On the (Non-)Equivalence of UC Security Notions

Oana Ciobotaru

Saarland University

Abstract. Over the years, various security notions have been proposed in the classical model where protocol participants are assumed either honest or arbitrarily malicious. Recently, the study of security notions has been extended towards comparing notions such as weak precise secure computation with the game-theoretic notion of universal implementation of a mediator.
Even though the study of implication relations among various security notions has been very prolific, important open problems remain. For example, for almost a decade it was not known [25] whether the notions of specialized simulator universally composable security and 1-bit specialized simulator universally composable security are equivalent or not. Another open problem [19,20] is finding security notions stronger than weak precise secure computation such that they are equivalent with appropriately defined game-theoretic notions. In this work, we give an answer to each of the above mentioned open problems.

1 Introduction

Nowadays we rely more and more often for everyday tasks on security protocols. Moreover, the number of contexts where the use of security protocols is required by law or expected by users has also grown rapidly in recent years. A wide range of security properties have been defined and implemented into real-world systems, but so far there is no unique notion that fulfills all requirements: For example, a given notion may ensure strong security guarantees, but comes at the price of inefficiency or it offers good scalability in practice, but there are scenarios where it is too permissive. In order to ensure the most appropriate security notion is chosen when designing a system that has security as one of its features, one should know very well how various security notions relate to each other.

Recently the view on security definitions has been extended [20] with the incipient study of the equivalence relation between weak precise secure computation and a weak variant of the game-theoretic notion of universal implementation for a trusted mediator. However, it is still left as an open problem [20,19] how to obtain such a comparisons for other, possibly stonger security notions.

1.1 Contribution

In this work we have a three fold contribution.

First, we relate the notion of weak stand-alone security[1] to the emerging game-theoretic concept of universal implementation [20,19]. In contrast to previous work, for our result we use a variant of universal implementation that discards the cost of computation. We are able to answer positively the open question from [20,19] regarding the existence of game-theoretic concepts that are equivalent to cryptographic security notions where the simulator does not depend on both the input distribution and the distinguisher.

Second, we study the propagation of weak security notion through the hierarchy security definitions. More precisely, we show that the notion weak security composed under concurrent general composition is equivalent to 1-bit specialized simulator UC security, which is a variant of UC security. Together with our first result, this implies that weak stand-alone security and stand-alone security are not equivalent.

Third, we present a separation result between two variants of UC security: 1-bit specialized simulator UC security and specialized simulator UC security. This solves an open question from [25] and comes in contrast with the well known equivalence result between 1-bit UC security and UC security [5]. Both variants of the UC security notion are obtained from the UC security definition by changing the order of quantifiers[2]. Thus, we continue the line of study started by [8,25]. In order to obtain the separation, we first show that the 1-bit specialized simulator UC security is equivalent to a seemingly weaker version of security, namely weak specialized simulator UC security[3].

[1] The difference between stand-alone security and weak stand-alone security is in the order of quantifiers. For stand-alone security, the simulator is universally quantified over all distinguishers and input distributions. As detailed in section 2, for our notion of weak security the simulator depends only on the distinguisher and not on the input distribution. This comes in contrast with [20], where the simulator for weak precise secure computation depends on both distinguisher and input distribution.

[2] This means that in contrast to the UC security definition, the simulator may depend on the environment.

[3] This notion, additionally to having the simulator depend on the environment, also has the simulator depend on the distinguisher that compares the views of the environment from the real and the ideal world.

The main proof technique used in our separation result is to employ a cryptographic tool called time-lock puzzles. Intuitively, this cryptographic tool can be used for comparing the computational power of two different polynomially bounded Turing machines. In order to achieve the separation result, we use time-lock puzzles from which we derive a result interesting also on its own, mainly a construction of a one-way function and a hard-core predicate.

1.2 Background and Related Work

The initial work [39] on general security definitions highlighted the need for a framework expressing security requirements in a formal way. The first formal definition of *secure computation* was introduced by Goldreich et al.[13]. The first approaches for formally defining security notions [15,16] have taken into account only the stand-alone model. In this model, the security of the protocol is considered with respect to its adversary, in isolation from any other copy of itself or from a different protocol. However, there are simple protocols [9] that fulfill stand-alone security, but are no longer secure under parallel or concurrent composition.

