Math. Struct. in Comp. Science (2009), *vol.* 19, *pp.* 193–205. © 2009 Cambridge University Press
doi:10.1017/S0960129508007330 Printed in the United Kingdom

A note on the feasibility of generalised universal composability†

ANDREW C. C. YAO‡, FRANCES F. YAO§ and YUNLEI ZHAO¶

‡*Institute for Theoretical Computer Science (ITCS), Tsinghua University, Beijing, China*
Email: andrewcyao@tsinghua.edu.cn
§*Department of Computer Science, City University of Hong Kong, Hong Kong, China*
Email: csfyao@cityu.edu.hk
¶*Software School, Fudan University, Shanghai, China*
Email: ylzhao@fudan.edu.cn

Received 19 October 2008

In this paper we study (interpret) the precise composability guarantee of the generalised universal composability (GUC) feasibility with global setups that was proposed in the recent paper Canetti *et al.* (2007) from the point of view of full universal composability (FUC), that is, composability with arbitrary protocols, which was the original security goal and motivation for UC. By observing a counter-intuitive phenomenon, we note that the GUC feasibility implicitly assumes that the adversary has *limited* access to arbitrary external protocols. We then clarify a general principle for achieving FUC security, and propose some approaches for fixing the GUC feasibility under the general principle. Finally, we discuss the relationship between GUC and FUC from both technical and philosophical points of view. This should be helpful in gaining a precise understanding of the GUC feasibility, and for preventing potential misinterpretations and/or misuses in practice.

1. Introduction

The original motivation for and security goal of the framework of universal composability (UC) (Canetti 2001), as implied by its name, was to provide cryptographic systems with a robust composability with *arbitrary* protocols (captured by an *unpredictable* environment). That is, a UC-secure protocol (henceforth referred to as the 'challenge protocol') should preserve its security even when it is composed concurrently with any arbitrary protocols in any arbitrary malicious (unpredictable) way in asynchronous networks like the Internet. These arbitrary protocols may be executed by the same parties or other parties, they may have potentially related inputs, and the scheduling of message delivery may be adversarially coordinated by the adversary controlling communication channels. Furthermore, the local outputs of a protocol execution may be used by other

† The work described in this paper was supported in part by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Number CityU 122105), by CityU Research Grant (9380039), by the National Basic Research Program of China Grant (973) 2007CB807900, 2007CB807901, by NSFC No. 60703091, by the Pujiang Program of Shanghai and by the Young Faculty Program of MSRA.
¶ Corresponding author. The work by this author was done in part while visiting Tsinghua university and City University of Hong Kong.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

protocols in an unpredictable way (Garay *et al.* 2003). It is very natural to expect such a security goal for UC-secure protocols, and it appears (as informal interpretations) in much existing work, and for the moment we will refer to it informally here as full UC (FUC) for presentational simplicity (the technical interpretation of FUC is deferred to Section 4).

Clearly, achieving cryptographic protocols having robust composability with arbitrary protocols is highly and naturally desirable in modern cryptography. This line of research also turns out to have one of the most complicated system complexities in cryptography (the security formulation and analysis within the UC framework are typically abstract and complicated). Due to the very high system complexity and subtle nature of UC, it is critical, for non-experts of UC as well as for the applications of UC theory and protocols in practice, to interpret *precisely* the actual security guaranteed by the UC formulations and implementations. It turns out that the interpretation itself can have subtle implications.

The recent papers Canetti *et al.* (2007) and Yao *et al.* (2007) showed that the traditional UC formulations in Canetti (2001) and Canetti and Rabin (2003) do not capture the natural FUC security goal (which was expected for UC, and was its original motivation). Roughly speaking, traditional UC-secure protocols guarantee composability with other different protocols *that do not share state information with it* (Canetti *et al.* 2007; Yao *et al.* 2007). This requirement may not be realistic when considering adversarial activities in asynchronous networks like the Internet (which was the original motivation for introducing UC), and limits the applicability of UC theory. For example, most developed UC-secure protocols are with trusted global setups, such as a common reference string (CRS) or public keys (PKs) that are drawn randomly and trustingly from some predefined distributions and are known to all parties, thus protocols trivially share common state information[†]. The necessity for trusted setups for UC security arises from the fact that large classes of functionalities cannot be UC realised in the plain model without assuming a majority of honest players – in particular, for the important case of two-party protocols, most of them cannot be UC realised in the plain model (Canetti 2001; Canetti and Fischlin 2001; Canetti *et al.* 2003; Lindell 2003; Lindell 2004).

