Universally Composable Zero-Knowledge Proof of Membership

Jesper Buus Nielsen*
University of Aarhus
Department of Computer Science
IT-parken, Aabogade 34
DK-8200 Aarhus N, Denmark
buus@daimi.au.dk

February 15, 2005

Abstract

Since its introduction the UC framework by Canetti has received a lot of attention. A contributing factor to its popularity is that it allows to capture a large number of common cryptographic primitives using ideal functionalities and thus can be used to give modular proofs for many cryptographic protocols. However, an important member of the cryptographic family has not yet been captured by an ideal functionality, namely the *zero-knowledge proof of membership*. We give the first formulation of a *UC zero-knowledge proof of membership* and show that it is closely related to the notions of straight-line zero-knowledge and simulation soundness.

1 Introduction

Since its introduction the UC framework [Can01] by Canetti have received a lot of attention (see e.g. [Can]). To define security of a protocol π (formally captured via a network of interactive Turing machines (ITMs)), one formalizes an abstract (and potentially much simpler) ITM $\mathcal{F}$, a so-called ideal functionality (IF), which obviously has the properties that we want π to have. The framework then contains a definition of when π is as secure as $\mathcal{F}$. We write this as $\pi \overset{c}{\triangleright} \mathcal{F}$. The framework then proves a composition theorem that if $\pi = \pi' \diamond \mathcal{G}$ is a protocol using the IF $\mathcal{G}$ and $\pi \overset{c}{\triangleright} \mathcal{F}$ and $\rho \overset{c}{\triangleright} \mathcal{G}$, then $(\pi' \diamond \rho) \overset{c}{\triangleright} \mathcal{F}$, where $\pi' \diamond \rho$ denotes the protocol where π uses ρ in place of $\mathcal{G}$.

The goal of having a composition theorem is two-fold. First, it guarantees that security defined via realizing an IF is *strong*. In fact, it is maintained in any context, and is thus called universally composable (UC) security in [Can01]. Second, it allows *modular proofs*. If one wants to prove a protocol secure, i.e. prove $\pi \overset{c}{\triangleright} \mathcal{F}$ for some IF $\mathcal{F}$, then one can write $\pi = \pi' \diamond \rho$ for some sub-protocol performing a well-defined task, capture this task by an IF $\mathcal{G}$ and then prove $(\pi' \diamond \mathcal{G}) \overset{c}{\triangleright} \mathcal{F}$ and $\rho \overset{c}{\triangleright} \mathcal{G}$.

A contributing factor to the popularity of the UC framework is that it allows to capture a large number of common cryptographic primitives like e.g. signatures and public-key encryption, proofs of knowledge and commitments by IFs. However, an important member of the cryptographic family has not yet been captured by an IF, namely the *zero-knowledge proof of membership*. (I.e. a proof that $x \in L$ without proving knowledge of a witness to $x \in L$.) The reason for this is that there only seems to be one natural IF for specifying a zero-knowledge (ZK) proof, called $\mathcal{F}_{\text{zk}}$ in [Can01], and that any protocol realizing $\mathcal{F}_{\text{zk}}$ must necessarily be a proof

*Supported by FICS, Foundations In Cryptology and Security, centre of the Danish National Science Research Council.

of knowledge. With respect to the first goal, this is not a problem. If we are only interested in strong security definitions, nothing is lost by saying that a UCZK proof of membership is a protocol realizing the IF $\mathcal{F}_{\text{zk}}$ for proof of knowledge. Clearly knowledge soundness implies membership soundness. With respect to the second goal of the UC framework, not having an IF capturing *just* a proof of membership means that many protocols using proofs of membership cannot be given a modular proof in the UC framework. Since ZK proofs of membership is used in a large number of protocols, this is a real restriction of the of the current state of affairs.

This paper gives the first definition of a *UCZK proof of membership* and initiates a study of the notion, by showing that the notion of UCZK proof of membership is strongly related to the notions of straight-line ZK and simulation soundness [Sah99, GMY03].

Related work. We could also have cast our study in the related *reactive simulatability* framework [PSW00] by Pfitzmann, Schunter and Waidner. But since most of the work on UCZK has been done in the UC framework, this seems to be the natural choice. We are not aware of any previous work on defining UCZK proof of membership in either framework.

2 The UC framework

We start by describing the UC framework. In the UC framework, security of protocols is defined in three steps. First, the process of executing a protocol in an environment is formalized. Next, an ideal process for carrying out the task that the protocol is intended to solve is formalized. This is done via a so-called IF, which is programmed to have the intended input-output behavior of the protocol, and leaking (on a so-called leakage tape) only the information that is not considered confidential. A protocol is said to securely realize an IF if the execution of the protocol can be simulated given only the information leaked by the IF in the ideal process.

The UC framework has several instantiations. We focus here on modeling an asynchronous network. The communication is public and authenticated. Parties may be broken into (i.e., become corrupted) throughout the computation, and once corrupted their behavior is arbitrary, i.e. we consider active, adaptive corruptions. Finally, all the involved entities are restricted to probabilistic polynomial time (PPT) computation.

Technically we present the variant of the UC framework from [Can], where the real-life adversary and the environment are one entity. We assume some knowledge of the UC framework on behalf of the reader and present the framework in a less intuitive order than in [Can]. Specifically we first specify the hybrid model and then derive the real-life model as a special case.

2.1 Working with ITMs

We model the entities in a protocol as interactive Turing machines (ITMs)[1] It is therefore convenient to have some definitions for working with ITMs.

Networks. We consider ITMs where the write-only output tapes and the read-only input tapes are named by labels $L \in \{0,1\}^*$. We call them in-tapes and out-tapes. By a collection of ITMs we mean a set of ITMs where no two ITMs have identically named in-tapes or identically named out-tapes. By a network we mean a collection of ITMs where each out-tape with name L is connected to the in-tape with the same name, if it exists in the collection. By connected we mean that the ITMs have access to a shared tape, which is the out-tape of the first machine and the in-tape of the other. The in-tapes and out-tapes of a network which are not connected to another tape are called the open tapes. We call a network with no open tapes a closed network. An ITM is also labeled by a bit $I \in \{0,1\}$ telling whether it is to act as the initial ITM. We require that a network contains at most one initial ITM and that a closed network contains exactly one initial ITM.

[1] See [Can] for an appropriate formalization. We will use several times that it holds for the formalization in [Can] that the composition of PPT ITMs gives PPT ITMs.

Composing and closing. For two networks $\mathcal{N}_1$ and $\mathcal{N}_2$ we let $\mathcal{N}_1 \diamond \mathcal{N}_2$ denote the network with the ITMs of both networks. If $\mathcal{N}_1$ and $\mathcal{N}_2$ does not make up a collection (because of conflicting tape names) we write $\mathcal{N}_1 \diamond \mathcal{N}_2 = \bot$. For any network $\mathcal{N}$ we then let $\mathcal{N}^{\mathtt{C}}$ be the set of PPT ITMs $\mathcal{Z}$ for which it holds that $\mathcal{N} \diamond \mathcal{Z}$ is a closed network. This means that $\mathcal{Z}$ must be an initial ITM iff $\mathcal{N}$ has no initial ITM.

Executing. A closed network $\mathcal{N}$ can be executed on a security parameter $k \in \mathbf{N}$ and auxiliary input $z \in \{0,1\}^*$: First the initial ITM M_0 is given input (k, z) and is activated, runs its code and writes some messages on its out-tapes. Then the receiver M on the last tape that M_0 wrote is activated next. If this is the first time M is activated, it is first given k as input. Then M writes messages on some of its out-tapes, and the iteration continues. If some ITM does not write on any out-tape, then M_0 is activated next. If at some point M_0 is activated and does not write on any out-tape, then the execution stops. The output of the execution is the first bit b on the work-tape of M_0, or 0 if this tape is empty. We write this as $b = \text{EXEC}_{\mathcal{N}}(k, z)$. This is a random variable and defines a distribution ensemble $\text{EXEC}_{\mathcal{N}} = \{\text{EXEC}_{\mathcal{N}}(k, z)\}_{k \in \mathbf{N}, z \in \{0,1\}}$. For two closed networks $\mathcal{N}_1$ and $\mathcal{N}_2$ we write $\mathcal{N}_1 \overset{\text{c}}{\approx} \mathcal{N}_2$ if $\text{EXEC}_{\mathcal{N}_1}$ and $\text{EXEC}_{\mathcal{N}_2}$ are computationally indistinguishable in the sense of [Can01]. We write $\mathcal{N}_1 \equiv \mathcal{N}_2$ if $\text{EXEC}_{\mathcal{N}_1}$ and $\text{EXEC}_{\mathcal{N}_2}$ identical ensembles.

