A final version of this paper appeared in Math. Struct. in Comp. Science, 2016. This is the preliminary full version.

A Second Note on the Feasibility of Generalized Universal Composability

ALONSO GONZÁLEZ[†] and ALEJANDRO HEVIA[‡]

Departamento de Ciencias de la Computación, Universidad de Chile.
ahevia@dcc.uchile.cl, alonso.gon@gmail.com.

Received 2 October 2014; revised 11 May 2016

Yao et al. (YYZ09a; YYZ07) claimed a potential limitation on the class of protocols that could be securely implemented in the Generalized Universal Composability (GUC) framework proposed by Canetti et al. (CDPW07). Specifically, Yao et al. presented a concrete attack on a GUC Zero Knowledge (GUCZK) protocol, a natural adaptation Blum's ZK proof for Directed Hamiltonicity using the general GUC feasibility of (CDPW07). Interestingly, the attack was not analyzed in the GUC model in (YYZ09a) but in the *FUC model*, a new UC-like framework proposed in the same work.
Nonetheless, Yao et al. (YYZ09a) argued that, in light of this attack, GUC would lose its concurrent general composability and proof of knowledge properties. Concretely, they argue that GUC composability would now be with respect to some adversaries with limited access to external arbitrary protocols.
In this work, we show that the claimed attack from Yao et al. is indeed harmless and does not contradict the security of the mentioned GUCZK protocol, thus restoring the general feasibility for GUC.

1. Introduction

The *Universal Composability framework* (UC), introduced by Canetti (Can01), allows designing protocols whose security is maintained even when composed with any other arbitrary protocol (i.e. general composition (Lin03)).

But such strong guarantee comes at a price: many interesting computational tasks, modeled as *ideal functionalities*, are unrealizable in the so-called *plain UC* as shown by Canetti et al. (CKL06). In particular, this result ruled out the possibility of realizing many functionalities, such as commitments and zero knowledge. Nevertheless, Canetti et al (CLOS02) showed that any functionality can be UC-realized in the $\mathcal{F}_{\mathsf{crs}}$-hybrid model, where a *Trusted Third Party* provides a *Common Reference String*.

In practice, trusted parties are obviously hard to obtain and consequently, it is desirable to restrict our trust to a single trusted party – one that could be reliably shared between

[†] The first author acknowledges financial support of CONICYT, CONICYT-PCHA/Doctorado Nacional/2013-21130937 and INRIA Chile, via NIC Chile Research Labs.
[‡] The second author thanks the support of NIC Chile Research Labs.

many protocols. But UC-secure protocols must be subroutine respecting, which means that the protocol an its subroutines should no provide (resp. receive) input (resp. output) to (resp. from) any other protocol. In the case of protocols that are run in the $\mathcal{F}_{\mathsf{crs}}$-hybrid model, this implies that $\mathcal{F}_{\mathsf{crs}}$ must be a subroutine of the protocol. Therefore, different subroutine respecting protocols must call different instances of $\mathcal{F}_{\mathsf{crs}}$.

Pass (Pas03) showed that if a shared $\mathcal{F}_{\mathsf{crs}}$ is used for realizing zero knowledge, it leads to situations where a zero knowledge protocol looses its deniability, a key and desirable property. Furthermore, Yao et al. (YYZ09b) showed that this setting could also lead to compromising the framework's general composability as well as the proof of knowledge property.

In order to solve this problem, the *Generalized Universal Composability (GUC) framework* was introduced by Canetti, Dodis, Pass, and Walfish (CDPW07). In this setting, protocols are not subroutine respecting but $\bar{\mathcal{G}}$-subroutine respecting instead – protocols are subroutine respecting except that they are allowed to call the *shared functionality* $\bar{\mathcal{G}}$. Canetti et al. (CDPW07) showed that a global Common Reference String is still insufficient for obtaining general feasibility for GUC (that is, realizing any functionality), thus a different setup assumption must be used. Consequently, the *Augmented CRS shared functionality* $\bar{\mathcal{G}}_{\mathsf{acrs}}$ was introduced in the same work (CDPW07). Using this setup assumption they showed how any ideal functionality could be realized.

Nevertheless, Yao et al. (YYZ09a; YYZ07) claimed that there exists a potential limitation of the general feasibility for generalized universal composability (GUC) framework. Specifically, they showed a concrete attack on the GUCZK protocol implied by the general GUC feasibility result of Canetti et al. (CDPW07). The attack's validity was first argued in the GUC model (YYZ07) but later re-stated in the so-called FUC mode (YYZ09a), a supposedly more general UC-like framework proposed in the same work (see Sect. 3.3 for a discussion of FUC). In any case, Yao et al. (YYZ09a; YYZ07) argued that this was evidence that GUC would lose its concurrent general composability. More concretely, they claimed that GUC composability would be restricted to some adversaries with limited access to external arbitrary protocols, as opposed to the general composability proposed in (CDPW07).

The consequences of this attack are significant. Canetti himself has wondered about the implications of the above result (Can07). Concretely, he conjectures that the attack is an indication that, in some situations, there are protocols that unintentionally (but inevitably) share more information than just public information.