Very recently, Canetti *et al.* (2007) reformulated UC security with the FUC goal in mind by explicitly considering the case when external arbitrary protocols can share any maliciously correlated state information with the challenge protocol. Such an augmented UC notion is named *generalised UC* (GUC) (Canetti *et al.* 2007). Canetti *et al.* (2007) also proposed some approaches for re-establishing GUC feasibility with some trusted setups (specifically, the key registration with knowledge (KRK) model and the augmented common reference string (ACRS) model). In this paper we consider the general feasibility of GUC in the ACRS model. The key difference between the ACRS model and the traditional CRS model is that the ACRS model additionally allows *corrupted* parties to ask the trusted CRS generator to obtain 'personalised' secret keys that are derived from

[†] As noted in Canetti *et al.* (2007), the approach proposed in Canetti and Rabin (2003) for handling universal composition with joint state (JUC) also does not fully work in this case.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

the common reference string, their public identities and some 'global secret' that is related to the common reference string and remains secret to all parties.

Due to the highly complex nature of GUC for both security formulation and security analysis, it is important to interpret the precise composability guarantee of the general feasibility proposed in Canetti *et al.* (2007) in accordance with the FUC security goal (which is expected for UC and was its original motivation). As GUC was introduced with the FUC goal in mind, and the GUC feasibility with ACRS setup currently has the strongest provable composability, one might easily be led to think that the GUC feasibility proposed in Canetti *et al.* (2007) would have naturally achieved the FUC security goal (that is, composability with arbitrary protocols sharing any maliciously correlated state information with the challenge protocol).

In this paper we clarify (from the point of view of FUC) the subtleties and potential limitations of the general GUC feasibility with global setups proposed in Canetti *et al.* (2007), by observing a counter-intuitive phenomenon. In outline, we note that the GUC feasibility proposed in Canetti *et al.* (2007) is with respect to adversaries with *limited* access to external arbitrary protocols. We also discuss the general principle for achieving the FUC goal, and the relationship between FUC and GUC from both technical and philosophical points of view. This should be helpful in gaining a precise understanding of the general GUC feasibility of Canetti *et al.* (2007), and in preventing potential misinterpretations and/or misuses in practice.

2. Preliminaries

In this section we will briefly recall some preliminaries. To save space, we will not give a presentation of the UC framework here – see Canetti (2001), Canetti and Rabin (2003), Canetti (2006) and Canetti *et al.* (2007) for details of the frameworks of traditional and generalised UC.

We will use standard notation and conventions for writing probabilistic algorithms, experiments and interactive protocols. If A is a probabilistic algorithm, $A(x_1, x_2, \ldots; r)$ is the result of running A on inputs $x_1, x_2, \ldots$ and coins r. We use $y \leftarrow A(x_1, x_2, \ldots)$ to denote the experiment of picking r at random and letting y be $A(x_1, x_2, \ldots; r)$. If S is a finite set, $x \leftarrow S$ is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set, $x \leftarrow \alpha$ is a simple assignment statement. We use $[R_1; \ldots; R_n : v]$ to denote the set of values of v that a random variable can assume due to the distribution determined by the sequence of random processes $R_1, R_2, \ldots, R_n$. We use $\Pr[R_1; \ldots; R_n : E]$ to denote the probability of event E after the ordered execution of random processes $R_1, \ldots, R_n$.

Let $\langle P, V \rangle$ be a probabilistic interactive protocol. We use the notation $(y_1, y_2) \leftarrow \langle P(x_1), V(x_2) \rangle(x)$ to denote the random process of running interactive protocol $\langle P, V \rangle$ on common input x, where P has private input x_1, V has private input x_2, y_1 is P's output and y_2 is V's output. We assume without loss of generality that the output of both parties P and V at the end of an execution of the protocol $\langle P, V \rangle$ contains the transcript of the communication exchanged between P and V during such execution.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

Definition 2.1 ((public-coin) interactive argument/proof system). A pair of interactive machines $\langle P, V\rangle$ is called an interactive argument system for a language L if both are probabilistic polynomial-time (PPT) machines and the following conditions hold:

— **Completeness**. For every $x \in L$, there exists a string w such that for every string z, $\Pr[\langle P(w), V(z)\rangle(x) = 1] = 1$.
— **Soundness**. For every polynomial-time interactive machine P^*, and for all sufficiently large n's and every $x \notin L$ of length n and every w and z, $\Pr[\langle P^*(w), V(z)\rangle(x) = 1]$ is negligible in n.

An interactive protocol is called a *proof* for L if the soundness condition holds against any (even power-unbounded) P^* (rather than only PPT P^*). An interactive system is called a public-coin system if at each round the prescribed verifier can only toss coins and send their outcome to the prover.

