Long-term Security and Universal Composability

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

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

Abstract. Algorithmic progress and future technology threaten today's cryptographic protocols. Long-term secure protocols should not even in future reveal more information to a—then possibly unlimited—adversary.

In this work we initiate the study of protocols which are long-term secure *and* universally composable. We show that the usual set-up assumptions used for UC protocols (e.g., a common reference string) are not sufficient to achieve long-term secure *and* composable protocols for commitments or general zero knowledge arguments. Surprisingly, nontrivial zero knowledge protocols are possible based on a coin tossing functionality: We give a long-term secure composable zero knowledge protocol proving the knowledge of the factorisation of a Blum integer.

Furthermore we give practical alternatives (e.g., signature cards) to the usual setup-assumptions and show that these allow to implement the important primitives commitment and zero-knowledge argument.

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

1 Introduction

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

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

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

To capture the threat of an adversary with increasing power we introduce the security notion of *long-term universal composability* (long-term-UC) with the intuition that the adversary becomes unlimited at some point of time after termination of the protocol. The protocols do not run after this point of time, but all information stored from past executions should not reveal any additional information to the then unlimited adversary. A surprising consequence of our work is that unconditionally hiding universally composable commitments [11] are not necessarily long-term-UC.

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

The idea of everlasting security has been considered with respect to memory bounded adversaries. Key exchange protocols and protocols for oblivious transfer have been developed in the bounded storage model [5,4]. These protocols can be broken by an adversary with more memory than assumed, however they cannot be broken in retrospect even by an unlimited adversary. A scheme using distributed servers of randomness (virtual satellites) to achieve everlasting security has been implemented [22]. In this scheme the access of the adversary to the communication of the parties is limited during the key exchange. It was shown by [12] that in the bounded-storage model composability cannot be taken for granted. They gave a key-exchange protocol that is secure in the bounded-storage model even if the initial key leaks after protocol termination, and then showed that if the initial key was generated by a computationally secure key exchange protocol, the resulting protocol is insecure. However, theirs was a purely negative result in that they did not give any criteria under which composition would be possible.

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

Zero knowledge proofs where the verifier cannot (ever) break the protocol and the prover can only on-line break the protocol where given in [2]. In [20] protocols achieving long-term security were stated, however, only secure function evaluation with constant input size was considered.

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

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

With exception of [12], previous work on long-term security did not take the problem of composability into account. When composability is required the situation changes drastically. E.g., an unconditionally hiding UC commitment is not long-term-UC and a straightforward adaption of e.g., the protocol of [2] using an unconditionally hiding UC commitment does not yield long-term-UC zero knowledge arguments.

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

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

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

1.1 Preliminaries

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

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

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

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

Cryptographic tools. In [21], it is shown that assuming the existence of a one-way permutation, an unconditionally hiding commitment scheme exists. This scheme has the additional properties that the unveil-phase consists of only one message, and that given the message, the committed value v, and the transcript of the interaction in the commit phase, there is a deterministic polynomial-time algorithm that checks whether the verifier accepts the value v.

Using that commitment-scheme in the zero-knowledge proof-system for graph-3-colourability from [16], we get a statistically witness indistinguishable argument of knowledge for any NP-relation given any one-way permutation.[4] Using a statistically witness indistinguishable argument of knowledge for any NP-relation and a unconditionally hiding commitment scheme, we can easily construct a statistically witness indistinguishable argument of knowledge for any MA-relation using any one-way permutation.[5]

2 Modelling long-term UC

We now present our modelling of universally composable long-term security (short long-term UC). We build on the Universal Composability framework [7]. In that modelling, a computationally limited entity called the environment has to distinguish between an execution of the protocol (with some adversary) and an execution of an ideal functionality (with some simulator). To define long-term security, we have to add the requirement that even if some entity gets unlimited computational power after the execution of the protocol, security is maintained. In the Universal Composability framework, this is quite easily done: We simply require that *after* the execution of the protocol (which is still performed against

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

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

computationally limited adversaries) even an unlimited entity could not distinguish between an execution of the real protocol or of the functionality, i.e., we require that the output of the environment is *statistically* indistinguishable.[6]

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

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

Note that the Universal Composition Theorem from [7] applies with a virtually unmodified proof.

