J. Cryptol. (2010) 23: 594–671
DOI: 10.1007/s00145-010-9068-8

Journal of
CRYPTOLOGY

Long-Term Security and Universal Composability*

Jörn Müller-Quade
IKS, Universität Karlsruhe, Karlsruhe, Germany
muellerq@ira.uka.de

Dominique Unruh
Saarland University, Saarbrücken, Germany
unruh@mmci.uni-saarland.de

Communicated by Oded Goldreich

Received 8 December 2007
Online publication 28 May 2010

Abstract. Algorithmic progress and future technological advances threaten today's cryptographic protocols. This may allow adversaries to break a protocol retrospectively by breaking the underlying complexity assumptions long after the execution of the protocol. Long-term secure protocols, protocols that after the end of the execution do not reveal any information to a then possibly unlimited adversary, could meet this threat. On the other hand, in many applications, it is necessary that a protocol is secure not only when executed alone, but within arbitrary contexts. The established notion of universal composability (UC) captures this requirement.

This is the first paper to study protocols which are simultaneously 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 zero-knowledge protocols.

We give practical alternatives (e.g. signature cards) to these usual setup-assumptions and show that these enable the implementation of the important primitives commitment and zero-knowledge protocols.

Key words. Universal Composability, Long-term security, Zero-knowledge, Commitment schemes.

1. Introduction

Computers and algorithms improve over time and so does the ability of an adversary to break cryptographic protocols. The VENONA project is an example where the NSA (USA) and the GCHQ (U.K.) stored Russian ciphertexts over the years until these ciphertexts could eventually be cryptanalysed. Official key length recommendations, e.g. by the Federal Office for Information Security (BSI) in Germany, are valid for no more

* A short version of this paper appeared at TCC 2007 [42].

than six years, and future technology like quantum computers could render even paranoid choices for the key length obsolete. This threatens in particular data that is required to remain confidential for many years (e.g. medical data or government secrets). In other words, while we can be reasonably confident that cryptographic assumptions are indeed valid for today's adversaries, it is very uncertain what computational problems will still be considered hard a decade from now. This leads to the requirement of *long-term security* (a.k.a. *everlasting security*). A long-term secure protocol is based on temporary assumptions, assumptions that only have to hold *during* the execution of the protocol. These protocols should be secure even if the adversary becomes computationally unlimited *after* the protocol execution. However, in contrast to unconditional security, which is not based on computational assumptions at all, for long-term security, computational assumptions need to hold *during* the execution of the protocol. This reflects the fact that we can judge more or less precisely what the *current* state-of-the-art is. A long-term secure protocol will be robust against future technological advances on the part of the adversary.

However, when considering protocol executions in complex environments (the Internet being the foremost example), long-term security alone will not be enough to guarantee the integrity and confidentiality of one's data. It may happen that a protocol is long-term secure when executed in isolation, but becomes insecure when executed in larger contexts, e.g. as a subprotocol of some other protocol or concurrently with instances of the same or other protocols. This is also bothersome if one tries to modularly build up a protocol from smaller ones and wants to prove the security of the separate building blocks individually. Therefore, a strong notion of long-term security should also have strong composability properties that guarantee the (long-term) security of the protocol even if this protocol is executed in a larger context.

While long-term security has been (explicitly or implicitly) considered in a number of works (see Sect. 1.4), to the best of our knowledge, so far there have been no general definitions of long-term security (i.e. not specific to a particular protocol task like commitment or key-exchange) and no positive composability results. In this paper, we investigate the requirements resulting from such a strong notion and give both positive and negative results for two important cryptographic building blocks, commitments and zero-knowledge protocols.

1.1. *The Notion of Long-Term UC Security*

In order to explain our definition of *long-term* UC security, we briefly recall the main idea behind the notion of *computational* UC security as presented in [11]. In the UC model, security is defined by comparison of a so-called real protocol π with its specification, the so-called ideal functionality $\mathcal{F}$. The ideal functionality $\mathcal{F}$ is a trusted machine that (usually) performs the desired protocol task directly, without involving any cryptography. For example, a functionality $\mathcal{F}_{\text{COM}}$ for commitments would take a value m from the sender in the commit-phase and notify the recipient that some value has been committed to (without revealing m). In the unveil-phase, the functionality $\mathcal{F}_{\text{COM}}$ would send m to the recipient (without allowing the sender to change m). Obviously, $\mathcal{F}_{\text{COM}}$ is a secure commitment by definition. We say π *computationally UC emulates* $\mathcal{F}$ if for any polynomial-time adversary $\mathcal{A}$ (attacking the real protocol π) there is a polynomial-time simulator $\mathcal{S}$ (attacking the ideal functionality $\mathcal{F}$) such that for any polynomial-time

machine $\mathcal{Z}$ (called the environment) the outputs of $\mathcal{Z}$ in an execution of the network $\pi + \mathcal{A} + \mathcal{Z}$ (the real model) and in an execution of the network $\mathcal{F} + \mathcal{S} + \mathcal{Z}$ (the ideal model) are *computationally indistinguishable*. If π computationally UC emulates $\mathcal{F}$, any attack performed by $\mathcal{A}$ can be mimicked by $\mathcal{S}$, hence intuitively π is as secure as $\mathcal{F}$ which in turn is secure by definition (as it is the specification of the protocol task at hand).

The notion of computational UC security has the advantage of giving strong compositionality guarantees: If π computationally UC emulates $\mathcal{F}$, then a protocol σ^{π} using π as a subprotocol computationally UC emulates the protocol $\sigma^{\mathcal{F}}$ which invokes $\mathcal{F}$ instead of π (this is the so-called *universal composition theorem* [11]). Hence it is enough to analyse the (partially idealised) protocol $\sigma^{\mathcal{F}}$ (which in most cases will be much simpler). Then, the security properties of $\sigma^{\mathcal{F}}$ carry over to the more complex protocol σ^{π}.

However, computational UC security does not guarantee long-term security: Since all machines (the adversary, the simulator, and the environment) are polynomially bounded, it is not excluded that the real protocol π can be broken by investing a superpolynomial amount of work after the protocol execution. This threat can be covered by requiring *statistical* UC security: The real protocol π *statistically UC emulates* the ideal functionality $\mathcal{F}$ if for any *possibly unbounded* adversary $\mathcal{A}$ there is a simulator $\mathcal{S}$ (whose running time is polynomially bounded in that of $\mathcal{A}$) such that for any *possibly unbounded* environment $\mathcal{Z}$ the outputs of $\mathcal{Z}$ in an execution of the network $\pi + \mathcal{A} + \mathcal{Z}$ and in an execution of the network $\mathcal{F} + \mathcal{S} + \mathcal{Z}$ are *statistically* indistinguishable. This definition is certainly strong enough to give long-term security guarantees. Yet, as this definition does not allow us to use any computational assumptions, it is too strong for most applications. For example, a commitment scheme satisfying this definition would have to be both statistically hiding and statistically binding; this is well-known to be impossible.

In order to extend the UC definition to encompass long-term security, we need to allow adversary, simulator, and environment to be computationally unlimited *after* the protocol execution (but not during the protocol execution). This could be modelled by explicitly introducing two execution phases, one in which the protocol runs and adversary, simulator, and environment are polynomially-bounded, and a second phase in which no protocol machine is allowed to run and adversary, simulator, and environment are unbounded.[1] Furthermore, the output of the environment should be *statistically* indistinguishable in the real and the ideal model. Yet, introducing these two phases unnecessarily complicates the details of the model. Instead, we can make use of a simple observation: It is a well-known fact that without loss of generality, we can assume that the adversary only passes messages between the protocol and the environment, but does not perform any computations on its own (a so-called dummy-adversary [11]). Hence, when the protocol stops, we may assume that the dummy-adversary stops execution, too. If the dummy-adversary does not send any messages anymore, the simulator may also stop execution (since there is nothing to simulate). Thus, we may assume that after protocol execution, neither adversary nor simulator runs, and the environment only applies a (possibly inefficient) function to its output, but is polynomially bounded otherwise. Yet, applying such a function to the output of the environment cannot increase

[1] One might, as in the case of statistical UC, restrict the simulator to be polynomially bounded in the running time of the adversary in the second phase. However, this does not make a difference for the argument below and leads to the same security definition.

the statistical distance between the output in the real and in the ideal model, so we can even omit this post-protocol computation and get the following definition of *long-term UC security*:

Definition 1.1 (Long-term UC security—informal). A real protocol π *long-term UC emulates* the ideal functionality $\mathcal{F}$ if for any *polynomial-time* adversary $\mathcal{A}$ there is a *polynomial-time* simulator $\mathcal{S}$ such that for any *polynomial-time* environment $\mathcal{Z}$ the outputs of $\mathcal{Z}$ in an execution of the network $\pi + \mathcal{A} + \mathcal{Z}$ and in an execution of the network $\mathcal{F} + \mathcal{S} + \mathcal{Z}$ are *statistically* indistinguishable.

Summarising, the definition of long-term UC security captures the intuitive requirement of being secure against attacks of unbounded computational power *after* the protocol execution. It is not hard to see that long-term UC security is strictly stronger than computational UC security and strictly weaker than statistical UC security (see Sect. 3.2). Yet, the universal composition theorem also holds for long-term UC security (with an almost unmodified proof, see Sect. 3.2).

1.2. *Impossibility Results*

1.2.1. *Impossibility Results for Commitments*

We investigate whether it is possible to give long-term UC secure implementations of commitments; more precisely, whether there are protocols that long-term UC emulate the functionality $\mathcal{F}_{\text{COM}}$ for commitments. (Note that long-term UC secure commitments are something stronger than statistically hiding, computationally UC secure commitments; cf. Sect. 1.2.3 below.) It is well-known that even with respect to computational UC, it is impossible to implement commitments without using any special setup assumptions [12,16]. Instead, one has to assume that the real protocol has access to an ideal functionality like a common reference string (CRS) or a public key infrastructure (PKI).

We show that even with these (standard) setup assumptions, it is impossible to implement a long-term UC secure commitment protocol. To give a rough idea for the reason of this impossibility, we sketch why it is impossible to implement a long-term UC secure commitment using a CRS: Assume a protocol π that long-term UC emulates a commitment between a sender and a recipient. Assume that the sender commits to a random bit b. Then, because a long-term UC secure commitment must be statistically hiding, the view of the recipient is almost independent of b. Since the CRS-functionality gives the same random value to the sender and the recipient, the view of the sender equals the view of the recipient, hence also the view of the sender is independent of b (here by view we mean the sequence of messages sent and received by the sender, not including its input b or its randomness). Furthermore, to satisfy the UC definition, the protocol π needs to be extractable (this holds even for computational UC). That is, when interacting with the sender, the simulator needs to be able to extract b from the messages sent by the sender. However, since the view of the sender is independent of b, this is impossible.

The only property of the CRS functionality that we actually used was that the communication from the functionality to the sender could be computed from the interaction

from the functionality to the recipient. Furthermore, although in the case of the CRS, this computation is efficient, we did not use this fact. Hence, the above argument can be generalised to cover a large class of setup assumptions, which we call *long-term revealing* (LTR).

Definition 1.2 (Long-term revealing—informal). Let P be a party identifier. A functionality $\mathcal{F}$ is called long-term revealing (LTR) for P if the following holds: In any protocol (where the protocol includes $\mathcal{F}$, and specifies some implementation of the party P, and possibly other machines), the communication[2] between P and $\mathcal{F}$ can be derived (possibly by applying a inefficient function) from the communication between $\mathcal{F}$ and all parties except P.

If a functionality $\mathcal{F}$ is LTR for P, any secrets that P and $\mathcal{F}$ have established during the protocol run will eventually be learned by a adversary that is unbounded after the protocol execution. (Note that the definition does not imply that any such secrets between P and $\mathcal{F}$ are established. If no such secrets are established, the definition holds vacuously.)

Thus, using a similar argument as in the case of the CRS, we get the following theorem:

Theorem 1.3 (Impossibility of commitments using LTR functionalities—informal). *No long-term UC secure protocol for commitments with sender C can be implemented using a functionality that is LTR for C.*

Many commonly used setup assumptions are LTR functionalities. Indeed:

- The CRS is LTR for all parties because all parties get the same value.
- For the same reason, a coin-toss is LTR for all parties.
- A PKI is LTR for all parties assuming that the public key uniquely determines the corresponding secret key.
- The commitment functionality is LTR for the recipient.

As a corollary we get that, with respect to long-term UC security, commitments cannot be "turned around": Since a commitment from R to C is LTR for C, it cannot be used to implement a long-term UC secure commitment from C to R.

1.2.2. *Impossibility Results for Zero-Knowledge Protocols*

We then show that, in general, it is impossible to implement long-term UC secure zero-knowledge protocols[3] using LTR functionalities. In order to describe the impossibility result, we first need to define a certain class of relations:

[2] We stress that this refers to the bi-directional communication between P and $\mathcal{F}$. In the case of the CRS, there was only communication from $\mathcal{F} = \mathcal{F}_{\mathrm{CRS}}$ to P.

[3] By a long-term UC secure zero-knowledge protocol we mean a protocol that long-term UC securely implements the zero-knowledge functionality $\mathcal{F}_{\mathrm{ZK}}$. This functionality takes a pair (x, w) from the prover, and if w is a witness for x, then the functionality gives x to the verifier.

Definition 1.4 (Essentially unique witnesses—informal). A relation R has *essentially unique witnesses* if there is a polynomial-time algorithm U_R (the *witness unifier*) with the following two properties: On input $(x, w) \in R$, U_R outputs a witness w' with $(x, w') \in R$. For $(x, w_1), (x, w_2) \in R$, the distributions of $U_R(x, w_1)$ and $U_R(x, w_2)$ are statistically indistinguishable.

In other words, for a relation with essentially unique witnesses it is possible to efficiently produce a "normal form" $w' = U_R(x, w)$ of a witness w in such a way that it is information-theoretically impossible to find out from which of x's witnesses the witness w' was derived.

A special case are relations where each statement has at most one witness (relations with unique witnesses). In this case, if $U_R(x, w)$ just outputs w, U_R fulfils Definition 1.4. Hence relations with unique witnesses are a special case of relations with essentially unique witnesses.

For relations that *do not* have essentially unique witnesses, we get the following impossibility result:

Theorem 1.5 (Impossibility of zero-knowledge protocols using LTR functionalities—informal). *If R does not have essentially unique witnesses, no long-term UC secure zero-knowledge protocol for the relation R with prover P can be implemented using a functionality that is LTR for P.*

The reader might wonder whether Theorem 1.5 can be strengthened to show that for any nontrivial relation R, long-term UC secure zero-knowledge protocols using LTR functionalities are impossible. However, this is not the case: We present a long-term UC secure zero-knowledge protocol for showing the knowledge of the factorisation of a Blum-integer using an LTR functionality. Assuming that the factorisation of Blum-integers is a hard problem, this constitutes a long-term UC secure zero-knowledge protocol for a nontrivial relation. Thus, although it is not clear whether this protocol has practical applications, it at least shows that the condition on R in Theorem 1.5 is necessary.

Yet, we can weaken the condition on R if we strengthen the conditions on the functionality: We call a functionality *offline* if it distributes some values to all parties at the very start of the protocol and then stops. Examples for offline functionalities are the CRS (which sends the same random value to all parties and then stops) and the PKI (which sends the secret key to a single party and the public key to all parties and then stops). We have the following theorem:

Theorem 1.6 (Impossibility of zero-knowledge protocols using offline LTR functionalities—informal). *Let R be a nontrivial relation.*[4] *Then no long-term UC secure zero-knowledge protocol for the relation R with prover P can be implemented using an offline functionality that is LTR for P.*

This rules out any useful long-term UC secure zero-knowledge protocol that uses only a CRS or a PKI.

[4] That is, a relation that cannot be decided in non-uniform polynomial time.

1.2.3. *Statistically Hiding, Computationally UC Secure Commitments*

Intuitively, a long-term UC secure commitment scheme gives the following two guarantees: First, it guarantees that the committed value stays secret even in the presence of adversaries that are computationally unbounded after the protocol execution; this would also be guaranteed if the protocol was statistically hiding. Second, long-term UC security guarantees composability within arbitrary protocols; this would also be guaranteed by computational UC security. This observation may lead to the following conjecture:

> **Conjecture (false):** If a commitment scheme is statistically hiding, and also computationally UC secure,[5] then it is long-term UC secure.

Indeed, commitment schemes that are both statistically hiding and computationally UC secure are known [18] (based on a CRS). So if this conjecture were true, it would immediately give us long-term UC secure commitment schemes based on a CRS. Unfortunately, from our impossibility results it follows that such schemes do not exist, hence the conjecture must be false.

In order to give a better intuition about why a statistically hiding, computationally UC secure commitment scheme may fail to be long-term UC secure, we roughly sketch why the scheme from [18] (called the DN-commitment hereafter) fails to be long-term UC secure.

The basic idea behind the DN-commitment is very roughly the following: The set of possible values of the CRS is partitioned into two computationally indistinguishable sets E and $E^{\complement}$. The DN-commitment scheme is designed in such a way that if the CRS is in E, then the commitment is statistically hiding. However, if the CRS is in $E^{\complement}$, then the commitment is extractable (given a certain trapdoor, the committed message can be extracted) and equivocable (given a certain trapdoor, the committed message can be changed after the commit phase). In particular, if the CRS is in $E^{\complement}$, the DN-commitment scheme is only computationally hiding. In a normal execution, the CRS will always be chosen from E, hence the scheme is statistically hiding. To show the computational UC property, however, one constructs a simulator that chooses the CRS from $E^{\complement}$. In this case, the simulator is able to use the extractability and equivocability to perform its simulation.

Since the simulator chooses the CRS from a different set $E^{\complement}$ than the CRS in the real execution, the simulated protocol execution is not statistically indistinguishable from the real execution. Hence the DN-commitment is not long-term UC secure in the sense of Definition 1.1 although it is both computationally UC secure and statistically hiding individually. The intuitive reason is that the DN-commitment is not *simultaneously* statistically hiding and computationally UC secure *within a single execution*.

This shows that the above conjecture is false, in general. One might still argue that this is a problem of our definition (i.e. that long-term UC security is too strong a notion),

[5] To avoid confusion, note that this is a different notion than requiring that the commitment scheme is both long-term *stand-alone* secure and computationally UC secure. Long-term stand-alone security would be defined in terms of the stand-alone model in secure multi-party computation (in which no interaction between the adversary and the environment takes place; see, e.g. [26, Chap. 7]) with the only modification that the simulation in the ideal model must be *statistically indistinguishable* from the outputs in the real model (instead of computationally indistinguishable). We leave it as an open question what the implications of long-term stand-alone secure, computationally UC secure commitments are.

and that statistically hiding, computationally UC secure commitment schemes are good enough for all purposes. To show that this is not the case, we present a simple (though contrived) protocol π with the following property: When π uses ideal commitments, Alice's input m is kept secret even in the presence of unbounded adversaries. Yet, when we instantiate the ideal commitments in π with the DN-commitment scheme, then m can be extracted by an adversary that is computationally bounded during the protocol execution, but unbounded afterwards. In other words, the DN-commitment scheme looses its long-term security when composed.

1.3. *Possibility Results*

As the commonly used setup assumptions do not allow us to construct long-term UC secure commitments and zero-knowledge protocols, we present two practically motivated setup assumptions (based on [38]) which do allow us to construct such protocols.

The first of these setup assumptions is a trusted pseudorandom function (TPF). A TPF represents a piece of trusted hardware that computes a pseudorandom function [28] but does not (explicitly) reveal the seed of that function. Of course, the seed may be implicitly determined by the answers to sufficiently many queries to the TPF, but finding the seed based on these answers is infeasible. We assume that all parties have access to TPFs implementing the same function.

The second setup assumption is a signature card. A signature card represents a piece of trusted hardware that computes signatures on arbitrary messages for its owner, but never (explicitly) reveals its signing key (as in the case of TPFs). The corresponding verification key, however, is publicly known. The practical advantage of assuming signature cards is that such signature cards are commercially available.

In the following exposition, we concentrate on signature cards, but analogous reasoning and results hold for TPFs.

At the first glance, it seems that signature cards are LTR for all parties and thus cannot be used to construct long-term UC secure commitments or zero-knowledge protocols: A computationally unbounded machine can compute the signature card's signing key from the verification key, and thus compute the answers to any query made by the owner of the signature card. However, upon closer inspection, it turns out that signature cards are not LTR for their owner: Although an unbounded machine could compute the answers to all queries made to the signature card, it cannot determine *which* queries were actually made.

We give a long-term UC secure zero-knowledge protocol π that uses signature cards. On input a statement x and a corresponding witness w, the prover obtains a signature σ on w from the signature card and then proves using a statistically witness indistinguishable argument of knowledge (not necessarily UC secure) that one of the following is true:

(a) The prover knows strings σ and w such that σ is a valid signature on w, and w is a witness for x, or
(b) The prover knows the secret signing key of the signature card.

We can show that this protocol is long-term UC secure even if polynomially many instances of the protocol are run concurrently, sharing a single signature card.[6] This is very important for practical applications as it would be impractical to expect the user to use a fresh signature card for each and every zero-knowledge protocol he executes.

Given long-term UC secure zero-knowledge protocols, we can construct long-term UC secure commitments using standard techniques.

Combining these results, we get the following theorem:

Theorem 1.7 (Long-term UC secure protocols from signature cards—informal). *Using a single signature card* (*and assuming the existence of one-way functions*), *we can execute an arbitrary* (*polynomially-bounded*) *number of long-term UC secure zero-knowledge protocols and commitments*.

The protocol is also guaranteed to stay secure if the adversary retrieves the signing key from the signature card after the protocol execution (e.g. by opening it).

Similar ideas are used to construct long-term UC secure zero-knowledge protocols and commitments schemes from TPFs. However, the resulting protocol is somewhat more complicated than in the case of signature cards.

1.4. *Related Work*

1.4.1. *Related to Long-Term Security*

Memory Bounded Adversaries The idea of long-term or 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 [7, 8]. 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. Rabin [44] presents a scheme using distributed servers of randomness (virtual satellites) to achieve everlasting security; in this scheme the access of the adversary to the communication between the parties and the distributed servers is limited during the key exchange. It was shown by [22] that protocols in the bounded storage model do not necessarily stay secure when composed with other protocols.

Quantum Cryptography Long-term security has been investigated in quantum cryptography. It is generally accepted (even though not formally proven) that a computationally secure authentication of a quantum key exchange yields a long-term secure key. Quantum protocols for bit commitment and oblivious transfer which become unconditionally secure, but rely on temporary computational assumptions were investigated and found to be impossible[7] (see, e.g. [6]).