1.1. *Our contribution*

In this work, we show that the claimed attack from Yao et al. is harmless and does not contradict the security of the implied GUCZK, thus restoring the general feasibility for GUC. More concretely, we show that the attack described in Yao et al. (YYZ09a) can be straightforwardly mapped into an attack in the GUC model. Then, we show that the same techniques used to prove the security of the general feasibility result for GUC can be used in order to simulate the mapped attack only with access to the ideal functionality.

1.2. *Organization*

The attack of Yao, the main focus of this work, applies to the GUC Zero-Knowledge protocol implied by the general feasibility result for GUC. In order to describe this protocol, we must introduce several primitives. In Sect. 2, after presenting some preliminaries about notation, terminology, and execution framework, we describe Identity-based Trapdoor Commitments, Blum's (honest verifier) zero-knowledge protocol, and Dense Pseudo-Random Ciphertext Secure Encryption Schemes, which are tools we need to describe the UC Adaptive Identity-based Commitments (UAIBC). Then, in Sect. 3.4 we describe the objective of Yao et al.'s attack, the UC Zero-Knowledge protocol implemented using the UAIBC protocol, as well as a discussion of why the attack is harmless, including exhibiting a simulator for the attack. We discuss the consequences and conclude in Sect. 4.

2. Preliminaries

We denote by $[n]$ the set $\{1, 2, \ldots, n\}$. We denote by $x \leftarrow v$ the instruction that assigns the variable x the value v. We denote by an integer k the security parameter.

We use PPT as shortcut for *probabilistic polynomial time Turing Machine*, that is, a Turing Machine who might use a *random tape* and whose execution time is bounded by some polynomial in the security parameter.

Given a probabilistic Turing Machine M we denote by $M(x; \rho)$ the output of M when executed with input x and randomness ρ. We use $y \stackrel{R}{\leftarrow} M(x)$ as shortcut for $y \leftarrow M(x; \rho)$, when ρ is uniformly sampled from $\{0,1\}^r$ and r is polynomially related to the security parameter. If D is a distribution, we denote by $x \stackrel{R}{\leftarrow} D$ the experiment of sampling x according to D.

We use $\mathsf{poly}(k)$ to denote a function $n : \mathbb{N} \to \mathbb{N}$ polynomially bounded in the security parameter, i.e. there exists a polynomial p such that $n(k) < p(k)$ for any large enough integer k. We use $\mathsf{negl}(k)$ to denote a function $\mu : \mathbb{N} \to \mathbb{R}_+$ which is asymptotically bounded by the inverse of any polynomial, that is, for any polynomial p and for any integer k large enough $\mu(k) < 1/p(k)$.

2.1. *Universal Composition (UC) and Generalized Universal Composition (GUC)*

In this section, we briefly recall the Universal Composability framework and the Generalized Universal Composability framework. The interested reader is refereed to (Can01; CDPW07; CLOS02) for more details.

The Universal Composability framework (UC)

The UC framework is a general tool for modeling security of cryptographic protocols. The desired properties of cryptographic protocols are defined in terms of tasks or *ideal functionalities*. The ideal functionality is a trusted third party that obtains inputs directly from the parties, performs certain instructions on these inputs, and provides the appropriate outputs back to the parties. A protocol securely implements a given cryptographic

task if running the protocol against a realistic (i.e. real-life) adversary "emulates" the execution of an ideal functionality. In the ideal functionality, the task is computed by the trusted party directly interacting with the parties against a very limited adversary called the *ideal adversary* or *simulator*. The notion of "emulation" involves a distinguisher $\mathcal{Z}$ which not only provides the inputs to the parties and sees their outputs but also interacts with the adversary, with the goal of telling whether it is interacting with a real protocol and the real-life adversary, or with the functionality and the ideal adversary. Good emulation means no such environment is successful. See details and proofs in (Can01).

The main advantage of UC security is that it composes, that is, the security of any protocol is maintained even when the protocol is being executed concurrently with other, possibly adversarially chosen, protocols. But there is a restriction: the protocol must be **subroutine respecting**. This means that the protocol itself and all its subroutines do not provide any input or output to any other protocol. In other words, the protocol and the subroutines called by the protocol are independent of all other protocols and cannot share state with other protocols.

The Generalized UC framework (GUC):

Being subroutine respecting, however, is an unrealistic limitation when protocols need access to a trusted party. This is the case when *setup assumptions* are used (which are required when realizing more interesting functionalities (CKL06)), as noted by Canetti et al. (CDPW07). The goal of the GUC framework is to overcome the UC limitation of requiring subroutine respecting protocols.

In the GUC framework (CDPW07), subroutine respecting protocols are extended so they can be $\bar{\mathcal{G}}$*-subroutine respecting*: protocols can be subroutine respecting except that are allowed to call the *shared functionality* $\bar{\mathcal{G}}$. The GUC framework models $\bar{\mathcal{G}}$-subroutine respecting protocols by giving the environment direct access to the shared functionality. This small change lets the environment simulate protocols that share state with the analyzed protocol.

The definitions of execution and emulation in GUC are almost identical to those of UC but notation changes. From now on, we denote by EXEC both the UC-execution as well as the GUC-execution, and by UC-emulation we also refer to GUC-emulation. Analogously to UC, it is possible to prove a composition theorem for $\bar{\mathcal{G}}$-subroutine respecting protocols.