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

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

2.1 On the minimality of the security notion

At this point one might wonder whether this definition is possibly stricter than necessary, especially in view of the various impossibility results presented below. However, if one is willing to accept stand-alone security (i.e., simulation-based security *without* an environment, see e.g. [15]), with the extra requirement that the outputs of the parties and the adversary/simulator are *statistically* indistinguishable in real and ideal model (long-term stand-alone security), as a minimal security notion, we can argue as follows: If we want this minimal security *and*

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

[7] See [7] for details.

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

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

composability simultaneously, the proof from [18][10] states that the minimal security notion satisfying these two requirements is a security notion similar to Definition 1, with the only difference that the simulator is allowed to depend on the environment (specialised-simulator long-term UC). Since all our impossibility results also apply for this weaker notion (we never use the fact that the simulator does not depend on the environment), we see that we cannot find an essentially more lenient security notion than Definition 1 if we accept long-term stand-alone security as a minimal security notion.

2.2 Functionalities

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

We assume the following conventions in specifying functionalities:

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

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

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

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

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

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

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

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

the coin.[11] As we will see below, a coin toss is more powerful than a CRS in the context of long-term UC.[12]

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

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

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

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

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

We call C the sender and R the recipient.

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

We call P the prover and V the verifier.

3 Commitment

In this section we will examine the possibility of long-term-UC realising commitments. It will turn out, that commitment cannot be long-term-UC realised using CRS or coin-toss, nor with an arbitrary PKI. In particular unconditionally hiding

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

[12] Although, in contrast, a UC secure (without long-term) coin toss can be realised using a CRS under reasonable complexity assumptions, see [9].

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

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

UC commitments, which are possible with a CRS [11], are not necessarily long-term UC.[15] Note that the incompleteness of the CRS stands in stark contrast to the situation of (non-long-term) UC. In [10] it was shown that given a CRS, any functionality has a UC secure realisation. Furthermore, in [1] it was shown that the same holds for a PKI.[16] However, given a ZK functionality, commitments can be realised even with respect to long-term UC.

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

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

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

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

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

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

Theorem 9 (Impossibility of commitment with OTS functionalities). *Let $\mathcal{F}$ be a functionality that is OTS for party C. Then there is no nontrivial*

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

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

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

protocol that long-term-UC realises commitment with sender C ($\mathcal{F}_{\mathrm{COM}}^{C\rightarrow R}$) in the $\mathcal{F}$-hybrid model.

If one is willing to assume NP $\not\subseteq$ P/poly, this theorem is an immediate consequence of Lemma 17 stating that $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT},C\rightarrow R}$ (ZK for SAT with the sender C being the prover) is possible from $\mathcal{F}_{\mathrm{COM}}^{C\rightarrow R}$, and Corollary 15 stating that $\mathcal{F}_{\mathrm{ZK}}^{\mathrm{SAT},C\rightarrow R}$ cannot be realised using $\mathcal{F}$ (both shown in Section 4). However, in the full version [19] we give a direct proof (similar in spirit to that of Theorem 14) for this theorem that does not depend on NP $\not\subseteq$ P/poly.

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

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

Proof. Immediate from Lemma 8 and Theorem 9. □

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

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

The protocol π looks as follows:

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

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

The actual proof is given in the full version [19].

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

4 Zero-Knowledge

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

4.1 Using OTS functionalities

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

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

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

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

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

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

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

The proof is given in the full version [19].

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

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

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

The proof is given in the full version [19].

Note that we cannot expect an analogous result in the case that $\mathcal{F}$ is OTS for the verifier V, since commitments are OTS for the recipient and Lemma 17 show that $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$ can be long-term-UC implemented using commitments with the verifier V as recipient.

