w in the sense of Definition 12. However, $\bar{U}$ does not completely fulfil the conditions for a witness unifier, since we have shown the above only for x_k with $|x_k| \leq m(k)$. But by defining $U_R(k,x,w) := \bar{U}(\max\{k,|x|\},x,w)$ we get a witness unifier in the sense of Definition 12 (since $m(k) \geq k$ and thus $|x_k| \leq m(\max\{|x_k|,k\})$). So R has essentially unique witnesses, which leads to a contradiction and therefore shows the Theorem. □

B.5 Proof of Theorem 16

Proof (of Theorem 16). Let f_k be a one-way permutation on $\{0,1\}^k$. Let SWIAOK be a system for statistically witness indistinguishable arguments of knowledge (such systems exist for any MA-relation under the assumptions of the theorem, cf. Section 1.1). By R we denote the relation specified in the theorem. Then the protocol π between P and V using two instances of $\mathcal{F}_{\mathrm{CT}}$ is defined as follows:

1. P is invoked with input (p, q, n).
2. P checks whether $nR(p, q)$. Otherwise he aborts.
3. P and V invoke the first instance of $\mathcal{F}_{\mathrm{CT}}$ and receive a random k bit string $\bar{r}$.
4. P sends n to V.
5. P proves using the SWIAOK the knowledge of $p, q, \bar{r}^*$, s.t. $nR(p, q)$ or $f_k(\bar{r}^*) = \bar{r}$.
6. P and V invoke the second instance of $\mathcal{F}_{\mathrm{CT}}$ and receive a random bit string r of length $k \cdot (|n| + k)$. They split r into strings $r_1, \dots, r_k$ of length $|n| + k$.
7. For each r_i, the prover selects a random square root s_i of r_i modulo n, i.e. a uniformly distributed $s_i \in \{s_i \in \{0, \dots, n-1\} : s_i^2 \equiv r_i \bmod n\}$. If for some i no such s_i exists, let $s_i := \perp$.[33]
8. P sends $s_1, \dots, s_n$ to V.
9. V checks whether $s_i^2 \equiv r_i \bmod n$ for all $s_i \neq \perp$, and whether $\#\{i : s_i \neq \perp\} > k/5$. If so, V outputs n.

To show that this protocol π long-term-UC realises $\mathcal{F}_{\mathrm{ZK}}^R$ for the relation R given in the theorem, we have to prove the following three claims:

- The protocol is nontrivial, i.e., on input (p, q, n) with $nR(p, q)$ for the prover, the verifier outputs n if all messages are scheduled and both parties are uncorrupted (this roughly corresponds to the completeness of the proof-system)
- There is a simulator for the case that the prover P is corrupted (this roughly corresponds to the knowledge-soundness of the proof-system).
- There is a simulator for the case that the verifier V is corrupted (this roughly corresponds to the zero-knowledge-property of the proof-system).

We start by showing, that if the prover gets input (p, q, n) with $nR(p, q)$, the verifier outputs n (in the uncorrupted case). The protocol contains only two steps in which the verifier might abort, during the SWIAOK (Step 5) and during the checks at the end (Step 9). Because of the completeness of the SWIAOK, and since indeed $nR(p, q)$, the verifier will abort only with negligible probability during Step 5. To see that the verifier accepts in Step 9, it is necessary to see that with overwhelming probability, more than $k/5$ of the r_i are squares modulo n. Since $n = pq$ is a Blum-integer, we have $p, q \geq 3$. For random $r' \in \mathbb{Z}_n$, $r' \bmod p$ and $r' \bmod q$ are independently uniformly distributed. At least $1/2$ of all $r' \bmod p \in \mathbb{Z}_p$ are squares (because 0 and half of the invertible elements are squares), the same holds for q. Since r' is a square modulo n if and only if it is a square modulo p and modulo q, it follows that r' is a square with probability at least $1/4$. Further, for random r_i of length $|n| + k$, $r_i \bmod n$ is almost uniformly distributed on $\mathbb{Z}_n$. So the probability that r_i is a square modulo n is at least $\frac{1}{4} - \mu$ for some negligible

[33] This can easily be done efficiently using the factorisation of n.

μ. Therefore the probability, that at least $k/5$ of k independently chosen r_i are squares is overwhelming. This concludes the proof of the nontriviality of π.

W.l.o.g., we can assume a dummy-adversary.[34]

We now consider the case that the prover P is corrupted. Then we have to find a simulator $\mathcal{S}_P$, s.t. the interaction between the environment (posing as the prover and relaying through the dummy-adversary) and the simulator $\mathcal{S}_P$ is indistinguishable from the interaction between the environment and the verifier. Furthermore, when the verifier would output n, the simulator has to send (p, q, n) to the ideal functionality $\mathcal{F}^R_{\mathrm{ZK}}$ so that it will output n.

We construct the simulator $\mathcal{S}_P$ as follows:

- $\mathcal{S}_P$ simulates an honest and unmodified instance of the verifier V.
- When prover and verifier invoke the first coin-toss, the resulting value $\bar{r}$ is chosen uniformly from $\{0,1\}^k$ (as would $\mathcal{F}_{\mathrm{CT}}$ is the real model).
- When prover and verifier invoke the second coin-toss, the resulting value r is chosen as the concatenation of $r_1, \dots, r_n$. To chose the r_i, the algorithm Q from Lemma 32 is invoked and returns $(r_i, \tilde{s}_i)$ where s_i is a random root of r_i if r_i is a square modulo n.
- When the verifier V outputs n in Step 9, the simulator checks the following:
  - Is n a square. Then $\sqrt{n}$ is a nontrivial factor of n.
  - Is r_i not invertible modulo n for some i? Then $\gcd(r_i, n)$ is a nontrivial factor of n.
  - Is $\gcd(s_i - \tilde{s}_i, n)$ a nontrivial factor of n for some i with $s_i \neq \bot$?