Joining. Sometimes we consider a network $\mathcal{N}$ as a single ITM $\langle \mathcal{N} \rangle$. It has the code and tapes of the machines in $\mathcal{N}$. The connected tapes of $\mathcal{N}$ are now internal work-tapes. The ITM $\langle \mathcal{N} \rangle$ passes messages and activation around internally according to the rules of executing a network. In particular, if $\mathcal{Z} \in \mathcal{N}^{\mathtt{C}}$, then $\mathcal{N} \diamond \mathcal{Z} \equiv \langle \mathcal{N} \rangle \diamond \mathcal{Z}$.

Communication properties. After an execution $\text{EXEC}_{\mathcal{N}}(k, z)$ we can write down the generated communication $\text{COMM}_{\mathcal{N}}(k, z) = ((L_1, m_1), \ldots, (L_l, m_l))$, where m_i is the i'th message sent and L_i the tape it was sent on. We later need to talk about properties of communication holding except with negligible probability. For this purpose, call $P : (\{0,1\}^* \times \{0,1\}^*)^* \to \{0,1\}$ a communication property and let $P_{\mathcal{N}}(k, z) = P(\text{COMM}_{\mathcal{N}}(k, z))$. This defines a distribution ensemble $P_{\mathcal{N}}$. We say that $\mathcal{N}$ has the property P, written $\mathcal{N} \models^{\text{c}} P$, if $P_{\mathcal{N}} \overset{\text{c}}{\approx} 1$. Finally we say that an open network $\mathcal{N}$ has a communication property P iff $\mathcal{N} \diamond \mathcal{Z} \models^{\text{c}} P$ for all $\mathcal{Z} \in \mathcal{N}^{\mathtt{C}}$. For any $C \in (\{0,1\}^* \times \{0,1\}^*)^*$ and $\mathcal{L} \subseteq \{0,1\}^*$ we let $C^{|\mathcal{L}}$ be the sequence C with all entries (L_i, m) with $L_i \notin \mathcal{L}$ deleted. We say that P only depends on the tapes $\mathcal{L}$ if $P(C_1) = P(C_2)$ for all C_1, C_2 where $C_1^{|\mathcal{L}} = C_2^{|\mathcal{L}}$. For communication properties $P_1, \ldots, P_m$ and $P : \{0,1\}^m \to \{0,1\}$ we define a communication property $P(P_1, \ldots, P_m)(C) = P(P_1(C), \ldots, P_m(C))$. It is straight-forward to verify that $\mathcal{N} \models^{\text{c}} P(P_1, \ldots, P_m)$ for all tautologies P and that $\mathcal{N} \models^{\text{c}} P_1 \wedge P_2$ iff $\mathcal{N} \models^{\text{c}} P_1$ and $\mathcal{N} \models^{\text{c}} P_2$. Also, if P only depends on the open tapes of $\mathcal{N}$, then $\mathcal{N} \models^{\text{c}} P$ iff $\langle \mathcal{N} \rangle \models^{\text{c}} P$. Finally, if $\mathcal{N}_1 \models^{\text{c}} P_1$ and $\mathcal{N}_2$ is a PPT ITM such that $\mathcal{N}_1 \diamond \mathcal{N}_2 \neq \bot$, then $\mathcal{N}_1 \diamond \mathcal{N}_2 \models^{\text{c}} P_1$. Otherwise there would exist $\mathcal{Z} \in (\mathcal{N}_1 \diamond \mathcal{N}_2)^{\mathtt{C}}$ such that $(\mathcal{N}_1 \diamond \mathcal{N}_2) \diamond \mathcal{Z} \not\models P$. But then $\mathcal{N}_1 \diamond (\mathcal{N}_2 \diamond \mathcal{Z}) \not\models P$, and as P only depends on tapes of $\mathcal{N}_1$ it follows that $\mathcal{N}_1 \diamond \langle \mathcal{N}_2 \diamond \mathcal{Z} \rangle \not\models P$. So, since $\langle \mathcal{N}_2 \diamond \mathcal{Z} \rangle \in \mathcal{N}_1{}^{\mathtt{C}}$ we have that $\mathcal{N}_1 \not\models P$, a contradiction.

2.2 The hybrid model

An n-party protocol is a network of ITMs. It has a name $P \in \{0,1\}^*$ and contains n PPT ITMs $\mathcal{P}_1, \ldots, \mathcal{P}_n$ called the parties. Each $\mathcal{P}_i$ has an in-tape $P\text{-}\mathtt{in}_i$ and an out-tape $P\text{-}\mathtt{out}_i$, for receiving inputs to the protocol and delivering outputs. Besides this the network has an IF $\mathcal{F}$ with some name $F \neq P$. This is an ITM with in-tape $F\text{-}\mathtt{in}_i$ and out-tapes $F\text{-}\mathtt{out}_i$ for $i = 1, \ldots, n$. Besides this $\mathcal{F}$ has an out-tape $F\text{-}\mathtt{leak}$ for leaking values and an in-tape $F\text{-}\mathtt{infl}$ allowing to influence the behavior of $\mathcal{F}$. Each party $\mathcal{P}_i$ has an in-tape $F\text{-}\mathtt{out}_i$ and an out-tape $F\text{-}\mathtt{in}_i$ connecting it to $\mathcal{F}$. A protocol π can have several distinctly named IFs $\mathcal{F}_1, \ldots, \mathcal{F}_m$ with similar connections.

As an example of an IF, consider the IF $\mathcal{F}_{\text{at}}$ for authenticated message transmission. If it receives a message (j, m) on $\texttt{at-in}_i$, then it stores (i, j, m), writes (i, j, m) on $\texttt{at-leak}$ and sends the empty string on $P\texttt{-out}_i$. If it receives $(\texttt{deliver}, i, j, m)$ on $\texttt{at-infl}$ and at least one value of the form (i, j, m) is stored, then it deletes a copy of (i, j, m) and sends (i, m) on $\texttt{at-out}_j$. In Fig. 1, right, top a protocol π with name zk is shown, with two parties $P = 1$ and $V = 2$, and two IFs $\mathcal{F}_{\text{at}}$ and $\mathcal{F}_{\text{crs}}$ (with names at and crs). In Fig. 1, right, bottom, the same protocol in $\pi \diamond \mathcal{Z}$ for $\mathcal{Z} \in \pi^{\mathsf{C}}$.

Required behavior. In general an IF $\mathcal{F}$ with name F is required to work as follows: If it receives a value on $F\texttt{-in}_i$ it possibly writes a value on $F\texttt{-leak}$ and then sends a value on $F\texttt{-out}_i$. If it receives a value on $F\texttt{-infl}$, then it is allowed to send a value on any out-tape $F\texttt{-out}_i$. Note the $\mathcal{F}_{\text{at}}$ has this behavior.

A party $\mathcal{P}$ connected to IFs with names $F_1, \ldots, F_m$ is required to work as follows: If it receives an input on $P\texttt{-in}_i$ or any $F_j\texttt{-out}_i$ it sends a message on $P\texttt{-out}_i$ or one of the out-tapes $F_j\texttt{-in}_i$. It is only allowed to write on one tape. We furthermore reserve a symbol $\texttt{corrupt}$. If $\mathcal{P}$ receives $\texttt{corrupt}$ on $P\texttt{-in}_i$, then in some fixed order it sends $\texttt{corrupt}$ on each of the tapes $F_j\texttt{-in}_i$ and waits until a value $state_{F_j,i}$ is received on $F_j\texttt{-out}_i$. Then it sends $(state_{P,i}, state_{F_1,i}, \ldots, state_{F_m,i})$ on $P\texttt{-out}_i$, where $state_{P,i}$ is all of $\mathcal{P}$'s previous states. After this $\mathcal{P}$ goes into a special corrupted behavior, where whenever it receives (L, m) on $P\texttt{-in}_i$ and L is one of it out-tapes, it sends m on L and whenever it receives m on some tape $F_j\texttt{-out}_i$, it sends $(F_j\texttt{-out}_i, m)$ on $P\texttt{-out}_i$. I.e. the ITM connected to $P\texttt{-in}_i$ and $P\texttt{-out}_i$ is now 'connected' to all the tapes $F\texttt{-in}_i$ and $F\texttt{-out}_i$.

For an IF to have the same communication pattern as a protocol it will also output a value $state_{F,i}$ when $\texttt{corrupt}$ is input on $F\texttt{-in}_i$. It will let the the entity connected to $F\texttt{-infl}$ decide this value. Specifically, when $\mathcal{F}$ receives $\texttt{corrupt}$ on some $F\texttt{-in}_i$, then it sends $(\texttt{corrupt}, i)$ on $F\texttt{-leak}$ and waits to get back a value $state_{F,i}$ on $F\texttt{-infl}$. Then it sends $state_{F,i}$ on $F\texttt{-out}_i$.