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

Corollary 15. *Let $\mathcal{F}$ be a functionality that is OTS for party P. If one-way-functions (secure against* uniform *adversaries) exist,* or *if NP $\not\subseteq$ P/poly, there is no nontrivial long-term-UC secure protocol for ZK with prover P for SAT in the $\mathcal{F}$-hybrid model.*

Proof. Immediate from Lemma 13 and Theorem 14. □

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

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

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

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

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

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

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

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

Now for the case that the *prover is corrupted*. In this case, the simulator needs to interact with the environment incorporating the prover and to extract

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

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

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 $\mathcal{F}_{\text{ZK}}^{R}$ which ensures (by definition of R) that n is a Blum-integer. So in this case there is no problem. However, if the prover is corrupted, the simulator will have to choose the coin-toss $r_1, \dots, r_n$. If n is not a Blum-integer, he might learn this later on (since he learns (p, q) in case of a successful proof), but then it might already be too late, because the simulator sends the r_i to the environment before the end of the proof (the algorithm Q does not guarantee $r_1, \dots, r_n$ to be (almost) uniformly distributed if n is not a Blum-integer). To overcome this difficulty, we add an additional step to the beginning of the protocol. *Before the coin-toss is invoked*, the prover proves that n is indeed a Blum-integer. If the prover succeeds in this proof, the simulator can use the algorithm Q without danger, otherwise the simulator may abort (since the verifier would have done so, too). However, this introduces the additional difficulty that in case of a corrupted verifier, the simulator has to perform that proof, too, and without knowledge of the witness. To achieve this, we make use of the FLS-technique [13]: Prover and verifier first invoke another instance of the coin-toss functionality (in this case, a CRS would be sufficient, too) and then the prover proves using a statistically witness indistinguishable argument of knowledge to the verifier that either n is a Blum-integer or that he knows a the preimage of the coin-toss t under a one-way permutation f. Then the simulator can simulate this proof by simply choosing $t = f(u)$ for uniform u. Since $f(u)$ is uniformly distributed, this is indistinguishable from what an honest prover knowing the witness would produce. After having successfully performed this first step, prover and verifier proceed with the protocol as described above.

The actual proof for Theorem 16 is given in the full version [19].

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

Lemma 17 (ZK from commitment). *Let R be a NP-relation. Then there is a long-term-UC secure protocol π for ZK with relation R (i.e., $\mathcal{F}_{\text{ZK}}^{R,P\to V}$) using a polynomial number of commitments from prover P to verifier V (i.e., $\mathcal{F}_{\text{COM}}^{P\to V}$).*

Proof. [9] gives a UC secure protocol that realises $\mathcal{F}_{\text{ZK}}^{R,P\to V}$ using $\mathcal{F}_{\text{COM}}^{P\to V}$ where R is the relation for the Hamilton cycle problem. Their result even holds unconditionally (i.e., even when the environment is unlimited *during* the execution of the protocol) and therefore in particular with respect to long-term UC. Since the Hamilton cycle problem is NP-complete, the lemma follows. □

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

4.2 Using offline functionalities

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

Definition 18 (Offline functionalities). *We call a functionality $\mathcal{F}$ offline, if it has the following form: When $\mathcal{F}$ runs with parties $P_1, \dots, P_n$, upon its first activation, it chooses values $(c, c_{P_1}, \dots, c_{P_n})$ according to a fixed distribution and sends c to the adversary. When receiving any input from P_i, send c_{P_i} to P_i.*

Lemma 19. *CRS and PKI are offline functionalities.*

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

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

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

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

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

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

The full proof is given in the full version [19].

A natural question arising in this context is whether this impossibility result can be made stronger. In particular, one might ask whether such an impossibility result already holds if $\mathcal{F}$ is OTS for P *or* for V. Further one might ask, whether the theorem can be strengthened to state impossibility of ZK for *uniformly nontrivial* relations. These questions are discussed in the full version.

Lemma 8 tells us that at least for some commonly used encryption schemes, $\mathcal{F}^G_{\mathrm{PKI}}$ is OTS for all parties (here and in the following G denotes the key generation algorithm) and therefore cannot be used for long-term-UC realising commitment or zero-knowledge[21]. However, in general this is not the case. As we show in the full version, there exist special public key schemes for which a PKI can be used for constructing ZK and commitment protocols.

5 Other setup-assumptions

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

5.1 Trusted devices implementing a random oracle

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

[21] Except for nonuniformly trivial relations, see Theorem 20.

functionality that evaluates a pseudorandom function with a secret seed (representing e.g. a (temporarily) trusted device).