Micali and Rogaway [30] introduce the first study of protocol composition, which the authors call *reducibility*. The first security definition expressed as a comparison with an ideal process, as well as the corresponding sequential composition theorem for the stand-alone model are provided in [3]. A general definition of security for evaluating a probabilistic function on the parties' inputs is given in [4]. It is shown that security is preserved under a subroutine substitution composition operation, which is a non-concurent version of universal composition: Only a single instance of the protocol is active at any point in time.

The framework of *universally composable security*, for short UC security [5] allows for specifying the requirements for any cryptographic task and within this framework protocols are guaranteed to maintain their security even in the presence of an unbounded number of arbitrary protocol instances that run concurrently in an adversarially controlled manner.

The notion of *specialized simulator UC* security has been introduced in [25] and it was shown that this is equivalent to *general concurrent composability* when the protocol under consideration is composed with one instance of any possible protocol. Changing the order of quantifiers in the context of security definitions has been previously used in [8,19,20] for strengthening or weakening given security notions. A more detailed review about the existing implication relations among different security notions can be found in section 5.

In parallel with the UC framework, the notion of reactive security has been developed [33,21,32,34,35]. The framework addresses for the first time *concurrent* composition in a computational setting: it is shown that security is preserved when a single instance of a subroutine protocol is composed concurrently with the calling protocol. The framework has been extended in [2] to deal with the case where the number of parties and protocol instances depends on the security parameter. More about the differences between reactive simulatability and universal composability notions can be read in the related work section from [5].

Our study of the relation between security and game-theoretic notions has been triggered by the recently emerging field of rational cryptography, where users are assumed to only deviate from a protocol if doing so offers them an advantage. Rational cryptography is centered around (adapted) notions of game theory such as computational equilibria [7]. A comprehensive line of work already exists developing novel protocols for cryptographic primitives such as rational secret sharing and rational secure multiparty computation [1,10,11,17,18,24].

Historically, game theory and its computational aspects have been first studied in more detail in [31] (i.e., players are modeled as finite automata) and in [28] (players are defined as Turing machines). Later, [7,38] study the rational cryptographic problem of implementing mediators using polynomially bounded Turing machines. Another direction, [19,20,37] considers that computation is costly for players and investigates how this affects their utilities and the design of appropriate protocols. In [37], a player's strategy is defined as a finite automaton whose complexity (i.e., number of states) influences players utilities. In [20,19] similar considerations are made: both the input and the complexity of the machine (which is a Turing machine this time) are taken into account. This complexity can be interpreted, for example, as the running time or the space used by the machine for a given input. Their work develops a game-theoretic notion of protocol implementation and they show a special case of their definition is equivalent to a weak variant of precise secure computation.

1.3 Organization

This work is structured as follows: In section 2 we review security notions and in section 3 we revise the game-theoretic notion of universal implementation. In section 4 we prove our separation result between specialized simulator UC security and 1-bit specialized simulator UC security. In section 5 we show our equivalence relation between weak security under 1-bounded concurrent general composition and 1-bit specialized simulator UC security. In section 5.1 we present the equivalence between our weak security notion and the game-theoretic notion of strong universal implementation. In section 6 we present out conclusions and future directions.

2 Review of Security Notions

In this work we consider all parties and adversaries run in polynomial time in the security parameter k and not in the length of input. In this section we review two models of security under composition: concurrent general composition and universal composability. Both frameworks require the notion of (computational) indistinguishability given below.

Definition 1 (Computational Indistinguishability). *We call distribution ensembles* $\{X(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ *and* $\{Y(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ *computationally indistinguishable and we write* $X \equiv Y$*, if for every probabilistic polynomial time interactive Turing machine (PPITM)* $\mathcal{D}$ *there exists a function* ϵ*, negligible in* k*, and* k_0 *such that for every* $z \in \{0,1\}^*$

$$|(Pr(\mathcal{D}(X(k,z)) = 1) - (Pr(\mathcal{D}(Y(k,z)) = 1)| < \epsilon(k),$$

for every $k \geq k_0$*.*

In the following, we call PPITM $\mathcal{D}$ *a* distinguisher*.*

A variant of this definition, which we call *indistinguishability with respect to a given adversary* $\mathcal{D}$ and we denote by $\stackrel{\mathcal{D}}{\equiv}$, is analogous to the definition above, where "for every probabilistic distinguisher $\mathcal{D}$" is replaced with "for distinguisher $\mathcal{D}$". Such a definition will be used in relation with our notion of weak security.