2.2.1 The real-life model

Consider now a protocol π with only the IF $\mathcal{F}_{\text{at}}$ and consider an environment $\mathcal{Z}$ for π. In the execution $\text{EXEC}_{\pi \diamond \mathcal{Z}}$ the environment $\mathcal{Z}$ can give the protocol inputs and see outputs from the protocol. When $\mathcal{Z}$ activates a party $\mathcal{P}_i$ on $P\texttt{-in}_i$, then $\mathcal{P}_i$ might send some message on $\mathcal{F}_{\text{at}}$. After each message sent $\mathcal{P}_i$ gets back the activation and at some point sends a message on $P\texttt{-out}_i$. For each message m sent to some $\mathcal{P}_j$, the IF $\mathcal{F}_{\text{at}}$ wrote (i, j, m) on $\texttt{at-leak}$. This means that $\mathcal{Z}$ has access to the messages sent by $\mathcal{P}_i$. It can then use $\texttt{at-infl}$ to deliver these messages. If it does, then the receiver $\mathcal{P}_j$ is activated by $\mathcal{F}_{\text{at}}$ and the execution continues. At some point $\mathcal{Z}$ might also send $\texttt{corrupt}$ to some party $\mathcal{P}_i$ in response to which it sees the internal state of $\mathcal{P}_i$ and can now send messages on behalf of $\mathcal{P}_i$ through $P\texttt{-in}_i$. Except for some minor 'syntactical' differences,[2] this is exactly the real-life execution in [Can]. I.e., in the notation of [Can], $\text{REAL}_{\pi,\mathcal{Z}} = \text{EXEC}_{\pi \diamond \mathcal{Z}}$.

2.3 Simulating a protocol given an ideal functionality

Assume now that we want to express that a protocol π is as secure as some IF $\mathcal{F}$. We do this by requiring that no environments can distinguish whether it is interacting with π or $\mathcal{F}$. Specifically, for any IF with name F we call π a protocol for $\mathcal{F}$ if it has the same name (and thereby the same protocol tapes) and it does not use an IF named F (to avoid that both networks have tapes named $F\texttt{-leak}$ and $F\texttt{-infl}$). Consider an environment $\mathcal{Z}$ for π, i.e. such that $\pi \diamond \mathcal{Z}$ is closed. We want to compare $\pi \diamond \mathcal{Z}$ to $\mathcal{F} \diamond \mathcal{Z}$. We therefore introduce another ITM $\mathcal{S} \in (\mathcal{F} \diamond \mathcal{Z})^{\mathsf{C}}$. This

[2] When e.g. $\mathcal{Z}$ corrupts P_i, it will see $\mathcal{F}_{\text{at}}$ output $(\texttt{corrupt}, i)$ on $\texttt{at-leak}$ and must input a value $state_{\text{at},i}$ on $\texttt{at-infl}$. It then gets $state_{\text{at},i}$ back on $\texttt{at-out}_i$ together with the state of $\mathcal{P}_i$. This weird loop-back does not occur in [Can], but changes noting. The reason for IFs taking $state_{F,i}$ from $F\texttt{-infl}$ will be clear later, when a simulator is connected to $F\texttt{-infl}$.

gives us a closed network $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$. In the following we call such an $\mathcal{S}$ a simulator from $\mathcal{F}$ to π if it is in addition PPT; We write $\mathcal{S} \in [\pi \triangleright \mathcal{F}]$. We then compare the closed networks $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$ and $\pi \diamond \mathcal{Z}$ (see Fig. 1 for an example of a simulator and a network $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$).

Assume that π has IFs with names $F_1, \ldots, F_m$. The job of $\mathcal{S}$ in $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$ is to simulate to $\mathcal{F}$ (over F_j-`leak` and F_j-`infl`) the view $\mathcal{Z}$ would have of π on the sequence of inputs $\mathcal{Z}$ is giving to $\mathcal{F}$. In doing this $\mathcal{S}$ only gets the allowed leakage (over F-`leak`). Furthermore, it is $\mathcal{F}$ which gives back outputs to $\mathcal{Z}$, over F-`out`$_i$, and $\mathcal{S}$ can only control these outputs through F-`infl`. I.e. it only has the allowed influence. Finally, if $\mathcal{Z}$ inputs `corrupt` on F-`in`$_i$ in $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$, then $\mathcal{F}$ outputs (`corrupt`, i) to $\mathcal{S}$ on F-`leak`. Then $\mathcal{S}$ gets to return a value $state_{F,i}$ on F-`infl` and $\mathcal{F}$ outputs $state_{F,i}$ on F-`out`$_i$. In $\pi \diamond \mathcal{Z}$, when $\mathcal{Z}$ inputs `corrupt` on F-`in`$_i$ the value $state_{F,i}$ returned is the internal state of $\mathcal{P}_i$. So, in $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$, the simulator $\mathcal{S}$ must simulate the internal state of corrupted parties. Again, except for some 'syntactical' differences, this is exactly the ideal process in [Can]. I.e., in the notation of [Can] $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = \text{EXEC}_{(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}}$.

2.3.1 Conditions on environments

In [Can] the notion of a class of environments is used. When defining UCZK proof of membership, we need a similar notion, where we prove security in a set of environments with a given property. Opposed to [Can] we need to consider properties holding except with negligible probability. We therefore take some care in formalizing this and reproving the composition theorem.

We let a conditioned IF $\mathcal{F} = (\mathcal{F}', P)$ consist of an IF $\mathcal{F}'$ and a communication property P which only depends on the tapes F-`leak` and F-`infl`.[3] The intuition is that $\mathcal{F}$ must only be influenced over F-`infl` by values such that the communication over F-`infl` and F-`leak` is in $P(\mathcal{F})$. For a protocol π with IFs $\mathcal{G}_1, \ldots, \mathcal{G}_m$ we let $P(\pi) = \wedge_{i=1}^{m} P(\mathcal{G}_i)$. We then require that an environment for π has the property $P(\pi)$ and that a simulator $\mathcal{S}$ (for $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$) has the property $P(\mathcal{F})$. Since $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$ will be compared to $\pi \diamond \mathcal{Z}$, $\mathcal{S}$ will be running with an environment $\mathcal{Z}$ with the property $P(\pi)$. It is therefore sufficient that $\mathcal{S}$ has the property $P(\mathcal{F})$ as long as the property $P(\pi)$ holds.

Definition 1 *Let π be a protocol for an IF $\mathcal{F}$. We say that π realizes $\mathcal{F}$, written $\mathcal{F} \overset{\text{c}}{\triangleright} \pi$, if there exists $\mathcal{S} \in [\pi \triangleright \mathcal{F}]$ such that $\mathcal{S} \overset{\text{c}}{\models} (P(\pi) \rightarrow P(\mathcal{F}))$ and $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z} \overset{\text{c}}{\approx} \pi \diamond \mathcal{Z}$, for all $\mathcal{Z} \in \pi^{\mathsf{C}}$ where $\mathcal{Z} \overset{\text{c}}{\models} P(\pi)$.*

The network $\mathcal{F} \diamond \mathcal{S}$ can be consider an IF with an interface $\mathcal{S}$ for translating leakage from $\mathcal{F}$ to leakage of the form in π and translating valid influence on π into valid influence on $\mathcal{F}$ (see Fig. 1, top, left). The environment $\mathcal{Z}$ then serves as an "interactive distinguisher" between the two networks $\mathcal{F} \diamond \mathcal{S}$ and π, and is restricted to using influence valid according to $P(\pi)$.

2.4 Protocol composition

Let π be a protocol with parties $\mathcal{P}_1^\pi, \ldots, \mathcal{P}_n^\pi$ using an IF $\mathcal{G}$ (maybe among others). Let ρ be a protocol for $\mathcal{G}$ with parties $\mathcal{P}_1^\rho, \ldots, \mathcal{P}_n^\rho$ (ρ maybe itself using some IFs). Assume that all involved IFs have distinct names. We describe a composed protocol $\pi^{\mathcal{G} \mapsto \rho}$. We can write π as $\pi' \diamond \mathcal{G}$. Since ρ is a protocol for $\mathcal{G}$ we can define a network $\pi' \diamond \rho$, where the parties $\mathcal{P}_i^\pi$ now connect to $\mathcal{P}_i^\rho$ instead of $\mathcal{G}$ (via G-`in`$_i$ and G-`out`$_i$). This is however not formally a protocol, as each 'party' is now of the form $\mathcal{P}_i^\pi \diamond \mathcal{P}_i^\rho$ and thus not an ITM. We can however consider $\mathcal{P}_i^\pi \diamond \mathcal{P}_i^\rho$ as a single ITM and let $\mathcal{P}_i = \langle \mathcal{P}_i^\pi \diamond \mathcal{P}_i^\rho \rangle$. By the communication pattern enforced on parties, $\mathcal{P}_i$ can be verified to have the behavior required of a party. We let $\pi^{\mathcal{G} \mapsto \rho}$ denote $\pi' \diamond \rho$ with each $\mathcal{P}_i^\pi \diamond \mathcal{P}_i^\rho$ replaced by $\mathcal{P}_i$.