Definition 2.2 (statistically/perfectly binding bit commitment scheme). A pair of PPT interactive machines $\langle P, V\rangle$ is called a statistically/perfectly binding bit commitment scheme, if it satisfies the following:

Completeness. For any security parameter n, and any bit $b \in \{0,1\}$, we have

$$\Pr[(\alpha,\beta) \leftarrow \langle P(b), V\rangle(1^n); (t,(t,v)) \leftarrow \langle P(\alpha), V(\beta)\rangle(1^n) : v = b] = 1\,.$$

Computationally hiding. For all sufficiently large n's and any PPT adversary V^*, thecf following two probability distributions are computationally indistinguishable:

$$[(\alpha,\beta) \leftarrow \langle P(0), V^*\rangle(1^n) : \beta]$$

$$[(\alpha',\beta') \leftarrow \langle P(1), V^*\rangle(1^n) : \beta']\,.$$

Statistically/perfectly binding. For all sufficiently large n's and *any* adversary P^*, the following probability is negligible (or equals 0 for perfectly binding commitments):

$$\begin{aligned}\Pr[(\alpha,\beta) &\leftarrow \langle P^*, V\rangle(1^n); (t,(t,v))\\ &\leftarrow \langle P^*(\alpha), V(\beta)\rangle(1^n); (t',(t',v'))\\ &\leftarrow \langle P^*(\alpha), V(\beta)\rangle(1^n)\\ &\quad : v, v' \in \{0,1\} \bigwedge v \neq v']\,.\end{aligned}$$

That is, no (*even computational power unbounded*) adversary P^* can decommit the same transcript of the commitment stage to both 0 and 1.

One-round perfectly binding (computationally hiding) commitments can be based on any one-way permutation OWP (Blum 1982; Goldreich *et al.* 1991). Loosely speaking, given an OWP f with a hard-core predict b (*cf.* Goldreich (2001)), on a security parameter n one commits a bit σ by uniformly selecting $x \in \{0,1\}^n$ and sending $(f(x), b(x) \oplus \sigma)$ as a commitment, while keeping x as the decommitment information. Statistically binding commitments can also be based on any one-way function (OWF), but run in two rounds (Naor 1991; Hastad *et al.* 1999).

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

Proof of Knowledge (POK). The concept of 'proof of knowledge' was introduced informally in Goldwasser *et al.* (1985), and later treated formally in Bellare and Goldreich (1992), Goldreich (2001) and Bellare and Goldreich (2006). POK systems play a fundamental role in the design of cryptographic protocols, and enable a formal complexity theoretic treatment of what it means for a machine to 'know' something. Very roughly, for an interactive protocol, POK means that a possibly malicious prover can convince the honest verifier that an $\mathcal{NP}$ statement is true if and only if it, in fact, 'knows' (that is, possesses) a witness to the statement (rather than just convincing him of the language membership of the statement, that is, the fact that a corresponding witness exists).

Blum's protocol for DHC (Blum 1986). The n-parallel repetitions of Blum's basic protocol for proving the knowledge of a Hamiltonian cycle on a given directed graph G (Blum 1986) is just a 3-round public-coin (witness-indistinguishable) POK system for $\mathcal{NP}$ under any one-way permutation (as its first round involves one-round perfectly binding commitments of a random permutation of G). The following describes Blum's *basic* protocol for DHC:

Common input. A directed graph $G = (V, E)$ with $q = |V|$ nodes.
Prover's private input. A directed Hamiltonian cycle C_G in G.
Round-1. The prover selects a random permutation π of the vertices V and commits (using a perfectly binding commitment scheme) the entries of the adjacency matrix of the resulting permutated graph using π. That is, it sends a q-by-q matrix of commitments so that the $(\pi(i), \pi(j))^{th}$ entry is a commitment to 1 if $(i, j) \in E$, and is a commitment to 0 otherwise.
Round-2. The verifier uniformly selects a bit $b \in \{0, 1\}$ and sends it to the prover.
Round-3. If $b = 0$, the prover sends π to the verifier and reveals all commitments to him (and the verifier checks that the revealed graph is indeed isomorphic to G using π); If $b = 1$, the prover reveals to the verifier only the commitments to entries $(\pi(i), \pi(j))$ with $(i, j) \in C_G$ (and the verifier checks that all revealed values are 1 and the corresponding entries form a simple q-cycle).