[6] This is not automatically guaranteed by the universal composition theorem since that theorem requires that the individual instances of a composed protocol do not share any functionalities. See [14] for a detailed discussion of this issue.

[7] Unless additional assumptions are made, such as bounded quantum storage [19] or the availability of a piece of trusted hardware.

Zero-Knowledge, Commitments, Function Evaluation Statistical zero-knowledge arguments [5] are, in fact, long-term secure zero-knowledge protocols (but not necessarily UC secure, of course). As soon as the protocol terminates, an adversary cannot profit from additional computational power; an unbounded verifier cannot learn anything because the argument system is *statistical* zero-knowledge, and an unbounded prover cannot cheat because the proof is already finished.

Similarly, statistically hiding commitments [43] are actually long-term secure commitments since after the end of the protocol, even with unlimited computational power, no party can learn the committed value (unless an unveil is performed) and no party can change the committed value (since the protocol has already ended).

Müller-Quade [41] stated protocols for secure function evaluation achieving long-term security; however, they only considered secure function evaluation with constant input size. Their protocols do not allow for universal composition (they do, however, compose sequentially).

Forward Security Another related topic is that of forward security, which was introduced in the context of key exchange in [20,32]. Forward security requires that past session keys remain computationally secure even if some long-term secret is revealed to the adversary. This notion is related to and seems to be implied by long-term UC because in most protocols the session keys can be computed by unlimited adversaries and can therefore be considered to be known after the protocol execution.

1.4.2. *Related to Composition*

Composition of general (not long-term secure) protocols as well as particular classes of protocols like zero-knowledge has been extensively studied. For example, [23] presented a zero knowledge scheme that does not compose in parallel and introduced the notion of witness indistinguishable proofs. This security property is weaker than zero-knowledge but closed under parallel composition. More general impossibility results regarding the both sequential and parallel composability of zero-knowledge protocols were independently shown in [27].

The work [13] provides a security definition for key exchange which allows us to build secure channels from key exchange, which was not guaranteed by earlier definitions. The security requirement of non-malleability of cryptographic protocols [21] becomes necessary when the protocols are to be used in certain larger applications.

Composition theorems which are independent of the surrounding application were given in [10] for sequential composition and in [2,11] for universal composition (UC).

To the best of our knowledge, with exception of the negative result [22], previous work on *long-term security* did not take the problem of composability into account.[8]

[8] Wehner and Wullschleger [46] investigate sequential composition in the quantum bounded storage model. Their constructions, however, do not allow for composition with protocols that are based on computational assumptions (because the simulator in their constructions is not polynomial-time). Thus their work does not apply to long-term security in our sense, namely as a combination of computational assumptions during and unlimited security after the protocol execution.

1.5. *Organisation*

Section 2 (Preliminaries) We present elementary notation and nomenclature (Sect. 2.1) and list the cryptographic tools used throughout this work (Sect. 2.2).

Section 3 (Modelling Composable Long-Term Security) We develop the definition of long-term UC. First, we review the UC framework on which our definition is based (Sect. 3.1). In Sect. 3.1, we introduce and motivate our definition of long-term UC security. Further, we state a few implicit conventions used in the rest of the paper (Sect. 3.3) and define the functionalities used in the rest of this work (Sect. 3.4).

Section 4 (Commitments) In this section, we discuss long-term UC secure commitments. We introduce long-term revealing setup assumptions and show the impossibility of realising long-term UC secure commitments from these (Sect. 4.1). Then we show that given a long-term UC zero-knowledge scheme, long-term UC commitments can be realised (Sect. 4.2). In Sect. 4.4, we explain why statistically hiding, computationally UC secure commitment schemes are not necessarily long-term UC secure.

Section 5 (Zero-Knowledge Protocols) We discuss long-term UC secure zero-knowledge protocols. In Sect. 5.1, we introduce the class of relations with essentially unique witnesses and show that using long-term revealing setup assumptions, relations without essentially unique witnesses cannot have long-term UC secure zero-knowledge protocols. In Sect. 5.2, we extend our impossibility result by showing that if a long-term revealing setup assumption is used offline, we can only construct long-term UC zero-knowledge protocols for (almost) trivial relations. In Sect. 4.3, we investigate in more detail whether a PKI can be used for constructing long-term UC zero-knowledge protocols.

Section 6 (Possibility Results from Non-standard Setup Assumptions) We investigate non-standard (but practically motivated) setup assumptions that enable the implementation of long-term UC secure commitments and zero-knowledge schemes. Namely, in Sect. 6.1 we show how to do this using trusted hardware that evaluates pseudorandom functions, and in Sect. 6.2 we show how to use signature cards.

Section 7 (Conclusions) We conclude the paper and propose directions for further research.

2. Preliminaries

2.1. *Notation*

We call a function f *negligible*, if for any positive polynomial p and sufficiently large k, $f(k) \leq 1/p(k)$. We call f *overwhelming*, when $1 - f$ is negligible. We call f *non-negligible* if it is not negligible. We call f *noticeable* if there is a positive polynomial such that for sufficiently large k we have $f(k) \geq 1/p(k)$.

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 such that $|w| \leq p(|x|)$ 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. 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 an NP-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 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 $p \equiv q \equiv 3 \bmod 4$. An integer $p \geq 3$ is called a *safe prime* if both p and $\frac{p-1}{2}$ are prime.

An EF-CMA secure signature scheme is a scheme that has existential unforgeability under chosen message attacks [31]. We do not require perfect completeness (i.e. when choosing a key pair, signing and then verifying, the probability of a successful verification is overwhelming but not necessarily 1).

Unless otherwise stated, all security assumptions are against nonuniform adversaries (e.g. EF-CMA security means EF-CMA security with respect to nonuniform adversaries).

2.2. *Cryptographic Tools*

In [33], it is shown that assuming the existence of a one-way function, a statistically 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. If both parties are honest, the verification succeeds with probability 1 (perfect correctness).

Using that commitment-scheme in the zero-knowledge argument-system for graph-3-colourability from [30], we get a statistically witness indistinguishable argument of knowledge statistically witness indistinguishable argument of knowledge for any NP-relation given any one-way function.[9]

For some results, we will need the existence of EF-CMA secure signature schemes. These also exist under the assumption that one-way functions exist [45].

3. Modelling Composable Long-Term Security

3.1. *UC Framework*

Since our work builds on the UC framework [11], in this section we revisit the main points of that model. The reader familiar with the UC framework can safely skip this section and proceed to Sect. 3.2 on p. 609.

In general, the fact that a protocol is secure with respect to some security notion does not necessarily imply that the protocol stays secure with respect to that notion when composed with other (secure) protocols. Therefore, an important property of security notions is the ability to guarantee secure composition. One security notion that gives very strong composability guarantees is the Universal Composability framework (UC) from [11]. (As well as the independently proposed and essentially identical Reactive Simulatability framework [2].)

[9] The resulting scheme is, of course, even zero-knowledge, not only witness-indistinguishable, but we do not need that property here.

Overview Security in the UC framework is defined by comparison of a real protocol π with some ideal protocol ρ. In most cases, this ideal protocol ρ will consist of a single machine, a so-called ideal functionality. Such a functionality can been seen as a trusted machine that implements the intended behaviour of the protocol. For example, a functionality $\mathcal{F}$ for commitment would expect a value m from a party C. Upon receipt of that value, the recipient R would be notified by $\mathcal{F}$ that C has committed to some value (but $\mathcal{F}$ would not reveal that value). When C sends an unveil request to $\mathcal{F}$, the value m will be sent to R (but $\mathcal{F}$ will not allow C to unveil a different value). For more examples of functionalities, see Sect. 3.4 or [11].

Given a real protocol π and an ideal protocol ρ, we say that π realises ρ (also called "implements", "emulates", or "is as secure as") if for any adversary $\mathcal{A}$ attacking the protocol π there is a simulator $\mathcal{S}$ performing an attack on the ideal protocol ρ such that no environment $\mathcal{Z}$ can distinguish between π running with $\mathcal{A}$ and ρ running with $\mathcal{Z}$. Here $\mathcal{Z}$ may choose the protocol inputs and read the protocol outputs and may communicate with the adversary or simulator (but $\mathcal{Z}$ is, of course, not informed whether it communicates with the adversary or the simulator). There are two important differences between the UC model and the so-called stand-alone model for secure multiparty computation (see, e.g. [26, Chap. 7]). First, the environment may communicate with the adversary during the protocol execution, and second, the environment does not need to choose the inputs at the beginning of the protocol execution; it may adaptively send inputs to the protocol parties at any time, and it may choose these inputs depending upon the outputs and the communication with the adversary. These modifications are the reason for the very strong composability properties of the UC model.

Definitions In order to formulate these properties, let us first give more details on the above intuition. In the UC framework, all protocol machines and functionalities, as well as the adversary, the simulator and the environment are modelled as interactive Turing machines (ITM). Throughout a protocol execution, an integer k called the security parameter is accessible to all parties. At the beginning of the execution of a network consisting of π, $\mathcal{A}$, and $\mathcal{Z}$, the environment $\mathcal{Z}$ is invoked with an initial input z. From then on, every machine M that is activated can send a message m to a single other machine M'. Then that machine M' is activated and given the message m and the id of the originator M'. If in some activation a machine does not send a message, the environment $\mathcal{Z}$ is activated again. Additionally the environment may issue corruption requests for some party P. From then on, the machines corresponding to the party P are controlled by the adversary (i.e. it can send and receive messages in the name of that machine, and it can read the internal state of that machine). Finally, at some point the environment $\mathcal{Z}$ gives some output m which can be an arbitrary string. By $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ we denote the distribution of that output m on security parameter k and initial input z. Analogously, we define $\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k,z)$ for an execution involving the protocol ρ, the simulator $\mathcal{S}$, and the environment $\mathcal{Z}$.

We distinguish two different flavours of corruption. We speak of static corruption if the environment $\mathcal{Z}$ may only send corruption requests before the begin of the protocol, and of adaptive corruption if $\mathcal{Z}$ may send corruption requests at any time in the protocol, even depending on messages learned during the execution. In this paper, we will restrict our attention to the less strict security model using static corruption.

If the ideal protocol ρ consists of an ideal functionality $\mathcal{F}$, for technical reasons we assume the presence of so-called dummy parties that forward messages between the environment $\mathcal{Z}$ and the functionality $\mathcal{F}$. For example, assume that $\mathcal{F}$ is a commitment functionality. In an ideal execution, $\mathcal{Z}$ would send a value m to the party C (since it does not know of $\mathcal{F}$ and therefore will not send to $\mathcal{F}$ directly). Then C would forward m to $\mathcal{F}$. Then $\mathcal{F}$ notifies R that a commitment has been performed. This notification is then forwarded to $\mathcal{Z}$. With these dummy parties we have, at least syntactically, the same messages as in the real execution: $\mathcal{Z}$ sends m to C and receives a commit notification from R. Second, the dummy-parties allow a meaningful corruption in the ideal model. If $\mathcal{Z}$ corrupts some party P, in the ideal model the effect would be that the simulator controls the corresponding dummy party P and thus can read and modify messages to and from the functionality $\mathcal{F}$ in the name of P. Thus if we write $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, this is essentially an abbreviation for $\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}$ where the ideal protocol ρ consists of the functionality $\mathcal{F}$ and the dummy-parties.

For full details, see [11].

Having defined the families of random variables $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ and $\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k,z)$, we can now define security via indistinguishability.

Definition 3.1 (Universal composability [11]). A protocol π *UC realises* a protocol ρ, if for any polynomial-time adversary $\mathcal{A}$ there exists a polynomial-time simulator $\mathcal{S}$ such that, for any polynomial-time environment $\mathcal{Z}$, 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 $\{\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}}$ are computationally indistinguishable.

Note that in this definition, it is also possible to only consider environments $\mathcal{Z}$ that give a single bit of output. As demonstrated in [11], this gives rise to an equivalent definition. However, in the case of long-term UC below, this will not be the case, so we stress the fact that we allow $\mathcal{Z}$ to output arbitrary strings. In particular, an environment machine can output its complete view.

Natural variants of this definition are statistical UC, where all machines (environment, adversary, simulator) are computationally unbounded and the families of random variables are required to be statistically indistinguishable, and perfect UC, where all machines are computationally unbounded and the families of random variables are required to have the same distribution. In these cases, one often additionally requires that if the adversary is polynomial-time, so is the simulator.

Composition For some protocol σ, and some protocol π, by σ^{π} we denote the protocol where σ invokes (up to polynomially many) instances of π.[10] That is, in σ^{π} the machines from σ and from π run together in one network, and the machines from σ access the inputs and outputs of π. (In particular, $\mathcal{Z}$ then talks only to σ and not to the subprotocol π directly.) See [11] for details. A typical situation would be that $\sigma^{\mathcal{F}}$ is some protocol that makes use of some ideal functionality $\mathcal{F}$ (say, a commitment) and then σ^{π} would be the protocol resulting from implementing that functionality by some

[10] For simplicity, we assume throughout this work that the session ids assigned to these instances are $\{1,\ldots,p\}$ for some polynomial p.

protocol π (say, a commitment protocol). One would hope that such an implementation results in a secure protocol σ^π. That is, if π realises $\mathcal{F}$ and $\sigma^\mathcal{F}$ realises $\mathcal{G}$, then σ^π realises $\mathcal{G}$. Fortunately, this is the case:

Theorem 3.2 (Universal composition theorem [11]). *Let π, ρ, and σ be polynomial-time protocols. Assume that π UC realises ρ. Then σ^π UC realises σ^ρ.*

For a proof, see [11] or our proof sketch of Theorem 3.4 below. The intuitive reason for this theorem is that σ can be considered as an environment for π or ρ, respectively. Since Definition 3.1 guarantees that π and ρ are indistinguishable by any environment, security follows.

In a typical application of this theorem, one would first show that π realises $\mathcal{F}$ and that $\sigma^\mathcal{F}$ realises $\mathcal{G}$. Then using the composition theorem one gets that σ^π realises $\sigma^\mathcal{F}$ which in turn realises $\mathcal{G}$. Since the realises-relation is transitive (as can be easily seen from Definition 3.1), it follows that σ^π realises $\mathcal{G}$.

This composition theorem is the main feature of the UC framework. It allows us to build up protocols from elementary building blocks. This greatly increases the manageability of security proofs for large protocols. Furthermore, it guarantees that the protocol can be used in arbitrary contexts.

Analogous theorems also hold for statistical and perfect UC.

Dummy-Adversary When proving the security of a given protocol in the UC setting, a useful tool is the so-called dummy-adversary. The dummy-adversary $\tilde{\mathcal{A}}$ is the adversary that simply forwards messages between the environment $\mathcal{Z}$ and the protocol (i.e. it is a puppet of the environment that does whatever $\mathcal{Z}$ instructs it to do). In [11], it is shown that UC security with respect to the dummy-adversary implies UC security. The intuitive reason is that since $\tilde{\mathcal{A}}$ does whatever $\mathcal{Z}$ instructs it to do, it can perform arbitrary attacks and is therefore the worst-case adversary given the right environment (remember that we quantify over all environments).

We very roughly sketch the proof idea. Let protocols π and ρ and some adversary $\mathcal{A}$ be given. Assume that π UC realises ρ with respect to the dummy-adversary $\tilde{\mathcal{A}}$. We want to show that π UC realises ρ with respect to $\mathcal{A}$. Given an environment $\mathcal{Z}$, we construct an environment $\mathcal{Z}_\mathcal{A}$ which simulates $\mathcal{Z}$ and $\mathcal{A}$. Note that an execution of $\mathrm{EXEC}_{\pi,\tilde{\mathcal{A}},\mathcal{Z}_\mathcal{A}}$ is essentially the same as $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ (up to a regrouping of machines). Then there is a simulator $\tilde{\mathcal{S}}$ such that $\mathrm{EXEC}_{\pi,\tilde{\mathcal{A}},\mathcal{Z}_\mathcal{A}}$ and $\mathrm{EXEC}_{\rho,\tilde{\mathcal{S}},\mathcal{Z}_\mathcal{A}}$ are indistinguishable. Let $\mathcal{S}$ be the simulator that internally simulates the machines $\mathcal{A}$ and $\tilde{\mathcal{S}}$ and forwards all actions performed by $\mathcal{A}$ as instructions to $\tilde{\mathcal{S}}$ (remember that $\tilde{\mathcal{S}}$ simulates $\tilde{\mathcal{A}}$, so it expects such instructions). Then $\mathrm{EXEC}_{\rho,\tilde{\mathcal{S}},\mathcal{Z}_\mathcal{A}}$ is again the same as $\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}$ up to a regrouping of machines. Summarising, we have that $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}$ are indistinguishable. For details, see, e.g. [11].

A nice property of this technique is that it is quite robust with respect to changes in the definition of UC security. For example, it also holds with respect to statistical and perfect UC security, as well as with respect to our notion of long-term UC that is defined below (in all cases the proof is virtually unmodified).

3.2. *Long-Term UC*

We now present our model of universally composable long-term security (long-term UC). We build on the Universal Composability framework [11] described in the preceding section. 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.

The most immediate solution would be to introduce two phases in the execution of the network. In the first phase, all machines are computationally limited, and the protocol runs normally. In the second phase, the environment, the adversary, and the simulator are allowed to be computationally unlimited, but the protocol machines are not allowed to send any more messages. Thus the first phase models the protocol execution, and the second phase models the time after the protocol execution during which the data retrieved in the protocol may be analysed and computational assumptions may be broken. Finally, the output of the environment is required to be *statistically* indistinguishable in the real and the ideal model. (The choice of statistical instead of computational indistinguishability is because the distinction also happens after the protocol execution.)

Although this approach would probably work and give a reasonable model of long-term UC security, the introduction of different phases introduces major changes to the technical parts of the UC framework. For example, a machine model must be defined that allows us to switch the computational power from polynomial to unlimited, it must be specified who decides when the first phase ends (does the protocol have an output that it has finished execution, does the environment decide?), scheduling issues must be solved (when the second phase begins, how are the machines notified of this fact, in which order are they activated by the notification?), etc. We believe that all these issues can be solved. However, changes in scheduling or the definition of running-time have shown to interact nontrivially with other parts of the definition of UC and with the composition theorem (see, e.g. [1,35]). And many of the design decisions might be quite arbitrary or hard to motivate. Therefore, instead of using this two-phase model, we strive for a simpler definition of long-term UC security that introduces minimal changes with respect to the original UC framework and that in particular does not need to change the network, machine or running-time model.

First observe that in the two-phase model, we can assume the adversary to be the dummy-adversary that only forwards messages between the environment and the protocol (cf. the discussion on the preceding page). The intuitive reason for this is that any attack performed by the adversary itself can also be "remote-controlled" by the environment using a dummy-adversary. This argument is not invalidated by the fact that the adversary becomes unlimited at some point since the environment will become unlimited at that point, too, and will therefore still be able to simulate the adversary's behaviour. Since in the second phase, the protocol will not send any messages or receive any messages, we can assume without loss of generality that the dummy-adversary stops communicating after the first phase, too. Since the simulator has to mimic the behaviour of the adversary, the simulator will also stop communicating after the first phase. So we have arrived at a model, where in the second phase the only machine that actually performs any computations is the environment, and no communication takes place.

Then any post-protocol computation of the environment would be nothing more than the application of a function to the view, so the statistical distance between the environment's output in the real and the ideal model will be maximal for the environment that just outputs its unmodified view. Thus we can even get rid of the environment's internal computation in the second phase by allowing the environment to output its complete view.

By this argument, we have found a very simple way of adding the long-term security requirement to the UC framework: We simply require that *after* the execution of the protocol (which is still performed against computationally limited adversaries and environments) 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 in the real and ideal model.[11] Note that this is the only modification with respect to the original UC framework and the modification is actually smaller than that between computational and statistical UC. In particular, we can expect many properties derived for the original UC framework to easily carry over to the long-term UC framework.

In the following definition, 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. See Sect. 3.1 for details.

Definition 3.3 (Long-term UC). A protocol π *long-term UC realises* an ideal protocol ρ, if for any polynomial-time adversary $\mathcal{A}$ there exists a polynomial-time simulator $\mathcal{S}$ such that, for any polynomial-time environment[12] $\mathcal{Z}$, 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 $\{\mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}}$ are *statistically* indistinguishable.

Relation to Other UC Variants Obviously, this definition is stronger than computational UC as in Definition 3.1 because we replaced computational by statistical indistinguishability. Furthermore, statistical security (following [11], we assume the variant that requires the simulator to be polynomial-time if the adversary is; see p. 607) implies long-term UC security: In the case of statistical security, statistical indistinguishability holds for all environments and simulators, thus in particular for polynomial-time ones. And for polynomial-time adversaries, the simulator is guaranteed to be polynomial-time, too. Thus the conditions for long-term UC security are fulfilled.

Thus long-term UC lies between computational and statistical UC. Indeed, these implications are strict (assuming, e.g. the existence of one-way functions) as shown by the following examples: Let π be a protocol that expects a message m from the environment and passes that message to the adversary if and only if the adversary solves some computationally hard puzzle (e.g. inverting a one-way function). Let ρ be a protocol that expects m from the environment but does not give m to the adversary. Then π long-term UC realises ρ but does not statistically UC realise ρ.

[11] Here it is important that we are using the formulation of the UC framework where the environment is not restricted to outputting a single bit.

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

Furthermore, let π' be a protocol that outputs a pseudorandom string, and let ρ' be a protocol that outputs a truly random string. Then π' computationally UC realises ρ' but does not long-term UC realise ρ'.

The fact that statistical UC implies long-term UC is useful since one can use statistical UC results and combine them with long-term UC results using the composition theorem (see Theorem 3.4 below) to construct a larger long-term UC secure protocol. For example, from a commitment functionality, it is easy to construct a statistically secure coin toss.[13] Combined with a long-term UC secure protocol for commitment, we get a long-term UC secure coin toss protocol.

Composition Due to the minimal change in the security definition, the composition theorem from [11] (Theorem 3.2 above) carries over to our Definition 3.3:

Theorem 3.4 (Universal composition theorem). *Let π, ρ, and σ be polynomial-time protocols. Assume that π* long-term *UC realises ρ. Then σ^{π}* long-term *UC realises σ^{ρ}.*

For the full proof, we refer to [11], the only modification that needs to be made to their proof of the composition theorem is to replace all occurrences of terms of the form $|\Pr[\mathrm{EXEC}_{\ldots} = 1] - \Pr[\mathrm{EXEC}_{\ldots} = 1]|$ (comparing the probability of the environment outputting 1) by $\Delta(\mathrm{EXEC}_{\ldots}, \mathrm{EXEC}_{\ldots})$ where Δ denotes the statistical distance (comparing the actual distributions of environment's output). Except for this modification, the proof from [11] is unchanged. However, for completeness we give a short proof sketch of Theorem 3.4.