  If one these tests succeed, the simulator knows a nontrivial factor of n and can send (p, q, n) to $\mathcal{F}_{\mathrm{ZK}}$ (which fulfil $nR(p, q)$ if n is a Blum-integer).

By the knowledge-soundness of the SWIAOK and using the fact that no polynomially-bound machine can find an $\bar{r}^* = f_k^{-1}(\bar{r})$, for polynomially-bounded environments, we can assume that if the simulator verifier does not abort in Step 5, n is a Blum-integer. So, by Lemma 32, the r_i are almost uniformly distributed on $\{0,1\}^{|n|+k}$. So r (as chosen by the simulator) is statistically indistinguishable from a uniform r of length $k(|n| + k)$. Since the verifier behaves as would an honest verifier, it follows that the interaction with the real V is statistically indistinguishable from that with the simulator.

It is left to show that with overwhelming probability the simulator $\mathcal{S}_P$ sends (p, q, n) with $nR(p, q)$ to $\mathcal{F}_{\mathrm{ZK}}$ when the simulated verifier V outputs n. By the soundness of the SWIAOK, we can assume that n is a Blum-integer. Therefore it is left to show that the probability is negligible that the three tests performed by $\mathcal{S}_P$ fail. This would mean that all r_i are invertible modulo n, and that n is not a square. Since n is a Blum-integer, each r_i then has four roots, and since the $\tilde{s}_i$ are chosen (almost) independently of s_i (Lemma 32 guarantees that $\tilde{s}_i$ is an almost uniformly distributed root of r_i), for each $s_i \neq \bot$ with probability $\frac{1}{2}$ it is $s_i \neq \pm\tilde{s}_i$. So with overwhelming probability for at least one s_i we have $s_i \neq \pm\tilde{s}_i$, and in consequence $\gcd(s_i, \tilde{s}_i)$ is a nontrivial factor of n. So the simulator $\mathcal{S}_P$ successfully simulates.

[34] I.e., an adversary, that simply follows the instructions of the environment, cf. [Can05].

We now come to the case that the verifier V is corrupted. In this case, the simulator $\mathcal{S}_V$ gets an n from the functionality $\mathcal{F}_{\text{ZK}}$ which is guaranteed to be a Blum-integer, but the simulator does not get the factorisation of n. Now the simulator $\mathcal{S}_V$ has to interact with the environment in a way that is statistically indistinguishable from the interaction of the honest verifier with the environment (through the dummy-adversary). We construct the simulator $\mathcal{S}_V$ as follows:

- When the first coin-toss is requested, the simulator chooses its value $\bar{r}$ as $\bar{r} := f_k(\bar{r}^*)$ for uniformly chosen $\bar{r}^* \in \{0,1\}^k$.
- When the second coin-toss is requested, the simulator invokes the algorithm Q from Lemma 32 k times and gets $r_1, \dots, r_k$ and $s_1, \dots, s_k$. The value r of the second coin-toss is then the concatenation of the r_i.
- $\mathcal{S}_V$ simulates the prover P with the following modifications:
  - When performing the SWIAOK in Step 5 of the protocol, instead of using p, q as the witness (which is unknown), we use $\bar{r}^*$ as chosen above as a witness (for the rhs $\bar{r} = f_k(\bar{r}^*)$ of the statement to be proven).
  - Instead of trying to find square roots of the r_i in Step 7 (which is infeasible without the factorisation of n) we use the s_i returned by the algorithm Q.

Since n is always a Blum-integer, Lemma 32 guarantees, that the r_i and s_i have an indistinguishable distribution from that in an interaction with the real prover (since in the latter case the r_i would be uniformly distributed and the s_i would be random roots of the r_i or $s_i = \bot$ if no such root exists). Further, since the SWIAOK is statistically witness indistinguishable, the proof of the honest prover (which uses witness p, q) and the proof of the simulated prover (which uses witness $\bar{r}^*$) are statistically indistinguishable. Combining these facts, it is straightforward to see that the interaction between with the real and with the simulated prover are statistically indistinguishable.

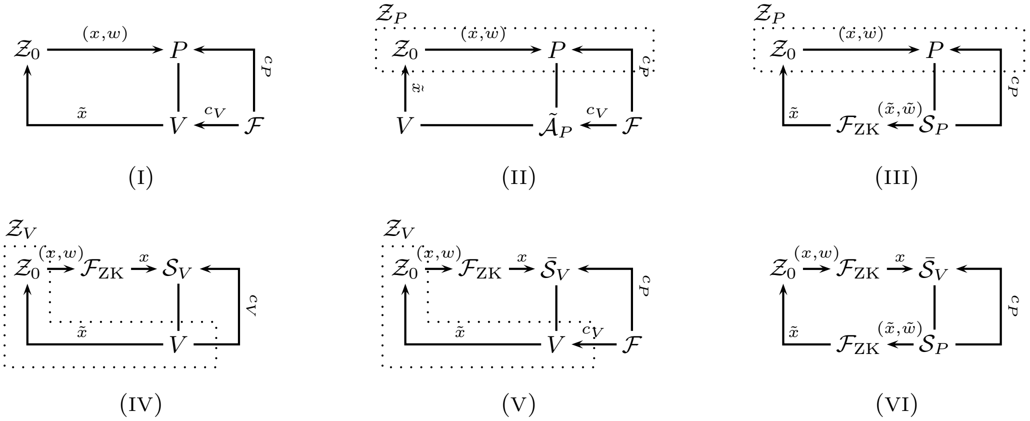
So π long-term-UC realises $\mathcal{F}_{\text{ZK}}^R$. □

B.6 Proof of Theorem 21

Proof (of Theorem 21). To show the Theorem, we assume that there is a protocol π consisting of prover P and verifier V that nontrivially long-term-UC realises $\mathcal{F}_{\text{ZK}}^{R,P\to V,m}$ with $m(k) \geq k$ using the offline functionality $\mathcal{F}$, and that $\mathcal{F}$ is OTS for party P and for party V.