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

Definition 21 (Trusted pseudorandom function (TPF)). *Let f_s be an efficiently computable family of deterministic functions $f_s : \{0,1\}^{l(|s|)} \to \{0,1\}^{l(|s|)}$ with polynomially bounded l.*

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

At this point, one should note that the UC definition (and therefore our variant, too) implicitly assumes that when using a TPF, that TPF is accessed only by the protocol (and the adversary), but that it cannot be directly accessed by the environment. This in particular rules out that different protocols share a single TPF. A more detailed analysis of the consequences of this assumption can be found in [17,8]. However, we show that using a single TPF we can perform an *arbitrary number* of zero knowledge arguments or commitments, so that at least we do not need a large number of TPFs when constructing a larger protocol that performs many ZK arguments or commitments.

Theorem 22 (ZK from TPF). *Assume that a one-way permutation exists. Let f_s be a pseudorandom function (as in [14]), and R an NP-relation. Then there is a nontrivial protocol π using one instance of $\mathcal{F}^f_{\mathrm{TPF}}$ that long-term-UC realises unlimited number of instances of $\mathcal{F}^R_{\mathrm{ZK}}$ (i.e., ZK for the relation R).*

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

Using this extractable commitment we modify the zero knowledge protocol for graph-3-colourability of [16]. 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 case of a corrupted verifier the simulator can extract the edge which will later be investigated before committing to the colouring. So the simulator can easily commit to a fake colouring and still pass the test at the edge in question.

In both cases the communication between the parties, the adversary and the environment are statistically indistinguishable in the real protocol and in this simulation and we achieve a long-term-UC zero knowledge argument for graph-3-colouring and hence for all NP-statements. The complete proof can be found in the full version [19].

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

Corollary 23 (Commitments from TPF). *Assume that a one-way permutation exists. Let f_s be a pseudorandom function. Then there is a nontrivial protocol π using one instance of $\mathcal{F}_{\mathrm{TPF}}^{f}$ that long-term-UC realises an unlimited number of instances of $\mathcal{F}_{\mathrm{COM}}$ (i.e., commitments).*

Proof. Immediate from Lemma 11 and Theorem 22. □

5.2 Signature cards

One disadvantage of the TPF-assumption from the foregoing section is that trusted hardware implementing a pseudorandom function are unlikely to be available for practical use.[22] However, another kind of trusted device is already available commercially today: the signature card. A signature card is a tamperproof device with an built-in secret key. Upon request, this card signs an arbitrary document, but *never* reveals the secret key. The corresponding public key can be obtained from some certification authority. These properties are required e.g. from the German signature law [23].

These properties are captured by the following ideal functionality (based on [17]):

Definition 24 (Signature Card (SC)). *Let $\mathfrak{S} = (KeyGen, Sign, Verify)$ be a signature scheme. Let H be a party. Then the functionality $\mathcal{F}_{\mathrm{SC}}^{H,\mathfrak{S}}$ (signature card for scheme $\mathfrak{S}$ with holder H) behaves as follows: Upon the first activation, $\mathcal{F}_{\mathrm{SC}}^{H,\mathfrak{S}}$ chooses a public/secret key pair (pk, sk) using the key generation algorithm $KeyGen(1^k)$. Upon a message (pk) from a party P or the adversary, send pk to that party or the adversary, resp. Upon a message $(sign, m)$* from the holder *H, produce a signature σ for m using the secret key sk and send σ to H.*[23]

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 21. However, in [17] techniques where introduced that allow to share

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

[23] The definition from [17] 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.

a single signature card in different protocols. It would be interesting to explore whether their approach can also be applied to our scenario.

It was shown in [17] 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 25 (ZK from a signature card). *Assume that a one-way permutation exists. Let $\mathfrak{S}$ be an EF-CMA secure signature scheme. Let R be any MA-relation. Then there is a nontrivial protocol π that long-term-UC realises an unbounded number of instances of $\mathcal{F}_{\mathrm{ZK}}^{R,P\to V}$ (i.e., ZK for the relation R with prover P) using a single instance of $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ (i.e., a signature card for $\mathfrak{S}$ with P as the holder).*

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

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

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

The full proof is given in the full version [19].