Proof Sketch. Our goal is to prove that under the assumptions of Theorem 3.4, σ^{π} long-term UC realises σ^{ρ}. Assume that σ invokes at most n instances of the respective subprotocol π or ρ. Since π long-term UC realises ρ, there is a polynomial-time simulator $\tilde{\mathcal{S}}$ such that for any environment $\mathcal{Z}$ we have that $\mathrm{EXEC}_{\pi,\tilde{\mathcal{A}},\mathcal{Z}}$ and $\mathrm{EXEC}_{\rho,\tilde{\mathcal{S}},\mathcal{Z}}$ are statistically indistinguishable. Here $\tilde{\mathcal{A}}$ is the dummy-adversary that just forwards messages between the environment and the protocol (see Sect. 3.1, p. 608). In the following, we will call this simulator $\tilde{\mathcal{S}}$ the dummy-simulator.

Let a polynomial-time adversary $\mathcal{A}$ be given (that is supposed to attack σ^{π}). We construct a simulator $\mathcal{S}$ that simulates the adversary $\mathcal{A}$ and n instances $\tilde{\mathcal{S}}_1, \ldots, \tilde{\mathcal{S}}_n$ of the dummy-simulator $\tilde{\mathcal{S}}$. The simulated adversary $\mathcal{A}$ is connected to the environment and to the protocol σ, but all messages between $\mathcal{A}$ and the i-th instance π_i of π are routed through the corresponding dummy-simulator $\mathcal{S}_i$ (which is then supposed to transform these messages into a form suitable for instances of ρ). The simulator $\mathcal{S}$ is depicted by the dashed box in network (II) in Fig. 1.

We have to show that for any environment $\mathcal{Z}$ we have that $\mathrm{EXEC}_{\sigma^{\pi},\mathcal{A},\mathcal{Z}}$ and $\mathrm{EXEC}_{\sigma^{\rho},\mathcal{S},\mathcal{Z}}$ are statistically indistinguishable, i.e. that the output of $\mathcal{Z}$ in the networks (I) and (II) in Fig. 1 is statistically indistinguishable.

For this, we construct a hybrid environment $\mathcal{Z}_{\sigma,i}$. (It is depicted as the dashed box in network (III) in Fig. 1.) This environment simulates the machines $\mathcal{Z}$, $\mathcal{A}$, the protocol σ, instances $\pi_1, \ldots, \pi_{i-1}$ of the real protocol π, and instances $\tilde{\mathcal{S}}_{i+1}, \ldots, \tilde{\mathcal{S}}_n$ and

[13] Alice commits to a random string r_1; Bob sends a random strings r_2 to Alice; Alice unveils. The result is $r_1 \oplus r_2$.

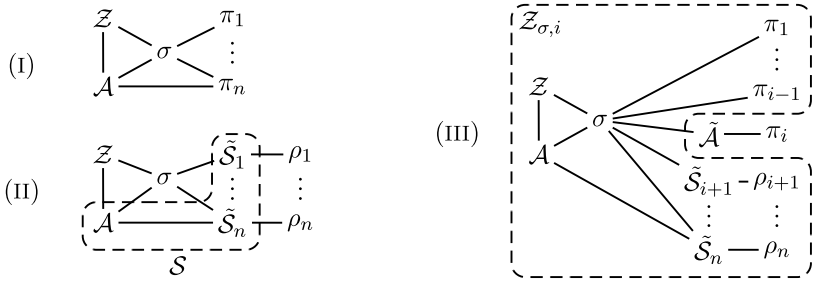

Fig. 1. Networks occurring in the proof sketch of Theorem 3.4. Network (I) represents the real model, (II) the ideal model, and (III) the hybrid case. For simplicity, not all connections between machines are depicted. For example, $\mathcal{A}$ also has connections to the instances of π, and σ also has direct connections to the instances of ρ.

$\rho_{i+1}, \ldots, \rho_n$ of the dummy-simulator and the ideal protocol ρ, respectively. The communication between $\mathcal{Z}$, $\mathcal{A}$, and σ is forwarded. Communication between one of these and the j-th protocol instance is forwarded as follows: If $j < i$, the communication is simply forwarded to π_j. If $j > i$, the communication is routed through the corresponding dummy-simulator $\tilde{\mathcal{S}}_j$ (which is then supposed to transform these messages into a form suitable for ρ_i). And finally, if $j = i$, the communication is passed to the outside of $\mathcal{Z}_{\sigma,i}$ (where it then will be passed to some adversary/simulator or to a protocol instance, depending on whether it was a message from $\mathcal{A}$ or σ).

We will now show that there is a negligible function μ such that the statistical distance between $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,i},\tilde{\mathcal{A}},\pi}(k,z)$ and $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,i},\tilde{\mathcal{S}},\rho}(k,z)$ is bounded by $\mu(k)$ for any security parameter k and any $i = 1, \ldots, n$. For this, construct an environment $\mathcal{Z}_\sigma$ which expects as its initial input a pair (i,z) with $i \in \{1,\ldots,n\}$, and then runs $\mathcal{Z}_{\sigma,i}$ with input z. Since the dummy-simulator $\tilde{\mathcal{S}}$ is a good simulator for the dummy-adversary $\mathcal{A}$, we have that $\mathrm{EXEC}_{\mathcal{Z}_\sigma,\tilde{\mathcal{A}},\pi}(k,(i,z))$ and $\mathrm{EXEC}_{\mathcal{Z}_\sigma,\tilde{\mathcal{S}},\rho}(k,(i,z))$ are statistically indistinguishable, so there exists a negligible function μ such that the statistical distance between $\mathrm{EXEC}_{\mathcal{Z}_\sigma,\tilde{\mathcal{A}},\pi}(k,(i,z)) = \mathrm{EXEC}_{\mathcal{Z}_{\sigma,i},\tilde{\mathcal{A}},\pi}(k,z)$ and $\mathrm{EXEC}_{\mathcal{Z}_\sigma,\tilde{\mathcal{A}},\pi}(k,(i,z)) = \mathrm{EXEC}_{\mathcal{Z}_{\sigma,i},\tilde{\mathcal{S}},\rho}(k,z)$ is bounded by $\mu(k)$ for all i.[14]

The game $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,i},\tilde{\mathcal{A}},\pi}(k,z)$ is depicted as network (III) in Fig. 1 (except that we denoted the external copy of π with π_i). Observe that $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,i+1},\tilde{\mathcal{S}},\rho}(k,z)$ (note the changed index $i+1$) contains the same machines (when unfolding the simulation performed by $\mathcal{Z}_{\sigma,i}$ into individual machines) up to the fact that the communication with the i-th instance of π is routed through the dummy-adversary $\tilde{\mathcal{A}}$. However, the latter just forwards messages, so $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,i},\tilde{\mathcal{A}},\pi}(k,z) = \mathrm{EXEC}_{\mathcal{Z}_{\sigma,i+1},\tilde{\mathcal{S}},\rho}(k,z)$.

Since the triangle inequality holds for the statistical distance, it follows that the statistical distance between $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,n},\tilde{\mathcal{A}},\pi}(k,z)$ and $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,1},\tilde{\mathcal{S}},\sigma}(k,z)$ is bounded by $n \cdot \mu(k)$ which is negligible. Thus these two families of distributions are statistically indistinguishable. Moreover, $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,n},\tilde{\mathcal{A}},\pi}(k,z)$ and $\mathrm{EXEC}_{\mathcal{Z},\mathcal{A},\sigma^\pi}$ describe the same

[14] Here we explicitly used the auxiliary input of $\mathcal{Z}$. The composition theorem, however, also holds if no auxiliary input is used. In this case, we let $\mathcal{Z}_\sigma$ choose the value i randomly. See, e.g. [36] for details.

game (up to unfolding of simulated submachines and up to one instance of the dummy-adversary). Similarly, $\mathrm{EXEC}_{\mathcal{Z}_{\sigma,1},\tilde{\mathcal{S}},\sigma}(k,z)$ and $\mathrm{EXEC}_{\mathcal{Z},\mathcal{S},\sigma^\rho}$ describe the same game (up to unfolding of simulated submachines). Thus $\mathrm{EXEC}_{\mathcal{Z},\mathcal{A},\sigma^\pi}$ and $\mathrm{EXEC}_{\mathcal{Z},\mathcal{S},\sigma^\rho}$ are statistically indistinguishable. Since the construction of $\mathcal{S}$ does not depend on $\mathcal{Z}$, we have that σ^π long-term UC realises σ^ρ. □

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, if one is willing to accept stand-alone security (i.e. simulation-based security *without* an environment, see e.g. [26]), with the extra requirement that the outputs of the parties and of the adversary/simulator are *statistically* indistinguishable in real and ideal model (long-term stand-alone security), as a minimal security requirement, we can argue as follows: If we want to fulfil this minimal security requirement *and* simultaneously get universal composability, the proof from [39] states[15] that the minimal security notion satisfying these two requirements is a security notion similar to Definition 3.3, 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 3.3 if we accept long-term stand-alone security as a minimal security notion.

3.3. *Conventions*

In all our results, we assume that *secure channels* are given for free (i.e. we are in the secure-channel network-model).[16] Further, security always denotes security with respect to *static adversaries*, i.e. parties are not corrupted *during* the protocol execution. Obviously, our negative results also apply to the stricter setting with adaptive adversaries. However, we believe that even our positive results can be adapted to that setting.

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

3.4. *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

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

[16] 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 recipient). 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.

delivery. So a phrase like "upon input x from P_1, $\mathcal{F}$ sends y to P_2" should be understood as "upon input x 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 3.5 (Common reference string (CRS)). Let $\mathcal{D}_k$ ($k \in \mathbb{N}$) be an efficiently samplable distribution on $\{0,1\}^*$. At the beginning of the protocol, the functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ chooses a value r according to the distribution $\mathcal{D}_k$ (k being the security parameter) and sends r to the adversary and all parties P_i.[17]

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 $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}_k}$.

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.[18] As we will see below, a coin toss is more powerful than a CRS in the context of long-term UC.[19]

Definition 3.6 (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.

Note that here we differ in notation from [11]. Canetti [11] defines the coin-toss to be a one-bit CRS (i.e. as $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{U}}$ where $\mathcal{U}$ is the uniform distribution on $\{0,1\}$). In particular, all parties learn the result of the coin-toss at the beginning of the protocol. In contrast, with our functionality $\mathcal{F}_{\mathrm{CT}}$ the result of the coin-toss stays secret until both P_1 and P_2 explicitly allow it to be released. This make the coin-toss functionality much more powerful than the CRS, see, e.g. [37].

[17] Here, we are slightly inexact: Strictly speaking, in the UC framework, a functionality cannot generate output without being explicitly activated. Instead, the functionality would choose and send r when first activated by some party. For simplicity, however, we use the slightly more inexact formulation here and in Definitions 3.7 and 5.8.

[18] 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.

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

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

Definition 3.7 (Public key infrastructure (PKI)). Let G be a PPT-algorithm that upon input 1^k outputs two string sk and pk.[20] When $\mathcal{F}_{\text{PKI}}^{G}$ runs with parties $P_1, \ldots, P_n$, at the beginning of the protocol, $\mathcal{F}_{\text{PKI}}^{G}$ chooses independent key pairs $(sk_i, pk_i) \leftarrow G(1^k)$ for $i = 1, \ldots, n$ and sends $(pk_1, \ldots, pk_n)$ to the adversary, and it sends $(sk_i, pk_1, \ldots, pk_n)$ to the party P_i for $i = 1, \ldots, n$.

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

Definition 3.8 (Commitment (COM)). Let C and R be two parties. The functionality $\mathcal{F}_{\text{COM}}^{C \to R, m}$ 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 3.9 (Zero-knowledge (ZK)). Let R be an NP-relation, and let P and V be two parties. The functionality $\mathcal{F}_{\text{ZK}}^{R, P \to V, m}$ behaves as follows: Upon the first input of (x, w) from P satisfying xRw and $|x| \leq m(k)$, send x to V.

We call P the prover and V the verifier.

4. Commitments

4.1. *Impossibility Results*

In this section, we will examine the possibility of long-term UC realising commitments. It will turn out that commitments cannot be long-term UC realised using CRS or coin-toss, nor with an arbitrary PKI. Note that the incompleteness of the CRS stands in stark contrast to the situation of (non-long-term) UC. In [15], it was shown that given a CRS, any functionality has a UC secure realisation. Furthermore, in [3] it was shown that the same holds for a PKI.[21] 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 4.1 (Long-term revealing). Let P be a party identifier. For a given network S, let $trans_S$ denote the transcript of all communication between a functionality $\mathcal{F}$ and all other machines (including the adversary) in an execution of $S \cup \mathcal{F}$. Let $trans_S \setminus P$ denote the transcript of all communication between $\mathcal{F}$ and all machines except P.

[20] That is, G is a key generation algorithm.

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

We say a functionality $\mathcal{F}$ is *long-term revealing* (*LTR*) *for party* P if the following holds for any network S: There is a deterministic function f_S (not necessarily efficiently computable) such that with overwhelming probability we have $trans_S = f_S(k, trans_S \setminus P)$.

The intuition behind this definition is that if $\mathcal{F}$ is long-term revealing (LTR) for P, then any secrets that P and $\mathcal{F}$ share may eventually become public. The following lemma gives some examples:

Lemma 4.2. *Coin toss* ($\mathcal{F}_{\mathrm{CT}}$) *and CRS* ($\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ *with any* $\mathcal{D}$) *are LTR for all parties. Commitment* ($\mathcal{F}_{\mathrm{COM}}$) *and ZK* ($\mathcal{F}_{\mathrm{ZK}}$) *are LTR for the recipient/verifier. If* G *is a key generation algorithm, such that the secret key depends deterministically on the public key* (*e.g. RSA, ElGamal*[22]), *the PKI* $\mathcal{F}_{\mathrm{PKI}}^{G}$ *is LTR for all parties.*[23]

Proof. In the case of coin toss and CRS, the adversary learns the random value r when 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 $pk_1, \ldots, pk_n$ sent to the adversary. □

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

Theorem 4.3 (Impossibility of commitment with LTR functionalities). *Let* $\mathcal{F}$ *be a functionality that is LTR for party* C. *Then there is no nontrivial*,[24] *polynomial-time*[25] *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 $NP \not\subseteq \mathrm{P/poly}$, this theorem is an immediate consequence of Lemma 5.7 stating that $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT},C\to R}$ (ZK for SAT with the sender C being the prover) can be realised with $\mathcal{F}_{\mathrm{COM}}^{C\to R}$, and Corollary 5.4 stating that $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT},C\to R}$ cannot be realised using $\mathcal{F}$ (both shown in Sect. 5 below). However, we instead give a direct proof (similar in spirit to that of Theorem 5.3 below) for this theorem that does not depend on the assumption $NP \not\subseteq \mathrm{P/poly}$.

Proof of Theorem 4.3. 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) such that $A_{k,z}$ and

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

[23] More exactly, for any P, $\mathcal{F}_{\mathrm{PKI}}^{G}$ is LTR for P.

[24] A protocol is called nontrivial if it always gives output if all messages are delivered and no party is corrupted. See [1] for details and more exact definitions.

[25] If $\mathcal{F}$ is not polynomial-time, we call π polynomial-time if all machines except $\mathcal{F}$ are polynomial-time.

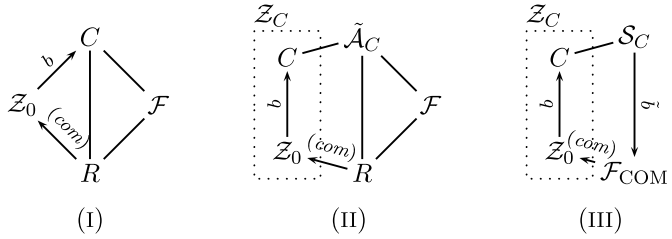

Fig. 2. Networks from the proof of Theorem 4.3.

$G(k, B_{k,z})$ 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. Fig. 2). To facilitate calculation, we use the following notation: $com_X^{k,z}(AB, CD, \ldots)$ 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. For example, $com_{\mathrm{II}}^{k,z}(R\mathcal{Z}_C, R\tilde{\mathcal{A}}_C, R\mathcal{F})$ 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 $\mathcal{F}_{\mathrm{COM}}^{C\to R,1}$ (i.e. one-bit commitment with sender C and recipient R). First, consider the following network I (depicted in Fig. 2, 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 $\tilde{\mathcal{A}}$ (see Sect. 3.1, p. 608). The environment $\mathcal{Z}_0$ behaves as follows: It takes an auxiliary input of the form (b, hold) 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 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, hold) (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 LTR for C. So, by Definition 4.1, 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 Fig. 2. Then the communication between $\mathcal{Z}_C$ and $\tilde{\mathcal{A}}_C$ consists of the following: (i) the communication of the simulated C with R 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 $\mathcal{S}_C$ such that in the network III depicted in Fig. 2 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, hold), 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

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

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

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

By definition of $\lhd$ and (1), there is a probabilistic function G such that

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

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

An interesting corollary of Theorem 4.3 is that long-term UC commitments cannot be turned around, i.e. using one (or even multiple) long-term UC commitments from A to B, one cannot long-term UC realise a commitment from B to A.

Corollary 4.4 (Commitments cannot be turned around). *There is no nontrivial, polynomial-time 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 4.2 and Theorem 4.3. □

4.2. *Commitments from Zero-Knowledge Protocols*

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

Lemma 4.5 (Commitment from ZK). *Assume that a one-way function exists. Then there is a nontrivial, polynomial-time protocol π that long-term UC realises $\mathcal{F}_{\mathrm{COM}}^{C\to R}$ (commitment with sender C) and that uses two instances of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT},C\to R}$ (ZK for SAT with the sender C being the prover).*

The (simplified) protocol π is the following:

- *To commit* to v, the sender C first commits to v using a statistically hiding commitment scheme. Then C proves (using the first instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$) that he knows the content of the commitment.
- *To unveil*, the sender C sends v to the recipient and proves (using the second instance of $\mathcal{F}_{\mathrm{ZK}}$) that he can unveil the commitment as v.

Intuitively, the long-term UC security of this protocol stems from the following two facts. Equivocality: the simulator can unveil to any value v' since he controls the second instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$. Extractability: due to the binding property of the commitment, the sender must send unveil information for the same value v to both instances of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$. Since the simulator controls the first instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$, he can extract the committed value v from the witness that is sent to the first instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$.

Proof of Lemma 4.5. Given one-way functions, there exists a computationally binding and statistically hiding commitment *COM* with a non-interactive, deterministic unveil phase (see Sect. 2). Let $V(c, v, u)$ denote the output of the unveil phase's verification algorithm when c is the transcript of the commit phase, v the (claimed) committed value, and u the unveil information.

Given a transcript c of the commit phase and a value v, we defined the circuit V^c by $V^c(v', u') := V(c, v', u')$, and the circuit $V^{c,v}$ by $V^{c,v}(u') := V(c, v', u')$. (Informally, the satisfiability of V^c shows that the commitment c can be unveiled, and the satisfiability of $V^{c,v}$ shows that c can be unveiled as v.)

We use the following protocol π:

- *Commit phase:* When the sender C receives a value v from the environment, the sender C and the recipient R execute the commit phase of *COM* on value v. Let c denote the transcript of the commit phase (known to C and R), and u the unveil information (only known to C).
- The sender C sends $(V^c, (u, v))$ to the first instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$. (That is, he proves that the commitment c can be unveiled.) When the recipient R receives V^c from the first instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$, he outputs (`committed`) to the environment.
- *Unveil phase:* When receiving (`unveil`) from the environment, the sender C sends v to the recipient R and sends $(V^{c,v}, u)$ to the second instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$. When the recipient R receives v from C and $V^{c,v}$ from $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$, he outputs v to the environment (i.e. he accepts the unveiled value v).

We have to show that π long-term UC realises $\mathcal{F}_{\mathrm{COM}}^{C\to R}$ in the following three cases:

Sender and Recipient Are Uncorrupted If all messages are delivered, when the sender C gets input v, the recipient R will output (*committed*), and when C gets input (*unveil*), R will output v. Thus π is a nontrivial protocol.

The Sender C Is Corrupted, the Recipient R Is Uncorrupted Without loss of generality, we can assume the dummy-adversary (see p. 608). Then, in the real model, the environment $\mathcal{Z}$ will send and receive protocol messages to and from R (through the dummy-adversary) and provide inputs to the instances of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$ (through the dummy-adversary), and get the outputs from R. In the ideal model, the environment $\mathcal{Z}$ will send and receive the protocol messages to and from the simulator $\mathcal{S}$ and provide the inputs for $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$ to the simulator $\mathcal{S}$. The simulator provides the inputs to the ideal functionality $\mathcal{F}_{\mathrm{COM}}^{C\to R}$ (in the name of the corrupted party C), and $\mathcal{Z}$ receives the outputs of $\mathcal{F}_{\mathrm{COM}}^{C\to R}$ (through the dummy-party R).

We construct the following simulator $\mathcal{S}$:

- *Commit phase:* To simulate the protocol interaction between R and $\mathcal{Z}$, it executes the recipient's part of *COM*. Let c denote the transcript of that interaction.
- When receiving $(V^c, (v', u'))$ from $\mathcal{Z}$ (intended for the first instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$), if $V^c(v', u') = 1$, the simulator sends v' to $\mathcal{F}_{\mathrm{COM}}^{C\to R}$. (Which then sends `committed`) to $\mathcal{Z}$.)
- *Unveil phase:* When receiving v from $\mathcal{Z}$ (intended for the recipient R), and receiving $(V^{c,v}, u)$ from $\mathcal{Z}$ (intended for the second instance of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$), if $V^{c,v}(u) = 1$, the simulator sends (`unveil`) to $\mathcal{F}_{\mathrm{COM}}^{C\to R}$. (Which then sends v' to $\mathcal{Z}$, not v.)

To see that with this simulator the real and the ideal model are statistically indistinguishable for any polynomial-time environment $\mathcal{Z}$, we introduce an intermediate game (called the intermediate model). This game proceeds like the ideal model, with the only difference that $\mathcal{F}_{\mathrm{COM}}^{C\to R}$ outputs the value v to $\mathcal{Z}$ instead of v'. By comparing the constructions of the real model and of the intermediate model, we see that even an unbounded environment $\mathcal{Z}$ has zero probability of distinguishing the real model and of the intermediate model.