Since $\mathcal{F}$ is an offline functionality, we can assume w.l.o.g. that each party accesses $\mathcal{F}$ only once, and that this is done upon its first activation. We call the value P gets c_P and the value V gets c_V. W.l.o.g. we can assume that the value c that the adversary gets is always the empty string (since if the protocol is secure and nontrivial using an $\mathcal{F}$ that gives some information to the adversary, it certainly is so, if that information is not given to the adversary).

Consider the following network I (shown in Figure 3, the dummy-adversary is omitted for simplicity): The parties P and V run uncorrupted with an environment $\mathcal{Z}_0$ and the dummy-adversary $\tilde{\mathcal{A}}$. The environment $\mathcal{Z}_0$ takes its auxiliary input (x, w) and sends (x, w) to the prover P. Then it instructs the dummy-adversary $\tilde{\mathcal{A}}$ to deliver all messages. We assume that all environments constructed in this proof simply output their view (i.e.,



(I) (II) (III)

(IV) (V) (VI)

Fig. 3. Networks from the proof of Theorem 21.

the transcript of all messages they sent or got and of all their internal states). Since the protocol is nontrivial, the verifier V eventually gives output if xRw and $|x| \leq m(k)$.

Now we corrupt P and simulate it honestly, i.e., we construct an environment $\mathcal{Z}_P$ that simulates $\mathcal{Z}_0$ and P and routes all messages from and to P through the dummy-adversary. The resulting network II is depicted in Figure 3. Since π is long-term-UC secure, there is a simulator $\mathcal{S}_P$, s.t. the output of $\mathcal{Z}_P$ is statistically indistinguishable in networks II and III (cf. Figure 3). Call the machines in the upper half of network I U, those in the lower half L. The upper half of network III consists of the same machines as that of network I, so we also call it U. The lower half of III we call $\tilde{L}$. Since $\mathcal{Z}_P$ consists of the machines U, the communication between U and $\tilde{L}$ is contained in $\mathcal{Z}_P$'s output, so the communication between U and L is statistically indistinguishable from that between U and $\tilde{L}$.

We can consider U, L and $\tilde{L}$ as single machines with security parameter k and auxiliary input x, w. Let then $\langle U, L\rangle^{k,x,w}$ denote the transcript of the communication between these machines. Then

$$\langle U, L\rangle^{k,x,w} \approx \langle U, \tilde{L}\rangle^{k,x,w},$$

where $\approx$ means statistical indistinguishability.

Now we go back to network I, corrupt V and simulate it honestly, i.e., we construct an environment $\mathcal{Z}_V$ simulating $\mathcal{Z}_0$ and V, and routing the communication through the dummy-adversary $\tilde{\mathcal{A}}_V$. W.l.o.g., we assume that $\mathcal{Z}_V$ queries the value c_V from the dummy-adversary $\tilde{\mathcal{A}}_V$ in its first activation, i.e., before invoking $\mathcal{Z}_0$ and in particular before using its auxiliary input. Then we construct the corresponding simulator $\mathcal{S}_V$. So we get the network IV shown in Figure 3. The communication of V and $\mathcal{Z}_0$ with the rest of the network is statistically indistinguishable for networks I and IV.

The simulator $\mathcal{S}_V$ has to provide the value c_V at its first activation, i.e., the choice of c_V and the internal state t of the simulator $\mathcal{S}_V$ after that step are chosen independently of the environment's auxiliary input x, w. So there is a family of probability distributions

$\mathcal{D}_k$ s.t. the (t, c_V) are distributed according to $\mathcal{D}_k$. Let $\mathcal{E}_k$ denote the distribution of (c_P, c_V) as chosen by $\mathcal{F}$. Since c_V is part of the communication observed by $\mathcal{Z}_V$, the distributions of c_V in as chosen by $\mathcal{E}_k$ and by $\mathcal{D}_k$ are statistically indistinguishable. Therefore, there is a probabilistic function D_k (not necessarily efficiently computable) s.t. when choosing $(c_P, c_V) \leftarrow \mathcal{E}_k$, the pair $(D_k(c_V), c_V)$ has statistically indistinguishable distribution from $\mathcal{E}_k$. Further, since $\mathcal{F}$ is OTS for V, there is a function f s.t. $c_V = f(c_P)$. Therefore, $(D_k(f(c_P)), c_V)$ is statistically indistinguishable from $\mathcal{D}_k$. So instead of using a simulator $\mathcal{S}_V$ that chooses (t, c_V) according to $\mathcal{D}_k$ and then sends c_V to V and keeps t for itself, we can use a modified simulator $\bar{\mathcal{S}}_V$ that instead receives c_P as chosen by an instance $\mathcal{F}$ and calculates $t := D_k(f(c_P))$. The machine V gets c_V from $\mathcal{F}$. The resulting network V is depicted in Figure 3. The communication of V and $\mathcal{Z}_0$ with the rest of the network is statistically indistinguishable for networks IV and V (and I, as seen above). Note that $\bar{\mathcal{S}}_V$ is not necessarily a polynomial-time machine.