2.2. *Full Universal Composition (FUC)*

We now briefly recall the FUC framework, proposed in Yao et al. (YYZ09a). Even though the presentation there is very informal, in what follows, we attempt to extract a somewhat more precise, UC-like, especification in order to reason about it. Below, quotes are from Yao et al. (YYZ09a).

Full universal composability (FUC) denotes a UC-like framework that attempts to capture composability with arbitrary protocols, *"which was the original security goal and motivation for UC."* In this context, the described framework seems to follow the UC design, with the following differences. In FUC, *"the adversary controls all communication*

channels among all executing protocols. In other words, in FUC the adversary has full access to external arbitrary protocols.". Accordingly, the FUC framework would allow the adversary to directly start new protocol executions (as opposed to UC where only the environment can do it) and control all the communication of these new protocols. Note that, like in UC, it is the environment and not the adversary who provides the inputs to the new protocol, otherwise the resulting notion of emulation is meaningless (see discussion after the environment constructed in Sect 3.2).

In Yao et al. (YYZ09a), it is claimed that *"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."*

UC/GUC/FUC Attacks: In order to understand the attack from (YYZ09a; YYZ07), we describe what an attack to some protocol Π is in any UC-like framework mentioned above (the concept is shared among all framworks). Intuitively, a protocol is not secure if there is an attack, where security in UC/GUC/FUC is defined by the concept of *protocol emulation* mentioned earlier. A protocol Π realizes a functionality $\mathcal{F}$ if for all adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for all environment $\mathcal{Z}$

$$|\Pr[\mathsf{EXEC}(\mathcal{Z},\mathcal{A},\Pi)=1]-\Pr[\mathsf{EXEC}(\mathcal{Z},\mathcal{S},\mathcal{F})=1]|\leq \mathsf{negl}(k)\,.$$

Consequently, there is an attack to protocol Π if there exists an adversary $\mathcal{A}$ such that for all $\mathcal{S}$ there exists $\mathcal{Z}$ where

$$|\Pr[\mathsf{EXEC}(\mathcal{Z},\mathcal{A},\Pi)=1]-\Pr[\mathsf{EXEC}(\mathcal{Z},\mathcal{S},\mathcal{F})=1]|>1/k^c$$

for some $c\in\mathbb{N}$.

2.3. *Identity Based Trapdoor Commitments*

Identity-based Trapdoor Commitments (IBTC) (Wal08) are commitment schemes parameterized by an identity ID and with the additional property that, with knowledge of some *trapdoor* associated to ID, one can violate the binding property.

Definition 1 (IBTC). An Identity-Based Trapdoor Commitment scheme IC is given by a 5-tuple of PPT algorithms $\mathsf{IC}=(\mathsf{Setup},\mathsf{Extract},\mathsf{Com},\mathsf{ECom},\mathsf{Equiv})$, with the following basic properties:

Setup: A deterministic algorithm which takes as input a (uniformly random) "master secret key" $MSK\in\{0,1\}^k$ and outputs a public key PK.

Extract: On input (ID,MSK) outputs a trapdoor SK_{ID} for identity ID.

Com: A deterministic algorithm which takes as input a tuple of the form (PK,ID,d,m), and outputs a commitment κ for message m under identity ID using random input d. The domain from which d is chosen is denoted by $\mathcal{D}$, and the domain where the commitment lies is denoted by $\mathcal{C}$. As a shorthand, we may write $\mathsf{Com}_{ID}(m;d)$ to denote $\mathsf{Com}(PK,ID,d,m)$.

ECom: On input (PK,ID,SK_{ID}) outputs a pair (κ,α), to be used with Equiv.

Equiv: On input $(PK, ID, SK_{ID}, \kappa, \alpha, m)$ produces a value $d \in \mathcal{D}$ such that $\mathsf{Com}_{ID}(m; d) = \kappa$. That is, d makes κ appear to be a commitment to m.

2.4. *Blum's Proof of Knowledge protocol for Directed Hamiltonian Cycle*

We recall Blum's protocol for Directed Hamiltonian Cycle (DHC) from (Blu86). Blum's protocol is 3-round public coin honest verifier protocol (a.k.a. Σ-protocol) in which the prover first commits to a random permutation of the edges in a graph, and then to the permutation itself. The verifier uniformly chooses between asking for an opening a Hamiltonian cycle of the permuted graph, or an opening of the graph and the permutation.

Common input: The common input for the prover and the verifier is a graph $G = (V, E)$, where $V = [n]$ and E is represented as a matrix E of size n^2, where each entry $e_{ij} \in \{0, 1\}$ for $i, j \in [n]$.

Secret input: The secret input of the prover is a Hamiltonian cycle C for G.

Round-1: The prover randomly samples a permutation over $[n]$ and commit to the matrix C of size n^2, such that each entry is given by $c_{ij} = \mathsf{Com}(e_{kl}, r_{ij})$, each $r_{ij} \in_R \{0, 1\}^{\mathsf{poly}(k)}$, for $k, l \in [n]$, $i = \pi(k)$, and $j = \pi(l)$. The prover also commit to a description of π.