Definition 2.3 (collision-resistant hash function). Let $\mathscr{H}$ be a family of hash functions: $\mathscr{K} \times D \to R$, where D and R, respectively, are the domain and range of $\mathscr{H}$, and $\mathscr{K}$ denotes the key space of $\mathscr{H}$. For a particular key $K \in \mathscr{K}$, $\mathscr{H}_K \colon D \to R$ is defined as $\mathscr{H}(K, \cdot)$. We say that $\mathscr{H}$ is collision resistant if, for a randomly chosen key $K \leftarrow \mathscr{K}$ and for any (even non-uniform) polynomial-time algorithm A, the probability that A, given the randomly chosen key K as its input, outputs two distinct points x_1 and x_2 in D such that $\mathscr{H}_K(x_1) = \mathscr{H}_K(x_2)$ is negligible. The probability is taken over the choice of K and the randomness of A.

Identity-based trapdoor commitments (IBTC) (Canetti *et al.* 2007). In the setting of IBTC, a single 'master key' is made public. Additionally, each party can obtain a private key that is associated to its identifier. Intuitively, an IBTC scheme is a commitment scheme with the additional property that a committer who *knows* the receiver's secret key can *equivocate* commitments (that is, it can open up commitments to any value) as it pleases. Furthermore, it should be the case that an adversary who obtains the secret keys of

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

multiple parties should still not be able to violate the binding property of commitments sent to parties for which it has not obtained the secret key – see Canetti *et al.* (2007) for the formal definition of IBTC. IBTC constructions are presented in Atenise and De Medeiros (2003) in the random oracle model, and in Canetti *et al.* (2007) under any one-way function in the standard model.

3. A counter-intuitive phenomenon

In this section we first briefly recall the GUCZK protocol implied by Canetti *et al.* (2007), Canetti and Fischlin (2001) and Canetti *et al.* (2002). We then present a counter-intuitive phenomenon with respect to the GUC feasibility of Canetti *et al.* (2007). Finally, we discuss the reasons underlying the phenomenon.

3.1. *The GUCZK protocol implied by Canetti* et al. *(2007), Canetti and Fischlin (2001) and Canetti* et al. *(2002)*

Canetti *et al.* (2007) re-established the general feasibility of GUC in the augmented CRS (ACRS) model as follows.

— First, it implements a GUC-secure commitment scheme in the ACRS model in a way that is reminiscent of the traditional UC-secure commitments of Canetti *et al.* (2002) in the traditional CRS model.
— Then, using the GUC-secure commitments as a building tool, GUCZK can be implemented by following the construction of traditional UCZK in the traditional CRS model of Canetti and Fischlin (2001).
— Finally, with GUCZK as a key tool, general feasibility of GUC for any cryptographic functionality can be finally re-established with some other techniques.

The GUCZK protocol implied by Canetti *et al.* (2007) and Canetti and Fischlin (2001) is a modification of Blum's protocol for DHC (recalled in Section 2) in which the statistically binding commitments used in its first round are replaced by the GUC-secure commitments of Canetti *et al.* (2007) in the ACRS model. The following is a brief summary of the structure of the GUCZK protocol implied by Canetti *et al.* (2007) and Canetti and Fischlin (2001) in the ACRS model (we omit the augmented CRS for presentational simplicity).

Common input: $x \in L$, where L is an $\mathcal{NP}$-language with $\mathcal{NP}$-relation R_L.

Prover's private input: w such that $(x, w) \in R_L$.

Main-proof stage: This consists of two phases:

Phase-1: This is a coin-tossing protocol in which the verifier V commits to a random string using the IBTC scheme and the prover P responds with a public random string. We use K to denote the output of the coin tossing.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

Phase-2: This is a modification of Blum's protocol for DHC. Specifically, the statistically binding bit commitment in the first round of Blum's protocol for DHC is replaced by:
For a bit $b \in \{0,1\}$, the prover P commits to b as follows:

— It first forms $(c,d) = IBTC(b)$ (that is, it commits to b by running the IBTC scheme, where c is the commitment and d is the decommitment information).

— Then, if $b = 0$, it forms $e = PKE_K(d)$ (that is, it encrypts the decommitment information d by a public-key encryption scheme with the coin-tossing output K as the public key), otherwise (that is, $b = 1$) e is set to be a random value.

3.2. *A simple state-information-sharing attack*

Consider the following protocol $\langle P', V' \rangle$:

Common input: $x \in L$.
Prover's private input: w such that $(x,w) \in R_L$.
Main-proof stage: This consists of two phases:

Phase-1: The verifier V' sends a string K to the prover P', where K is a random encryption public key.