We are left to show that the intermediate model and the ideal model are statistically indistinguishable for any polynomial-time environment $\mathcal{Z}$. The probability that $\mathcal{Z}$ distinguishes is bounded by the probability P_0 of the following event: $V^c(v', u') = 1$ and $V^{c,v}(u) = 1$ and $v \neq v'$ where v', u', v, u are chosen by $\mathcal{Z}$ and c is the transcript of the commit phase of *COM*. However, a polynomial-time environment $\mathcal{Z}$ producing such values v', u', v, u contradicts the binding property of *COM*. Thus P_0 is negligible, and the intermediate and the ideal model are indistinguishable.

This shows that for the case of a corrupted sender, π long-term UC realises $\mathcal{F}_{\mathrm{COM}}^{C\to R}$.

The Recipient R Is Corrupted, the Sender C Is Uncorrupted Without loss of generality, we can assume the dummy-adversary (see p. 608). Then, in the real model, the environment $\mathcal{Z}$ will send and receive protocol messages to and from C (through the dummy-adversary) and get the outputs from the instances of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$ (through the dummy-adversary), and provide the inputs for C. In the ideal model, the environment $\mathcal{Z}$ will send and receive the protocol messages to and from the simulator $\mathcal{S}$ and get the (claimed) outputs of $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT}}$ from the simulator $\mathcal{S}$. The simulator gets the outputs of the

ideal functionality $\mathcal{F}_{\text{COM}}^{C\to R}$ (in the name of the corrupted party R), and $\mathcal{Z}$ provides the inputs for $\mathcal{F}_{\text{COM}}^{C\to R}$ (through the dummy-party C).

We construct the following simulator $\mathcal{S}$:

- *Commit phase:* When receiving (`committed`) from $\mathcal{F}_{\text{COM}}^{C\to R}$, the simulator $\mathcal{S}$ simulates the protocol interaction between C and $\mathcal{Z}$ by executing the sender's part of *COM* on value $v' := 0$ (instead of the value v which was sent to $\mathcal{F}_{\text{COM}}^{C\to R}$ but not revealed to $\mathcal{S}$). Let c denote the transcript of that interaction.
- Then $\mathcal{S}$ sends V^c to $\mathcal{Z}$ (in the name of the first instance of $\mathcal{F}_{\text{ZK}}^{\text{SAT}}$).
- *Unveil phase:* When receiving v from $\mathcal{F}_{\text{COM}}^{C\to R}$, the simulator $\mathcal{S}$ sends $V^{c,v}$ to $\mathcal{Z}$.

To see that with this simulator, the real and the ideal model are statistically indistinguishable for any polynomial-time environment $\mathcal{Z}$, we introduce an intermediate game (called the intermediate model): This game proceeds like the ideal model, with the only difference that the simulator $\mathcal{S}$ runs the commit phase of *COM* on value $v' := v$ (where v is the value sent to the $\mathcal{F}_{\text{COM}}^{C\to R}$ by $\mathcal{Z}$). By comparing the constructions of the real model and of the intermediate model, we see that even an unbounded environment $\mathcal{Z}$ has zero probability of distinguishing the real model and of the intermediate model.

Since the intermediate and the ideal model only differ in the value that is used in the commit phase of *COM*, and since the unveil information produced by *COM* is never used, it follows directly from the statistical hiding property of *COM* that the intermediate and the ideal model are indistinguishable.

This shows that for the case of a corrupted recipient, π long-term UC realises $\mathcal{F}_{\text{COM}}^{C\to R}$. □

4.3. *Commitment Protocols from a PKI*

Lemma 4.2 tells us that at least for some commonly used encryption schemes, $\mathcal{F}_{\text{PKI}}^{G}$ is LTR 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.[26] However, in general this is not necessarily the case. So the question arises whether there are encryption schemes so that $\mathcal{F}_{\text{PKI}}^{G}$ 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 $\mathcal{F}_{\text{PKI}}^{G}$ 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 4.3 and 5.10 that covers all PKIs with "natural" encryption schemes. (Notice, however, that our example relies on the fact that our ElGamal variant does not include the full randomness used during key generation in the secret key. It can be argued that it would be more natural to use encryption schemes that do not erase their randomness during key generation.)

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 $x \in \{0,\ldots,2^{2k}-1\}$,[27] and a random invertible $g \in \mathbb{Z}_n$

[26] Except for nonuniformly trivial relations, see Theorem 5.10.

[27] This is probably the most uncanonical choice in our construction, since an x of length k and not $2k$ would usually be used for ElGamal.

(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 4.6. *Assume that random safe primes can be efficiently chosen and that factoring the product of two random safe primes is hard (for 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 sender C's secret key and (n, g, h) 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.

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 statistically hiding, because there are many different x' of length $2k$ satisfying $h = g^{x'} \bmod n$, and for a random such x', we have that $x' \bmod 3$ is almost uniformly 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 $b := c - x \bmod 3$.

Proof of Lemma 4.6. We use the protocol π given in the proof sketch above (directly after the statement of Lemma 4.6). Obviously, an honest sender C always succeeds in unveiling with an honest recipient 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.[28] 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 [4, 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 $x' \in \{0, \ldots, 2^{2k} - 1\}$ 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).

Since n is a *safe* prime, $\varphi(n) = 4p'q'$ 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 precisely 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 $\mathcal{F}_{\mathrm{COM}}^{1}$. □

Note that the protocol given here only shows that we cannot expect a generalisation of Theorem 4.3 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, not even further commitments are possible. It would be interesting to know whether this is an artifact of our particular example, or whether for any PKI there is a fixed upper bound on the number of long-term UC secure commitments that can be realised from it. In contrast to this situation, in Sect. 6 we present functionalities that can be used for an arbitrary number of commitments/ZK protocols.

From Lemma 4.6, we additionally get the following result:

Corollary 4.7. *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*

[28] 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.

functionality $\mathcal{F}$ that is LTR for B such that 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 ($\mathcal{F}_{\mathrm{ZK}}^{R,A\to B}$), 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 $\mathcal{F}_{\mathrm{PKI}}^{G_{amal}}$.[29] Since the protocol π from Lemma 4.6 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 LTR 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 5.7, 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 $\mathcal{F}_{\mathrm{CT}}^{m}$.[30] By Theorem 5.5, we get $\mathcal{F}_{\mathrm{ZK}}^{R,B\to A}$ for the NP-relation R from Theorem 5.5. Since by assumption factoring the product of two random safe primes is hard, R is nonuniformly nontrivial. □

4.4. *Statistically Hiding UC Commitments Are Not Always Long-Term UC*

In Theorem 4.3, we showed that no long-term UC secure commitment schemes exist that use only a CRS. On the other hand, Damgård and Nielsen [18] give a commitment scheme based on a CRS which is both computationally UC secure (in the sense of Definition 3.1) and statistically hiding. The latter property seems to imply some form of long-term security. Therefore, the commitment both securely composes and is long-term secure; we would hence intuitively expect it to be long-term UC secure. In this section, we resolve this seeming contradiction by explaining why the commitment scheme from [18] (DN-commitment for short) is not long-term UC and by arguing that the long-term security of that scheme actually gets lost when composing the protocol. Note that the present section is only aimed at giving an intuition about the problem of long-term UC commitments, it does not contain any formal statements or proofs.

The DN-Commitment We first give a highly simplified presentation of the DN-commitment scheme. This presentation contains many omissions, but it should be sufficient to understand both the idea underlying the DN-commitment scheme and the discussions in this section. The DN-commitment builds upon another commitment scheme called the mixed commitment. This is a non-interactive commitment COM_K parametrised by a public key $K \in \mathcal{K}$ and a system key N that has the following properties:

- For a uniformly chosen key $K \in \mathcal{K}$, COM_K is a computationally binding and hiding commitment scheme.
- There is a subset $E \subseteq \mathcal{K}$ of keys such that for a uniformly chosen key $K \in E$, we have that COM_K is a statistically hiding, equivocable commitment scheme. More precisely, the party choosing $K \in E$ can choose it together with a trapdoor that enables the sender to unveil to any value.

[29] That is, m public/secret key pairs are generated for each party.

[30] 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.

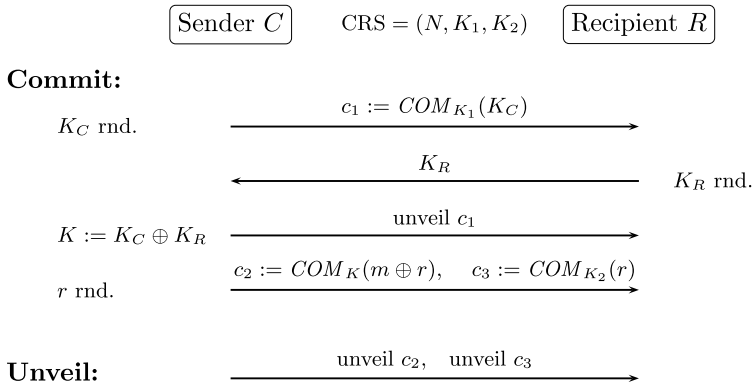

Fig. 3. The DN-commitment scheme.

- For a uniformly chosen key $K \in \mathcal{K} \setminus E$, we have that COM_K is an extractable commitment scheme, more precisely, a party knowing the trapdoor for the system key N can extract the message.
- Keys chosen uniformly from E are indistinguishable from keys chosen uniformly from $\mathcal{K}$. The size of E is only a negligible fraction of the size of $\mathcal{K}$ (i.e. random keys are almost always in $\mathcal{K} \setminus E$).

Then a DN-commitment to a message m is performed as follows (see also Fig. 3):

- The CRS is assumed to contain the system key N and two uniformly chosen keys $K_1, K_2 \in E$.
- **Commit phase.** The sender C chooses a uniform key K_C and sends a message $c_1 := COM_{K_1}(K_C)$ to the recipient R.
- The recipient chooses a uniform key K_R and sends it to the sender.
- The sender unveils c_1.
- Let $K := K_C \oplus K_R$. The sender chooses a uniform r and sends $c_2 := COM_K(m \oplus r)$ and $c_3 := COM_{K_2}(r)$ to the recipient.
- **Unveil phase.** The sender unveils c_2 and c_3.

Security of the DN-Commitment To see that this scheme is indeed UC secure, let us first assume a corrupted sender. In this case, the simulator will have to simulate the messages of the recipient R in such a manner that it can extract the message m already during the commit phase. For this, the simulator chooses the CRS differently: It chooses $K_2 \in \mathcal{K} \setminus E$ (instead of $K_2 \in E$). From the properties of mixed commitments it follows that this CRS is indistinguishable from the original one. Further, note that the first three messages constitute a secure coin-toss, so K will be uniformly distributed over $\mathcal{K}$, and therefore will be in $\mathcal{K} \setminus E$ with overwhelming probability. Thus both $K, K_2 \in \mathcal{K} \setminus E$ and therefore the simulator can extract (using the trapdoor for the system key N) the messages from c_2 and c_3 and compute m.

Now consider a corrupted recipient. In this case, the simulator will have to simulate the messages of the sender C in such a manner that it can unveil to any message m (which it may only learn after the commit phase). For this, the sender cheats during the

coin toss (the first three messages) to get a key $K \in E$. The simulator can do this because he knows the trapdoor for K_1 and can therefore unveil c_1 to any K_C after learning K_R. Since $K \in E$, the simulator can unveil c_2 to any value $\tilde{m}$ during the unveil phase. And since the unveiled value m is then computed as $\tilde{m} \oplus r$ by the recipient, this implies that the simulator can unveil to any value m.

Finally, we observe that the DN-commitment is statistically hiding: Since $K_2 \in E$, the commitment c_3 is statistically hiding. Therefore, even an unbounded recipient might learn $m \oplus r$ from c_2, but it cannot learn r from c_3, thus it will not learn m.

DN-Commitments Are Not Long-Term UC We know from Theorem 4.3 that the DN-commitment cannot be long-term UC secure since it only uses a CRS. However, a deduction from general principles is not so instructive when considering a concrete protocol. We therefore ask why this particular protocol is not long-term UC. The answer turns out to be simple: Although the protocol is statistically hiding, the simulator described above does one of the following:

- If the sender is corrupted, the simulator chooses the key K_2 contained in the CRS uniformly from $\mathcal{K} \setminus E$ instead of E. This gives a statistically distinguishable distribution of the CRS.
- If the recipient is corrupted, the simulator cheats when performing the coin toss such that the key K will always be chosen from E. Since in the real execution K would be uniformly distributed from $\mathcal{K}$, this leads to statistically distinguishable protocol executions in the real and the ideal model.

In other words, although the DN-commitment is statistically hiding, it is not possible to extract or to unveil to a different value (equivocation) without producing a statistically distinguishable view. Therefore, the DN-commitment is not long-term UC secure.

An alternative intuition is that both an extraction and an equivocation trapdoor need to be simultaneously present in a protocol execution. (Otherwise the environment could distinguish between executions with and without these trapdoors.) But the existence of an extraction trapdoor implies that the committed value is information-theoretically fixed, which in turn contradicts the equivocality.

Long-Term Security Is Lost Under Composition. The fact that the DN-commitment is not long-term UC secure may be interpreted in two ways. The first interpretation is that the DN-commitment it too weak for guaranteeing long-term security under composition. The second interpretation is that the DN-commitment has all properties we want, but the notion of long-term UC security is too restrictive. We will now argue that the first interpretation is the right one by giving an example where the DN-commitment looses its long-term security under composition. More precisely, we present a protocol π that statistically hides Alice's input m when using an ideal commitment functionality, but that completely reveals m to an unbounded adversary when using the DN-commitment instead of the ideal functionality. (The adversary needs to be unbounded only after the protocol execution, of course.)

In [18], all proposed instantiation of the mixed commitments schemes are of the following form: The system key N contains two groups G and H, a homomorphism $f : G \to H$, and an element $g \in H$. Let $F := f(G)$ and let $\mathrm{ord}_F\, g$ be the smallest positive

integer with $g^i \in F$ (i.e. $\mathrm{ord}_F\, g$ is the order of gF in the quotient group H/F). The system key satisfies $H = \{g^i f(r) : i \in \mathbb{Z}, r \in G\}$ (in other words, gF generates H/F), and the smallest prime factor of $\mathrm{ord}_F\, g$ is superpolynomial in the security parameter. In the following, we will assume for simplicity that $\mathrm{ord}_F\, g$ is actually prime.[31] A public key is an element K of $\mathcal{K} := H$. A commitment to m is computed as $COM_K^N(m) := K^m f(r)$ for random $r \in G$; to unveil, we reveal m and r. The set E of keys suitable for equivocation is $E = F$, hence all keys $K \in \mathcal{K} \setminus E$ are of the form $g^i f(r)$ for some $r \in G$ and $i \in \{1, \ldots, \mathrm{ord}_F\, g - 1\}$. We omit further details of the scheme (like the structure of an extraction or equivocation trapdoor) as they are not necessary for understanding the example.

Consider the following protocol π in the $\mathcal{F}_{\mathrm{COM}}$-hybrid model:

- Alice has an input m.
- Bob chooses a system key $\tilde{N}$ and a public key $\tilde{K} \in \mathcal{K} \setminus E$ (together with an extraction trapdoor). We denote by G, H, f, g the various components of the system key $\tilde{N}$. Then Bob sends $(\tilde{N}, \tilde{K})$ to Alice.
- Alice chooses v in the message space of the commitment at random.[32] Then Alice sends $c^* := COM_{\tilde{K}}^{\tilde{N}}(v)$ to Bob.
- Bob extracts v from c^* using the extraction trapdoor.
- Bob commits to v using $\mathcal{F}_{\mathrm{COM}}$.
- Alice unveils c^*.
- Bob unveils $\mathcal{F}_{\mathrm{COM}}$.
- Alice checks whether the message received from $\mathcal{F}_{\mathrm{COM}}$ is v. If so, Alice picks $t \in \{0, \ldots, \mathrm{ord}_F\, g\}$[33] and $u \in G$ and sends $\gamma := g^m \tilde{K}^t f(u)$ to Bob.

We claim that in the protocol π, Alice's input m is statistically hidden, even given an unbounded Bob. (Moreover, π statistically UC emulates the functionality $\mathcal{F}_{\mathrm{null}}$ that takes an input from Alice and erases it.) To see this, consider a malicious Bob. If Alice does not send γ, her input m is obviously hidden (as it is never used). Thus we can concentrate on the case where Bob lets $\mathcal{F}_{\mathrm{COM}}$ unveil as v. However, even an unlimited Bob can only do this if c^* is not statistically hiding. However, if $\tilde{K} \in E$, then $COM_{\tilde{K}}^{\tilde{N}}$ is equivocable and hence statistically hiding. Thus $\mathcal{F}_{\mathrm{COM}}$ will only unveil as v if $\tilde{K} \in \mathcal{K} \setminus E$ (except for negligible probability). But in this case, $\tilde{K} = g^i f(r)$ for some $i \in \{1, \ldots, \mathrm{ord}_F\, g - 1\}$ and $r \in G$. Hence every element of H can be expressed as $\tilde{K}^t f(u)$, and $g^m \tilde{K}^t f(u)$ is independent of m for random t and u. Thus π statistically hides m.

Now assume that $\mathcal{F}_{\mathrm{COM}}$ is implemented using a DN-commitment. We claim that the resulting protocol π^{DN} is not unconditionally hiding any more. Instead, m mod $\mathrm{ord}_F\, g$ can be (inefficiently) computed from the view of an efficient malicious Bob. This malicious Bob performs the following steps. Let (N, K_1, K_2) denote the CRS used in the DN-commitment. Then Bob sets $\tilde{N} := N$ and $\tilde{K} := K_2$ and sends $(\tilde{N}, \tilde{K})$ to Alice.

[31] The construction below also works if $\mathrm{ord}_F\, g$ is not a prime (but is guaranteed to have superpolynomial prime factors). Yet, to keep things simple, we assume in the following that $\mathrm{ord}_F\, g$ is prime.

[32] We assume that the message space of the commitment is superpolynomially large. Some of the instantiations in [18] do not have large message spaces; in this case, we extend the message space by sending several commitments in parallel.

[33] If $\mathrm{ord}_F\, g$ cannot be efficiently computed, a sufficiently large random $t \gg \mathrm{ord}_F\, g$ can be chosen instead.

Bob then receives $c^* = COM^{\tilde{N}}_{\tilde{K}}(v) = COM^{N}_{K_2}(v)$. However, Bob cannot extract v from c^*. Then Bob performs the DN-commitment as in Fig. 3, except that he uses $c_2 := COM_K(0)$ and $c_3 := c^*$. Then Alice unveils c^* as v (as prescribed by the protocol π). Bob can now open c_2 as 0 and $c_3 = c^*$ as v. Since $0 \oplus v = v$, this means that Bob unveils the DN-commitment as v. Thus Alice sends $\gamma = g^m \tilde{K}^t f(u) = g^m K_2^t f(u)$. Since K_2 is chosen to lie in E (by definition of the DN-commitment), we have that $K_2^t f(u) \in E = F$. Thus $\gamma = g^m K_2^t f(u)$ determines m modulo $\mathrm{ord}_F\, g$ (by definition of $\mathrm{ord}_F\, g$). Since γ is in the view of Bob, an unlimited machine can extract m mod $\mathrm{ord}_F\, q$ from the view of Bob. Thus π^{DN} is not long-term secure in any reasonable sense.

5. Zero-Knowledge Protocols

In the present section, we examine to what extent long-term UC secure zero-knowledge arguments can be implemented using various functionalities. Besides several impossibility results, we also find a quite surprising possibility result (Theorem 5.5).

5.1. *Impossibility when Using LTR Functionalities*

First, analogously to our investigations concerning commitments in Sect. 4, we will now examine whether long-term UC secure zero-knowledge protocols can be implemented using functionalities that are LTR for one of the parties.

Whether long-term UC realising zero-knowledge protocols for some relation R exist strongly depends on the relation R under consideration. For example, for trivial relations R, such zero-knowledge protocols do, of course, exist. The following definition specifies a class of relations which is going to play an important role in our results:

Definition 5.1 (Essentially unique witnesses). An NP-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 $(x_k, w_k) \in R$, the probability $P((x_k, U_R(1^k, x_k, w_k)) \in R)$ 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 $(x_k, w_k^1) \in R$ and $(x_k, w_k^2) \in R$, 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.

The notion of essentially unique witnesses can best be illustrated by considering the example of the discrete logarithm: Given a group $G = \langle g \rangle$, for some $x \in G$ a witness is some $w \in \mathbb{Z}$ such that $x = g^w$. Such a witness is not unique, since any $w' = w + n \,\mathrm{ord}\, g$ is a witness. However, in a sense all these witnesses are equivalent, since from one witness we can efficiently compute any other witness (assuming we know $\mathrm{ord}\, g$). So, essentially, the witness for x is unique. Therefore, the relation $x R w :\iff x = g^w$ has essentially unique witnesses. The above definition slightly generalises this example by also allowing negligible error probabilities.

A possible way to interpret the witness unifier is as a statistically witness indistinguishable argument that simply sends a witness in the clear, so relations with essentially unique witnesses are relations with trivial statistically witness indistinguishable arguments.

Our impossibility result below holds only for relations *without* essentially unique witnesses. Since this result would trivialise if there were no such relations, we first show that relations without essentially unique witnesses are indeed likely to exist:

Lemma 5.2. *If one-way-functions* (*secure against uniform adversaries*) *exist*, or *if* $NP \not\subseteq \mathrm{P/poly}$, *then SAT does not have essentially unique witnesses*.

Proof. Assume that SAT has essentially unique witnesses. Let R be the following relation: For two circuits f_1, f_2, one has $(f_1, f_2)Rw$ 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 5.1.

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). Thus w is a witness of f or of f'. Further, when the input of A is $h(\tilde{w})$ for a uniformly chosen $\tilde{w} \in \{0,1\}^n$, the circuits 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 $NP \not\subseteq \mathrm{P/poly}$. 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) such that 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 $f \in M$ such that $f(A(f, M)) = 1$ 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 $M_n^{3k} = \varnothing$.

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

- For each $i = 0, \ldots, 3k-1$, 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 $(f'_0, w'_0, \ldots, f'_{3k-1}, w'_{3k-1}) \leftarrow \bar{\mathcal{U}}_k^*$.
- For each $i = 0, \ldots, 3k-1$, 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^{**} such that 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 A.1, 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. □

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

Theorem 5.3 (Impossibility of zero-knowledge with LTR functionalities). *Let R be an NP-relation* without *essentially unique witnesses. Let $\mathcal{F}$ be a functionality that is LTR for party P. Then there is no nontrivial, polynomial-time*[34] *protocol that long-term UC realises zero-knowledge 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 then (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 5.1. Therefore, R must

[34] If $\mathcal{F}$ is not polynomial-time, we call π polynomial-time if all machines except $\mathcal{F}$ are polynomial-time.