Round-2: The verifier uniformly selects a bit $c \in_R \{0, 1\}$ and sends it to the prover.

Round-3: If $b = 0$, the prover sends to the opening to all commitments, and the verifier checks that the revealed graph is indeed isomorphic to G via π. If $b = 1$, the prover reveals to the verifier only the commitments to entries $\pi(i), \pi(j)$ such that $(i, j) \in C$, and the verifier checks that all revealed values are 1 and the corresponding entries form a simple n-cycle.

We note that the previous protocol slightly differs from the original protocol from (Blu86): in round-1 of the above protocol, the prover also sends a commitment to the permutation used. Yao et al. (YYZ09a) use the original protocol by Blum but this does not suffice for straight line witness extraction (a requirement for the simulation in UC), so we believe it is an error. Moreover, the protocol we describe here is the underlying protocol used in order to construct UCZK (CF01, Sect. 5), and allows straight-line witness extraction for malicious provers.

2.5. *Dense Pseudo-Random Ciphertexts Secure Encryption*

The protocol UAIBC from (CDPW07) made use of a Dense Pseudo-random Ciphertexts (PRC) Secure Encryption which is an encryption scheme where the the public keys and the ciphertexts are "close" to the uniform distribution on the respective support sets (DSP92; Wal08).

Definition 2 (Dense PRC Encryption). A public key encryption scheme $\mathsf{PRC} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, satisfying the standard correctness properties for public key encryption, is said to be Dense PRC secure if it satisfies the following additional constraints:

1 Public keys produced by Gen lie within some abelian group Φ, with efficiently computable inverse and group operations.

2 Membership in Φ is efficiently testable.
3 The uniform distribution on Φ is efficiently samplable.
4 The distribution of public keys produced by Gen is computationally indistinguishable from the uniform distribution on Φ.
5 For all PPT adversaries $\mathcal{A}$:

$$\Pr\left[\mathcal{A}^{LR(\cdot,0)}(1^k)=1\right]-\left[\mathcal{A}^{LR(\cdot,1)}(1^k)=0\right]\leq \mathsf{negl}(k),$$

where the answer to a query of the oracle $LR(m,b)$ is computed by obtaining $(\rho,\rho^{-1})\stackrel{R}{\leftarrow}\mathsf{Gen}(1^k)$ and returning the pair $\langle\rho,\mathsf{Enc}_\rho(m)\rangle$ if $b=0$, or returning the uniformly random pair $\langle U,R\rangle$ if $b=1$ (where U is chosen from a group Φ, and R from $\{0,1\}^{|E_U(m)|}$).

2.6. *Universally Composable Adaptive Identity Based Commitments (*UAIBC*)*

We recall the UAIBC protocol from (CDPW07). This is an interactive protocol where the committer party P_c commits to a bit b to the receiver party P_r. UAIBC allows extraction of commitments when the committer is corrupt, which means that the actual committed value can be efficiently computed, as well as commitment equivocability, that is, if the receiver is corrupt there is a procedure to open the committed value to an arbitrary value. In order to achieve these properties, UAIBC builds on IBTC, which allows equivocation, and a *coin tossing protocol*, which allows extraction.

Let $\mathsf{IC}=(\mathsf{Setup},\mathsf{Extract},\mathsf{Com},\mathsf{ECom},\mathsf{Equiv})$ be an IBTC scheme and $\mathsf{PRC}=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ a Dense PRC Encryption scheme. In the *commit phase* both parties engages in a *coin tossing protocol* (CT), where a completely random value is obtained by both parties. The CT protocol have the additional property that, when P_c is corrupted, the outcome can be set to any value. After the execution of the CT protocol, P_c commits to b by computing $\kappa=\mathsf{Com}_{P_r}(b;d)$, where $d\in_R\mathcal{D}$, and also computing the value

$$\psi=\begin{cases}\mathsf{Enc}_\rho(d) & \text{if } b=0,\\ c\in_R\mathcal{C} & \text{otherwise.}\end{cases}$$

A detailed description of UAIBC can be found in Fig. 1.

3. The attack to UAIBC

In (YYZ09a) Yao et al. claimed a potential limitation of the general feasibility for GUC. Specifically, they presented a concrete attack on the GUCZK protocol implied by the general GUC feasibility of (CDPW07), supposedly showing that this protocol could lose concurrent general composability and proof of knowledge properties. In this section, we review both the GUCZK protocol mentioned above as well as the attack from (YYZ09a).

3.1. *The implied GUC Zero Knowledge protocol*

Canetti and Fischlin (CF01) showed how to obtain composable zero knowledge from composable commitments, or more formally, how to realize $\mathcal{F}_{\mathsf{zk}}$ (see Fig. 2) in the $\mathcal{F}_{\mathsf{com}}$-hybrid

Fig. 1. The protocol UAIBC of Sect. 2.6.