[3] If not mentioned explicitly, then $P(\mathcal{F}) = \top$, where $\top(C) = 1$ for all C.

2.4.1 The composition theorem

In [Can] a very general composition theorem is proved, considering a.o.t. replacing any polynomial number of copies of the same IF and protocol emulation. For simplicity we consider only a special case. We reprove the theorem to deal with conditioned IFs.

Theorem 2 *Let $\mathcal{F}, \mathcal{G}$ be IFs and let π be a protocol using $\mathcal{G}$. If $\pi \overset{c}{\triangleright} \mathcal{F}$ and $\rho \overset{c}{\triangleright} \mathcal{G}$, then $\pi^{\mathcal{G}\mapsto\rho} \overset{c}{\triangleright} \mathcal{F}$.*

Proof: Assume that there exist simulators $\mathcal{S}_\pi \in [\pi \triangleright \mathcal{F}]$ and $\mathcal{S}_\rho \in [\rho \triangleright \mathcal{G}]$ such that

$$\mathcal{S}_\pi \models^{c} P(\pi) \rightarrow P(\mathcal{F}), \quad \mathcal{S}_\rho \models^{c} P(\rho) \rightarrow P(\mathcal{G}), \quad (\mathcal{F} \diamond \mathcal{S}_\pi) \diamond \mathcal{Z}_\pi \overset{c}{\approx} \pi \diamond \mathcal{Z}_\pi, \quad (\mathcal{G} \diamond \mathcal{S}_\rho) \diamond \mathcal{Z}_\rho \overset{c}{\approx} \rho \diamond \mathcal{Z}_\rho \quad (1)$$

for all $\mathcal{Z}_\pi \in \pi^{\mathtt{C}}$ and $\mathcal{Z}_\rho \in \rho^{\mathtt{C}}$ where $\mathcal{Z}_\pi \models^{c} P(\pi)$ and $\mathcal{Z}_\rho \models^{c} P(\rho)$. Let $\pi = \pi' \diamond \mathcal{G}$ and let $P(\pi')$ be the conjunction over the properties of the IFs of π except $\mathcal{G}$. Let $\mathcal{S} = \langle \mathcal{S}_\pi \diamond \mathcal{S}_\rho \rangle$. Observe that $\mathcal{S}_\rho$ closes the tapes G-`infl` and G-`leak` on $\mathcal{S}_\pi$ and opens tapes for each IF used by ρ. Therefore $\mathcal{S} \in [\pi^{\mathcal{G}\mapsto\rho} \triangleright \mathcal{F}]$. Furthermore, from (1) it follows that $\mathcal{S}_\pi \diamond \mathcal{S}_\rho \models^{c} (P(\rho) \rightarrow P(\mathcal{G})) \wedge (P(\pi) \rightarrow P(\mathcal{F}))$. Since $P(\pi) = P(\pi') \wedge P(\mathcal{G})$ it thus follows that $\mathcal{S}_\pi \diamond \mathcal{S}_\rho \models^{c} (P(\rho) \wedge P(\pi')) \rightarrow P(\mathcal{F}))$, or $\mathcal{S}_\pi \diamond \mathcal{S}_\rho \models^{c} P(\pi^{\mathcal{G}\mapsto\rho}) \rightarrow P(\mathcal{F})$. Since $P(\pi^{\mathcal{G}\mapsto\rho}) \rightarrow P(\mathcal{F})$ only depends on open tapes of $\mathcal{S}_\pi \diamond \mathcal{S}_\rho$ it follows that $\mathcal{S} = \langle \mathcal{S}_\pi \diamond \mathcal{S}_\rho \rangle \models^{c} P(\pi^{\mathcal{G}\mapsto\rho}) \rightarrow P(\mathcal{F})$. So, what remains is to prove that $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z} \overset{c}{\approx} \pi^{\mathcal{G}\mapsto\rho} \diamond \mathcal{Z}$ for all $\mathcal{Z} \in (\pi^{\mathcal{G}\mapsto\rho})^{\mathtt{C}}$ where $\mathcal{Z} \models^{c} P(\rho) \wedge P(\pi')$, which can be proven as follows:

$$\begin{aligned}
(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z} &\equiv (\mathcal{F} \diamond (\mathcal{S}_\pi \diamond \mathcal{S}_\rho)) \diamond \mathcal{Z} = (\mathcal{F} \diamond \mathcal{S}_\pi) \diamond (\mathcal{S}_\rho \diamond \mathcal{Z}) \equiv (\mathcal{F} \diamond \mathcal{S}_\pi) \diamond \mathcal{Z}_\pi \\
&\overset{c}{\approx} \pi \diamond \mathcal{Z}_\pi \equiv (\pi' \diamond \mathcal{G}) \diamond (\mathcal{S}_\rho \diamond \mathcal{Z}) = (\mathcal{G} \diamond \mathcal{S}_\rho) \diamond (\pi' \diamond \mathcal{Z}) \equiv (\mathcal{G} \diamond \mathcal{S}_\rho) \diamond \mathcal{Z}_\rho \\
&\overset{c}{\approx} \rho \diamond \mathcal{Z}_\rho \equiv \rho \diamond (\pi' \diamond \mathcal{Z}) = (\pi' \diamond \rho) \diamond \mathcal{Z} \equiv \pi^{\mathcal{G}\mapsto\rho} \diamond \mathcal{Z} \ ,
\end{aligned}$$

where $\mathcal{Z}_\pi = \langle \mathcal{S}_\rho \diamond \mathcal{Z} \rangle \in \pi^{\mathtt{C}}$ and $\mathcal{Z}_\rho = \langle \pi' \diamond \mathcal{Z} \rangle \in \rho^{\mathtt{C}}$. Of course we have to verify that $\mathcal{Z}_\pi \models^{c} P(\pi)$ and $\mathcal{Z}_\rho \models^{c} P(\rho)$. From (1) and $\mathcal{Z} \models^{c} P(\rho) \wedge P(\pi')$ we get that $\mathcal{S}_\rho \diamond \mathcal{Z} \models^{c} (P(\rho) \wedge P(\pi')) \wedge (P(\rho) \rightarrow P(\mathcal{G}))$ and thus $\mathcal{S}_\rho \diamond \mathcal{Z} \models^{c} P(\mathcal{G}) \wedge P(\pi')$. Since $P(\pi) = P(\mathcal{G}) \wedge P(\pi')$ only depends on open tapes of $\mathcal{S}_\rho \diamond \mathcal{Z}$ it follows that $\langle \mathcal{S}_\rho \diamond \mathcal{Z} \rangle \models^{c} P(\pi)$. From $\mathcal{Z} \models^{c} P(\rho) \wedge P(\pi')$, it follows that $\mathcal{Z} \models^{c} P(\rho)$ and $\pi' \diamond \mathcal{Z} \models^{c} P(\rho)$. Since $P(\rho)$ only depends on open tapes of $\pi' \diamond \mathcal{Z}$, it then follows that $\mathcal{Z}_\rho \models^{c} P(\rho)$. □

3 UC zero-knowledge proof of membership

We first specify a two-party ZK proof IF $\mathcal{F}_{\text{zk}}$ from a prover P to a verifier V and investigate what is required from a two-party CRS protocol to realize $\mathcal{F}_{\text{zk}}$. We see that a realization of $\mathcal{F}_{\text{zk}}$ is always a proof of knowledge and we see why. This leads us to the first definition of UC proof of membership, and we investigate how to realize this new notion.

3.1 The zero-knowledge ideal functionality

The IF $\mathcal{F}_{\text{zk}}$ [Can01] is parameterized by a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$. We require from the relation that there exists some polynomial p such that $(x,w) \in R$ implies that $|w| \leq p(|x|)$ and such that $(x,w) \in R$ can be checked in time $p(|x|)$. We let $L(R) = \{x \in \{0,1\}^* | \exists w \in \{0,1\}^* ((x,w) \in R)\}$ denote the language of the relation, and for $(x,w) \in R$ we call x the instance and call w the witness.

The IF has the tapes shown in Fig. 1. It ignores all inputs on $\texttt{zk-in}_V$ (except that it returns the activation on $\texttt{zk-out}_V$ as required). If it receives (x,w) on $\texttt{zk-in}_P$, it ignores the input if $(x,w) \notin R$ and otherwise stores x and writes x on `zk-leak`. Then it returns the activation on $\texttt{zk-out}_P$. If it receives a value $(\texttt{deliver}, x)$ on `zk-infl` and at least one value x is stored, then it deletes a copy of x and sends x on $\texttt{zk-out}_V$. From the point where P is corrupted, whenever it receives $(x,w) \in R$ on `zk-infl` it immediately sends x on $\texttt{zk-out}_V$.

3.2 Defining UC zero-knowledge proof of knowledge

We consider realizations of $\mathcal{F}_{\text{zk}}$ using two-party common reference string (CRS) protocols. This is a two-party protocol with a CRS IF $\mathcal{F}_{\text{crs}}$ and the IF $\mathcal{F}_{\text{at}}$ described in Section 2. See Fig. 1.