Phase-2: This is the modification of Blum's protocol for DHC. Specifically, the statistically binding bit commitment in the first round of Blum's protocol for DHC is replaced by:
For a bit $b \in \{0,1\}$, the prover P' commits to b as follows:

— It first forms $(c,d) = IBTC(b)$ (that is, commits to b by running the IBTC scheme, where c is the commitment and d is the decommitment information).

— Then, if $b = 0$ it forms $e = PKE_K(d)$ (that is, it encrypts the decommitment information d with the encryption public key K received in Phase-1), otherwise (that is, $b = 1$) e is set to be a random value.

The above protocol, which is designed to be composed with the GUCZK of Canetti *et al.* (2007), is essentially its Phase-2 sub-protocol[†]. The state-information-sharing attack then proceeds as follows:

— Static corruption: The adversary $\mathscr{A}$ corrupts P (the prover of the GUCZK protocol) and V' (the verifier of the above protocol) at the onset of computation.
— In the execution of the GUCZK protocol, referred to as the first session for presentational simplicity, $\mathscr{A}$ works just as the honest prover does until Phase-1 (that

[†] For presentational simplicity we have omitted the commitment receiver's identity in the input of IBTC of Phase-2. In actual implementations, the IBTC in Phase-2 of the above protocol, as well as the IBTC of the Phase-2 of the GUCZK of Canetti *et al.* (2007), commits also to the receiver's identity. But, as the above protocol (which is designed to be composed with the GUCZK of Canetti *et al.* (2007)) works in the plain model, no GUC-secure authentication and identification can be counted on. Actually, Canetti *et al.* (2007) clarifies the fact that authentication/identification is not the issue that would enable one to prevent the following attack.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

is, the coin tossing) finishes. We use K to denote the output of the coin tossing in the first session. $\mathscr{A}$ then suspends the first session.

— Now $\mathscr{A}$ runs the above protocol, referred to as the second session, and sends to the prover P' the value K as the Phase-1 message of the second session, where K is the output of the coin tossing of the first session.

— Then, $\mathscr{A}$ just copies messages received from P' in the second session to the verifier V of the first session.

Clearly, $\mathscr{A}$ can successfully finish the execution of the first session, even without knowing any corresponding $\mathscr{NP}$ witness to the statement that is proved in the first session.

Note that the above state information sharing attack is very simple, and perhaps not very natural, in that it just copies messages from one session to another session. But, in general, it is possible that messages sent in one session by the adversary are not directly copied from another session, but messages in the concurrent interleaving sessions are maliciously correlated. And, also, the protocol to be composed with the GUCZK of Canetti *et al.* (2007) could potentially be a naturally existing and widely used protocol, as demonstrated in Yao *et al.* (2007). Such a phenomenon is in conflict with our intuitive impression and expectations, as well as with some statements and informal interpretations in Canetti *et al.* (2007) about the composability guarantee of the GUC feasibility, which is the strongest composability currently guarantee available in the literature. This motivated us to make a deep investigation to clarify the underlying reasons and to establish the exact security guarantee provided by the GUC feasibility of Canetti *et al.* (2007).

3.3. *Comments and discussions*

A careful investigation shows that the general GUC feasibility with ACRS setup proposed in Canetti *et al.* (2007) relies on the following assumptions:

— The challenge protocol in the ACRS model can share, *by design*, state information with external protocols only through trusted functionality.

— The adversary only controls the communication channels among the instances of the challenge protocols, while having access to some input–output behaviours of external protocols (captured by an entity named the environment).

Putting this in other words, the adversary formulated in the general GUC feasibility (Canetti *et al.* 2007) has *limited* access to external concurrently executing protocols in order to attack the challenge protocol.

But for FUC, that is, composability with arbitrary protocols and unpredictable environments, it is assumed that the adversary controls all communication channels among all executing protocols. In other words, in FUC the adversary has full access to external arbitrary protocols. That is, the adversary considered in accordance with FUC, who has full access to the external world, is much more powerful than the adversary considered in accordance with the GUC feasibility of Canetti *et al.* (2007), who has limited access to the external world. In particular, this shows that the above state-information-sharing attack does not violate the security formulation and analysis in Canetti *et al.* (2007), as we consider a more powerful adversary here than is allowed for in the GUC feasibility

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

of Canetti *et al.* (2007). But, it does violate our intuitive impression and expectations, as well as some statements and informal interpretations in Canetti *et al.* (2007) about the GUC feasibility, which is the strongest composability guarantee currently available in the literature.