When c_V and c_P are chosen by $\mathcal{F}$, c_P can be deterministically calculated from c_V, since $\mathcal{F}$ is OTS for P. Therefore the communication of V, $\mathcal{Z}_0$ *and* $\mathcal{F}$ with the rest of the network is statistically indistinguishable for networks I and V. So if we call the upper half of V $\tilde{U}$, and the lower half L (it consists of the same machines as the lower half L of network I), we get

$$\langle U, L\rangle^{k,x,w} \approx \langle \tilde{U}, L\rangle^{k,x,w}.$$

Since all machines send only a polynomial number of messages, by Lemma 33 it follows that

$$\langle U, L\rangle^{k,x,w} \approx \langle \tilde{U}, \tilde{L}\rangle^{k,x,w}.$$

Let network VI be the network consisting of $\tilde{U}$ and $\tilde{L}$ (i.e., the upper half of network V and the lower half of network III). Since in network I the statement $\tilde{x}$ send from V to $\mathcal{Z}_0$ fulfils $\tilde{x} = x$ with overwhelming probability, the same holds for network VI. So the $\tilde{w}$ sent from $\mathcal{S}_P$ to $\mathcal{F}_{\text{ZK}}$ in network VI is a witness for x with overwhelming probability (i.e., $\tilde{x}Rw$) as long as xRw and $|x| \leq m(k)$. So the following algorithm finds witnesses for x with overwhelming probability (assuming x has a witness).

1. Simulate network VI up to the point where $\bar{\mathcal{S}}_V$ has evaluated $t := D_k(f(c_P))$ with security parameter $k := |x|$. Call the state of the network s_0. (This step is not efficient. Note that the auxiliary input of $\mathcal{Z}_0$ has not been used so far, so this step depends only on the length of x but not on its value.)
2. Continue the simulation of network VI from state s_0 using (x, w) as auxiliary input for $\mathcal{Z}_0$ where w is some witness for x. (Note that for this simulation, we do not need to explicitly find such a w, since the $\mathcal{F}_{\text{ZK}}$ in the upper half of network VI will not use the value of w as long as it fulfils xRw. So this step can be performed efficiently.)
3. Extract the $\tilde{w}$ sent by $\mathcal{S}_V$ to $\mathcal{F}_{\text{ZK}}$ from this simulation and output $\tilde{w}$.

Obviously, the output of this algorithm is a witness for x with overwhelming probability. However, Step 1 is not efficient. But since the auxiliary input of $\mathcal{Z}_0$ is not used in that step, the distribution $\mathcal{G}_k$ of s_0 only depends on $k := |x|$. So there is an algorithm A taking inputs x, s_0 (consisting simply of Steps 2 and 3) that has the following property: $A(x, \mathcal{G}_{|x|})$ is a witness for x with overwhelming probability. So by Lemma 31 witnesses for R can be found by a nonuniform deterministic polynomial-time algorithm, so R is nonuniformly deterministically trivial, which gives us a contradiction and proves the theorem. □

B.7 Proof of Corollary 22

Proof (of Corollary 22). To implement $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V,m}$ using a CRS we use the following protocol π (notation is as in the proof of Theorem 16 in Appendix B.5):

1. A CRS $\underline{\mathbf{r}} = (\bar{r}, r^{(0)}, \dots, r^{(m(k))})$ is provided by the functionality $\mathcal{F}_{\mathrm{CRS}}$, where $\bar{r}$ has length k, and $r^{(i)}$ has length $k \cdot (|\gamma(0^i)| + k)$ (and $|r^{(i)}| := 0$ for $\gamma(0^i) = \bot$).
2. P is invoked with input (p, q, n, x).
3. P checks whether $(n, x)R(p, q)$. Otherwise he aborts.
4. P sends (n, x) to V.
5. P proves using the SWIAOK the knowledge of $p, q, \bar{r}^*$, s.t. $(n, x)R(p, q)$ or $f_k(\bar{r}^*) = \bar{r}$.
6. $r^{(|x|)}$ is split into $r_1, \dots, r_k$, each of length $|n|+k$ (note that the lengths match, since $|r^{(|x|)}| = k \cdot (|\gamma(0^{|x|})| + k) = k \cdot (|n| + k)$).
7. For each r_i, the prover selects a random square root s_i of r_i modulo n, i.e. a uniformly distributed $s_i \in \{s_i \in \{0, \dots, n-1\} : s_i^2 \equiv r_i \bmod n\}$. If for some i no such s_i exists, let $s_i := \bot$.[35]
8. P sends $s_1, \dots, s_n$ to V.
9. V checks whether $s_i^2 \equiv r_i \bmod n$ for all $s_i \neq \bot$, and whether $\#\{i : s_i \neq \bot\} > k/5$. If so, V outputs n.

We only describe how the simulator chooses the CRS $\underline{\mathbf{r}}$: Like in the proof of Theorem 16, $\bar{r}$ is chosen randomly if P is corrupted, and $\bar{r} = f_k(\bar{r}^*)$ for random $\bar{r}^*$ if V is corrupted.