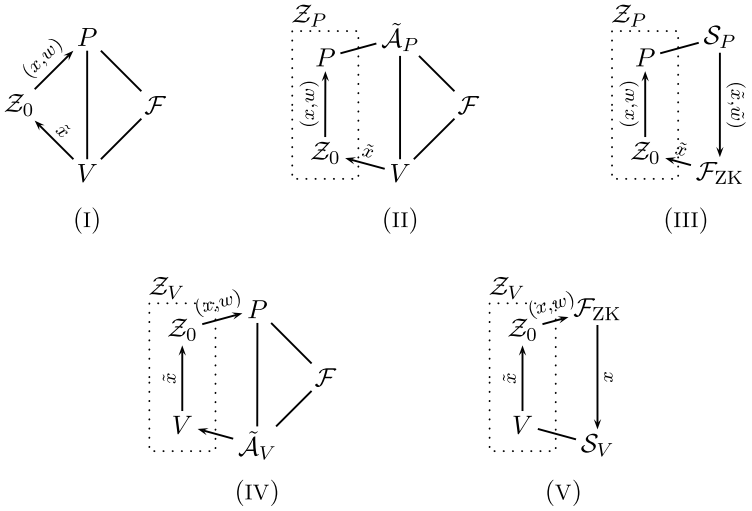

Fig. 4. Networks from the proof of Theorem 5.3.

have essentially unique witnesses, which gives the desired contradiction. The details are given in the following proof.

Proof of Theorem 5.3. In this proof, we again use the $\lhd$-notation and the $com_X^{k,z}(\ldots)$-notation presented in the proof of Theorem 4.3: 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) such that $A_{k,z}$ 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. Fig. 4). To facilitate calculation, we use the following notation: $com_X^{k,z}(AB, CD, \ldots)$ 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. For example, $com_{\mathrm{II}}^{k,z}(V\mathcal{Z}_P, V\tilde{\mathcal{A}}_P, V\mathcal{F})$ denotes all communication of party V in network II.

To produce a contradiction, we assume that there is a nontrivial protocol π that long-term UC realises $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V,m}$ 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 Fig. 4, 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 $\tilde{\mathcal{A}}$ (see Sect. 3.1, p. 608). The environment $\mathcal{Z}_0$ behaves as follows: It takes its auxiliary input (x, w) and 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 $|x| \leq m(k)$. 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 Fig. 4. Then the communication between $\mathcal{Z}_P$ and $\tilde{\mathcal{A}}_P$ 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 $\mathcal{S}_P$ such that in the network III depicted in Fig. 4 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 have that $com_{\mathrm{III}}^{k,x,w}(\mathcal{Z}_P\mathcal{S}_P)$ and $com_{\mathrm{II}}^{k,x,w}(\mathcal{Z}_P\tilde{\mathcal{A}}_P)$ are statistically indistinguishable and hence

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

Note that the following fact holds 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 $|x| \leq m(k)$ 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 LTR for P. Then, by Definition 4.1, 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 Fig. 4. 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 $\mathcal{S}_V$ such that the output of $\mathcal{Z}_V$ in networks IVand V(cf. Fig. 4) are statistically indistinguishable. It follows that

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

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 $x_k R w_k^1$ and $x_k R w_k^2$. 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,

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

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

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

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 5.1. 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 5.1 (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. □

Note that we cannot expect a result analogous to Theorem 5.3 in the case that $\mathcal{F}$ is LTR for the *verifier* V, since commitments are LTR for the recipient and Lemma 5.7 shows 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 zero-knowledge protocols for SAT:

Corollary 5.4. *Let $\mathcal{F}$ be a functionality that is LTR for party P. If one-way-functions (secure against uniform adversaries) exist,* or *if NP $\not\subseteq$ P/poly, then there is no nontrivial, polynomial-time long-term UC secure protocol for zero-knowledge with prover P for SAT ($\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT},P\to V}$) in the $\mathcal{F}$-hybrid model.*

Proof. Immediate from Lemma 5.2 and Theorem 5.3. □

At this point, one might ask why our impossibility result from Theorem 5.3 needs the restriction to relations without essentially unique witnesses. Would the following argumentation not show that given, say, a coin-toss, there is no long-term UC zero-knowledge 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, so that interaction both "contains" and does not "contain" w, which gives a contradiction. Surprisingly, this intuition is not sound as the following possibility result shows. It implies that in Theorem 5.3, the condition that the relation does not have essentially unique witnesses is indeed necessary.

Theorem 5.5 (Long-term UC 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, polynomial-time protocol π using two instances of $\mathcal{F}_{\mathrm{CT}}$ that long-term UC emulates $\mathcal{F}_{\mathrm{ZK}}^{R}$.*

Note that the main intent of this theorem is to show that our impossibility result from Theorem 5.3 cannot be strengthened. We do not claim that a protocol for $\mathcal{F}_{\mathrm{ZK}}^{R}$ (with R as in Theorem 5.5) has many applications: Since $\mathcal{F}_{\mathrm{ZK}}^{R}$ can be implemented using a functionality that is LTR, nothing can be implemented from $\mathcal{F}_{\mathrm{ZK}}^{R}$ that cannot be implemented from functionalities that are LTR. (It is, however, conceivable that $\mathcal{F}_{\mathrm{ZK}}^{R}$ might have some applications for implementing long-term UC secure identification schemes.)

To construct a protocol π as in Theorem 5.5, 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 such that 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 $r_1, \ldots, r_k$ 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.[35]
- The prover sends $s_1, \ldots, s_k$ 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 $r_1, \ldots, r_n$ that are statistically 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 $r_1, \ldots, r_n$, 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 A.2). 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 $r_1, \ldots, r_n$ 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 $r_1, \ldots, r_n$ 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).

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 $\mathcal{F}_{\mathrm{ZK}}^R$

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

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 $r_1, \ldots, r_n$. 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 $r_1, \ldots, r_n$ 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 [24]: 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 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 following proof gives the full details.

Proof of Theorem 5.5. Let f_k be a one-way permutation on $\{0,1\}^k$. Let SWIAOK be a statistically witness indistinguishable argument of knowledge (such argument systems exist for any NP-relation under the assumptions of the theorem, cf. Sect. 2). 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 $p, q, \bar{r}^*$ such that $nR(p, q)$ 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 $r_1, \ldots, r_k$ 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 $s_i \in \{s_i \in \{0, \ldots, n-1\} : s_i^2 \equiv r_i \bmod n\}$. If for some i no such s_i exists, let $s_i := \perp$.[36]
8. P sends $s_1, \ldots, s_n$ 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:

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

- 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 ZK-protocol).
- There is a simulator for the case that the prover P is corrupted (this roughly corresponds to the knowledge-soundness of the ZK-protocol).
- There is a simulator for the case that the verifier V is corrupted (this roughly corresponds to the zero-knowledge-property of the ZK-protocol).

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 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.

Without loss of generality, we can assume a dummy-adversary (see Sect. 3.1, p. 608).

We now consider the case that the prover P is corrupted. Then we have to find a simulator $\mathcal{S}_P$ such that 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 outputs n, the simulator has to send (p, q, n) to the ideal functionality $\mathcal{F}_{\text{ZK}}^{R}$ 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 $\mathcal{F}_{\text{CT}}$ is the real model).
- When prover and verifier invoke the second coin-toss, the resulting value r is chosen as the concatenation of $r_1, \ldots, r_k$. To chose the r_i, the algorithm Q from Lemma A.2 is invoked and returns $(r_i, \tilde{s}_i)$ where $\tilde{s}_i$ 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 $s_i \neq \perp$?

  If one of these tests succeeds, the simulator knows a nontrivial factor of n and can send (p, q, n) to $\mathcal{F}_{\text{ZK}}$ (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 simulated verifier does not abort in Step 5, then the number n given by the environment machine to the simulator is a Blum-integer. So, by Lemma A.2, 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 an honest verifier would, 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 A.2 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 $\gcd(s_i - \tilde{s}_i, n)$ is a nontrivial factor of n. So the simulator $\mathcal{S}_P$ successfully simulates.

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 $\mathcal{F}_{\mathrm{ZK}}$ 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 A.2 k times and gets $r_1, \ldots, r_k$ and $s_1, \ldots, s_k$. 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 $\bar{r}^*$ 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 f_k is a permutation, the value $\bar{r}$ has the same distribution as the value $\bar{r}$ produced in the real model. Since n is always a Blum-integer, Lemma A.2 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 with the real and with the simulated prover are statistically indistinguishable.

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

Actually, we can somewhat strengthen the result from Theorem 5.5 and get a long-term UC secure zero-knowledge protocol that does not only show the existence of a factorisation of n, but can also show that the factorisation satisfies some predicate. Such a zero-knowledge protocol might, for example, be useful when proving statements about RSA-based encryptions or signatures.

Corollary 5.6. *Assume that a one-way permutation exists. Let X be any predicate that can be decided in deterministic polynomial time. Let R be as in Theorem 5.5. 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 instances 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 5.5. 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 5.7 (ZK from commitment). *Let R be an NP-relation. Then there is a long-term UC secure protocol π for zero-knowledge with relation R (i.e. realising $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$) using a polynomial number of commitments from prover P to verifier V (i.e. $\mathcal{F}_{\mathrm{COM}}^{P\to V}$).*

Proof. Canetti and Fischlin [12] give 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 LTR for the recipient and thus Theorem 5.3 applies.

5.2. *Impossibility when Using Offline Functionalities*

In the preceding section, we saw that using a coin toss, long-term UC secure zero-knowledge protocols 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 5.8 (Offline functionalities). We call a functionality $\mathcal{F}$ offline, if it has the following form: When $\mathcal{F}$ runs with parties $P_1,\ldots,P_n$, at the beginning of the protocol, $\mathcal{F}$ chooses values $(c, c_{P_1},\ldots,c_{P_n})$ according to a fixed distribution (that may depend on the security parameter) and sends c to the adversary and sends c_{P_i} to P_i for $i = 1,\ldots,n$.

Lemma 5.9. *CRS and PKI are offline functionalities.*

Proof. For $\mathcal{F}_{\text{CRS}}$, set $c := c_i := r$ (cf. Definition 3.5), and for $\mathcal{F}_{\text{PKI}}$, set $c := (pk_1, \ldots, pk_n)$ and $c_i := (sk_i, pk_1, \ldots, pk_n)$ (cf. Definition 3.7). □

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 4.2) cannot be used for long-term UC secure ZK for any relation R unless that relation is trivial for nonuniform algorithms anyway.

Theorem 5.10 (Impossibility of ZK with LTR offline functionalities). *Let R be a nonuniformly nontrivial NP-relation.*[37] *Let $\mathcal{F}$ be an* offline *functionality that is LTR for party P and for party V. Then there is no nontrivial, polynomial-time protocol that long-term UC realises ZK for relation R with prover P and verifier V (i.e. $\mathcal{F}_{\text{ZK}}^{R,P\to V}$) 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 $\mathcal{F}_{\text{ZK}}^{R}$. Then there is a simulator $\mathcal{S}_1$ that is able to choose the CRS r_1 and calculate a corresponding trapdoor T_1 such that 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 such that he can simulate the verifier and—if the verifier accepts—extract a witness w. Since both r_1 and r_2 must, in the case of long-term UC, be 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.

Proof of Theorem 5.10. 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 LTR for party P and for party V.

Since $\mathcal{F}$ is an offline functionality, we can assume without loss of generality 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. Without loss of generality, 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).

[37] That is, there is no nonuniform deterministic polynomial-time algorithm that finds witnesses for R.

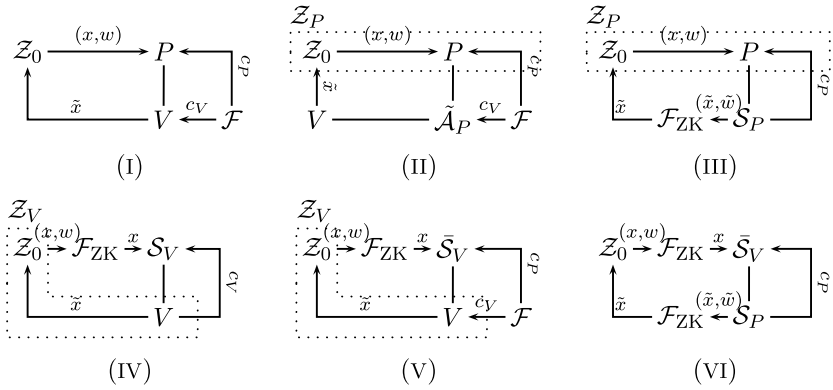

Fig. 5. Networks from the proof of Theorem 5.10.

Consider the following network I (shown in Fig. 5, the dummy-adversary is omitted for simplicity): The parties P and V run uncorrupted with an environment $\mathcal{Z}_0$ and the dummy-adversary $\tilde{\mathcal{A}}$ (see Sect. 3.1, p. 608). 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. 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 Fig. 5. Since π is long-term UC secure, there is a simulator $\mathcal{S}_P$ such that the output of $\mathcal{Z}_P$ is statistically indistinguishable in networks II and III (cf. Fig. 5). 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 $\tilde{\mathcal{A}}_V$. Without loss of generality, 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 simu-

lator $\mathcal{S}_V$. So we get the network IV shown in Fig. 5. 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 $\mathcal{D}_k$ such that 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) such that 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 LTR for V, there is a function f such that $c_V = f(c_P)$. 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 $\tilde{\mathcal{S}}_V$ 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 Fig. 5. 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 $\tilde{\mathcal{S}}_V$ 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 LTR 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 A.3, 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. $x R \tilde{w}$) as long as $x R w$ 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 $\tilde{\mathcal{S}}_V$ 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 $x R w$. 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 A.1, witnesses for R can be found by a nonuniform deterministic polynomial-time algorithm, so R is nonuniformly trivial, which gives us a contradiction and proves the theorem. □

A natural question arising in this context is whether the impossibility result from Theorem 5.10 can be made stronger. In particular, one might ask whether such an impossibility result already holds if $\mathcal{F}$ is LTR for P *or* for V. This, however, is refuted by Corollary 4.7. 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 5.11. *Assume that a one-way permutation exists. Let γ be an efficiently computable function from Σ^* to $\mathbb{N} \cup \{\bot\}$ such that $\gamma(x)$ 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 5.5. Let $(n, x) R_\gamma (p, q)$ iff $n R(p, q)$ 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 a γ exists, so that R_γ is uniformly nontrivial. So to strengthen Theorem 5.10, one would have to disprove the existence of such a γ. Since such a γ exists relative to an oracle if factoring Blum-integers is hard,[38] we get an alternative interpretation of Corollary 5.11: A strengthening of Theorem 5.10 to *uniformly* nontrivial relations would be non-relativizing or show that factoring is not hard.

We wish to stress that Corollary 5.11 should not be seen as a positive result since there is no candidate for γ. Also, it does not seem that a protocol realising $\mathcal{F}_{\mathrm{ZK}}^{R_\gamma}$ would have any applications. The purpose of the result is solely to show the limitations of Theorem 5.10.

The rough proof idea for Corollary 5.11 is the following: Recall why protocol π from Theorem 5.5 needs a coin-toss instead of a CRS. The simulator had to choose the value $r = (r_1, \ldots, r_k)$ of the second invocation of the coin-toss functionality in a manner so that it knew 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 a coin-toss, we use a different CRS $r^{(|x|)} = (r_1^{|x|}, \ldots, r_k^{|x|})$ 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 following proof makes these ideas more concrete.

[38] This follows from Lemma A.4 with $\mathcal{D}_n$ being the uniform distribution on n-bit Blum-integers and $x R w$ denoting that w is a nontrivial factor of x. Note, however, that this construction of γ is not constructive; we know of no concrete candidate.

Proof of Corollary 5.11. To implement $\mathcal{F}_{\mathrm{ZK}}^{R_\gamma, P\to V, m}$ using a CRS, we use the following protocol π (the notation is as in the proof of Theorem 5.5):

1. A CRS $\underline{\mathbf{r}} = (\bar{r}, r^{(0)}, \ldots, r^{(m(k))})$ 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 $(n, x) R_\gamma (p, q)$. Otherwise he aborts.
4. P sends (n, x) to V.
5. P proves using the SWIAOK the knowledge of $p, q, \bar{r}^*$ such that $(n, x) R_\gamma (p, q)$ or $f_k(\bar{r}^*) = \bar{r}$.
6. $r^{(|x|)}$ is split into $r_1, \ldots, r_k$, 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 $s_i \in \{s_i \in \{0, \ldots, n-1\} : s_i^2 \equiv r_i \bmod n\}$. If for some i no such s_i exists, let $s_i := \bot$.[39]
8. P sends $s_1, \ldots, s_n$ 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 5.5, the first component $\bar{r}$ is chosen randomly if P is corrupted, and $\bar{r} = f_k(\bar{r}^*)$ for random $\bar{r}^*$ if V is corrupted.

For $\mu = 0, \ldots, m(k)$, the simulator (both in case of a corrupted V and of a corrupted P) invokes the algorithm Q from Lemma A.2 on input $n := \gamma(0^\mu)$. Then Q outputs $r_1, \ldots, r_k$ of length $|n| + k$ together with random square roots $s_1, \ldots, s_k$ (or $s_i = \bot$, if no root exists). Then $r^{(\mu)}$ is chosen as the concatenation of $r_1, \ldots, r_k$. The s_i are stored. Finally, the CRS is set to $\underline{\mathbf{r}} := (\bar{r}, r^{(0)}, \ldots, r^{(m(k))})$.

Except for this modification, the simulators are constructed analogously to the simulators in the proof of Theorem 5.5, and the proof of security is analogous to that of Theorem 5.5 (note that for any $(n, x) \in L_{R_\gamma}$, 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). □

5.3. *Generalising LTR*

In Sect. 4.1, we have introduced the notion of long-term revealing (LTR) functionalities (Definition 4.1). We have shown that the cryptographic power of such functionalities is very limited in the long-term UC setting; they cannot even be used to implement a commitment or a ZK protocol for SAT. It seems, however, that the notion of LTR functionalities can be further generalised. For example, assume that a functionality $\hat{\mathcal{F}}$ is LTR for Alice, and that $\mathcal{F}$ behaves like $\hat{\mathcal{F}}$, except that it additionally tosses a random coin b and sends b to Alice. Then $\mathcal{F}$ is not LTR for Alice because given the view of Bob, one cannot compute b. It does seem, however, that the fact that Alice gets a bit b from $\mathcal{F}$ does not have any impact on the cryptographic power of $\mathcal{F}$. Alice could have chosen b herself.

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

Indeed, it is easy to show that $\mathcal{F}$ cannot be used to implement commitments or ZK protocols for SAT. There is a protocol π in the $\hat{\mathcal{F}}$-hybrid model that long-term UC realises $\mathcal{F}$. Namely, π picks b at random, sends b to Alice, and forwards all other communication to $\hat{\mathcal{F}}$. Thus, if there were a protocol σ long-term UC realising, say, a commitment in the $\mathcal{F}$-hybrid model, then the composition theorem (Theorem 3.2) would give us a protocol σ^{π} realising a commitment in the $\hat{\mathcal{F}}$-hybrid model. This would contradict Theorem 4.3 since $\hat{\mathcal{F}}$ is LTR for Alice.

To formulate this reasoning in a more general fashion, we introduce the following notation: Given a functionality $\mathcal{F}$ and a polynomial p, let $\mathcal{F}^p$ denote the *multi-session variant* of $\mathcal{F}$. That is, $\mathcal{F}^p$ internally simulates p copies of $\mathcal{F}$. When $\mathcal{F}^p$ receives a message (i, m) with $i \in \{1, \ldots, p\}$ from a machine M, it forwards m to the ith copy of $\mathcal{F}$. When the ith copy of $\mathcal{F}$ sends a message m' to M, $\mathcal{F}^p$ sends that message as (i, m) to M.

Definition 5.12 (Generalised LTR). We say a functionality $\mathcal{F}$ is generalised LTR for a party P if for every polynomial p, there exists a functionality $\hat{\mathcal{F}}$ that is LTR for P and there exists a nontrivial, polynomial-time[40] protocol π in the $\hat{\mathcal{F}}$-hybrid model that long-term UC realises $\mathcal{F}^p$.

Note that if $\mathcal{F}$ and $\hat{\mathcal{F}}$ are polynomial-time, it is sufficient to show that $\mathcal{F}$ can be realised from $\hat{\mathcal{F}}$; the realisability of $\mathcal{F}^p$ follows from the composition theorem (Theorem 3.4).

Examples for functionalities that are generalised LTR for a party P are the functionalities $\mathcal{F}_{\mathrm{ZK}}^{R}$, $\mathcal{F}_{\mathrm{ZK}}^{R'}$ and $\mathcal{F}_{\mathrm{ZK}}^{R_\gamma}$ (with prover P) as defined in Theorem 5.5, Corollary 5.6, and Corollary 5.11 (the ZK functionalities for Blum-integers). These functionalities are not LTR for P: If (p, q) is a witness for the factorisation of a Blum-integer, so is (q, p). Thus one cannot compute which of these two witnesses was used. Lemma 5.15 below shows that these functionalities are generalised LTR.

We stress that Definition 5.12 only deals with the case of being generalised LTR for a single party P. In a multi-party setting, it is not clear what it would mean for a functionality $\mathcal{F}$ to be generalised LTR for, say, parties P_1 and P_2 simultaneously. Does it mean that $\mathcal{F}$ is individually generalised LTR for P_1 and generalised LTR for P_2? Or should the functionality $\hat{\mathcal{F}}$ and the protocol π from Definition 5.12 be the same in both cases? Fortunately, for our results, we only need the notion of being generalised LTR for a single party P, so the issue does not arise.

Corollary 5.13. *Let $\mathcal{F}$ be a functionality that is generalised LTR for party P_1. Let R be an NP-relation* without *essentially unique witnesses. Then there is no nontrivial, polynomial-time protocol that long-term UC realises commitment with sender P_1 ($\mathcal{F}_{\mathrm{COM}}^{P_1 \to P_2}$) or zero-knowledge for the relation R with prover P_1 ($\mathcal{F}_{\mathrm{ZK}}^{R, P_1 \to P_2}$) in the $\mathcal{F}$-hybrid model.*

To show the corollary, we first need the following variant of the universal composition theorem (Theorem 3.4):

[40] If $\hat{\mathcal{F}}$ is not polynomial-time, we call π polynomial-time if all machines except $\hat{\mathcal{F}}$ are polynomial-time.