It is important to appreciate the fact that the GUC feasibility proposed in Canetti *et al.* (2007) is with respect to adversaries with limited access to external arbitrary protocols, *in order to prevent potential misinterpretation and misuses in practice*. We would also like to mention that the above state-information-sharing attack is only designed to draw attention to the potential security vulnerabilities, *in general*, when facing adversaries with full access to arbitrary external protocols. Even if the specific vulnerability demonstrated by the concrete attack might be viewed as rather unnatural in certain scenarios, the key point is that it does *not* imply that there are no natural security vulnerabilities, in general, when facing adversaries with full access to external arbitrary protocols.

4. FUC versus GUC

In this section we present a general principle for achieving FUC, and then, following this general principle, we discuss some approaches for fixing the GUC feasibility of Canetti *et al.* (2007). Finally, we discuss the relationship between GUC and FUC from both technical and philosophical points of view.

4.1. *A general principle for FUC*

In the real world of cryptographic protocols running concurrently with arbitrary (external) protocols in an asynchronous open network like the Internet, typically, any message received by an honest player in the challenge protocol could be maliciously dependent on not only the transcript of the challenge protocol, but also, and more harmfully, the whole transcript of the external arbitrary protocols running concurrently with the challenge protocol. This is just the (adversarial) scenario considered by FUC. That is, it is a natural adversarial strategy for the adversary to manage to make the challenge protocol maliciously related to the external arbitrary protocols by concurrent interleaving and modifying attacks, thus sharing some maliciously correlated state information amongst them.

The key point here is that the external arbitrary protocols (running concurrently with the challenge protocol) and the adversarial strategies employed by the adversary are *unpredictable*. Even if a given challenge protocol is proved to satisfy a certain level of composability with respect to an adversary having limited access to the external world (as in the UCZK of Garay *et al.* (2003) and the GUCZK implied by Canetti *et al.* (2007), Canetti and Fischlin (2001) and Canetti *et al.* (2002)), it does not imply that the sub-protocols or building tools used by the challenge protocol guarantee the same level of composability. An adversary could maliciously relate the (weaker) sub-protocols or building tools to the external world *in an unpredictable way*, so that some (*not necessarily global but possibly local and session-specific*) state information is maliciously shared amongst them.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

The general principle here is that for a protocol to be FUC secure, any sub-protocol used by the protocol, and even each message sent in the protocol, should also be FUC secure. In other words, we need the sub-protocols used by the protocol, and even each message sent in the protocol, to be session/source-authentic – that is, each message should not be maliciously modified, by the adversary, from/to the external arbitrary protocols. We note this is a very strong requirement. Following this general principle for FUC, in the next section we make some (informal) suggestions for fixing the GUC feasibility of Canetti *et al.* (2007).

4.1.1. *Suggestions for fixing GUC feasibility with session/source-authentic commitments.* According to the above general principle for FUC, in order to achieve a FUC-secure protocol, we may require the subprotocols used by the protocol, and even each message sent in the protocol, to be session/source-authentic – that is, they should not be maliciously changed, by the adversary, in passing between the arbitrary (external) protocols and the unpredictable environment. Then, with respect to the specific GUCZK implied by Canetti *et al.* (2007), we propose a fix, which provides very useful session/source-authentic non-modifiability in many applied scenarios (though it still does *not* necessarily render full UC security[†]).

The idea is to augment the underlying IBTC scheme used in the UCZK of Canetti *et al.* (2007) as follows. For any message m to be committed, we mask the message m, together with some public session/source-specific auxiliary information aux (in particular, aux could include the session ID, the committer ID, the receiver ID, and some partial transcript), using a collision-resistant hash function $\mathscr{H}$; and then, only the hashed value is committed. Specifically, in this case, the commitment is $IBTC(\mathscr{H}(m, aux))$. We call such commitments *session/source-authentic ID-based commitments*.

Note that for any commitment scheme Com, any message m and any (public) auxiliary information aux, the session/source-authentic commitment $Com(\mathscr{H}(m, aux))$ does not lose the hiding and binding properties, in contrast to $Com(m)$. Specifically, the hiding property of session/source-authentic commitments comes directly from that of the original commitment scheme Com; for the binding property, we note that the ability to equivocate session/source-authentic commitments implies the ability to break the collision resistance of the underlying hash function $\mathscr{H}$, or the ability to break the binding property of the underlying commitment scheme Com. In particular, this means that the augmented version of the GUCZK of Canetti *et al.* (2007), with the $IBTC$ replaced by session/source-authentic $IBTC$, retains the same GUC security in accordance with the GUC formulation of Canetti *et al.* (2007). But, the augmented version provides some extra non-modifiability/composability guarantees. In particular, the attack demonstrated in Section 3 fails in this case.