For $\mu = 0, \dots, m(k)$, the simulator (both in case of a corrupted V and of a corrupted P) invokes the algorithm Q from Lemma 32 on input $n := \gamma(0^\mu)$. Then Q outputs $r_1, \dots, r_k$ of length $|n| + k$ together with random square roots $s_1, \dots, s_k$ (or $s_i = \bot$, if no root exists). Then $r^{(i)}$ is chosen as the concatenation of $r_1, \dots, r_k$. The s_i are stored.

Aside from this modification, the simulators are constructed analogously to the simulators in the proof of Theorem 16, and the proof of security is analogous to that of Theorem 16 (note that for any $(n, x) \in L_R$, the $r^{(|x|)}$ used in the protocol will have been constructed using Q with argument $\tilde{n} := \gamma(0^{|x|})$, which satisfies $n = \tilde{n}$, since γ depends only on the length of its argument). □

B.8 Proof of Lemma 23

Proof (of Lemma 23). The protocol π is as follows:

[35] This can easily be done efficiently using the factorisation of n.

- Let (n, g, x) be C's secret key and (n, h, x) the corresponding public key (as provided by $\mathcal{F}_{\mathrm{PKI}}^{G_{amal}}$).
- To commit to a bit $b \in \{0, 1\}$, the sender C sends $c := x + b \bmod 3$ to the recipient R. Upon receipt of that message the recipient outputs (*committed*).
- To unveil b, the sender C sends (b, x) to the recipient R. The recipient R checks, that $x + b \equiv c \bmod 3$ and that $h \equiv g^x \bmod n$. If that check succeeds, the verifier outputs b.

Obviously, an honest C always succeeds in unveiling with an honest R. So the protocol is nontrivial.

Consider the case that the *sender C is corrupted*. In this case, the simulator $\mathcal{S}_C$ has to interact with the environment in such a way that the interaction with the simulator is indistinguishable from an interaction with the honest recipient R. Further, when the verifier accepts the commit-phase, the simulator has to enter a bit $\tilde{b}$ into the ideal functionality $\mathcal{F}_{\mathrm{COM}}^1$. When the verifier accepts the unveil-phase and outputs bit b, the simulator $\mathcal{S}_C$ has to unveil $\tilde{b}$ using $\mathcal{F}_{\mathrm{COM}}^1$. In order for $\mathcal{S}_C$ to be successful, it must be $\tilde{b} = b$ with overwhelming probability. We achieve this as follows: The simulator honestly simulates the recipient R and the PKI $\mathcal{F}_{\mathrm{PKI}}^{G_{amal}}$. In particular, $\mathcal{S}_C$ learns x and c (as defined in the description of the protocol).

When the recipient R outputs (*committed*), the simulator sets $\tilde{b} := c - x \bmod 3$ and uses this value to commit using $\mathcal{F}_{\mathrm{COM}}^1$.

Obviously, the interaction with $\mathcal{S}_V$ and with the real recipient R are statistically indistinguishable (since $\mathcal{S}_V$ performs an honest simulation). It remains to check that $b = \tilde{b}$ with overwhelming probability. If $b \neq \tilde{b}$, the value x' received from the sender C during unveil fulfils $x' \neq x$, but $x' \equiv x \bmod \varphi(n)$ (otherwise $h \neq g^{x'}$ and the recipient would not have accepted). But then $4(x' - x)$ is a multiple of $4 \operatorname{ord} g$, which again is a multiple of $\varphi(n)$ with high probability.[36] Since g, x and g^x can be chosen without knowledge of the factorisation of n, this implies that there is a PPT-algorithm that finds a multiple of $\varphi(n)$ given n. By [Bon99, Fact 1] this implies the possibility to factor n and thus contradicts the complexity assumption in the lemma.

Now, we come to the case where the *recipient R is corrupted*. In this case, the simulator $\mathcal{S}_R$ has to interact with the environment in a way that its communication is indistinguishable from an interaction with the honest sender C. However, the simulator learns the bit b to be unveiled only at the beginning of the unveil phase (in contrast to the sender that knows b already during commit, because it has to commit to b).

We construct this simulator $\mathcal{S}_R$ as follows:

- The PKI $\mathcal{F}_{\mathrm{PKI}}^{G_{amal}}$ is simulated honestly. However, the simulator stores the factorisation $n = pq$.
- To commit, the simulator sends a random $c' \in \{0, 1, 2\}$.
- To unveil to b, the simulator chooses a random $x' \in \{0, \dots, 2^{2k} - 1\}$ subject to the conditions $x' + b \equiv c' \bmod 3$ and $x' \equiv x \bmod \operatorname{ord} g$ (here x is part of the secret key chosen by $\mathcal{F}_{\mathrm{PKI}}^{G_{amal}}$, and $\operatorname{ord} g$ can be efficiently calculated using p, q).

[36] Here we use that n is a product of safe primes: In this case, $\varphi(n) = 4p'q'$ for large primes p', q', and the probability that $\operatorname{ord} g \mid 4$ or that only one of p', q' is a factor of $\operatorname{ord} g$ is negligible.

Since n is a safe prime, $\varphi(n) = 2p'q'$ where p', q' are primes greater 3 with overwhelming probability. So $3 \nmid \varphi(n)$. Therefore for a random solution x of $h \equiv g^x \bmod n$ it holds that $x \bmod 3$ is almost uniformly distributed over $\{0, 1, 2\}$ (since x is chosen from a set of size at least $2^k n$). So also the c chosen by the honest sender C is almost uniformly distributed on $\{0, 1, 2\}$. It follows that c and c' have statistically indistinguishable distributions. So, given some fixed value of c, x is a uniformly random element subject to $x + b \equiv c' \bmod 3$ and $g^x \equiv h \bmod n$. But this is exactly how the simulator chooses x', so the distribution of (c, x) and of (c', x') are statistically indistinguishable (given only the public key). So $\mathcal{S}_R$ is successful in presenting an indistinguishable interaction.