Lemma 5.14. *Let π, ρ, and σ be protocols. Assume that σ is polynomial-time and invokes exactly one instance of its subprotocol. Assume that π long-term UC realises ρ. Then σ^π long-term UC realises σ^ρ.*

Proof. Note that in comparison to Theorem 3.4, we do not require that π and ρ are polynomial-time, but instead we require that σ invokes only one subprotocol instance. In the proof of Theorem 3.4, we have only used that π and ρ are polynomial-time to show that the hybrid environment $\mathcal{Z}_\sigma$ is polynomial-time. In the case that σ invokes only one subprotocol instance, we have $n = 1$ in the notation of the proof of Theorem 3.4, and thus $\mathcal{Z}_\sigma$ expects an auxiliary input (i, z) with $i \in \{1, \ldots, n\}$, i.e. with $i = 1$. Then $\mathcal{Z}_\sigma$ behaves like $\mathcal{Z}_{\sigma,i} = \mathcal{Z}_{\sigma,1}$, and $\mathcal{Z}_{\sigma,1}$ does not simulate any instances of the protocols π or ρ (note again that $n = 1$). Thus $\mathcal{Z}_\sigma$ is polynomial-time if σ is, and we do not need that π and ρ are polynomial-time. □

Proof of Corollary 5.13. Assume for contradiction that there is a nontrivial, polynomial-time protocol $\tilde{\sigma}^{\mathcal{F}}$ that long-term UC realises $\mathcal{G} \in \{\mathcal{F}_{\mathrm{COM}}^{P_1 \to P_2}, \mathcal{F}_{\mathrm{ZK}}^{R, P_1 \to P_2}\}$ in the $\mathcal{F}$-hybrid model.

Then, since $\tilde{\sigma}^{\mathcal{F}}$ is polynomial-time, it invokes only a polynomial number p of instances of $\mathcal{F}$. Thus there is a protocol $\sigma^{\mathcal{F}^p}$ in the $\mathcal{F}^p$-hybrid model that long-term UC realises $\mathcal{G}$ and that invokes only one instance of $\mathcal{F}^p$.[41]

Since $\mathcal{F}$ is generalised LTR for P_1, by Definition 5.12 there is a functionality $\mathcal{F}$ that is LTR, and there is a nontrivial, polynomial-time protocol $\pi^{\hat{\mathcal{F}}}$ that long-term UC realises $\mathcal{F}^p$ in the $\hat{\mathcal{F}}$-hybrid model.

By Lemma 5.14, $\sigma^{\pi^{\hat{\mathcal{F}}}}$ long-term UC realises $\sigma^{\mathcal{F}^p}$. Since furthermore $\sigma^{\mathcal{F}^p}$ long-term UC realises $\mathcal{G}$, we have that $\sigma^{\pi^{\hat{\mathcal{F}}}}$ long-term UC realises $\mathcal{G}$. This contradicts Theorem 4.3 or Theorem 5.3. □

The notion of generalised LTR allows us, for example, to show that ZK protocols for relations with essentially unique witnesses cannot be used to construct commitments or ZK protocols for relations without essentially unique witnesses:

Lemma 5.15. *If R has essentially unique witnesses, then $\mathcal{F}_{\mathrm{ZK}}^{R, P \to V, m}$ is generalised LTR for P.*

Proof. Let U_R be the witness unifier for R. Let $L_R := \{x : \exists w : xRw\}$. For $x \in L_R$, let w_x denote the lexicographically smallest witness for x. Let $\mathcal{D}_{x,k}$ denote the distribution of w in the following experiment: $w' \leftarrow U_R(1^k, x, w_x)$, $w := \mathit{if}\ xRw'\ \mathit{then}\ w'\ \mathit{else}\ w_x$. Note that the support of $\mathcal{D}_{x,k}$ consists only of witnesses for x. From the properties of the witness unifier U_R (Definition 5.1), we have that $\mathcal{D}_{x,k}$ is statistically indistinguishable from $U_R(1^k, x, w)$ for all x, w with xRw.

Let $F_{x,k}$ be a function such that $F_{x,k}(w, r)$ with $w \leftarrow \mathcal{D}_{x,k}, r \leftarrow \{0,1\}^q$ and the uniform distribution on $\{0,1\}^t$ have statistical distance at most 2^{-k} (for some polyno-

[41] Here we use the fact that we assume the instances of $\mathcal{F}$ in $\tilde{\sigma}^{\mathcal{F}}$ to be numbered sequentially (cf. footnote 10). Hence only the session ids $1, \ldots, p$ are used; this matches the session ids supported by $\mathcal{F}^p$.

mially bounded q and t), and such that $F_{x,k}(w,r) \neq F_{x,k}(w',r')$ for all w, w', r, r' with $w \neq w'$. Such a function exists by Lemma A.5.

We define the functionality $\hat{\mathcal{F}}$ as follows: Upon the first input (x,w) from party P satisfying xRw and $|x| \leq m(k)$, compute $r \leftarrow \{0,1\}^q$, $f := F_{x,k}(w,r)$ and send (x,f) to party V.

Then $\hat{\mathcal{F}}$ is LTR for P: the values (x,w) are determined by (x,f). Similarly, $\hat{\mathcal{F}}^p$ is LTR for P for all polynomials p. Note that $\hat{\mathcal{F}}$ is not polynomial-time.

Consider the following protocol π in the $\hat{\mathcal{F}}^p$-hybrid model: Upon input (sid,x,w), party P computes $w' \leftarrow U_R(1^k,x,w)$ and sends (sid,x,w') to $\hat{\mathcal{F}}^p$. When party V receives (sid,x,f) from $\hat{\mathcal{F}}^p$, V outputs (sid,x).

We claim that π long-term UC realises $\mathcal{F}^p$ with $\mathcal{F} := \mathcal{F}_{\mathrm{ZK}}^{R,P\to V,m}$. In the case that both P and V are uncorrupted, this follows from the fact that $U_R(1^k,x,w)$ outputs a witness for x with overwhelming probability. In the case that P is corrupted, the additional output f sent by $\mathcal{F}^p$ is ignored by V, and the output x sent by $\hat{\mathcal{F}}^p$ is computed in the same way as $\mathcal{F}^p$ would have done.

The interesting case is when V is corrupted. The value f output by $\hat{\mathcal{F}}^p$ in the real model is computed as $w \leftarrow U_R(1^k,x,w), r \leftarrow \{0,1\}^q, f := F_{x,k}(w,r)$ for some x, w with xRw. Since $\mathcal{D}_{x,w}$ and $U_R(1^k,x,w)$ are statistically indistinguishable, computing f in this way is statistically indistinguishable from computing f by $w \leftarrow \mathcal{D}_{x,k}, r \leftarrow \{0,1\}^q, f := F_{x,k}(w,r)$. By construction of $F_{x,k}$, the distribution of f in this latter game is statistically indistinguishable from the uniform distribution on $\{0,1\}^t$. Thus the distribution of the value f, as sent by $\hat{\mathcal{F}}^p$ in the real model, is statistically indistinguishable from a uniformly chosen $f \in \{0,1\}^t$, even given x, w. Thus, in the case that V is corrupted, we can use the following simulator (assuming the dummy-adversary in the real model): Upon receiving (sid,x) from $\mathcal{F}^p$, pick $f \in \{0,1\}^t$ uniformly at random and send (sid,x,f) to the environment.

Thus π long-term (and even statistically) UC realises $\mathcal{F}^p$ in the $\hat{\mathcal{F}}^p$-hybrid model. Since $\hat{\mathcal{F}}^p$ is LTR and π is polynomial-time (i.e. P and V but not $\hat{\mathcal{F}}^p$ are polynomial-time), $\mathcal{F} = \mathcal{F}_{\mathrm{ZK}}^{R,P\to V,m}$ is generalised LTR. □

6. Possibility Results from Non-standard 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.

6.1. *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 because in real-life the random oracle must be implemented by 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 $\mathcal{F}_{\text{TPF}}$. Note, however, that all possibility results given in this section also hold (with identical proofs) when using a random oracle instead of $\mathcal{F}_{\text{TPF}}$.

Definition 6.1 (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 efficiently computable l.

Then the functionality *trusted pseudorandom function* (*TPF*) $\mathcal{F}_{\text{TPF}}^{f}$ 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, 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 [17,38]. However, we show that using a single TPF we can run an *arbitrary number* of zero-knowledge protocols or commitments, so that at least we do not need a large number of TPFs when constructing a larger protocol that executes many ZK protocols or commitments.

Theorem 6.2 (Zero-knowledge from TPF). *Assume that a one-way function exists. Let f_s be a pseudorandom function, and R an NP-relation. Then there is a nontrivial, polynomial-time protocol π using one instance of $\mathcal{F}_{\text{TPF}}^{f}$ that long-term UC realises an unlimited number of instances of $\mathcal{F}_{\text{ZK}}^{R}$ (i.e. ZK for the relation R).*

We give the proof idea first. First, a commitment scheme is constructed which is computationally binding, statistically hiding and extractable (however, this commitment is not necessarily UC). The extractable commitment is constructed from a given commitment which is statistically 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 $f_s(\cdot)$ can only be evaluated by using the functionality $\mathcal{F}_{\text{TPF}}$, a simulator can extract the committed value v from the calls which are placed to $\mathcal{F}_{\text{TPF}}$.

Using this extractable commitment, we modify the zero-knowledge argument for graph-3-colourability of [30]. 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 protocol for graph-3-colouring and hence for all NP-statements. The following proof gives the details of this approach.

Proof of Theorem 6.2. The proof proceeds in two steps. First, we construct from $\mathcal{F}_{\text{TPF}}$ 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 functions 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 Sect. 2. Partially following the construction from [34], 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 $\mathcal{F}_{\text{TPF}}$ 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 $\mathcal{F}_{\text{TPF}}$ with value u and commits to $(u, f_s(u))$ using COM_0.
- 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 $(u, f_s(u))$. If the checks were successful R outputs (unveil, v).

To extract the value v from a valid commitment the simulator keeps a list of all calls placed to the $\mathcal{F}_{\text{TPF}}$ 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 $\mathcal{F}_{\text{TPF}}$. 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 [30] 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.
- Perform the following $m \cdot k$ 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 accepts 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 proof, 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. Let us 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. □

According to Lemma 4.5, one commitment can be obtained from two invocations of a zero-knowledge scheme, and we can hence conclude:

Corollary 6.3 (Commitments from TPF). *Assume that a one-way function exists. Let f_s be a pseudorandom function. Then there is a nontrivial, polynomial-time protocol π using one instance of $\mathcal{F}_{\text{TPF}}^{f}$ that long-term UC realises an unlimited number of instances of $\mathcal{F}_{\text{COM}}$ (i.e. commitments).*

Proof. Immediate from Lemma 4.5 and Theorem 6.2. □

6.2. *Signature Cards*

One disadvantage of the TPF-assumption from the previous section is that trusted hardware implementing a pseudorandom function are unlikely to be available for practical use.[42] However, another kind of trusted device is already commercially available today: the signature card. A signature card is a tamper-proof device with a built-in signing key. Upon request, this card signs an arbitrary document, but *never* reveals the signing key. The corresponding verification key can be obtained from some certification authority by any party (including the adversary). These properties are required, e.g. by the German signature law [25].

These properties are captured by the following ideal functionality (based on [38]):

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

Definition 6.4 (Signature Card (SC)). Let $\mathfrak{S} = (KeyGen, Sign, Verify)$ be a signature scheme. Let H be a party. Then the functionality $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},H}$ (signature card for scheme $\mathfrak{S}$ with holder H) behaves as follows: Upon the first activation, $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},H}$ chooses a verification/signing key pair (pk, sk) using the key generation algorithm $KeyGen(1^k)$. Upon a message $(\mathtt{pk})$ from a party P or the adversary, send pk to that party or the adversary. Upon a message $(\mathtt{sign}, m)$ *from the holder* H, produce a signature σ for m using the signing key sk and send σ to H.

Note that the definition from [38] 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 from the definition of signature cards.

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 6.1. However, in [38], techniques were introduced that allow sharing a single signature card in different protocols. It would be interesting to explore whether their approach can be combined with our techniques.

It was shown in [38] 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 6.5 (Zero-knowledge from a signature card). *Assume that a one-way function exists. Let $\mathfrak{S}$ be an EF-CMA secure signature scheme. Let R be any NP-relation. Then there is a nontrivial, polynomial-time 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).*

The idea of the proof is as follows: To prove the existence of a witness w for some statement x, the prover P signs w 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 signing key sk' matching the verification 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 signing key sk' matching the verification 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 signing key sk matching the verification key pk. So the simulator can prove (ii) instead of (i). Since the argument system we use is statistically witness indistinguishable, the resulting interaction is statistically indistinguishable. We proceed to give the details of this approach.

Proof of Theorem 6.5. Let $\mathfrak{S} = (KeyGen, Sign, Verify)$ be an EF-CMA secure signature scheme, where $KeyGen(1^k)$ returns a key pair (pk, sk), $Sign(sk, m)$ returns a signature for m using signing key sk, and $Verify(pk, m, \sigma)$ returns 1 if σ is a valid signature for m with verification key pk (possibly only with overwhelming probability). $KeyGen(1^k; r)$, $Sign(sk, m; r)$ and $Verify(pk, m, \sigma; r)$ denote the output of the respective algorithms upon randomness r. Let R be an NP-relation. By SWIAOK, we mean the statistically witness indistinguishable argument of knowledge described in Sect. 2 (which exists under the assumptions of the theorem).

The Simplified Case Before showing the theorem in full generality, we assume some additional properties from the signature scheme $\mathfrak{S}$. These conditions are fulfilled by most practical signature schemes and lead to a much simpler protocol and proof. We give this simplified proof first since it already contains the main ideas of the proof below and since for most applications the slightly less general result will probably suffice. The additional conditions are:

- The verification algorithm is deterministic.
- The verification succeeds with probability 1 (and not only with overwhelming probability). That is, for all $m \in \{0,1\}^*$, $k \in \mathbb{N}$ we have that $\Pr[(pk, sk) \leftarrow KeyGen(1^k),\ \sigma \leftarrow Sign(sk, m) : Verify(pk, m, \sigma) = 1] = 1$.

Note that the latter condition implies that given a verification key pk (chosen by $KeyGen$) and given access to a corresponding signing oracle, it is hard to find some r such that $KeyGen(1^k; r) = (pk, sk')$ for some sk'. Otherwise the signing key sk' (although it is not necessarily the one originally generated with pk) would allow to produce arbitrary signatures since $KeyGen(1^k; r)$ is guaranteed to output a valid key pair for *all* randomnesses.

We now give the protocol π for the simplified case (one instance of that protocol is run for each instance of $\mathcal{F}_{\mathrm{ZK}}^{R}$):

- 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 $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$.
- P proves using the SWIAOK the knowledge of strings σ, w, r such that one of the following conditions holds:

  (i) $Verify(pk, w, \sigma) = 1$ and xRw, or
  (ii) $KeyGen(1^k; r) = (pk, sk')$ for some sk'.

  The prover P can perform this proof using σ, w as obtained above.
- If the verifier V accepts the SWIAOK, it outputs x.

We now proceed to show that π indeed is a nontrivial protocol that long-term UC realises an unbounded number of instances of $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$.

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

Without loss of generality, we can assume a dummy-adversary (see Sect. 3.1, p. 608).

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). In contrast to the prover, the simulator does not have access to a witness. However, the simulator is allowed to simulate the functionality $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ in an arbitrary way. The simulator $\mathcal{S}_V$ is constructed as follows:

- Simulate $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ honestly but store the randomness r used to generate the key pair of P.
- For each protocol instance, upon input x from the ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{R}$ (i.e. when the environment has sent some (x, w) to the functionality with xRw), simulate an honest prover P with input $(x, 0)$ with the following modification.
- The simulated prover P does not perform the check whether xRw holds, and as witness for the SWIAOK, the simulated prover P uses r (i.e. the prover proves (ii) instead of (i)).

For a witness w that does not satisfy xRw, the original prover and the modified simulated prover behave identically (namely, the original prover aborts and the modified simulated prover is never activated). If xRw, the original prover will use a witness σ, w with $\mathit{Verify}(pk, w, \sigma)$ for the SWIAOK with probability 1 (since the signature scheme outputs valid signatures with probability 1) and the modified simulated prover uses some r such that $\mathit{KeyGen}(1^k; r) = (pk, sk')$ for some sk' with probability 1. Since the SWIAOK is statistically witness indistinguishable, the interaction between the prover P and the environment is statistically indistinguishable in both situations. This even holds if multiple instances of the protocol are executed concurrently since witness indistinguishability composes concurrently. Thus the view of the environment is statistically indistinguishable in a run of the real and the ideal model.

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 $\mathcal{F}_{\mathrm{ZK}}^{R}$. We construct the simulator $\mathcal{S}_P$ as follows:

- Simulate $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ 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 $\mathcal{F}_{\mathrm{ZK}}^{R}$. Otherwise abort.

Obviously, as long as $\mathcal{S}_P$ does not abort, this interaction is statistically indistinguishable from an interaction with the real verifier. We therefore only have to show that $\mathcal{S}_P$ aborts with negligible probability. Therefore, assume that for some environment, $\mathcal{S}_P$ aborts with non-negligible probability. In this case, the environment can be transformed into a nonuniform polynomial-time ITM M_1 with access to a signing oracle and the verification key pk that performs the SWIAOK specified in the description of the protocol π. With non-negligible probability, the proof succeeds (for some x chosen by M_1)

and M_1 never sends a query $\tilde{w}$ with $x R \tilde{w}$ to the signing oracle. (Otherwise the simulator would not abort.) By applying the knowledge-extractor of the SWIAOK to M_1, we get a machine M_2 which queries the signing oracle and that with non-negligible probability outputs a witness for (i) or (ii) while never querying some $\tilde{w}$ with $x R \tilde{w}$ (the latter condition is preserved since the knowledge-extractor uses M_1 as a black-box and therefore does not access the signing oracle directly). Since finding an r satisfying (ii) is hard (as discussed above), and finding σ, w satisfying (i) would break the EF-CMA property of $\mathfrak{S}$ (as w with $x R w$ is not queried from the signing oracle), we have a contradiction. Thus $\mathcal{S}_P$ aborts only with negligible probability, and therefore the view of the environment is statistically indistinguishable in the real and in the ideal model.

Thus π is a nontrivial protocol that long-term UC realises an unbounded number of instances of $\mathcal{F}_{\mathrm{ZK}}^{R, P \to V}$.

The General Case In addition to the notation given at the beginning of the proof, we use the following notation. For a given security parameter k and a given input x, let $l_1 - 1$ be a polynomial upper bound on $|sk|$, $l_2 - 1$ a polynomial upper bound on $|w|$ for any witness w with $x R w$, and $l_3 - 1$ a polynomial upper bound on the length of $\sigma = \mathit{Sign}(sk, w)$ for any witness w with $x R w$. Such bounds always exist since the signature scheme is efficiently computable and R is polynomially-balanced. In the following, we will always tacitly assume that sk, σ and w (and similarly named variables) are padded in an efficiently invertible fashion to lengths l_1, l_2, and l_3, respectively, when used in a commitment or as a witness for an ZK-protocol. Let l_4 be a polynomial upper bound on the length of the randomness used in computing $\mathit{Verify}(pk, w, \mathit{Sign}(sk, w))$.

Let COM be the statistically hiding commitment scheme from [33] mentioned in Sect. 2 (that scheme exists under the assumptions of this theorem). Let V_{COM} be the corresponding deterministic verification algorithm such that $V_{\mathit{COM}}(c, v, u) = 1$ if the recipient accepts the unveil phase. Here c denotes the messages sent during the commit phase, v the value the commitment supposedly contains, and u the unveil message. We further assume that COM has a property that we call *corruption-correctness*. This property states that when the committing party is honest but the recipient is dishonest, and the commit phase completes successfully (i.e. the committing party does not abort during the commit), then with probability 1 we have that $V_{\mathit{COM}}(c, v, u) = 1$.[43] The commitment scheme from [33] is easily seen to be corruption-correct.

[43] One might think that any commitment scheme with a deterministic verification function can be directly transformed into a corruption-correct commitment scheme by letting the committee C check directly after the commit whether $V_{\mathit{COM}}(c, v, u) = 1$. However, an example for a commitment protocol that is not corruption-correct and that becomes insecure after this transformation is the following scheme COM': Let $f : \{0,1\}^k \to \{0,1\}^k$ be a one-way permutation such that the first bit of its input is hardcore. To commit to v, perform $\mathit{COM}(v)$ and then expect a value circuit $y \in \{0,1\}^k$ from the recipient R. An honest recipient chooses $y := f(1\|r)$ for random r. Define $V_{\mathit{COM}'}(c, v, u) := (V_{\mathit{COM}}(c_1, v, u) \wedge f(0\|v) = y)$ where $c =: (c_1, y)$. This scheme is not corruption-correct. Further, if the committee aborts when $V_{\mathit{COM}'}(c, v, u) \neq 1$, then R could send $y := f(1\|v')$ and thus find out whether the committed value is v', so the resulting scheme would not be hiding any more. This shows that this naive transformation does not work.

A transformation that works in general would be to first commit to random data r, then to check whether V_{COM} succeeds, and only then to send an r' with $r \oplus r' = v$.

Note that with a slight modification, the scheme COM' is an example for a commitment scheme with which our protocol π given below becomes insecure. Namely, instead of $y \stackrel{?}{=} f(0\|v)$ we check $y \stackrel{?}{=} f(0\|w)$ where $v =: (w, \sigma)$. Then the verifier could send a value y such that one particular witness w' is not accepted

We now describe the protocol π for implementing $\mathcal{F}_{\mathrm{ZK}}^{R}$ (one instance of that protocol π is run for each instance of $\mathcal{F}_{\mathrm{ZK}}^{R}$, all instances of π share a single instance of $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$):

- The prover P is activated with input (x, w).
- P checks whether xRw. Otherwise, it aborts.
- P sends x to V.
- P obtains a signature σ for w from $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$.
- Using COM, P commits to $(r^1_{Sign}, r^1_{Verify})$ where r^1_{Sign}, r^1_{Verify} are both uniformly chosen from $\{0,1\}^{l_4}$. Let c_1 denote the messages sent during the commitment.
- Using COM, P commits to 0^{l_1}. Let c_2 denote the messages sent during the commitment.
- Using COM, P commits to (w, σ). Let c_3 denote the messages sent during the commitment.
- V sends $(r^2_{Sign}, r^2_{Verify}, m)$ where r^2_{Sign}, r^2_{Verify} are both uniformly chosen from $\{0,1\}^{l_4}$ and m from $\{0,1\}^{k}$.
- P proves using the SWIAOK the knowledge of strings $(sk, r^1_{Sign}, r^1_{Verify}, w, \sigma, u_1, u_2, u_3)$ such that one of the following conditions holds:

  (i) $V_{COM}(c_1, (r^1_{Sign}, r^1_{Verify}), u_1) = 1$, and $V_{COM}(c_3, (w, \sigma), u_3) = 1$, and $Verify(pk, w, \sigma; r_{Verify}) = 1$, and xRw, or

  (ii) $V_{COM}(c_1, (r^1_{Sign}, r^1_{Verify}), u_1) = 1$, and $V_{COM}(c_2, sk, u_2) = 1$, and $Verify(pk, m, Sign(sk, m; r_{Sign}); r_{Verify}) = 1$

  where $r_{Sign} := r^1_{Sign} \oplus r^2_{Sign}$ and $r_{Verify} := r^1_{Verify} \oplus r^2_{Verify}$. The honest prover P performs this proof using $r^1_{Sign}, r^1_{Verify}, w, \sigma, u_1, u_3$ as obtained above as its witness.
- If the verifier V accepts the SWIAOK, it outputs x.