On the usefulness of session/source-authentic commitments. The usefulness of the session/source-authentic commitments lies in the observation that even if the original

[†] Actually, we have some doubts about the possibility of FUC in general.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

commitment scheme *Com* is very weak with respect to man-in-the-middle attacks, the session/source-authentic commitments using the *hash-then-commit* paradigm still provide very reasonable and practical non-modifiability guarantees. This is from the fact that: given a session/source-authentic commitment c (for example, one eavesdropped from one session), it could be possible for the adversary to maliciously modify c into a commitment c' (possibly in another concurrent session). But, it is intuitively hard for the adversary to successfully open c' to satisfy the additional session/source-specific restriction imposed using the collision-resistant hashing. Thus, we suggest session/source-authentic commitments could be very useful, in particular, in many applied scenarios involving commitments, for providing practical and reasonable non-modifiability/composability guarantees against man-in-the-middle attacks.

4.2. *FUC vs. GUC: a technical view*

Note that the difference between FUC and GUC is that FUC considers an adversary having full access to external arbitrary protocols besides the challenge protocols, and, in particular, one that controls all communication channels between all (external and challenge) protocols.

Technically, FUC security means that what can be performed by an adversary having full access to external arbitrary protocols against the challenge protocol that securely implements an ideal functionality, can also be performed by a PPT simulator (with full access to arbitrary external protocols) in the ideal world against the ideal functionality. Here, the entity environment formulated in UC/GUC can be waived.

Within the framework of GUC, one way to give the adversary the ability to control (either by merely observing or by maliciously modifying) network messages transmitted in communication channels (among arbitrary external protocols as well as the challenge protocols) is to define a special state-sharing functionality through which the adversary, cooperating with the environment, can control all network messages transmitted in the communication channels among arbitrary external protocols and the challenge protocols. In this case, the combination of the adversary and the environment can be seen as a single adversary from the point of view of the FUC framework. The typical functionality enabling the adversary to see (but not necessarily being able to modify) all external messages is the functionality of message transmit authenticator (MT-authenticator). This poses the fundamental problem: can a FUC-secure MT-authenticator exist in general (with certain setups)? Recall that an adversary in the FUC framework fully controls all communication channels and can do whatever he wishes. In a way, FUC amounts to GUC with respect to *any* state-sharing functionalities, while the GUC feasibility proposed in Canetti *et al.* (2007) is with respect to *some specific* state-sharing functionalities.

4.3. *FUC vs. GUC: a philosophical view*

From a philosophical point of view, FUC security, that is, composability with arbitrary protocols, which was the original motivation and goal of UC, can be viewed as the *ultimate* goal for protocol composition in arbitrary adversarial settings, and we are currently on

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

an asymptotic path to this ultimate security goal. The line of research working towards this ultimate goal can be viewed as one of the most ambitious, as well as the hardest, in cryptography. No matter whether this ultimate security goal can be achieved in the future, we have been witnessing a series of celebrated and accumulating accomplishments while travelling along these lines, and these have cast light on the extremely complicated and subtle nature of protocol composition, as well as having deepened our knowledge of protocol composition. As we work our way along this asymptotic trajectory, it is also important for us to be able to interpret precisely where we are currently.

Acknowledgements

We would like to thank Ran Canetti, Yevgeniy Dodis and Shabsi Walfish for discussions that were both very valuable and thoughtful.

References

Atenise, G. and De Medeiros, B. (2003) Identity-Based Chameleon Hash and Applications. Cryptology ePrint Archive, Report No. 2003/167.

Blum, M. (1982) Coin Flipping by Telephone. In: *Proc. IEEE Spring COMPCOM* 133–137.

Blum, M. (1986) How to Prove a Theorem so No One Else can Claim It. In: *Proceedings of the International Congress of Mathematicians*, Berkeley, California 1444–1451.

Bellare, M. and Goldreich, O. (1992) On Defining Proofs of Knowledge. In: Brickell, E. F. (ed.) Advances in Cryptology – Proceedings of CRYPTO 1992. *Springer-Verlag Lecture Notes in Computer Science* **740** 390–420.

Bellare, M. and Goldreich, O. (2006) On Probabilistic versus Deterministic Provers in the Definition of Proofs of Knowledge. *Electronic Colloquium on Computational Complexity* **13** (136).

Canetti, R. (2001) Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: *IEEE Symposium on Foundations of Computer Science* 136–145.