The IF $\mathcal{F}_{\text{crs}}$ is parameterized by a PPT function $D : \{0,1\}^* \to \{0,1\}^*$. The first time it is activated on any tape ($\texttt{crs-in}_P$, $\texttt{crs-in}_V$ or $\texttt{crs-infl}$) it computes $crs = D(r)$ for uniformly random $r \in \{0,1\}^k$, writes crs on all three out-tapes, and returns the activation on the corresponding out-tape ($\texttt{crs-out}_P$, $\texttt{crs-out}_V$ respectively $\texttt{crs-leak}$). From then on it ignores all inputs.

The two parties have the following behavior. The first time $X \in \{P, V\}$ is activated, if a value crs is written on $\texttt{crs-out}_X$, then X copies it to its work-tape. Otherwise, it activates on $\texttt{crs-in}_X$, waits to get back crs on $\texttt{crs-out}_X$, and the writes crs on its work-tape. From now on we describe two-party CRS protocols from the point where crs is written on the work tape of P and V.

Definition 3 *Let π be a two-party CRS protocol for $\mathcal{F}_{\text{zk}}$. We say that π is a two-party UCZK proof of knowledge for R if $\pi \overset{\text{c}}{\triangleright} \mathcal{F}_{\text{zk}}$.*

3.3 Realizing UC zero-knowledge proof of knowledge

In this section we justify why we called a CRS realization of $\mathcal{F}_{\text{zk}}$ a proof of *knowledge*, by proving that it is indeed always so. This is not a new observation, see e.g. [Can], but to appreciate our upcoming definition of UCZK proof of membership it is important to get a feeling why it is the case. At the same time we get to do some work, which we can reuse when we discuss how to realize UCZK proofs of membership.

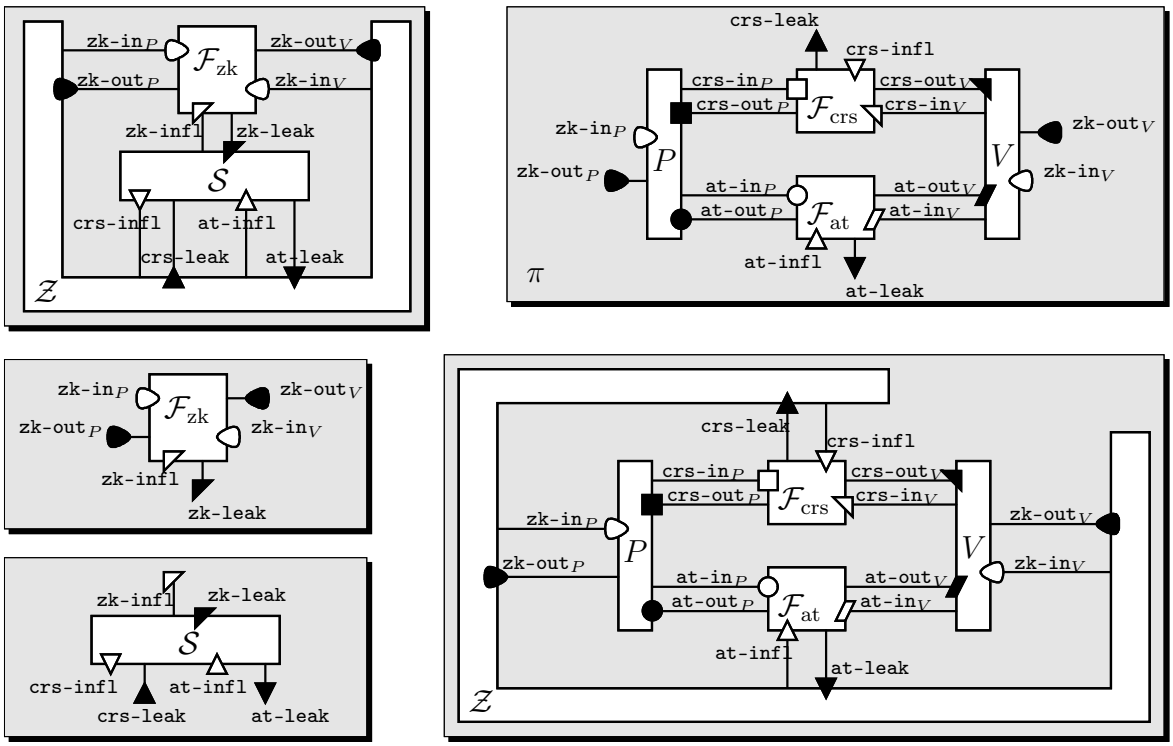

Figure 1: Left, middle: the ZK IF $\mathcal{F}_{\text{zk}}$. Right, top: a two-party protocol π with the authenticated transfer IF $\mathcal{F}_{\text{at}}$ and the CRS IF $\mathcal{F}_{\text{crs}}$. Right, bottom: an environment $\mathcal{Z} \in \pi^{\mathsf{C}}$ and $\pi \diamond \mathcal{Z}$. Left, bottom: a simulator $\mathcal{S} \in [\pi \triangleright \mathcal{F}_{\text{zk}}]$. Left, top: the same environment, now experimenting with $\mathcal{F}_{\text{zk}} \diamond \mathcal{S}$.

3.3.1 Two-party common reference string protocols

We assume that the protocol is specified by an PPT computable function $D : \{0,1\}^* \to \{0,1\}^*$ and two ITMs Pro and Ver. For $crs \leftarrow D$, security parameter $k \in \mathbf{N}$ and $(x,w) \in R$, an interaction between $\mathrm{Pro}(k, crs, x, w; r_{\mathrm{Pro}})$ and $\mathrm{Ver}(k, crs, x; r_{\mathrm{Ver}})$ proceeds as follows: First copies of Pro and Ver are initialized with fresh randomness r_{Pro} respectively r_{Ver} and inputs (k, crs, x, w) respectively (k, crs, x). Then messages $m_1, m_2, \ldots$ are exchanges in rounds, and at some point $\mathrm{Ver}(k, crs, x)$ terminates with output $b \in \{0,1\}$, where $b = 1$ indicates acceptance.

From $(D, \mathrm{Pro}, \mathrm{Ver})$ we derive a two-party CRS protocol π as follows. It uses the IF $\mathcal{F}_{\mathrm{crs}}^{D}$. The prover P starts by initializing a counter $pid = 0$, and whenever it gets an input $(x,w) \in R$ on $\texttt{zk-in}_P$ it lets $pid = pid + 1$, starts a new copy $\mathrm{Pro}^{pid}(k, crs, x, w)$ of Pro and sends (pid, x) to V over $\mathcal{F}_{\mathrm{at}}$. Then V initializes a new copy $\mathrm{Ver}^{pid}(k, crs, x)$, and P and V lets these two copies interact by exchanging messages of the form (pid, m_i) over $\mathcal{F}_{\mathrm{at}}$. If some copy $\mathrm{Ver}^{pid}(k, crs, x)$ terminates with output 1, then V outputs x on $\texttt{zk-out}_V$. To discuss the security of this protocol, it is convenient to use some security definitions for the triple $(D, \mathrm{Pro}, \mathrm{Ver})$.

3.3.2 Straight-line zero-knowledge

We first define the notion of corruptible straight-line ZK. As for straight-line ZK [FS89, SD98] it is defined by comparing two games, the cheating verifier game and the simulation, but now the simulator must also be able to simulate the internal state of the ZK proof when given the witness.

The cheating verifier game. Let Ver^* be any ITM, and consider the game $[\mathrm{Pro}, \mathrm{Ver}^*](k)$, which proceeds as follows: Generate $crs \leftarrow D(r_{\mathrm{crs}} \in_R \{0,1\}^k)$ and then let Ver^* interact with any number of $\mathrm{Pro}(k, crs, x, w; r_{\mathrm{Pro}})$ for $(x,w) \in R$ of its choice. The cheating verifier Ver^* gets to schedule the interaction with the copies of Pro, and at some point Ver^* chooses to end the game. In response to this it is given the randomness r_{Pro} used by all the copies $\mathrm{Pro}(k, crs, x, w; r_{\mathrm{Pro}})$ and outputs a guess $b \in \{0,1\}$.

The simulation. Now let SimPr be any ITM. The game $[\mathrm{SimPr}, \mathrm{Ver}^*](k)$ proceeds exactly as $[\mathrm{Pro}, \mathrm{Ver}^*](k)$, except that SimPr is given $(k, crs, x, r_{\mathrm{crs}})$ as input. When Ver^* chooses to end the game, then for each $\mathrm{SimPr}(k, crs, x, r_{\mathrm{crs}})$, the witness w is input to $\mathrm{SimPr}(k, crs, x, r_{\mathrm{crs}})$ to generate an output r'_{Pro}, which is given to Ver^*. Then Ver^* outputs a guess $b \in \{0,1\}$.