Summarising, we have that π long-term-UC realises $\mathcal{F}^1_{\mathrm{COM}}$. □

B.9 Proof of Theorem 26

Proof (of Theorem 26).

The proof proceeds in two steps. First we construct from $\mathcal{F}_{\mathrm{TPF}}$ a (not necessarily long-term-UC secure) commitment which is computationally binding, unconditionally hiding, and extractable. In the second step we construct a simple zero knowledge protocol using this extractable commitment.

Constructing an extractable commitment. Given the prerequisite that one-way permutations exist there also exists a bit commitment scheme COM_0 which is computationally binding, unconditionally hiding, and where the unveil information can deterministically be verified, see Subsection 1.1. Partially following the construction from [HMQ04] we turn this commitment scheme into a commitment scheme COM_1 which has the additional property of extractability, i.e., if an uncorrupted recipient accepts the commit phase then the simulator can extract a value v from the information the environment gives to the adversary and the probability that a value different from v can later be unveiled is negligible. (Note that the newly constructed commitment need not be long-term-UC secure as it may not be equivocable).

The protocol COM_1 looks as follows:

- *To commit* to v, the sender C calls $\mathcal{F}_{\mathrm{TPF}}$ with value v and receives $f_s(v)$ then C commits to $v, f_s(v)$ using COM_0 and obtains unveil information u. Next C calls $\mathcal{F}_{\mathrm{TPF}}$ with value u and commits to $u, f_s(u)$.
- The recipient outputs (commit) after having received two commitments.
- *To unveil* the sender sends v, u and the unveil information for the second commitment.
- The recipient checks if u is the correct unveil information for $v, f_s(v)$ and verifies if the second commitment was correctly unveiled to (unveil, v).

To extract the value v from a valid commitment the simulator keeps a list of all calls placed to the $\mathcal{F}_{\mathrm{TPF}}$ functionality. The values v and u must be in this list, because it is infeasible to generate a commitment (which can be unveiled) without querying $\mathcal{F}_{\mathrm{TPF}}$. As all machines are polynomially limited during the protocol execution there are only polynomially many candidates for v and u. By trying to unveil the first instance of

COM_0 with all possible candidates for u the simulator can identify the value v or the commitment cannot be unveiled.

Long-term UC zero knowledge based on extractable commitments. It is sufficient to prove the existence of a long-term UC ZK protocol for graph-3-colourability, which we will construct using the above computationally binding, unconditionally hiding, and extractable commitment. We modify the zero knowledge protocol for graph-3-colourability from [GMW91] to obtain the following long-term-UC protocol π (one instance of that protocol is run for each instance of $\mathcal{F}_{\text{ZK}}^{R}$).

- The prover P gets as input a graph with m edges and a colouring and aborts if it is not a valid 3-colouring.
- The prover sends the graph to the verifier V.
- DO $m \cdot k$ times in parallel
  - The verifier commits (using COM_1) to a randomly chosen edge (v_1, v_2) of the graph.
  - The prover chooses a random permutation π of the three colours in his witness and commits (using COM_1) to $(v, \pi(c_v))$ for each vertex v with colour c_v.
  - The verifier unveils the edge (v_1, v_2).
  - The prover unveils the two corresponding vertices $(v_1, \pi(c_{v_1}))$, $(v_2, \pi(c_{v_2}))$.
  - The verifier checks if $\pi(c_{v_1}) \neq \pi(c_{v_2})$.
- The verifier outputs (accept) if all $m \cdot k$ parallel checks were successful.

The protocol always works for uncorrupted parties and is hence nontrivial. Next we consider the two cases of a corrupted verifier and of a corrupted prover.

The verifier V is corrupted: We construct a simulator $\mathcal{S}$ as follows:

- The simulator $\mathcal{S}$ runs a simulated copy of the real adversary $\mathcal{A}$ which he connects
  - to the environment $\mathcal{Z}$,
  - to a simulated honest prover P (one for each instance of π) with a modification as detailed below, and
  - to a simulated functionality $\mathcal{F}_{\text{TPF}}$.
- When $\mathcal{S}$ receives a message from (an instance of) $\mathcal{F}_{\text{ZK}}$ that a graph G is 3-colourable then $\mathcal{S}$ starts the simulation of the corresponding honest prover P.
- In each of the $m \cdot k$ parallel executions
  - Whenever the simulated prover accepted a commitment from $\mathcal{A}$ the simulator $\mathcal{S}$ extracts (if possible) the edge (v_1, v_2) from this commitment.
  - The simulated prover is modified to commit to a random colouring (not necessarily a 3-colouring) with $c_{v_1} \neq c_{v_2}$ (if an edge could be extracted).

In case the environment does not give a valid witness to the uncorrupted prover the simulation is clearly statistically indistinguishable from the real protocol. We can in the following assume that the graph in question is 3-colourable.

As the commitment scheme used in the protocol is extractable the simulator can either extract an edge (v_1, v_2) from the commitment of the simulated real adversary or the commitment cannot (can only with negligible probability) be unveiled to an edge. So far the communication of the environment with the protocol is statistically indistinguishable for the real and the ideal model. Next the prover commits to a random colouring instead of a true 3-colouring, but still the communication with the environment remains statistically indistinguishable for the real and the ideal model, because the commitment scheme is unconditionally hiding. If the simulated real adversary fails to unveil the commitment then the protocol will abort and the simulation is statistically indistinguishable from the real protocol. Else the extracted edge (v_1, v_2) must equal the unveiled edge (v_1', v_2'), because of the extractability of the commitment. Then the simulated prover will unveil $(v_1, c_{v_1}), (v_2, c_{v_2})$ with the colours being random, but unequal and hence statistically indistinguishable from what is unveiled by the uncorrupted real prover.