Definition 2 (Indistinguishability with respect to a Given Distinguisher). *We say the following ensembles* $\{X(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ *and* $\{Y(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ *are computationally indistinguishable with respect to a given PPITM* $\mathcal{D}$ *and we write* $X \stackrel{\mathcal{D}}{\equiv} Y$*, if there exists a function* ϵ*, negligible in* k *and* k_0 *such that for every* $z \in \{0,1\}^*$

$$|(Pr(\mathcal{D}(X(k,z)) = 1) - (Pr(\mathcal{D}(Y(k,z)) = 1)| < \epsilon(k),$$

for every $k \geq k_0$*.*

2.1 Universal Composability

The standard method for defining security notions is by comparing a real world protocol execution to an ideal world process execution. In the real world execution, a protocol interacts with its adversary and possibly with other parties. In the ideal world execution, an idealized version of the protocol (called ideal functionality) interacts with an ideal world adversary (usually called simulator) and possibly with other parties. The ideal functionality is defined by the security requirements that we want our protocol to fulfill.

On an intuitive level, given an adversary, the purpose of the simulator is to mount an attack on the ideal functionality; any PPITM distinguisher may try to tell apart the output of the interaction between the ideal functionality and the simulator and the output of the interaction between the protocol and its adversary. If for every adversary, a simulator exists such that the two outputs cannot be told apart by any PPITM distinguisher, then our initial protocol is as secure as the ideal functionality, with respect to what is called *the stand-alone model*.

Definition 3 (Stand-alone Security). *Let ρ be a protocol, $\mathcal{F}$ an ideal functionality and k a security parameter. We say ρ* securely implements $\mathcal{F}$ *if for every PPITM real-model* adversary $\mathcal{A}$ *there exists a PPITM ideal-model* simulator $\mathcal{S}$ *such that for every protocol input x and for every auxiliary input z given to the adversary with $x, z \in \{0,1\}^{poly(n)}$, we have*

$$\{IDEAL^{\mathcal{F}}_{\mathcal{S}}(k,x,z)\}_{k\in\mathbb{N}} \equiv \{REAL_{\rho,\mathcal{A}}(k,x,z)\}_{k\in\mathbb{N}}.$$

By $IDEAL^{\mathcal{F}}_{\mathcal{S}}(k,x,z)$ we denote the output of $\mathcal{F}$ and $\mathcal{S}$ after their interaction and $REAL_{\rho,\mathcal{A}}(k,x,z)$ denotes the output of ρ and adversary $\mathcal{A}$ after their interaction.

If in Definition 3 we allow the simulator to depend also on the distinguisher, we obtain the notion of *weak stand-alone security.* More formally, we have the following definition:

Definition 4 (Weak Stand-alone Security). *Let ρ be a protocol, $\mathcal{F}$ an ideal functionality and k a security parameter. We say ρ* computes $\mathcal{F}$ with respect to weak security *if for every PPITM real-model* adversary $\mathcal{A}$ *and for every PPITM distinguisher $\mathcal{D}$ there exist a PPITM simulator $\mathcal{S}$ such that for every $x, z \in \{0,1\}^{poly(n)}$ we have*

$$\{IDEAL^{\mathcal{F}}_{\mathcal{S}}(k,x,z)\}_{k\in\mathbb{N}} \stackrel{\mathcal{D}}{\equiv} \{REAL_{\rho,\mathcal{A}}(k,x,z)\}_{k\in\mathbb{N}}.$$

Sometimes, for brevity of notation, we compact the definition above into the relation:

$$\{IDEAL(k, \mathcal{S}, \mathcal{F})\}_{k \in \mathbb{N}} \stackrel{\mathcal{D}}{\equiv} \{REAL(k, \rho, \mathcal{A})\}_{k \in \mathbb{N}}. \quad (1)$$

There are examples [9] of protocols secure in the stand-alone model that do not remain secure even when two of its instances run concurrently. More stringent security definitions take into account that a protocol interacts not only with its adversary, but also with other (possibly polynomially many) protocols or with (polynomially many) copies of itself. This is intuitively captured by the universal composability (UC) security framework [5].

The definition of universal composability follows the paradigm described above, however it introduces an additional adversarial entity which is called environment. The environment, usually denoted by $\mathcal{Z}$, is present in both the UC real world and UC ideal world. The environment represents everything that is external to the current execution of the real-world protocol or to the ideal functionality.