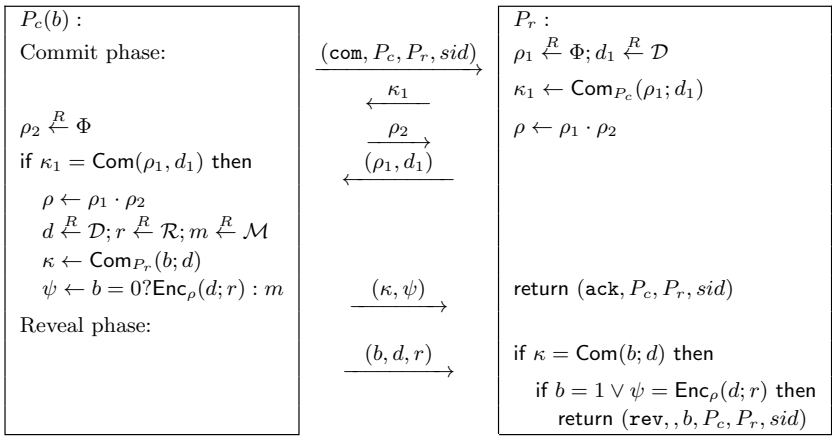


Fig. 2. The Zero-Knowledge ideal functionality

Functionality $\mathcal{F}_{\mathsf{zk}}$
$\mathcal{F}_{\mathsf{zk}}$, parameterized by a binary relation $\mathcal{R}$ and running with a prover P, a verifier V , and an adversary $\mathcal{S}$, proceeds as follows. Upon receipt of a message $(\mathtt{ZK\text{-}prover}, P, V, x, w)$ from the prover P: If $(x, w) \in \mathcal{R}$, then send $(\mathtt{ZK\text{-}proof}, P, V, x)$ to V and $\mathcal{S}$, and halt. Otherwise halt.

model. Their protocol is a simple adaptation of Blum's protocol, where commitment calls are replaced with calls to $\mathcal{F}_{\mathsf{com}}$.

In GUC, the same protocol works for obtaining GUCZK from UAIBC. The GUCZK implied by (CDPW07) and (YYZ09a) is described in Fig. 3. We observe, however, that the protocol from Fig. 3 is NOT directly obtained by replacing each call to $\mathcal{F}_{\mathsf{com}}$ by the protocol UAIBC. Instead and following (YYZ09a), only one CT is executed. Nonetheless, if one CT protocol is executed for each call to $\mathcal{F}_{\mathsf{com}}$, it is easy to see that the attack from (YYZ09a) even modified for many calls to CT remains harmless.

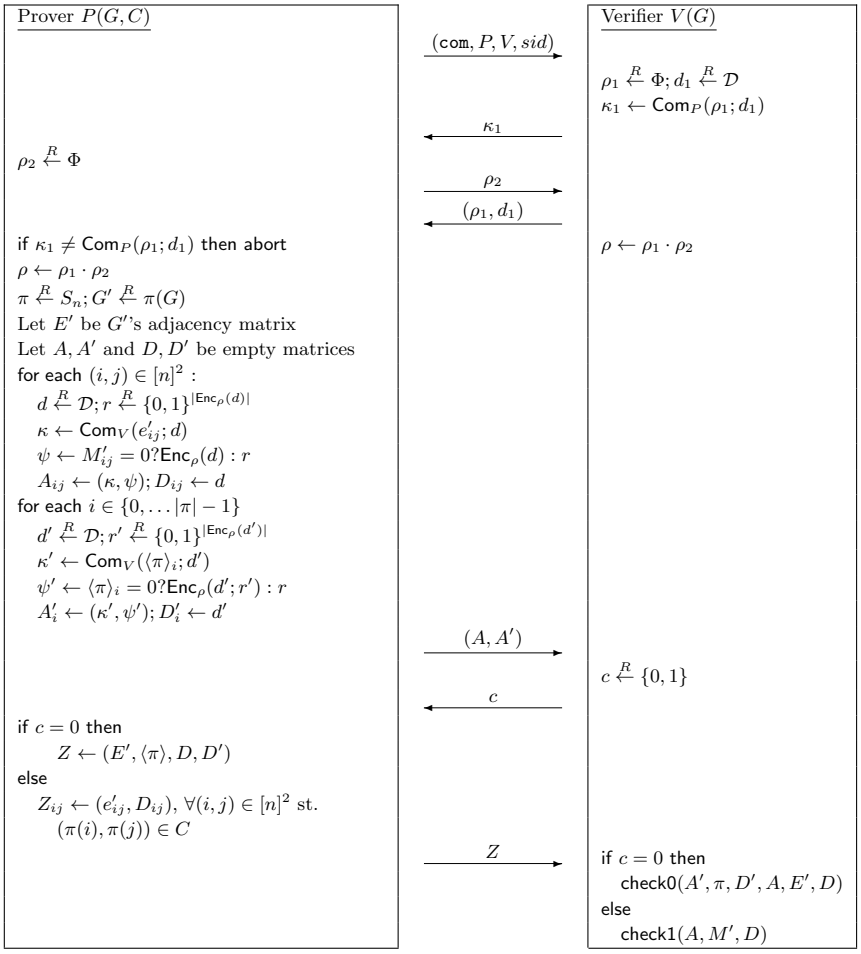
3.2. *The attack from (YYZ09a)*

We now describe the attack from (YYZ09a), which was described in the FUC framework. The attack from (YYZ09a) is a *chosen protocol attack*, that is, an adversarial protocol is designed in order to be composed with protocol from Fig. 3. Also, notice that the protocol from (YYZ09a), described in Fig. 4, is exactly as the protocol from Fig. 3, except that the coin tossing is replaced by a random value chosen by the verifier. This resemblance is exploited by an adversary – who (supposedly) does not know a Hamiltonian circuit for some graph G – to obtain a convincing proof for the GUCZK protocol.