Corollary 26 (Commitments from a signature card). *Assume that a one-way permutation exists. Let $\mathfrak{S}$ be an EF-CMA secure signature scheme. Then there is a nontrivial protocol π that long-term-UC realises an unbounded number of instances of $\mathcal{F}_{\mathrm{COM}}^{C\to R}$ (i.e., commitment with sender C) using a single instance of $\mathcal{F}_{\mathrm{SC}}^{\mathfrak{S},P}$ (i.e., a signature card for $\mathfrak{S}$ with P as the holder).*

Proof. This is an immediate consequence of Theorem 25 and Lemma 11. □

6 Conclusions

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

Further research in this directions might include the following:

- Which protocol tasks can or cannot be long-term-UC realised using commitments and zero-knowledge proofs.
- What other setup-assumptions might be useful in the context of long-term UC. In particular, under which assumptions can OT (and therefore any functionality) be realised?
- Our investigations were in the secure-channels communication-model. If only authenticated channels are present, the important issue of key exchange occurs. What setup-assumptions are necessary to implement the latter?
- The protocols presented here were not optimised for efficiency. To what extend can efficient protocols be found for the tasks discussed in this work?
- In [17] techniques were presented that allow to share a single signature card between different protocols. Can these techniques be applied to our setting, too?
- Much work on unconditional and long-term security has been done in the field of quantum cryptography. How does long-term UC behave in the presence of quantum communication. Can some of the impossibility results given in this work be avoided? In particular, quantum communication could solve the problem of key exchange mentioned above.

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

References

1. Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Symposium on Foundations of Computer Science, Proceedings of FOCS 2004, 17-19 October 2004, Rome, Italy*, pages 186–195. IEEE Computer Society, October 2004.
2. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge, 1988. JCSS, 37:156-189.
3. Gilles Brassard, Claude Crépeau, Dominic Mayers, and Louis Salvail. Defeating classical bit commitments with a quantum computer. Los Alamos preprint archive quant-ph/9806031, May 1999.
4. Christian Cachin, Claude Crépeau, and Julien Marcil. Oblivious transfer with a memory-bounded receiver. In *34th Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2002*, pages 493–502. ACM Press, 2002.
5. Christian Cachin and Ueli Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology, Proceedings of CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer-Verlag, 1997.
6. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001*, pages 136–145. IEEE Computer Society, 2001.

7. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive, December 2005. Full and revised version of [6].
8. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. These proceedings.
9. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology, Proceedings of CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 2001.
10. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2002*, pages 494–503. ACM Press, 2002. Extended abstract.
11. Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology, Proceedings of CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer-Verlag, 2002.
12. Stefan Dziembowski and Ueli Maurer. On generating the initial key in the bounded-storage model. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology, Proceedings of EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 126–137. Springer-Verlag, 2004.
13. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.
14. Oded Goldreich. *Foundations of Cryptography – Volume 1 (Basic Tools)*. Cambridge University Press, August 2001.
15. Oded Goldreich. *Foundations of Cryptography – Volume 2 (Basic Applications)*. Cambridge University Press, May 2004.
16. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):690–728, 1991.
17. Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. Universally composable zero-knowledge arguments and commitments from signature cards. In *Proceedings of the 5th Central European Conference on Cryptology, MoraviaCrypt '05*, 2005.
18. Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *44th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2003*, pages 394–403. IEEE Computer Society, 2003.
19. Jörn Müller-Quade and Dominique Unruh. Long-term security and universal composability, 2006. Full version of this paper, IACR ePrint 2006/422.
20. Jörn Müller-Quade. Temporary assumptions—quantum and classical. In *The 2005 IEEE Information Theory Workshop On Theory and Practice in Information-Theoretic Security*, 2005. abstract.
21. Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, March 1998.
22. Michael O. Rabin. Hyper-encryption by virtual satellite. Science Center Research Lecture Series, December 2003.
23. Gesetz über Rahmenbedingungen für elektronische Signaturen. Bundesgesetzblatt I 2001, 876, May 2001.