Definition 4 *We say that $(D, \mathrm{Pro}, \mathrm{Ver})$ is corruptible straight-line ZK if there exists a PPT* SimPr *such that $|\Pr[[\mathrm{Pro}, \mathrm{Ver}^*](k,z) = 1] - \Pr[[\mathrm{SimPr}, \mathrm{Ver}^*](k,z) = 1]|$ is negligible in k for all PPT Ver^* and all $z \in \{0,1\}^*$.*

3.3.3 Weak simulation knowledge soundness

We also define a notion of weak simulation knowledge soundness. For this purpose, let A be any ITM, and assume that two ITMs SimPr and Ext are given. Consider the following game $[\mathrm{SimPr}, \mathrm{Ver}, \mathrm{Ext}, A](k,z)$, running in two stages. In the verifier stage A first gets (k,z) and then gets to interact with copies of SimPr as in $[\mathrm{SimPr}, \mathrm{Ver}^*](k)$, but is not required to specify $(x,w) \in R$ to start a new copy $\mathrm{SimPr}(k, crs, x, r_{\mathrm{crs}})$; It can specify any $x \in \{0,1\}^k$. To end the verifier stage correctly it must however for each copy $\mathrm{SimPr}(k, crs, x, r_{\mathrm{crs}})$ supply w such that $(x,w) \in R$. Then w is given to $\mathrm{SimPr}(k, crs, x, r_{\mathrm{crs}})$ and A is given the reply r'_{Pro}. If A ends the verifier stage correctly, then the game enters the prover stage. Here A interacts with copies of $\mathrm{Ver}(k, x, crs)$ for x of its own choosing. If it makes a copy $\mathrm{Ver}(k, x, crs)$ accept, then Ext is run on k, r_{crs} and the interaction that $\mathrm{Ver}(k, x, crs)$ had with A, and Ext outputs a value w. If it ever happens that $(x,w) \notin R$, then the game outputs 1; Otherwise it outputs 0.

Definition 5 *We say that $(D, \mathrm{Pro}, \mathrm{Ver})$ is a weak[4] simulation knowledge sound, corrupt-*

[4] The weakening over the definitions in [Sah99, GMY03] is that A only sees simulated proofs for *true* statements.

ible straight-line ZK protocol if it is a corruptible straight-line ZK protocol and it holds for the simulator SimPr *demonstrating this that there exists a PPT ITM* Ext *such that* $\Pr[[\text{SimPr}, \text{Ver}, \text{Ext}, A](k, z) = 1]$ *is negligible in* k *for all PPT* A *and all* $z \in \{0,1\}^*$.

3.3.4 Security of the two-party common reference string protocol

Now assume that we have a weak simulation knowledge sound, corruptible straight-line ZK protocol $(D, \text{Pro}, \text{Ver})$ with the corresponding SimPr and Ext. We construct a simulator $\mathcal{S}$ for the derived protocol π. The reader is encouraged to inspect Fig. 1 for the structure of the simulation $(\mathcal{F}_{\text{zk}} \diamond \mathcal{S}) \diamond \mathcal{Z}$ and the protocol $\pi \diamond \mathcal{Z}$ during the description.

Initialization. The first time $\mathcal{Z}$ activates on $\texttt{zk-in}_P$, $\texttt{zk-in}_V$ or `crs-infl`, the simulator $\mathcal{S}$ generates $crs = D(r_{\text{crs}} \in_R \{0,1\}^k)$, writes crs on `crs-leak`, stores (crs, r_{crs}) and sets $pid = 0$; This gives $\mathcal{Z}$ the exact same view as in $\pi \diamond \mathcal{Z}$.

Two honest parties. As long as no party is corrupted, whenever $\mathcal{Z}$ sends $(x, w) \in R$ on $\texttt{zk-in}_P$ to $\mathcal{F}_{\text{zk}}$ it outputs x to $\mathcal{S}$ on `zk-leak`. Then $\mathcal{S}$ lets $pid \leftarrow pid + 1$, initializes $\text{SimPr}^{pid}(k, crs, x, r_{\text{crs}})$ and shows $(P, V, (pid, x))$ on `at-leak`. If $\mathcal{Z}$ at some point inputs $(\texttt{deliver}, P, V, (pid, x))$ on `at-infl` then $\mathcal{S}$ initializes $\text{Ver}^{pid}(k, crs, x)$ and runs $\text{SimPr}^{pid}(k, crs, x, r_{\text{crs}})$ and $\text{Ver}^{pid}(k, crs, x)$ together, and shows the exchanged messages m_i on `at-leak` (letting $\mathcal{Z}$ schedule the execution via `at-infl`). If a copy $\text{Ver}(k, crs, x)$ accepts, then $\mathcal{S}$ sends $(\texttt{deliver}, x)$ on `zk-infl`. If a copy $\text{Ver}(k, crs, x)$ rejects, then $\mathcal{S}$ *gives up* the simulation.

Corrupted prover. If at any point P is corrupted, i.e. $\mathcal{Z}$ inputs `corrupt` on $\texttt{zk-in}_P$, then $\mathcal{S}$ receives $(\texttt{corrupt}, P)$ on `zk-leak` along with $(x, w) \in R$ for every $\text{SimPr}^{pid}(k, crs, x, r_{\text{crs}})$. Then $\mathcal{S}$ inputs each w to the corresponding $\text{SimPr}^{pid}(k, crs, x, r_{\text{crs}})$ to get r'_{Pro} and concatenates these to a string r'_P. Then it sends r'_P on `zk-infl`. In response to this, $\mathcal{F}_{\text{zk}}$ outputs r'_P to $\mathcal{Z}$ on $\texttt{zk-out}_P$. Note that in $\pi \diamond \mathcal{Z}$, if $\mathcal{Z}$ inputs `corrupt` on $\texttt{zk-in}_P$ it would instead received the randomness r_P used by P in the proofs $\text{Pro}^{pid}(k, crs, x, w, r_{\text{Pro}})$.

From this point on, if $\mathcal{Z}$ inputs a value m on $\texttt{zk-in}_P$, then by construction of $\mathcal{F}_{\text{zk}}$ the value (P, m) is output on `zk-leak` to $\mathcal{S}$. Recall that $\mathcal{Z}$ should 'think' that it runs in $\pi \diamond \mathcal{Z}$. So, if $\mathcal{S}$ receives (P, m) with $m = (\texttt{at-in}_P, m')$, then it outputs (P, V, m) on `at-leak`, and if $\mathcal{Z}$ inputs $(\texttt{deliver}, P, V, m')$ on `at-infl`, then P delivers m' in the simulated protocol π. So, now it is $\mathcal{Z}$ which interacts with the copies $\text{Ver}^{pid}(k, crs, x)$, and it might create new copies by sending (pid, x) on behalf of the corrupted prover. If a copy $\text{Ver}^{pid}(k, crs, x)$ accepts, then $\mathcal{S}$ outputs (x, w) on `zk-infl`, where w is the results of applying Ext to the accepting interaction. Because P is corrupted this makes $\mathcal{F}_{\text{zk}}$ output x on $\texttt{zk-out}_V$, exactly as in $\pi \diamond \mathcal{Z}$, *unless* $(x, w) \notin R$, in which case $\mathcal{S}$ *gives up* the simulation.

Corrupted verifier. If at any point V is corrupted (before or after P is corrupted), i.e. $\mathcal{S}$ receives $(\texttt{corrupt}, V)$ on `zk-leak`, then it lets r_V be the concatenation of the r_{Ver} used by the copies $\text{Ver}^{pid}(k, crs, x)$ and sends r_V on `zk-infl`. Thus r_V ends up at $\mathcal{Z}$, exactly as in $\pi \diamond \mathcal{Z}$. From this point on it is then $\mathcal{Z}$ who runs V. If P is still honest it therefore gets to interact with copies of SimPr as it desires.

Analysis. If $\mathcal{S}$ never gives up the simulation, then $\mathcal{Z}$ in $(\mathcal{F}_{\text{zk}} \diamond \mathcal{S}) \diamond \mathcal{Z}$ is essentially participating in the game $[\text{SimPr}, \mathcal{Z}](k, z)$, and that $\mathcal{Z}$ in $\pi \diamond \mathcal{Z}$ is in the same way essentially participating in the game $[\text{Pro}, \mathcal{Z}](k, z)$. So, it would follow directly from the straight-line ZK that $(\mathcal{F}_{\text{zk}} \diamond \mathcal{S}) \diamond \mathcal{Z} \overset{\text{c}}{\approx} \pi \diamond \mathcal{Z}$ if $\mathcal{S}$ gives up with negligible probability. If $\mathcal{S}$ gives up the simulation then the honest verifier either 1) rejected a simulated conversation with SimPr for $x \in L(R)$, or 2) accepted a conversation with $\mathcal{Z}$, and Ext output w such that $(x, w) \notin R$. The first case clearly happens with non-negligible probability if the proof system is correct. The second case happens with negligible probability by the simulation knowledge soundness. It follows that $\pi \overset{\text{c}}{\triangleright} \mathcal{F}_{\text{zk}}$.