The attack from (YYZ09a) is as follows:

1 Static corruption: the adversary $\mathcal{A}$ corrupts P (the prover of the GUCZK protocol) and V' (the verifier of the protocol from Fig. 4) at the onset of computation.

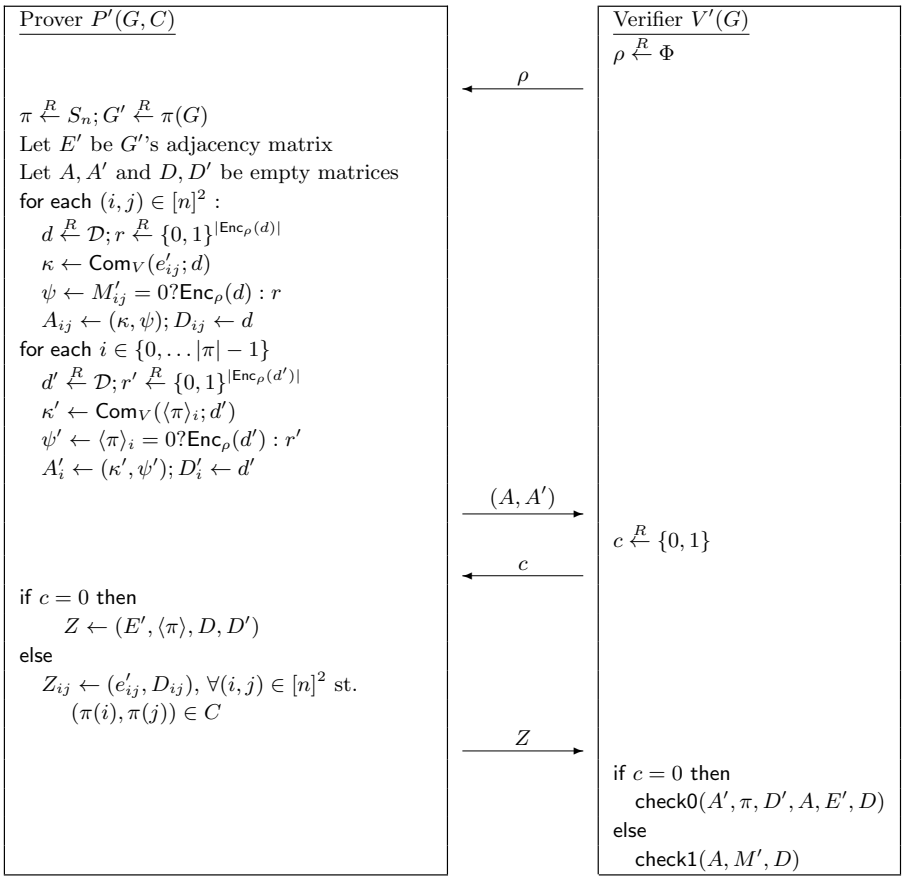
Fig. 3. The GUCZK implied by UAIBC. The common input is a graph $G = (V, E)$, where $V = [n]$ and E is a n^2-matrix whose entries e_{ij} lay in $\{0, 1\}$. The secret prover's input is a Hamiltonian Circuit C for G. In the protocol, the permutation π is uniformly chosen at random from S_n, the set of permutations over $[n]$, and $\langle\pi\rangle \in \{0,1\}^*$ is a binary representation of π. In the verifier's description, we use two functions for checks: function check0 checks if value A' opens to π and D' (bit by bit), then checks if value A opens to E' and D, and finally that $\pi(G) = G'$. Similarly function check1, checks if each A_{ij} opens to M'_{ij}, D_{ij}, and if they encode a n-cycle.



2 In the execution of the GUCZK protocol (which we call "the first session"), $\mathcal{A}$ works just as the honest prover does until Phase-1 (i.e., the coin-tossing) finishes. Denote by ρ the output of the coin-tossing in the first session. Adversary $\mathcal{A}$ suspends the first session.

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

Fig. 4. Adversarial protocol from (YYZ09a).



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

The previous adversary is supposed to allow an environment $\mathcal{Z}$ to distinguish an execution of the GUCZK protocol from an execution of $\mathcal{F}_{\mathsf{zk}}$ for any simulator $\mathcal{S}$. Although not explicitly given in (YYZ09a), the implied environment $\mathcal{Z}$ from (YYZ09a) is as follows:

1 Activate P' (the prover from the second session) with input (G, C), where G is a graph and C is a Hamiltonian circuit for G.
2 If V (the verifier from the first session) outputs $(\texttt{ZK-proof}, P, V, G)$, then output 1.