The prover is corrupted: We construct a simulator $\mathcal{S}$ as follows:

- The simulator $\mathcal{S}$ runs a simulated copy of the real adversary $\mathcal{A}$ which he connects
  - to the environment $\mathcal{Z}$,
  - to a simulated unmodified honest verifier V (one for each instance of π), and
  - to a simulated functionality $\mathcal{F}_{\text{TPF}}$.
- Whenever the simulated verifier V accepts a commit phase the simulator extracts the values of the commitments of the simulated real adversary.
- As soon as the simulated honest verifier accepts the zero knowledge argument the simulator enters a witness for the 3-colouring into (the corresponding instance of) the functionality $\mathcal{F}_{\text{ZK}}$ if one of the $m \cdot k$ colourings extracted from the commitments is a 3-colouring.

The communication of the environment $\mathcal{Z}$ with the adversary is clearly statistically indistinguishable in the real and in the ideal model, as the simulator runs a faithful simulation of the real model. It remains only to be proven that the simulator can enter (with overwhelming probability) a witness to the ideal zero knowledge functionality if the simulated honest verifier accepts the proof. Lets assume no proper 3-colouring could be extracted, then (with overwhelming probability) there exists at least one edge in each of the $m \cdot k$ parallel executions where the colours cannot be unveiled to be unequal, because it is infeasible to unveil something different from the extractable value. Then the probability that the protocol will not abort is negligible, namely at most $(1 - 1/m)^{m \cdot k} \in O(e^{-k})$. Hence the probability that the simulator can extract a witness if the simulated verifier accepted is overwhelming and the protocol is proven long-term-UC for a corrupted prover. □

B.10 Proof of Theorem 29

Proof (of Theorem 29). Let $\mathfrak{S} = (\mathit{KeyGen}, \mathit{Sign}, \mathit{Verify})$, where $\mathit{Sign}(sk, m)$ returns a signature for m using secret-key sk, and $\mathit{Verify}(pk, m, \sigma)$ returns 1 if σ is a valid signature for m with public key pk. By SWIAOK, we mean the statistically witness indistinguishable argument of knowledge described in Section 1.1 (which exists under the assumptions of the theorem).

We first describe the protocol π for implementing $\mathcal{F}_{\text{ZK}}^{R}$ (one instance of that protocol is run for each instance of $\mathcal{F}_{\text{ZK}}^{R}$):

- The prover P is activated with input (x, w).
- P checks, whether xRw. Otherwise, he aborts.
- P sends x to V.
- P obtains a signature σ for w from $\mathcal{F}_{\text{SC}}^{P,\mathfrak{S}}$.
- P proves using the SWIAOK the knowledge of strings σ, w, sk', s.t. one of the following holds:
  (i) $\mathit{Verify}(pk, w, \sigma) = 1$ and xRw, or
  (ii) For random $m \in \{0,1\}^k$, it is $\mathit{Verify}(pk, m, \mathit{Sign}(sk, m)) = 1$.
  The prover P can perform this proof using σ and w as obtained above.
- If the verifier V accepts the SWIAOK, it outputs x.

Obviously, if no-one is corrupted and xRw, and all messages are delivered, the verifier V outputs x with overwhelming probability, so the protocol is nontrivial.

W.l.o.g., we can assume a dummy-adversary.[37]

Let us first consider the case that the *verifier V is corrupted.* In this case, the simulator $\mathcal{S}_V$ has to interact with the environment in a way that is statistically indistinguishable from the interaction of the honest prover with the environment (through the dummy-adversary). However, in contrast to the prover, the simulator does not have access to a witness. The simulator $\mathcal{S}_V$ is constructed as follows:

- Simulate $\mathcal{F}_{\text{SC}}^{\mathfrak{S}}$ honestly.
- For each protocol instance, upon input x from the ideal functionality $\mathcal{F}_{\text{ZK}}^{R}$ (i.e., the environment sent (x, w) to the functionality with xRw), simulate an honest prover P with input $(x, 0)$ with the following modification:
- As witness for the SWIAOK, the simulated prover P uses $sk' := sk$ where sk is the secret key used by the simulator $\mathcal{F}_{\text{SC}}^{\mathfrak{S}}$ (i.e., the prover proves (ii) instead of (i)).

Since the SWIAOK is statistically witness indistinguishable, the resulting interaction (even if several instances of P are simulated in parallel) is statistically indistinguishable from an interaction with an honest prover.

Now we consider the case that the *prover P is corrupted.* In this case, the simulator $\mathcal{S}_P$ has to interact with the environment in a way that is statistically indistinguishable from the interaction of an honest verifier with the environment (through the dummy-adversary). Additionally however, if the honest verifier would output x, the simulator has to send (x, w) with xRw to the ideal functionality $\mathcal{F}_{\text{ZK}}^{R}$. We construct the simulator $\mathcal{S}_P$ as follows:

- Simulate $\mathcal{F}_{\text{SC}}^{\mathfrak{S}}$ honestly. However, whenever a string m is signed, store m in a list M.
- For each instance of the protocol, simulate the verifier V honestly.
- When the simulated verifier V outputs x, check whether xRw for some $w \in M$. If so, send (x, w) to $\mathcal{F}_{\text{ZK}}^{R}$. Otherwise, fail, abort and panic.