Theorem 6 *If* $(D, \text{Pro}, \text{Ver})$ *is a weak simulation knowledge sound, corruptible straight-line ZK*

protocol, then the derived two-party CRS protocol is a UCZK proof of knowledge.

3.3.5 The other direction.

We argued that corruptible straight-line ZK and simulation knowledge soundness are sufficient conditions for protocols of the particular form that we considered to be UCZK proofs of knowledge. It should however be clear now that if $(\mathcal{F}_{\text{zk}} \diamond \mathcal{S}) \diamond \mathcal{Z} \overset{\text{c}}{\approx} \pi \diamond \mathcal{Z}$, for any two-party CRS protocol π and any simulator $\mathcal{S}$, then $\mathcal{S}$ can simulate crs on `crs-leak` to $\mathcal{Z}$ and then simulate honest proofs under this crs for $(x, w) \in R$ given just x (as $\mathcal{F}_{\text{zk}}$ only leaks x to $\mathcal{S}$); that it can patch the internal state of simulated proof to be consistent with w (as it must simulate the internal state of P on $\texttt{zk-out}_P$ when P is corrupted); and that it can compute a witness w for all accepted proofs given by $\mathcal{Z}$ under crs even after having showed simulated proof to $\mathcal{Z}$ (as it has to output $(x, w) \in R$ on `zk-infl` to make $\mathcal{F}_{\text{zk}}$ output x on $\texttt{zk-out}_V$). And since $\mathcal{S}$ do not have the opportunity to rewind $\mathcal{Z}$ in $(\mathcal{F}_{\text{zk}} \diamond \mathcal{S}) \diamond \mathcal{Z}$, the simulation and extraction must be straight-line. It follows that π has some form of corruptible straight-line ZK and simulation knowledge soundness.

3.4 Defining UC zero-knowledge proof of membership

As demonstrated above the problem with $\mathcal{F}_{\text{zk}}$ (w.r.t. not wanting to capture a proof of *knowledge*) is that when P is corrupted, the simulator has to input $(x, w) \in R$ to $\mathcal{F}_{\text{zk}}$ whenever a proof from $\mathcal{Z}$ for some x is accepted. Our basic approach to modeling a ZK proof of membership will therefore be to specify that when P is corrupted, then $\mathcal{F}_{\text{zk}}$ only expects an input x on `zk-infl` and then outputs x to V. To prevent that the environment influences $\mathcal{F}_{\text{zk}}$ to output $x \notin L(R)$ to V we then simply add to $\mathcal{F}_{\text{zk}}$ the influence condition that whenever x is input on `zk-infl`, then $x \in L(R)$. Formally, let $\mathcal{F}'_{\text{zkm}}$ be the IF which works exactly as $\mathcal{F}_{\text{zk}}$, except that if P is corrupted, then whenever $\mathcal{F}'_{\text{zkm}}$ receives $(\texttt{deliver}, x)$ on `zk-infl`, it outputs x on $\texttt{zk-out}_V$. Let P_{zkm} be the communication property which outputs 0 iff there is an entry $(\texttt{zk-infl}, (\texttt{deliver}, x))$ in the communication sequence for which $x \notin L(R)$. Let $\mathcal{F}_{\text{zkm}} = (\mathcal{F}'_{\text{zkm}}, P_{\text{zkm}})$.

Definition 7 *A two-party CRS protocol π is a UCZK proof of membership for R if $\pi \overset{\text{c}}{\triangleright} \mathcal{F}_{\text{zkm}}$.*

Note that we cannot let $\mathcal{F}_{\text{zkm}}$ itself check whether $x \in L(R)$, as this cannot necessarily be done in PPT. Using such a functionality in a protocol π would give sever problems as $\mathcal{Z}$ in $\pi \diamond \mathcal{Z}$ would have access to `zk-infl` and `zk-leak` and therefore essentially an oracle for the language $L(R)$.

3.5 Using UC zero-knowledge proof of membership

We discuss how to use a UCZK proof of membership. Let ρ be a UCZK proof of membership and let $\pi = \gamma \diamond \rho$ be a protocol using ρ. To prove $\pi \overset{\text{c}}{\triangleright} \mathcal{F}$, all we have to prove is $\gamma \diamond \mathcal{F}_{\text{zkm}} \overset{\text{c}}{\triangleright} \mathcal{F}$. Assuming for simplicity that γ uses no other conditioned IF and that $\mathcal{F}$ is not conditioned, this comes down to proving that there exists a simulator $\mathcal{S}$ such that $\mathcal{S} \overset{\text{c}}{\models} P_{\text{zkm}} \rightarrow \top$ and $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z} \overset{\text{c}}{\approx} \pi \diamond \mathcal{Z}$ for all $\mathcal{Z} \in (\gamma \diamond \mathcal{F}_{\text{zkm}})^{\mathsf{C}}$ for which $\mathcal{Z} \overset{\text{c}}{\models} P_{\text{zkm}}$. Since $P_{\text{zkm}} \rightarrow \top$ is a tautology, what is left is proving $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z} \overset{\text{c}}{\approx} \pi \diamond \mathcal{Z}$. Since $\mathcal{Z} \overset{\text{c}}{\models} P_{\text{zkm}}$ it follows that in the simulation $(\mathcal{F} \diamond \mathcal{S}) \diamond \mathcal{Z}$, $\mathcal{Z}$ will not send $x \notin L(R)$ over `zk-infl` (except with negligible probability). So, it is sufficient to prove that $\mathcal{S}$ can simulate runs of $\gamma \diamond \mathcal{F}_{\text{zkm}}$ where only true instances are accepted.

3.6 Realizing UC zero-knowledge proof of membership

We now discuss how to construct a UCZK proof of membership. We use the protocol π derived from $(D, \text{Pro}, \text{Ver})$ in Section 3.3.1. Recall that in the proof that π was a UCZK proof of knowledge, the only place where Ext was used was when P was corrupted and $\mathcal{Z}$ gave an acceptable proof for some x (in which case Ext was used to compute the witness w to output

on `zk-infl` as (x, w)). So, if we want to prove $(\mathcal{F}'_{\text{zkm}} \diamond \mathcal{S}') \diamond \mathcal{Z} \overset{c}{\approx} \pi \diamond \mathcal{Z}$, we can do the simulation without the extractor. In particular, let $\mathcal{S}'$ work as the simulator in Section 3.3.4, except that whenever $\mathcal{Z}$ gives an acceptable proof for some x, the simulator $\mathcal{S}'$ simply sends $(\texttt{deliver}, x)$ on `zk-infl`. As P is corrupted, this always makes $\mathcal{F}'_{\text{zkm}}$ output x on $\texttt{zk-out}_V$, as desired. In particular, we now only need that $(D, \text{Pro}, \text{Ver})$ is corruptible straight-line ZK to prove $(\mathcal{F}'_{\text{zkm}} \diamond \mathcal{S}') \diamond \mathcal{Z} \overset{c}{\approx} \pi \diamond \mathcal{Z}$. This might seem puzzling, but the clue is that to prove $\pi \overset{c}{\triangleright} \mathcal{F}_{\text{zkm}}$, we also have to prove that $\mathcal{S}' \models^{c} P(\pi) \rightarrow P(\mathcal{F}_{\text{zkm}})$, (or equivalently that $\mathcal{S}' \models^{c} P_{\text{zkm}}$, as $P(\pi) = \top \wedge \top$), and to prove this we need a notion of soundness.

3.6.1 Weak simulation membership soundness

To see what it takes to prove $\mathcal{S}' \models^{c} P_{\text{zkm}}$ it is convenient to have a notion of weak simulation membership soundness. This is defined via a game $[\text{SimPr}, \text{Ver}, A](k, z)$, which proceeds like the game $[\text{SimPr}, \text{Ver}, \text{Ext}, A](k, z)$ for weak simulation knowledge soundness. But now no extractor is given. Instead A wins the game if in the prover stage it makes a copy of the verifier accept $x \notin L(R)$.

Definition 8 *We say that* $(D, \text{Pro}, \text{Ver})$ *is a weak simulation membership sound, corruptible straight-line ZK protocol if it is a corruptible straight-line ZK protocol and it holds for the simulator* SimPr *demonstrating this that* $\Pr[[\text{SimPr}, \text{Ver}, A](k, z) = 1]$ *is negligible in* k *for all PPT* A *and all* $z \in \{0,1\}^*$.