We remark that it is crucial that the Hamiltonian Circuit is given as input by the environment and not the adversary, because otherwise the claim *"$\mathcal{A}$ can successfully finish the execution of the first session, but without knowing any corresponding NP witness to the statement being proved in the first session"* is meaningless.

We conclude that the claim of (YYZ09a) can be rephrased as *there does not exist simulator $\mathcal{S}$ that makes $\mathcal{Z}$ output 1*, because C, the witness for G such that $(C, G) \in \mathcal{R}$, remain hidden to $\mathcal{S}$.

On the role of identities. In the first session, the role of prover as well as of verifier is given by the identities. With this information, an honest verifier will compute IBTC with identity P. The fact P is corrupted allows the simulator to simulate and then equivocate the commitment computed by V with identity P in the Coin Tossing (CT) protocol. On the other hand, in the second session, the identities univocally define V as the verifier (although this entity is actually V', but this is a maliciously constructed protocol). This role assignment is unavoidable as otherwise V will fail when checking the validity of commitments (computed by P') received after the CT protocol.

3.3. *Mapping the attack from FUC to GUC*

The attack from (YYZ09a) is designed in the FUC framework. We recall that, there, the adversary is claimed to be more powerful since *"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."*. Furthermore, is claimed that *"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."*

However, at least for the attack from (YYZ09a), the additional power of the adversary seems of no real use in the following sense. Given the previous FUC adversary $\mathcal{A}$, the previous FUC environment $\mathcal{Z}$, and any FUC simulator $\mathcal{S}$, one can construct a GUC adversary $\mathcal{A}'$, a GUC environment $\mathcal{Z}'$, and a GUC simulator $\mathcal{S}'$ such that the output of $\mathcal{Z}$ and $\mathcal{Z}'$ follows exactly the same distribution when executed with the GUCZK protocol or $\mathcal{F}_{\mathsf{zk}}$.

The GUC adversary $\mathcal{A}'$ internally simulates $\mathcal{A}$ and all communication with external entities is forwarded to the corresponding external entity in the GUC execution, except communication from P' to V' which is received from $\mathcal{Z}'$ (instead of directly from P'). The environment $\mathcal{Z}'$ internally simulates $\mathcal{Z}$ and also internally simulates P' with input (G, C) and forwards all communication from P' to $\mathcal{A}'$ (and communication with external entities is forwarded by $\mathcal{Z}'$). The simulator $\mathcal{S}'$ internally simulates $\mathcal{S}$ and communication from P' to V' is received from $\mathcal{Z}'$ (and communication with external entities is forwarded by $\mathcal{S}'$).

Note that in execution of GUCZK with $\mathcal{Z}, \mathcal{A}$ or $\mathcal{Z}', \mathcal{A}'$, as well as execution of $\mathcal{F}_{\mathsf{zk}}$ with $\mathcal{Z}, \mathcal{S}$ or $\mathcal{Z}', \mathcal{S}'$, the only difference is that in FUC party P' is an independent ITM, while in GUC executions is simulated by $\mathcal{Z}'$. However, in the GUC executions, messages from P' are computed exactly in the same way as in FUC executions. Therefore, in GUC executions, all input received by the simulated $\mathcal{Z}$s follows exactly the same distribution as in the FUC executions. We conclude that the output of $\mathcal{Z}$ and $\mathcal{Z}'$ follow exactly the same distribution in each of the different executions.

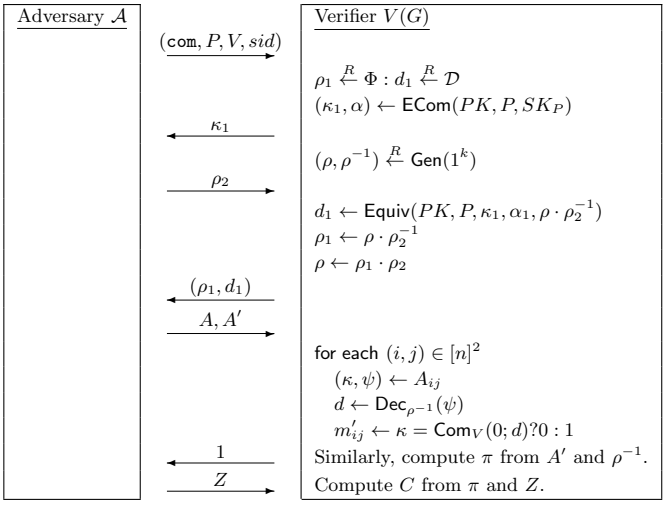
From here onwards, we focus on the attack in the GUC framework since, as we have shown, it has the same effect as the attack in the FUC model.

3.4. *A simulator for Yao's attack*

In (YYZ09a) is claimed that "$\mathcal{A}$ can successfully finish the execution of the first session, but without knowing any corresponding NP witness to the statement being proved in the first session". We claim that $\mathcal{A}$ definitively knows the witness for the statement being proved, and there exists a simulator $\mathcal{S}$ who is able to extract the witness used by P' by simply interacting with $\mathcal{A}$.