Obviously, as long as $\mathcal{S}_V$ does not fail, this interaction is indistinguishable from an interaction with the real verifier. We therefore only have to show that $\mathcal{S}_V$ fails with

[37] I.e., an adversary, that simply follows the instructions of the environment, cf. [Can05].

negligible probability. Therefore, assume that for some environment, $\mathcal{S}_V$ fails with non-negligible probability. In this case, the environment can be transformed into a nonuniform polynomial-time algorithm that, given an $(x, w) \in R$ as input and access to a public key pk and a signing oracle (the simulated $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S}}$) succeeds with non-negligible probability in performing the SWIAOK without signing a witness for x. By the argument-*of-knowledge*-property, using the knowledge extractor we get a nonuniform polynomial-time algorithm that, given access to a public key pk and a signing oracle, has the following properties:[38]

- It never signs any w' with xRw'.
- With non-negligible probability it outputs σ, w, sk' s.t. (i) or (ii) (from the definition of π) holds.

However, finding an sk' s.t. (ii) holds (even with non-negligible probability) contradicts the EF-CMA security of $\mathfrak{S}$. Furthermore, finding a signature σ for some w' that has not been signed using the oracle (even with non-negligible probability) also contradicts the EF-CMA security, (i) cannot be fulfilled by an nonuniform polynomial-time algorithm either. So by contradiction, the simulator $\mathcal{S}_P$ fails only with negligible probability. □

References

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge, 1988. JCSS, 37:156-189.

[BCMS99] Gilles Brassard, Claude Crépeau, Dominic Mayers, and Louis Salvail. Defeating classical bit commitments with a quantum computer. Los Alamos preprint archive quant-ph/9806031, May 1999.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Symposium on Foundations of Computer Science, Proceedings of FOCS 2004, 17-19 October 2004, Rome, Italy*, pages 186–195. IEEE Computer Society, October 2004.

[Bon99] D. Boneh. Twenty years of attacks on the rsa cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999. Online available at `http://crypto.stanford.edu/~dabo/abstracts/RSAattack-survey.html`.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001. Full version online available at `http://www.eccc.uni-trier.de/eccc-reports/2001/TR01-016/revisn01.ps`.

[Can05] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive, December 2005. Full and revised version of [Can01], online available at `http://eprint.iacr.org/2000/067.ps`.

[CCM02] Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *34th Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2002*, pages 493–502. ACM Press, 2002.

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil Vadhan, editor, *Theory of Cryptography, Proceedings of TCC 2007*, Lecture Notes in Computer Science. Springer-Verlag, 2007. To appear, full version online available at `http://eprint.iacr.org/2006/432`.

[38] Note that the environment together with the signing-oracle and the key-generator are polynomial-time, we can apply the knowledge-extractor the environment. Since the knowledge-extractor has only black-box access to the environment, the resulting algorithm does not perform any oracle queries the environment would not perform.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology, Proceedings of CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 2001. Full version online available at `http://eprint.iacr.org/2001/055.ps`.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2002*, pages 494–503. ACM Press, 2002. Extended abstract, full version online available at `http://eprint.iacr.org/2002/140.ps`.

[CM97] Christian Cachin and Ueli Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology, Proceedings of CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer-Verlag, 1997.

[DM04] Stefan Dziembowski and Ueli Maurer. On generating the initial key in the bounded-storage model. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology, Proceedings of EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 126–137. Springer-Verlag, 2004.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology, Proceedings of CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer-Verlag, 2002. Full version online available at `http://eprint.iacr.org/2001/091`.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):690–728, 1991. Online available at `http://www.wisdom.weizmann.ac.il/~oded/X/gmw1j.pdf`.

[Gol01] Oded Goldreich. *Foundations of Cryptography – Volume 1 (Basic Tools)*. Cambridge University Press, August 2001. Previous version online available at `http://www.wisdom.weizmann.ac.il/~oded/frag.html`.

[Gol04] Oded Goldreich. *Foundations of Cryptography – Volume 2 (Basic Applications)*. Cambridge University Press, May 2004. Previous version online available at `http://www.wisdom.weizmann.ac.il/~oded/frag.html`.

[HMQ04] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *Theory of Cryptography, Proceedings of TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 58–76. Springer-Verlag, 2004.

[HMQU05] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. Universally composable zero-knowledge arguments and commitments from signature cards. In *Proceedings of the 5th Central European Conference on Cryptology, MoraviaCrypt '05*, 2005. Online available at `http://iaks-www.ira.uka.de/home/unruh/publications/signaturecards.pdf`.

[Lin03] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *44th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2003*, pages 394–403. IEEE Computer Society, 2003. Online available at `http://eprint.iacr.org/2003/141`.

[MQ05] Jörn Müller-Quade. Temporary assumptions—quantum and classical. In *The 2005 IEEE Information Theory Workshop On Theory and Practice in Information-Theoretic Security*, 2005. abstract.

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, March 1998.

[Rab03] Michael O. Rabin. Hyper-encryption by virtual satellite. Science Center Research Lecture Series, December 2003. Online available at `http://athome.harvard.edu/dh/hvs.html`.

[Sig01] Gesetz über Rahmenbedingungen für elektronische Signaturen. Bundesgesetzblatt I 2001, 876, May 2001. Online available at `http://bundesrecht.juris.de/bundesrecht/sigg_2001/inhalt.html`.