Note that in this protocol, much of the complexity (namely, having to perform the commitments and to send m and r^2_{Sign}, r^2_{Verify}) arises from the need to show the knowledge of a signing key in (ii) and from the fact that the verification algorithm is probabilistic. For most signature schemes, the verification is deterministic and deciding whether a given sk is a signing key for some given verification key pk can be decided in deterministic polynomial time. Then (ii) can be replaced by the statement "sk is a signing key for the verification key pk", and additional messages can be omitted, resulting in a much simpler protocol (and proof) like the one presented for the simplified case above. In general, however, such a deterministic check is not possible, so the protocol has to be constructed as described above.

We now proceed to show that π indeed is a nontrivial protocol that long-term UC realises an unbounded number of instances of $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$.

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

Without loss of generality, we can assume a dummy-adversary (see Sect. 3.1, p. 608).

any more. Then the SWIAOK in the protocol below fails iff this witness w' is used, so the protocol is not even computationally witness-indistinguishable any more.

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). In contrast to the prover, the simulator does not have access to a witness. However, the simulator is allowed to simulate the functionality $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ in an arbitrary way. To construct the simulator $\mathcal{S}_V$, we first define several variants of the honest prover P. These differ from P in the values they commit to in commitments c_2 and c_3, and whether they prove (i) or (ii) in the SWIAOK. In the following table, we give these provers, and for comparison, also the original prover P. By sk we denote the signing key chosen by $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$.

Prover	Value in c_2	Value in c_3	Witness in SWIAOK
P	0^{l_1}	(w,σ)	$r^1_{Sign}, r^1_{Verify}, w, \sigma, u_1, u_3$ for (i)
P_1	sk	(w,σ)	$r^1_{Sign}, r^1_{Verify}, w, \sigma, u_1, u_3$ for (i)
P_2	sk	(w,σ)	$sk, r^1_{Sign}, r^1_{Verify}, u_1, u_2$ for (ii)
P_3	sk	$(0^{l_2}, 0^{l_3})$	$sk, r^1_{Sign}, r^1_{Verify}, u_1, u_2$ for (ii)

Note that P_1, P_2, P_3 are not valid provers in the real model since they use the signing key sk of $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$. However, we only use them in the simulation and in intermediate games. Let further P_3' be constructed like P_3, except that P_3' does not check whether xRw holds for its input.

The simulator $\mathcal{S}_V$ is constructed as follows:

- Simulate $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ honestly.
- For each protocol instance, upon input x from the ideal functionality $\mathcal{F}_{\mathrm{ZK}}^{R}$ (i.e. when the environment has sent some (x,w) to the functionality with xRw), simulate the prover P_3' with input $(x,0)$.

Note that $\mathcal{S}_V$ knows the signing key used by $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ since it simulates $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$. Therefore, it can simulate P_3'.

Since P_3 does not use the witness w except for checking xRw, and since $\mathcal{S}_V$ invokes P_3' only if xRw holds, we have that the view of the environment is the same in the ideal model with $\mathcal{S}_V$ and in the real model with P_3.[44] (Even when multiple instances of P_3' or P_3, respectively, are executed concurrently.)

Since *COM* is statistically hiding, and since P_3 never uses the unveil information u_3 for c_3, we have that the view of the environment is statistically indistinguishable in the real model with P_3 and in the real model with P_2.

Note that in an execution of the real model with prover P_2, we always have that $V_{COM}(c_1, (r^1_{Sign}, r^1_{Verify}), u_1) = 1$, and $V_{COM}(c_2, sk, u_2) = 1$, and $V_{COM}(c_3, (w,\sigma), u_3) = 1$ (since *COM* is corruption-correct). Further, since (pk, sk) is an honestly generated key pair, we have that for any message m, $Verify(pk, m, Sign(sk, m)) = 1$ holds with

[44] Note that P_3 is not a valid protocol machine since it accesses the signing key sk. The execution of the real model with P_3 is well-defined, nevertheless.

overwhelming probability. Since r_{Sign} and r_{Verify} are produced by a coin-toss, we have with overwhelming probability that $Verify(pk, m, Sign(sk, m; r_{Sign}); r_{Verify}) = 1$ holds. With analogous reasoning and using the fact that σ is an honestly generated signature on w, also $Verify(pk, w, \sigma; r_{Verify}) = 1$ holds with overwhelming probability. Furthermore, since P_2 aborts unless $x R w$ holds, we also have that $x R w$ holds. Thus when P_2 executes the SWIAOK, with overwhelming probability both $r^1_{Sign}, r^1_{Verify}, sk, u_1, u_2$ and $r^1_{Sign}, r^1_{Verify}, w, \sigma, u_1, u_3$ are valid witnesses for the SWIAOK, thus the environment's view in the real model with P_2 and with P_1 is statistically indistinguishable (since the SWIAOK is statistically witness indistinguishable).

Since *COM* is statistically hiding, and since P_1 never uses the unveil information u_2 for c_2, we have that the view of the environment is statistically indistinguishable in the real model with P_1 and in the real model with P.

It follows that the environment's view is statistically indistinguishable between the real model with honest prover P and the ideal model with the simulator $\mathcal{S}_V$.

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 $x R w$ to the ideal functionality $\mathcal{F}^R_{\text{ZK}}$. We construct the simulator $\mathcal{S}_P$ as follows:

- Simulate $\mathcal{F}^{\mathfrak{S}}_{\text{SC}}$ 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 $x R w$ for some $w \in M$. If so, send (x, w) to $\mathcal{F}^R_{\text{ZK}}$. Otherwise, abort.

Obviously, as long as $\mathcal{S}_P$ does not abort, this interaction is statistically indistinguishable from an interaction with the real verifier. We therefore only have to show that $\mathcal{S}_P$ aborts with negligible probability. Assume therefore that for some environment, $\mathcal{S}_P$ aborts with non-negligible probability. In this case, the environment can be transformed into a nonuniform polynomial-time ITM M_1 with access to a signing oracle and the verification key pk such that M_1 has the following properties:

- It outputs some x and commits to three values (resulting in communications c_1, c_2, c_3), then it waits for a tuple $(r^2_{Sign}, r^2_{Verify}, m)$ and then it performs the SWIAOK specified in the description of the protocol π.
- If $(r^2_{Sign}, r^2_{Verify}, m)$ is uniformly distributed, with non-negligible probability the SWIAOK is accepted by the verifier *and* the machine does not send any query $\tilde{w}$ with $x R \tilde{w}$ to the signing oracle. (Otherwise the simulator would not abort.)

Since the SWIAOK is an argument of knowledge, we can apply the knowledge-extractor to M_1 and transform the residual prover[45] performing the SWIAOK into a machine that outputs a witness for the SWIAOK. This results in a nonuniform polynomial-time ITM M_2 with the following properties:

[45] By the residual prover we mean the machine that gets as input the state s (including the random tape) of M_1 immediately before M_1 performs the SWIAOK and that then performs only the SWIAOK as M_1 would have done on that state s.

- It outputs some x and commits to three values (resulting in communications c_1, c_2, c_3), then it waits for a tuple $(r^2_{Sign}, r^2_{Verify}, m)$ and then it outputs a tuple $(sk, r^1_{Sign}, r^1_{Verify}, w, \sigma, u_1, u_2, u_3)$.
- If $(r^2_{Sign}, r^2_{Verify}, m)$ is uniformly distributed, one of the following holds:
  - With non-negligible probability we have that (i) is fulfilled *and* the machine M_2 does not send any query $\tilde{w}$ with $x R \tilde{w}$ to the signing oracle.
  - With non-negligible probability we have that (ii) holds.

The property that the machine never queries a $\tilde{w}$ with $x R \tilde{w}$ is preserved because the knowledge-extractor performs a black-box reduction: As long as the residual prover given as an oracle to the knowledge-extractor does not query a given value $\tilde{w}$, the knowledge-extractor does not either.

We now construct a machine M_3 that performs the following steps:

- It executes M_2, and when M_2 expects a tuple $(r^2_{Sign}, r^2_{Verify}, m)$, the machine M_3 uniformly chooses values $(r^{2\prime}_{Sign}, r^{2\prime}_{Verify}, m')$ and passes these to M_2. When M_2 outputs a tuple $(sk', r^{1\prime}_{Sign}, r^{1\prime}_{Verify}, w', \sigma', u'_1, u'_2, u'_3)$, the machine M_3 stores these values.
- Then M_3 chooses randomly $\tilde{r}_{Sign}, m$.
- Then M_3 chooses randomly $\tilde{r}_{Verify}$.
- Then M_3 rewinds M_2 to the point where it waits for a pair $(r^2_{Sign}, r^2_{Verify}, m)$, then it sets $r^2_{Sign} := \tilde{r}_{Sign} \oplus r^{1\prime}_{Sign}$ and $r^2_{Verify} := \tilde{r}_{Verify} \oplus r^{1\prime}_{Verify}$ and passes $(r^2_{Sign}, r^2_{Verify}, m)$ to M_2. Then M_3 waits for M_2 to output a tuple $(sk, r^1_{Sign}, r^1_{Verify}, w, \sigma, u_1, u_2, u_3)$.

Let (i′) and (ii′) be defined like (i) and (ii) but with respect to the variables $sk', r^{1\prime}_{Sign}$, $r^{1\prime}_{Verify}, w', \sigma', u'_1, u'_2, u'_3$ instead of $sk, r^1_{Sign}, r^1_{Verify}, w, \sigma, u_1, u_2, u_3$. Since in both executions of M_2, the inputs $(r^{2\prime}_{Sign}, r^{2\prime}_{Verify}, m')$ and $(r^2_{Sign}, r^2_{Verify}, m)$ are uniformly distributed, one of the following holds with non-negligible probability:

- Both (i) and (i′) are fulfilled and during the first execution of M_2, the value w' is not queried from the signing oracle.
- Both (ii) and (ii′) are fulfilled and during the first execution of M_2, the value m is not queried from the signing oracle.

The fact that M_3 does not query m during the first execution of M_2 (except with negligible probability) stems from the fact that m is chosen randomly after the first execution of M_2.

Since *COM* is computationally binding, the probability is negligible that (i) and (i′) are fulfilled and $(r^{1\prime}_{Sign}, r^{1\prime}_{Verify}, w', \sigma') \neq (r^1_{Sign}, r^1_{Verify}, w, \sigma)$ holds. Similarly, the probability is negligible that (ii) and (ii′) are fulfilled and $(sk', r^{1\prime}_{Sign}, r^{1\prime}_{Verify}) \neq (sk, r^1_{Sign}, r^1_{Verify})$ holds.

Thus one of the following holds with non-negligible probability:

- Case 1: Both (i) and (i′) are fulfilled, and $(r^{1\prime}_{Sign}, r^{1\prime}_{Verify}, w', \sigma') = (r^1_{Sign}, r^1_{Verify}, w, \sigma)$, and during the first execution of M_2, the value w' is not queried from the signing oracle.

- Case 2: Both (ii) and (ii′) are fulfilled, and $(sk', r^{1'}_{Sign}, r^{1'}_{Verify}) = (sk, r^{1}_{Sign}, r^{1}_{Verify})$, and during the first execution of M_2, the value m is not queried from the signing oracle.

Let $\sigma_m := Sign(sk', m; \tilde{r}_{Sign})$. In Case 1, we have with non-negligible probability that w' is not queried during the first execution of M_2 and

$$\begin{aligned} Verify(pk, w', \sigma'; \tilde{r}_{Verify}) &= Verify(pk, w', \sigma'; r^{1'}_{Verify} \oplus r^{2}_{Verify}) \\ &\overset{\text{(i)}}{=} Verify(pk, w, \sigma; r^{1}_{Verify} \oplus r^{2}_{Verify}) = 1. \end{aligned}$$

In Case 2, we have with non-negligible probability that m is not queried during the first execution of M_2 and

$$\begin{aligned} &Verify(pk, m, \sigma_m; \tilde{r}_{Verify}) \\ &\quad = Verify(pk, m, Sign(sk', m; \tilde{r}_{Sign}); \tilde{r}_{Verify}) \\ &\quad = Verify(pk, m, Sign(sk', m; r^{1'}_{Sign} \oplus r^{2}_{Sign}); r^{1'}_{Verify} \oplus r^{2}_{Verify}) \\ &\quad \overset{\text{(ii)}}{=} Verify(pk, m, Sign(sk, m; r^{1}_{Sign} \oplus r^{2}_{Sign}); r^{1}_{Verify} \oplus r^{2}_{Verify}) = 1. \end{aligned}$$

Thus, if we let M_3 terminate with output $(m, \sigma_m), (w', \sigma')$ directly after choosing $\tilde{r}_{Sign}, m$, we have that with non-negligible probability one of σ_m, σ' will be successfully verified and the corresponding message m or w', respectively, will not have been queried from the signing oracle. Since M_3 is a nonuniform polynomial-time machine, this contradicts the EF-CMA security of $\mathcal{S}$. Thus the assumption that $\mathcal{S}_P$ aborts with non-negligible probability was false, and thus the environment's view is statistically indistinguishable between the real model with the honest verifier V and the ideal model with the simulator $\mathcal{S}_P$. Thus π is a nontrivial protocol that long-term UC realises an unbounded number of instances of $\mathcal{F}_{\text{ZK}}^{R, P \to V}$. □

From Theorem 6.5, we can also easily deduce that longterm UC secure commitments can be realised using a signature card:

Corollary 6.6 (Commitments from a signature card). *Assume that a one-way function exists. Let $\mathcal{S}$ be an EF-CMA secure signature scheme. Then there is a nontrivial, polynomial-time protocol π that long-term UC realises an unbounded number of instances of $\mathcal{F}_{\text{COM}}^{C \to R}$ (i.e. commitment with sender C) using a single instance of $\mathcal{F}_{\text{SC}}^{\mathcal{S}, P}$ (i.e. a signature card for $\mathcal{S}$ with P as the holder).*

Proof. This is an immediate consequence of Theorem 6.5 and Lemma 4.5. □

Note also that the results in this section also cover the case that the adversary retrieves the signing key of the signature card (e.g. by opening it) *after* the protocol execution. Since we assume the adversary to be computationally unbounded after the execution, he

could determine the signing key from the verification key after the protocol execution. Thus by retrieving the signing key, he gains no additional information.[46]

7. Conclusions

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

Avenues for future research include:

- Characterise which protocol tasks can be long-term UC realised using commitments and/or the zero-knowledge functionality. Note that the result that any functionality can be realised from a commitment or a zero-knowledge functionality [15] has been proven under computational assumptions and thus does not necessarily carry over to the setting of longterm UC security.
- As seen in Sect. 4.4, statistically hiding UC-commitments are not necessarily long-term UC secure. However, they seem to provide some kind of long-term security which could provide limited compositional guarantees. It is an open problem to formalise the additional security guarantees a protocol composed from statistically hiding commitments has.
- Find other setup assumptions that are useful in the context of long-term UC. In particular, determine under which assumptions oblivious transfer (OT), and therefore any functionality, can be realised.
- Our investigations were in the secure-channels communication-model. If only authenticated channels are present, the important issue of key exchange occurs. Hence, it is necessary to identify setup assumptions that allow to implement key exchange.
- The protocols presented here were not optimised for efficiency. For practical applicability, we need efficient protocols for the tasks discussed in this work. For example, the Camenisch–Lysyanskaya signature scheme [9] allows for very efficient (non-UC) zero-knowledge protocols. Thus efficient variants of our protocols from Sect. 6.2 might be possible.
- In [38], techniques were presented that enable different protocols to share a single signature card. This techniques might be applicable in our setting, too.
- Find a transformation for converting a protocol that is *passively* secure against unbounded adversaries into a long-term UC secure protocol based on signature cards or TPFs. This might be achieved by using ZK-protocols to prove the correctness of each message (as in the GMW compiler [29]). The ZK-protocols could then be implemented using our constructions.

[46] Similar reasoning also applies to TPFs: Here the secret seed is information-theoretically determined by the answer to sufficiently many random queries.

- Much work on unconditional and long-term security has been done in the field of quantum cryptography. Investigate how long-term UC behaves in the presence of quantum communication. This may allow avoiding some of the impossibility results presented in this work. In particular, quantum communication could solve the problem of key exchange mentioned above.

Acknowledgements

We thank Oana Ciobotaru, Yevgeniy Dodis, Oded Goldreich, and Dennis Hofheinz as well as the anonymous referees of TCC 2007 and the Journal of Cryptology for valuable discussions and many helpful suggestions. We thank Catherine Howell for proofreading and helping to improve our English. Part of this work has been funded by the Cluster of Excellence "Multimodal Computing and Interaction" (MMCI).

Appendix A. Technical Lemmas

In this appendix, we prove five technical lemmas that are used in several of the proofs in the main part of this paper.

In the proof of Theorem 5.10, we need to turn a specific non-efficient algorithm into an efficient nonuniform algorithm. For this, we need the construction from the following lemma.

Lemma A.1. *Let R be an NP-relation. Let A be a 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. Without loss of generality 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 a 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 $p(k)$) such that $\tilde{A}(x, \mathcal{E}_{|x|})$ 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 $x_k \in L_k$ such that $G(x_k, e_k) \neq 1$. Then for random $x \in L_k$ and $e \leftarrow \mathcal{E}_k$, we would have

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

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. □

In Theorem 5.5, the existence of a long-term UC zero-knowledge proof for Blum-integers is stated. To prove this, there must be a simulator able to generate random square roots modulo this Blum-integer. The following lemma gives an efficient algorithm for this task.

Lemma A.2. *There is a PPT-algorithm Q such that $Q(1^k,n)$ outputs two values, $r\in\{0,\ldots,2^{k|n|}-1\}$ and $s\in\{0,\ldots,n-1\}\cup\{\perp\}$ such that the following holds if n is a Blum-integer*:

- *The distribution of r is almost uniformly distributed on $\{0,\ldots,2^{k|n|}-1\}$.*[47]
- *If r is a quadratic residue mod n, then s is an almost uniformly distributed root of r modulo n (and $s=\perp$ otherwise).*

Proof. First, we remind the reader of some facts: The Legendre-symbol $(\frac{a}{p})$ for prime p is defined as $(\frac{a}{p})=+1$ if a is a quadratic residue modulo p, $(\frac{a}{p})=-1$ if a is a quadratic non-residue modulo p, and $(\frac{a}{p})=0$ if $a\equiv 0 \bmod p$. The Jacobi-symbol $(\frac{a}{pq})$ for different primes p,q is defined as $(\frac{a}{pq}):=(\frac{a}{p})(\frac{a}{q})$. 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 $J:=(\frac{r_0}{n})$.
- If $J=-1$, output $(r_0,\perp)$.
- If $J=0$ and $r_0\neq 0$, factor n.[48] Output $(r_0,R(p,q,r_0))$.

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

[48] This can be done efficiently, since if $(\frac{r_0}{n})=0$ we have that $\gcd(r_0,n)$ is a factor of n.

- If $r_0 = 0$, return $(0, 0)$.
- If $J = +1$, choose a uniformly random invertible $s \in \mathbb{Z}_n$,[49] and with probability $\frac{1}{2}$, output (s^2, s), otherwise $(-s^2, \bot)$.

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

Let $S_1 \subseteq \mathbb{Z}_n$ be the set of all r_0 with $(\frac{r_0}{n}) \in \{-1, 0\}$. 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, $s = \bot$ (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 $(\frac{r_0}{n}) = +1$. 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) such that r is uniformly distributed on S_2, and if r is a square, s is a uniformly chosen root (and $s = \bot$ otherwise).

It follows that for uniformly chosen $r_0 \in \mathbb{Z}_n$, $Q'(n, r_0)$ outputs (r, s) such that 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 we define algorithm Q'':

- 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) such that 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 $\bar{r} \in \{0, \ldots, n-1\}$ be a representative of $r \in \mathbb{Z}_n$.
- Let $d := \lfloor 2^{k|n|}/n \rfloor$, and choose a uniformly random $e \in \{0, \ldots, d-1\}$.
- Return $(\bar{r} + en, s)$.

Obviously, for uniform $r \in \mathbb{Z}_n$, $\bar{r} + en$ is almost uniformly distributed on $\{0, \ldots, 2^{k|n|} - 1\}$. So if n is a Blum-integer, $Q(k, n)$ outputs (r, s) such that r is almost uniformly distributed on $\{0, \ldots, 2^{k|n|} - 1\}$, 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

[49] How to do this is discussed later.

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. □

The next lemma shows that it is impossible to construct machines U and L with the following property: U can deviate from its normal program in a way that is not noticed when running with L. L can deviate in a way that is not noticed when running with U. Yet, if both machines deviate simultaneously, this will be noticed. (This roughly corresponds to the fact that the AND function cannot be computed privately in the presence of semi-honest adversaries.) Doing the calculations here shortens the proof of Theorem 5.10.

Lemma A.3. *Let $U, \tilde{U}, L, \tilde{L}$ 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 an 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

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

Note that at a first glance this lemma looks like a special case of Theorem 1 of [40]. However, [40] assumes $\langle U, L\rangle^{k,z} = \langle \tilde{U}, L\rangle^{k,z} = \langle U, \tilde{L}\rangle^{k,z}$ while we will need the lemma to hold also in the case of statistical indistinguishability.

Proof. In the following, we omit k, z for readability. Without loss of generality, 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 the messages 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 ith 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 Δ denote the statistical distance of random variables, and let

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

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).

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

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

The following lemma gives a criterion for the existence of oracles relative to which a certain hardness assumption holds.