Given that P is corrupted, $\mathcal{S}$ retrieves SK_P from $\bar{\mathcal{G}}_{\mathsf{acrs}}$ and internally simulates $\mathcal{A}$ controlling corrupted P and internally simulates the honest verifier V. The key observation is that the knowledge of SK_P allows the simulator to equivocate the IBTC with identity P, thus the outcome of the CT can be set to a value ρ for which the simulator knows ρ^{-1}. The knowledge of ρ^{-1} allows to extract the Hamiltonian circuit used by the prover P' of the second session. The detailed description of the simulator is given in Fig. 5.

Fig. 5. Simulated adversary and simulated verifier.



On adaptive corruptions: The attack of Yao et al. only considers static corruptions. We can attempt to augment Yao et al.'s attack considering adaptive corruptions. Specifically one might consider an adversary that corrupts P, the prover in the first session, only after it receives the commitment κ_1 from V. In this case, the simulator is no longer able to equivocate, because it does not know the secret key of the prover.

But the UAIBC protocol is executed assuming secure channels, thus the simulator needs to send κ_1 only when the adversary corrupts P. In this case, the simulator knows the secret key of the prover when κ_1 is computed, and thus, is able to equivocate it.

4. Conclusion

Yao et al. presented potential limitation of the general feasibility for GUC proposed in (CDPW07). They claimed that the $\bar{\mathcal{G}}$-subroutine respecting limitation on state informa-

tion sharing is still unrealistic to reflect adversarial activities happening in asynchronous networks like the Internet. They argue that the inputs of an honest player could be maliciously dependent on the transcript of the challenge protocol and on the whole transcript of the external protocols. They refer to this as *unpredictable* environment and adversary, and they claimed that the general GUC feasibility proposed in (CDPW07) is w.r.t. *predictable* environment and adversarial strategies.

We note that the $\bar{\mathcal{G}}$-subroutine respecting limitation is rather a design decision. For example, if the designer of the challenge protocol decides that the protocol should share state via other way than $\bar{\mathcal{G}}$, then he designer need to assume that the protocol is $\bar{\mathcal{G}}, \bar{\mathcal{G}}'$-subroutine respecting for some shared functionality $\bar{\mathcal{G}}'$. Perhaps this modeling might have reduced expressive power (and it is interesting to find an example not expressible in this way), but we could not find an example of this on Yao et al.'s work.

We also note that the *unpredictable* environment an adversary strategy is already expressible on GUC. Certainly, there is no impediment for an environment to give inputs to the challenged protocol that depend not only on the challenge protocol but also on the whole transcript of external protocols. Furthermore, such strategy can be simulated by an environment an adversary executed on EUC (CDPW07, Thm. 1), which implies that the unpredictable strategy can be derived only from queries to the shared functionality and the ideal functionality.

In conclusion, Yao et al. also showed a concrete attack on the GUCZK protocol implied by the general GUC feasibility of (CDPW07) arguing that it could lose concurrent general composability and proof of knowledge properties. We showed that the attack is harmless because it can be simulated using the techniques of Canetti et al. (CDPW07).

We praise Yao et al.'s motivation of studying the GUC framework and the general feasibility of GUC, yet we believe that the question of identifying and characterizing the exact limits of the framework is still an open question.

References

Blum, M. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, volume 1, page 2, 1986.

Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

Canetti, R. Obtaining universally composable security: Towards the bare bones of trust. In *Advances in Cryptology–ASIACRYPT 2007*, pages 88–112. Springer, 2007.

Canetti, R., Dodis, Y., Pass, R., and Walfish, S. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.

Canetti, R. and Fischlin, M. Universally composable commitments. Cryptology ePrint Archive, Report 2001/055, 2001. http://eprint.iacr.org/.

Canetti, R., Kushilevitz, E., and Lindell, Y. On the limitations of universally composable two-party computation without set-up assumptions. *JCRYPTOL: Journal of Cryptology*, 19, 2006.

Canetti, R., Lindell, Y., Ostrovsky, R., and Sahai, A. Universally composable two-party and multi-party secure computation. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2002.

De Santis, A. and Persiano, G. Zero-knowledge proofs of knowledge without interaction. In IEEE, editor, *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 427–436, Pittsburgh, PN, October 1992. IEEE Computer Society Press.

Lindell, Y. General composition and universal composability in secure multi-party computation. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

Pass, R. On deniability in the common reference string and random oracle model. In *CRYPTO: Proceedings of Crypto*, 2003.

Walfish, S. *Enhanced security models for network protocols*. PhD thesis, New York University, New York, NY, USA, 2008. AAI3310580.

Yao, A.C.C., Yao, F. F., and Zhao, Y. A note on the feasibility of generalized universal composability. In Jin-Yi Cai, S.Barry Cooper, and Hong Zhu, editors, *Theory and Applications of Models of Computation*, volume 4484 of *Lecture Notes in Computer Science*, pages 474–485. Springer Berlin Heidelberg, 2007.

Yao, A.C.C., Yao, F. F., and Y. Zhao. A note on the feasibility of generalised universal composability. *Mathematical Structures in Computer Science*, 19(1):193–205, 2009.

Yao, A.C.C., Yao, F. F., and Y. Zhao. A note on universal composable zero-knowledge in the common reference string model. *Theor. Comput. Sci*, 410(11):1099–1108, 2009.