Canetti, R. (2006) Security and Composition of Cryptographic Protocols: A Tutorial. *Distributed Computing column of SIGACT News* **37** (3,4). (Also available from Cryptology ePrint Archive, Report 2006/465.)

Canetti, R., Dodis, Y., Pass, R and Walfish, S. (2007) Universal Composable Security with Global Setup. In: Vadhan, S. (ed.) Theory of Cryptography (TCC). *Springer-Verlag Lecture Notes in Computer Science* **4392** 61–85.

Canetti, R., Dodis, Y., Walfish, S and Zhao, Y. (2007) Personal communications.

Canetti, R. and Fischlin, M. (2001) Universal Composable Commitments. In: Kilian, J. (ed.) Advances in Cryptology – Proceedings of CRYPTO 2001. *Springer-Verlag Lecture Notes in Computer Science* **2139** 19–40.

Canetti, R., Kushilevitz, E. and Lindell, Y. (2003) On the Limitations of Universal Composition Without Set-Up Assumptions. In: E. Biham (ed.) Advances in Cryptology – Proceedings of EUROCRYPT 2003. *Springer-Verlag Lecture Notes in Computer Science* **2656** 68–86.

Canetti, R., Lindell, Y., Ostrovsky, R. and Sahai, A. (2002) Universally Composable Two-Party and Multi-Party Secure Computation. In: *ACM Symposium on Theory of Computing* 494–503.

Canetti, R. and Rabin, T. (2003) Universal Composition with Joint State. In: M. Yung (ed.) Advances in Cryptology – Proceedings of CRYPTO 2002. *Springer-Verlag Lecture Notes in Computer Science* **2729** 265–281.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press

Garay, J. A., MacKenzie, P. and Yang, K. (2003) Strengthening Zero-Knowledge Protocols Using Signatures. *Journal of Cryptology* (to appear). (A preliminary version appears in Biham, E. (ed.) Advances in Cryptology – Proceedings of EUROCRYPT 2003. *Springer-Verlag Lecture Notes in Computer Science* **2656** 177–194.)

Goldreich, O. (2001) *Foundation of Cryptography – Basic Tools*, Cambridge University Press.

Goldreich, O., Micali, S. and Wigderson, A. (1986a) Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design. In: *IEEE Symposium on Foundations of Computer Science* 174–187.

Goldreich, O., Micali, S. and Wigderson, A. (1986b) How to Prove all $\mathcal{NP}$-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. In: Odlyzko, A. M. (ed.) Advances in Cryptology – Proceedings of CRYPTO 1986. *Springer-Verlag Lecture Notes in Computer Science* **263** 104–110.

Goldreich, O., Micali, S. and Wigderson, A. (1991) Proofs that Yield Nothing but their Validity or All Languages in $\mathcal{NP}$ have Zero-Knowledge Proof Systems. *Journal of the Association for Computing Machinery* **38** (1) 691–729. (Preliminary versions appear in Goldreich *et al.* (1986a) and Goldreich *et al.* (1986b).)

Goldwasser, S., Micali S. and Rackoff, C. (1985) The Knowledge Complexity of Interactive Proof-Systems. In: *ACM Symposium on Theory of Computing* 291–304.

Hastad, J., Impagliazzo, R., Levin, L. A. and Luby, M. (1999) Construction of a Pseudorandom Generator from any One-Way Function. *SIAM Journal on Computing* **28** (4) 1364–1396.

Lindell, Y. (2003) General Composition and Universal Composability in Secure Multi-Party Computation. In: *IEEE Symposium on Foundations of Computer Science* 394–403.

Lindell, Y. (2004) Lower Bounds for Concurrent Self Composition. In: M. Naor (ed.) Theory of Cryptography (TCC) 2004. *Springer-Verlag Lecture Notes in Computer Science* **2951** 203–222.

Naor, M. (1991) Bit Commitment Using Pseudorandomness. *Journal of Cryptology* **4** (2) 151–158.

Pass, R. (2003) On Deniabililty in the Common Reference String and Random Oracle Models. In: Boneh, D. (ed.) Advances in Cryptology – Proceedings of CRYPTO 2003. *Springer-Verlag Lecture Notes in Computer Science* **2729** 316–337.

Yao, A. C. C., Yao, F. F. and Zhao, Y. (2007) A Note on Universal Composable Zero-Knowledge in the Common Reference String Model. In: Cai, J., Cooper, S. B. and Zhu, H. (eds.) Theory and Applications of Models of Computation – Proceedings of TAMC 2007. *Springer-Verlag Lecture Notes in Computer Science* **4484** 462–473.

https://doi.org/10.1017/S0960129508007330 Published online by Cambridge University Press