Lemma A.4. *Let $\mathcal{D}_n$ be an efficiently computable distribution on bitstrings, i.e. there is a probabilistic algorithm that on input n samples an element of $\mathcal{D}_n$ in time polynomial in n. Let R be a relation. Assume that for any PPT algorithm A, we have that $\Pr[x\,R(A(1^n, x)) : x \leftarrow \mathcal{D}_n]$ is negligible in n. Then there is a function $f : \mathbb{N} \to \{0,1\}^*$ such that for any PPT oracle algorithm B, we have that $\Pr[(f(n))\,R(B^f(1^n, f(n)))]$ is negligible in n.* (*We assume that the input to f in an oracle query is encoded in unary.*)

Proof. Let $\mathcal{F}$ be the distribution of functions $f : \mathbb{N} \to \{0,1\}^*$ resulting from choosing each image $f(n)$ independently according to $\mathcal{D}_n$. Then for any PPT oracle algorithm B, we have that $\Pr[(f(n))\,R(B^f(1^n, f(n))) : f \leftarrow \mathcal{F}]$ is negligible in n. If this were not the case for some B, we could construct a PPT algorithm A from B as follows: A expects inputs $(1^n, x)$. When B queries $f(n)$, A uses the value x instead. When B queries $f(i)$ with $i \neq n$, A chooses a random $y \leftarrow \mathcal{D}_i$ (which is then reused for subsequent queries $f(i)$ with the same i). Then $\Pr[x\,R(A(1^n, x)) : x \leftarrow \mathcal{D}_n] = \Pr[(f(n))\,R(B^f(1^n, f(n))) : f \leftarrow \mathcal{F}]$ is not negligible in contradiction to the assumption in the statement of the lemma.

Fix some PPT oracle algorithm B and some $c \in \mathbb{N}$. Since $\Pr[(f(n))\,R(B^f(1^n, f(n))) : f \leftarrow \mathcal{F}]$ is negligible, for any $\alpha > 0$, there is an $n_\alpha \in \mathbb{N}$ such that for all $n \ge n_\alpha$,

$$\Pr\big[\big(f(n)\big)R\big(B^f\big(1^n, f(n)\big)\big) : f \leftarrow \mathcal{F}\big] \le \frac{\alpha n^{-c-2}}{\sum_{i\ge 1}^{\infty} \frac{1}{i^2}}. \tag{A.1}$$

With $P_n^B(f) := \Pr[(f(n))\,R(B^f(1^n, f(n)))]$ we have that $\Pr[(f(n))\,R(B^f(1^n, f(n))) : f \leftarrow \mathcal{F}] = \mathrm{E}_f[P_n^B(f)]$ where E_f denotes the expectation over choices $f \leftarrow \mathcal{F}$. From

Markov's inequality, it follows that

$$\Pr\left[P_n^B(f) \geq n^{-c} : f \leftarrow \mathcal{F}\right] \leq \frac{E_f[P_n^B(f)]}{n^{-c}} \overset{(A.1)}{\leq} \frac{\alpha}{n^2 \sum_{i \geq 1}^{\infty} \frac{1}{i^2}} \tag{A.2}$$

for all $\alpha > 0$ and all $n \geq n_\alpha$.

Then we have for any $\alpha > 0$,

$$\begin{aligned}
&\Pr\left[\left(\forall m \geq 1.\ \exists n \geq m.\ P_n^B(f) \geq n^{-c}\right) : f \leftarrow \mathcal{F}\right] \\
&\leq \Pr\left[\left(\exists n \geq n_\alpha.\ P_n^B(f) \geq n^{-c}\right) : f \leftarrow \mathcal{F}\right] \\
&\leq \sum_{n \geq n_\alpha} \Pr\left[P_n^B(f) \geq n^{-c} : f \leftarrow \mathcal{F}\right] \\
&\overset{(A.2)}{\leq} \sum_{n \geq n_\alpha} \frac{\alpha}{n^2 \sum i^{-2}} \leq \alpha.
\end{aligned}$$

Since this holds for any $\alpha > 0$, we have $\Pr[(\forall m \geq 1.\ \exists n \geq m.\ P_n^B(f) \geq n^{-c}) : f \leftarrow \mathcal{F}] = 0$.

$$\begin{aligned}
&\Pr\left[\left(\exists B.\ P_n^B(f) \text{ not negligible} : f \leftarrow \mathcal{F}\right)\right] \\
&\leq \sum_B \Pr\left[P_n^B(f) \text{ not negligible} : f \leftarrow \mathcal{F}\right] \\
&= \sum_B \Pr\left[\left(\exists c \in \mathbb{N}.\ \forall m \geq 1.\ \exists n \geq m.\ P_n^B(f) \geq n^{-c}\right) : f \leftarrow \mathcal{F}\right] \\
&\leq \sum_B \sum_c \Pr\left[\left(\forall m \geq 1.\ \exists n \geq m.\ P_n^B(f) \geq n^{-c}\right) : f \leftarrow \mathcal{F}\right] \\
&= \sum_B \sum_c 0 = 0.
\end{aligned}$$

Here we quantify over PPT oracle machines B. For the last equality, note that there are only countably many B, c.

Thus, when choosing $f \leftarrow \mathcal{F}$, with probability 1, for all PPT B, $P_n^B(f)$ is negligible. In particular, there exists an f such that for all PPT B, $P_n^B(f)$ is negligible. This shows the claim. □

The following lemma allows us to encode an arbitrarily distributed random variable in an almost uniformly distributed one.

Lemma A.5. *Let W be a random variable with range D. Fix $k > 0$ and $m \geq \log \#D$. Let $t := m + k + 1$, and $q := t + k + 1$. Let R and U be uniformly distributed random variables with ranges $\{0,1\}^q$ and $\{0,1\}^t$, respectively. Assume that R is independent of W.*

Then there is a function F such that $\Delta(F(W,R),U) \leq 2^{-k}$, where Δ denotes the statistical distance, and such that $F(w,r) \neq F(w',r')$ for all w, w', r, r' with $w \neq w'$.

Proof. We identify the bitstrings in the ranges of R and U with integers $\{0,\dots,2^t-1\}$ and $\{0,\dots,2^q-1\}$, respectively. We identify D with $\{0,\dots,\#D-1\}$. Let $T := 2^t$ and $Q := 2^q$. We abbreviate $\Pr[W=w]$ as P_w. For $w \in D$, let $l_w := P_w T$ and $s_w := \sum_{w'<w} \lceil l_{w'} \rceil$ and $s^* := \sum_{w\in D} \lceil l_w \rceil$.

For an integer $n > 0$, let $f_n(r) := r \bmod n$ if $0 \le r < n\lfloor \frac{Q}{n} \rfloor$ and $f_n(r) := \bot$ otherwise. Then for any n, $f_n(R)$ is uniformly distributed on $\{0,\dots,n-1\}$ under the condition that $f_n(R) \neq \bot$, and $\Pr[f_n(R)=\bot] = 1 - \frac{n}{Q}\lfloor \frac{Q}{n} \rfloor = \frac{n}{Q}(\frac{Q}{n} - \lfloor \frac{Q}{n} \rfloor) \le \frac{n}{Q}$.

We define the function F as follows:

$$F(w,r) := \begin{cases} s_w + f_{\lceil l_w \rceil}(r) & \text{if } f_{\lceil l_w \rceil}(r) \neq \bot \text{ and } s_w + f_{\lceil l_w \rceil}(r) < T, \\ \bot & \text{otherwise.} \end{cases}$$

The requirement that $F(w,r) \neq F(w',r')$ for all w, w', r, r' with $w \neq w'$ is satisfied because $s_w \le s_w + f_{\lceil l_w \rceil}(r) < s_w + \lceil l_w \rceil = s_{w+1}$ for all w, r.

For any $z \in \{0,\dots,s^*-1\}$, we have

$$\Pr\big[F(W,R)=z\big] = \Pr[W=w]\Pr\big[f_{\lceil l_w \rceil}(R) = z - s_w\big] \le P_w / \lceil l_w \rceil \le P_w / l_w = 2^{-t},$$

where w is the unique value such that $s_w \le z < s_{w+1}$. For $z \ge s^*$ we have $\Pr[F(W,R)=z] = 0 \le 2^{-t}$. Hence for any $z \neq \bot$, we have that $\Pr[F(W,R)=z] \le \Pr[U=z]$.

To compute an upper bound on $\Delta(F(W,R),U)$, note the following general fact for any random variables X, Y with countable ranges: $\Delta(X,Y) = \sum_{z\in B}(\Pr[X=z] - \Pr[Y=z])$ where $B := \{z : \Pr[X=z] > \Pr[Y=z]\}$. Applying this fact to $F(W,R)$ and U, we have $B = \{\bot\}$ and

$$\Delta\big(F(W,R),U\big) = \sum_{z\in B}\big(\Pr\big[F(W,R)=z\big] - \Pr[U=z]\big) = \Pr\big[F(W,R)=\bot\big]. \quad \text{(A.3)}$$

Moreover, we have

$$\Pr\big[F(W,R)=\bot\big] \le \Pr\big[f_{\lceil l_W \rceil}(R)=\bot\big] + \Pr\big[s_W + f_{\lceil l_W \rceil}(R) \ge T \wedge f_{\lceil l_W \rceil}(R) \neq \bot\big].$$

Since $\Pr[f_n(R)=\bot] \le \frac{n}{Q}$ for any n, we have

$$\Pr\big[f_{\lceil l_W \rceil}(R)=\bot\big] \le \sum_{w\in D} P_w \frac{\lceil l_w \rceil}{Q} \le \max_{w\in D} \frac{\lceil l_w \rceil}{Q} \le \frac{T}{Q}. \quad \text{(A.4)}$$

We proceed to bound $\Pr[s_W + f_{\lceil l_W \rceil}(R) \ge T \wedge f_{\lceil l_W \rceil}(R) \neq \bot]$. First, remember that for any $w \in D$, $f_{\lceil l_w \rceil}(R)$ is uniformly distributed on $\{0,\dots,\lceil l_w \rceil\}$ under the condition $f_{\lceil l_w \rceil}(R) \neq \bot$. Let I_b denote the uniform distribution on the real interval $[0,b)$. Then if b is an integer, $\lfloor I_b \rfloor$ is the uniform distribution on $\{0,\dots,b-1\}$. Hence

$$\begin{aligned}
&\Pr\big[s_w + f_{\lceil l_w \rceil}(R) \ge T \wedge f_{\lceil l_w \rceil}(R) \neq \bot\big] \\
&\quad \le \Pr\big[s_w + f_{\lceil l_w \rceil}(R) \ge T \,|\, f_{\lceil l_w \rceil}(R) \neq \bot\big] \\
&\quad = \Pr\big[s_w + \lfloor I_{\lceil l_w \rceil} \rfloor \ge T\big] \le \Pr\big[s_w + I_{\lceil l_w \rceil} \ge T\big] \\
&\quad \le \Pr[s_w + I_{l_w+1} \ge T] \le \Pr[s_w + I_{l_w} \ge T - 1] \\
&\quad \le \Pr[s'_w + I_{l_w} \ge T - 1 - \#D]
\end{aligned}$$

with $s'_w := \sum_{w'<w} l_{w'}$. Thus

$$\Pr\big[s_W + f_{\lceil l_W \rceil}(R) \geq T \wedge f_{\lceil l_W \rceil}(R) \neq \perp\big] \leq \Pr[s'_W + I_{l_W} \geq T - 1 - \#D]. \tag{A.5}$$

But $s'_W + I_{l_W}$ is the uniform distribution on $[0, T)$ (note for this that P_w is proportional to $l_w = P_w T$). Hence $\Pr[s'_W + I_{l_W}(R) \geq T - 1 - \#D] = \frac{1+\#D}{T}$ and thus

$$\Pr\big[s_W + f_{\lceil l_W \rceil}(R) \geq T \wedge f_{\lceil l_W \rceil}(R) \neq \perp\big] \overset{(A.5)}{\leq} \frac{1 + \#D}{T}. \tag{A.6}$$

Summarising,

$$\begin{aligned}
\Delta\big(F(W, R), U\big) &= \Pr\big[F(W, R) = \perp\big] \\
&\overset{(A.3)}{\leq} \Pr\big[f_{\lceil l_W \rceil}(R) = \perp\big] + \Pr\big[s_W + f_{\lceil l_W \rceil}(R) \geq T \,|\, f_{\lceil l_W \rceil}(R) \neq \perp\big] \\
&\overset{(A.4),(A.6)}{\leq} \frac{T}{Q} + \frac{1 + \#D}{T} \leq 2^{-k}.
\end{aligned}$$

□

References

[1] M. Backes, D. Hofheinz, J. Müller-Quade, D. Unruh, On fairness in simulatability-based cryptographic systems, in *Proceedings of the 2005 ACM Workshop on Formal Methods in Security Engineering* (ACM, New York, 2005), pp. 13–22. Full version online available at http://eprint.iacr.org/2005/294. The definition of nontrivial protocols is only contained in the full version

[2] M. Backes, B. Pfitzmann, M. Waidner, The reactive simulatability (RSIM) framework for asynchronous systems. *Inform. Comput.* (2007). Preliminary version available at http://eprint.iacr.org/2004/082

[3] B. Barak, R. Canetti, J.B. Nielsen, R. Pass, Universally composable protocols with relaxed set-up assumptions, in *45th Symposium on Foundations of Computer Science*. Proceedings of FOCS 2004, Rome, Italy, 17–19 October 2004 (IEEE Computer Society, Los Alamitos, 2004), pp. 186–195

[4] D. Boneh, Twenty years of attacks on the RSA cryptosystem. *Not. Am. Math. Soc. (AMS)* **46**(2), 203–213 (1999). Online available at http://crypto.stanford.edu/~dabo/abstracts/RSAattack-survey.html

[5] G. Brassard, D. Chaum, C. Crépeau, Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.* **37**, 156–189 (1988)

[6] G. Brassard, C. Crépeau, D. Mayers, L. Salvail, Defeating classical bit commitments with a quantum computer. Los Alamos preprint arXiv:quant-ph/9806031, May 1999

[7] C. Cachin, U. Maurer, Unconditional security against memory-bounded adversaries, in *Advances in Cryptology*, ed. by B.S. Kaliski Jr. Proceedings of CRYPTO '97. LNCS, vol. 1294 (Springer, Berlin, 1997), pp. 292–306

[8] C. Cachin, C. Crépeau, J. Marcil, Oblivious transfer with a memory-bounded receiver, in *34th Annual ACM Symposium on Theory of Computing*. Proceedings of STOC 2002 (ACM, New York, 2002), pp. 493–502

[9] J. Camenisch, A. Lysyanskaya, A signature scheme with efficient protocols, in *Proc. 3rd International Conference on Security in Communication Networks (SCN)*. LNCS, vol. 2576 (Springer, Berlin, 2002), pp. 268–289

[10] R. Canetti, Security and composition of multiparty cryptographic protocols. *J. Cryptol.* **13**(1), 143–202 (2000)

[11] R. Canetti, Universally composable security: a new paradigm for cryptographic protocols, in *42th Annual Symposium on Foundations of Computer Science*. Proceedings of FOCS 2001 (IEEE Computer Society, Los Alamitos, 2001), pp. 136–145. Full and revised version is online available at http://eprint.iacr.org/2000/067

[12] R. Canetti, M. Fischlin, Universally composable commitments, in *Advances in Cryptology*, ed. by J. Kilian. Proceedings of CRYPTO '01. LNCS, vol. 2139 (Springer, Berlin, 2001), pp. 19–40. Full version online available at http://eprint.iacr.org/2001/055

[13] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in *Advances in Cryptology—EUROCRYPT 2001*, ed. by B. Pfitzmann. LNCS, vol. 2045 (Springer, Berlin, 2001), pp. 453–474

[14] R. Canetti, T. Rabin, Universal composition with joint state, in *Advances in Cryptology: CRYPTO 2003*. LNCS, vol. 2729 (Springer, Berlin, 2003), pp. 265–281

[15] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, Universally composable two-party and multi-party secure computation, in *34th Annual ACM Symposium on Theory of Computing*. Proceedings of STOC 2002 (ACM, New York, 2002), pp. 494–503. Extended abstract, full version online available at http://eprint.iacr.org/2002/140

[16] R. Canetti, E. Kushilevitz, Y. Lindell, On the limitations of universally composable two-party computation without set-up assumptions, in *Advances in Cryptology*, ed. by E. Biham. Proceedings of EUROCRYPT '03. LNCS, vol. 2656 (Springer, Berlin, 2003), pp. 68–86. Full version online available at http://eprint.iacr.org/2004/116

[17] R. Canetti, Y. Dodis, R. Pass, S. Walfish, Universally composable security with global setup, in *Theory of Cryptography*. Proceedings of TCC 2007. LNCS, vol. 4392 (Springer, Berlin, 2007), pp. 61–85

[18] I. Damgård, J.B. Nielsen, Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor, in *Advances in Cryptology*, ed. by Y. Moti. Proceedings of CRYPTO '02. LNCS, vol. 2442 (Springer, Berlin, 2002), pp. 581–596. Full version online available at http://eprint.iacr.org/2001/091

[19] I.B. Damgård, S. Fehr, L. Salvail, C. Schaffner, Cryptography in the bounded quantum-storage model, in *Proceedings of FOCS 2005* (2005), pp. 449–458. A full version is available at http://arxiv.org/abs/quant-ph/0508222

[20] W. Diffie, P.C. van Oorschot, M.J. Wiener, Authentication and authenticated key exchanges. *Des. Codes Cryptogr.* **2**(2), 107–125 (1992)

[21] D. Dolev, C. Dwork, M. Naor, Non-malleable cryptography (extended abstract), in *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing* (ACM, New York, 1991), pp. 542–552

[22] S. Dziembowski, U. Maurer, On generating the initial key in the bounded-storage model, in *Advances in Cryptology*, ed. by C. Cachin, J. Camenisch. Proceedings of EUROCRYPT '04. LNCS, vol. 3027 (Springer, Berlin, 2004), pp. 126–137

[23] U. Feige, A. Shamir, Witness indistinguishable and witness hiding protocols, in *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing* (ACM, New York, 1990), pp. 416–426

[24] U. Feige, D. Lapidot, A. Shamir, Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM J. Comput.* **29**(1), 1–28 (1999)

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

[26] O. Goldreich, *Foundations of Cryptography—Volume 2 (Basic Applications)* (Cambridge University Press, Cambridge, 2004). Preliminary version online available at http://www.wisdom.weizmann.ac.il/~oded/frag.html

[27] O. Goldreich, H. Krawczyk, On the composition of zero-knowledge proof systems. *SIAM J. Comput.* **25**(1), 169–192 (1996)

[28] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)

[29] O. Goldreich, S. Micali, A. Wigderson, How to play any mental game—a completeness theorem for protocols with honest majority, in *Nineteenth Annual ACM Symposium on Theory of Computing*. Proceedings of STOC 1987 (ACM, New York, 1987), pp. 218–229. Extended abstract

[30] O. Goldreich, S. Micali, A. Wigderson, Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM* **38**(3), 690–728 (1991). Online available at http://www.wisdom.weizmann.ac.il/~oded/X/gmw1j.pdf

[31] S. Goldwasser, S. Micali, R.L. Rivest, A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* **17**(2), 281–308 (1988). Online available at http://theory.lcs.mit.edu/~rivest/GoldwasserMicaliRivest-ADigitalSignatureSchemeSecureAgainstAdaptiveChosenMessageAttacks.ps

[32] C. Günther, An identity-based key-exchange protocol, in *Advances in Cryptology—EUROCRYPT '89*, ed. by J.-J. Quisquater, J. Vandewalle. vol. 434 (Springer, Berlin, 1990), pp. 29–37

[33] I. Haitner, O. Reingold, Statistically-hiding commitment from any one-way function, in *STOC '07: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing* (ACM, New York, 2007), pp. 1–10

[34] D. Hofheinz, J. Müller-Quade, Universally composable commitments using random oracles, in *Theory of Cryptography*, ed. by M. Naor. Proceedings of TCC 2004. LNCS, vol. 2951 (Springer, Berlin, 2004), pp. 58–76

[35] D. Hofheinz, J. Müller-Quade, D. Unruh, Polynomial runtime in simulatability definitions, in *18th IEEE Computer Security Foundations Workshop*. Proceedings of CSFW 2005 (IEEE Computer Society, Los Alamitos, 2005), pp. 156–169. Online available at http://iaks-www.ira.uka.de/home/unruh/publications/hofheinz05polynomial.html

[36] D. Hofheinz, J. Müller-Quade, R. Steinwandt, On modeling IND-CCA security in cryptographic protocols. *Tatra Mt. Math. Publ.* **33**, 83–97 (2006). Full version available at http://eprint.iacr.org/2003/024

[37] D. Hofheinz, J. Müller-Quade, D. Unruh, On the (im)possibility of extending coin toss, in *Advances in Cryptology*, ed. by S. Vaudenay. Proceedings of EUROCRYPT '06. LNCS, vol. 4004 (Springer, Berlin, 2006), pp. 504–521. Full version available at http://eprint.iacr.org/2006/177

[38] D. Hofheinz, D. Unruh, J. Müller-Quade, Universally composable zero-knowledge arguments and commitments from signature cards. *Tatra Mt. Math. Publ.* **37**, 93–103 (2007)

[39] Y. Lindell, General composition and universal composability in secure multi-party computation, in *44th Annual Symposium on Foundations of Computer Science*. Proceedings of FOCS 2003 (IEEE Computer Society, Los Alamitos, 2003), pp. 394–403. Online available at http://eprint.iacr.org/2003/141

[40] U.M. Maurer, K. Pietrzak, R. Renner, Indistinguishability amplification, in *Proceedings of Crypto'07*, ed. by A. Menezes. LNCS, vol. 4622 (Springer, Berlin, 2007), pp. 130–149

[41] J. Müller-Quade, Temporary assumptions—quantum and classical, in *The 2005 IEEE Information Theory Workshop On Theory and Practice in Information-Theoretic Security* (2005). Abstract

[42] J. Müller-Quade, D. Unruh, Long-term security and universal composability, in *Theory of Cryptography, Proceedings of TCC 2007*. LNCS, vol. 4392 (Springer, Berlin, 2007), pp. 41–60

[43] M. Naor, R. Ostrovsky, R. Venkatesan, M. Yung, Perfect zero-knowledge arguments for NP using any one-way permutation. *J. Cryptol.* **11**(2), 87–108 (1998)

[44] M.O. Rabin, Hyper-encryption by virtual satellite. Science Center Research Lecture Series, December 2003. Online available at http://athome.harvard.edu/programs/hvs/

[45] J. Rompel, One-way functions are necessary and sufficient for secure signatures, in *Twenty-Second Annual ACM Symposium on Theory of Computing*. Proceedings of STOC 1990 (ACM, New York, 1990), pp. 387–394

[46] S. Wehner, J. Wullschleger, Composable security in the bounded-quantum-storage model, in *ICALP 2008, Track C*. LNCS (Springer, Berlin, 2008), pp. 604–615. Full available at http://arxiv.org/abs/0709.0492v1