3.6.2 Security of the two-party common reference string protocol

We return to the analysis of the derived protocol π. What remains is to prove that $\mathcal{S}' \models^{c} P_{\text{zkm}}$. So, we consider any $\mathcal{Z} \in \mathcal{S}'^{\mathsf{C}}$ and prove that when $\mathcal{S}'$ outputs $(\texttt{deliver}, x)$ on $\texttt{zk-infl}_F$ in $\text{EXEC}_{\mathcal{S}' \diamond \mathcal{Z}}$, then $x \in L(R)$, except with negligible probability. In $\mathcal{S}' \diamond \mathcal{Z}$, the 'refuter' $\mathcal{Z}$ connects to all the open tapes of $\mathcal{S}'$; In particular, $\mathcal{Z}$ has direct access to `zk-infl` and `zk-leak` on $\mathcal{S}'$ and does not go through $\mathcal{F}_{\text{zkm}}$ as in $(\mathcal{F}_{\text{zkm}} \diamond \mathcal{S}') \diamond \mathcal{Z}$ (see Fig. 1). This means that $\mathcal{Z}$ has the following powers when 'refuting' $\mathcal{S}' \models^{c} P_{\text{zkm}}$: First it can send a number of $x \in \{0,1\}^*$ to $\mathcal{S}'$ on `zk-leak`, which makes $\mathcal{S}'$ ('thinking' that x arrived from $\mathcal{F}_{\text{zkm}}$) show $\mathcal{Z}$ a simulated proof for x over `at-leak` and `at-infl`. At some point $\mathcal{Z}$ can then input $(\texttt{corrupt}, P)$ to $\mathcal{S}'$ on `zk-leak` along with the witnesses w for all previously inputs x.[5] Then $\mathcal{Z}$ can act as the corrupted prover in a number of proofs to the copies of Ver run by $\mathcal{S}'$. If an acceptable proof is given for some x, then $\mathcal{S}'$ by construction outputs $(\texttt{deliver}, x)$ on `zk-infl`. If $x \notin L(R)$, then $\mathcal{Z}$ refuted $\mathcal{S}' \models^{c} P_{\text{zkm}}$. Except for some 'syntactical' differences, this is exactly the weak simulation membership soundness game $[\text{SimPr}, \text{Ver}, \mathcal{Z}](k, z)$, and it follows that if $(D, \text{Pro}, \text{Ver})$ is weak simulation membership sound, then $\mathcal{S}' \models^{c} P_{\text{zkm}}$.

Theorem 9 *If* $(D, \text{Pro}, \text{Ver})$ *is a weak simulation membership sound, corruptible straight-line ZK protocol, then the derived two-party CRS protocol is a UCZK proof of membership.*

3.6.3 An efficient UC zero-knowledge proof of membership

As an example of a weak simulation membership sound, corruptible straight-line ZK protocol we take the protocol [Dam00] by Damgaard, which is based on Σ-protocols.

A corruptible Σ-protocol is described by three PPT algorithms A, Z, B. The prover sends the first message $a = A(x, w; r_a \in_R \{0,1\}^k)$, the verifier sends a challenge $e \in \{0,1\}^k$, the prover returns $z = Z(x, w, e, r_a)$, and the verifier checks that $B(x, a, e, z) = 1$. We require that A, Z, B have the following properties. Correctness: $B(x, A(x, w; r_a), e, Z(x, w, e, r_a)) = 1$ for

[5] We can make assume that if $\mathcal{S}'$ receives $(\texttt{corrupt}, P)$ on `zk-leak` and does not receive witnesses for all simulated proofs, then it terminates. This does not change its behavior in $(\mathcal{F}'_{\text{zkm}} \diamond \mathcal{S}') \diamond \mathcal{Z}$, where the witnesses are sent by $\mathcal{F}'_{\text{zkm}}$, but forces the 'refuter' $\mathcal{Z}$ to supply the witnesses in $\mathcal{S}' \diamond \mathcal{Z}$.

all inputs with $(x, w) \in R$. Special soundness: if there exist a, e_1, e_2, z_1, z_2 with $e_1 \neq e_2$ and $B(x, a, e_1, z_1) = B(x, a, e_2, z_2) = 1$, then $x \in L(R)$. Special corruptible honest verifier ZK: there exists a PPT ITM hvs which given $x \in L(R)$ and $e \in \{0,1\}^k$ outputs (a, z), and which when later given w such that $(x, w) \in R$ outputs r_a such r_a is uniformly random and $a = A(x, w; r_a)$ and $z = Z(x, w, e, r_a)$.

The protocol. The protocol in [Dam00] works as follows. The CRS crs is a public key for a perfect hiding trapdoor commitment scheme, and the value r_{crs} such that $crs = D(r_{\text{crs}})$ is the trapdoor of the trapdoor commitment scheme. In the protocol, $\text{Pro}(k, crs, x, w)$ computes $a = A(x, w; r_a)$ and sends $c = \text{commit}_{crs}(a; r_c)$. Then $\text{Ver}(k, crs, x)$ returns $e \in_R \{0,1\}^k$, and $\text{Pro}(k, crs, x, w)$ computes $z = Z(x, w, e, r_a)$ and sends (c, a, r_c, z). Then $\text{Ver}(k, crs, x)$ accepts iff $c = \text{commit}_{crs}(a; r_c)$ and $B(x, a, e, z) = 1$.

Corruptible straight-line zero-knowledge. We describe the simulator $\text{SimPr}(k, crs, x, r_{\text{crs}})$. To simulate a proof given x, send a trapdoor commitment c to the verifier and get back $e \in \{0,1\}^k$. Then compute $(a, z) \leftarrow \text{hvs}(x, e)$, use r_{crs} to compute r_c such that $c = \text{commit}_{crs}(a; r_c)$ and send (c, a, r_c, z). When given w, use hvs to compute r_a and output (r_a, r_c).

In [Dam00] only static security was considered and therefore the simulation of r_a was not required, or described. It is however straight-forward to verify that the proof in [Dam00] generalizes to prove that the protocol is corruptible straight-line ZK.

Weak simulation membership soundness. We prove that the protocol is weak simulation membership sound. Assume there exists PPT A which gives an accepting proof for $x \notin L(R)$ in the prover stage of $[\text{SimPr}, \text{Ver}, A]$ with non-negligible probability. Using the rewinding technique from [Dam00] we can construct a PPT algorithm B which shows A one run of the verifier stage and then uses A to compute a, a', r_c, r'_c with $a \neq a'$ and $\text{commit}_{crs}(a; r_c) = \text{commit}_{crs}(a'; r'_c)$ with non-negligible probability. In the verifier stage B uses r_{crs} to compute trapdoor openings of some commitments c, but only once for each c, and r_{crs} is not used on the prover stage.

It follows that if commit is weak simulation sound in the sense that no PPT B can compute a double opening even after seeing a number of trapdoor openings of trapdoor commitments c to values a of B's choice,[6] then the protocol is weak simulation membership sound.

Theorem 10 *The protocol in [Dam00] based on a weak simulation sound trapdoor commitment scheme and a corruptible Σ-protocol is a weak simulation membership sound, straight-line corruptible ZK protocol.*

Any trapdoor commitment scheme can be transformed into a weak simulation sound trapdoor commitment scheme by committing to m as $c = (c_1, c_2) = (\text{commit}_{pk_1}(r), \text{commit}_{pk_2}(m \oplus r))$ for independent keys pk_1 and pk_2 and $r \in_R \{0,1\}^{|m|}$. Furthermore, many languages considered in the literature have very efficient corruptible Σ-protocols, so it follows from Theorems 9 and 10 that there exist very efficient UCZK proofs of membership for many languages. This includes, graph isomorphism, equality of discrete logarithms (in e.g. RSA groups), quadratic residuosity, linear relations between homomorphic encryptions (like [Pai99]), and many others.

4 Conclusion

We gave the first definition of UCZK proof of membership (which is not at the same time a proof of knowledge), showed that the notion is closely related to the notions of straight-line ZK and membership simulation soundness, and sketched an efficient realization of the new notion.

[6] The weakening over the notion of *simulation soundness* [GMY03] is that B is only shown one opening of each c.

References

[Can] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive 2000/067.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, Nevada, 14–17 October 2001. IEEE.

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

[FS89] Uri Feige and Adi Shamir. Zero-knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *Advances in Cryptology - Crypto '89*, pages 526–544, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science Volume 435.

[GMY03] Juan A. Garay, Philip MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In Eli Biham, editor, *Advances in Cryptology - EuroCrypt 2003*, pages 177–194, Berlin, 2003. Springer-Verlag. Lecture Notes in Computer Science Volume 2656.

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

[PSW00] Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Secure reactive systems. Technical Report RZ 3206, IBM Research, Zürich, May 2000.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553, New York City, NY, 17–19 October 1999. IEEE.

[SD98] Amit Sahai and Cynthia Dwork. Concurrent zero-knowledge: Reducing the need for timing constraints. In Hugo Krawczyk, editor, *Advances in Cryptology - Crypto '98*, Berlin, 1998. Springer-Verlag. Lecture Notes in Computer Science Volume 1462.