The main difference between the execution of UC real and UC ideal world, is that in the latter the ideal functionality cannot be directly accessed by the environment. Parties involved in the ideal execution give their inputs to the ideal functionality which computes some outputs and sends back these values. Since the ideal world parties perform no computation they are called the dummy parties for the ideal functionality. The ideal $\mathcal{F}$ together with its corresponding dummy parties represent an ideal process. The adversary for the protocol is not considered to be a part of the environment, but it could be controlled by the environment.

In order to determine whether a protocol securely implements a given task, first we define the ideal process for carrying out that task. Intuitively, in an ideal process for a given task, all parties give their inputs directly to the *ideal functionality for that task* which can be regarded as a formal specification of the security requirement of the task. According the universal composability security definition, a protocol securely implements a task if any damage that can be caused by an adversary while interacting with the protocol and the environment, can also be caused by an adversary interacting with the ideal process for that task and the environment. Intuitively, the entity assessing the amount of damage is the environment. Since there is no damage we can cause to the ideal functionality, the protocol considered must also be secure. We say that the protocol runs in a real-world model and the ideal functionality runs in the ideal-world model.

Real-world Protocols More formally, let ρ be a cryptographic protocol. The real-world model for the execution of protocol ρ contains the following PPITMs: a PPITM $\mathcal{Z}$ called the environment, a set of PPITMs representing the parties running the protocol ρ and an adversary PPITM $\mathcal{A}$. We now have a more detailed look at each of these PPITMs and their interaction.

The environment $\mathcal{Z}$ represents everything that is external to the current execution of ρ and it is modeled as an PPITM with auxiliary input. Throughout the course of the protocol execution, the environment can provide inputs to parties running ρ and to the adversary. These inputs can be a part of the auxiliary input of $\mathcal{Z}$ or can be adaptively chosen by the environment. Also $\mathcal{Z}$ receives all the outputs that are generated by the parties and the adversary. The only interaction between the environment $\mathcal{Z}$ and the parties is when the environment sends the inputs and receives the outputs. Finally, at the end of the execution of ρ, the environment outputs all the messages received.

The adversary can receive inputs from $\mathcal{Z}$ at any moment during the protocol execution and it can send replies to $\mathcal{Z}$ at any time. In order to capture any possible adversarial behaviour, $\mathcal{A}$ and $\mathcal{Z}$ can communicate freely throughout the course of the protocol and they can exchange information after any message sent between the parties and after any output made by a party.

Next, we look at the notion of corruption. By considering a PPITM P corrupted we mean that from that point on that adversary has access to all the inputs and communication messages send or received by P, and for any communication model, $\mathcal{A}$ can decide to alter such messages in any way it wants. Moreover, all the past incoming or outgoing messages of P are known to $\mathcal{A}$.

In order for $\mathcal{A}$ to corrupt a PPITM P, it first informs $\mathcal{Z}$ by sending it a corruption message $(\mathit{corrupt}, P)$. Thus $\mathcal{Z}$ is aware at any given moment about the corruption state of all PPITMs. Depending on the moment when the adversary $\mathcal{A}$ can corrupt a PPITM, there are two corruption models: static and adaptive. In the static corruption model, the adversary $\mathcal{A}$ is allowed to corrupt only in the beginning of the protocol, before the respective PPITMs receive their inputs from $\mathcal{Z}$. In contrast, if $\mathcal{A}$ is allowed to corrupt at any given moment during the protocol execution, then the adversary is called adaptive. Another way to look at the corruption model is by inspecting whether the adversary is passive, (i.e., only learns all inputs and communication messages a corrupted PPITM sends and receives), or if $\mathcal{A}$ is active. The latter case implies $\mathcal{A}$ is allowed to modify any input a corrupted PPITM gets and also any communication message sent.

In order to simplify the presentation, we use an equivalent definition for the static corruption model. As in the standard static case, the moment of corruption is fixed in the beginning, we can skip sending and receiving the corruption messages. Instead, we assume the corrupted PPITMs are fixed from the start and the adversary does not have to choose them. Then, the previous static adversary definition is equivalent to the latter formulation, which we use in this work.

Besides corruption, the adversary may interfere with the communication between honest parties. The most basic UC model ensures that all messages are handed to the adversary and the adversary delivers messages of its choice to all PPITMs. This model makes no assumption on the communication properties: authenticity, secrecy or synchrony of the messages delivered. For the more specialised models of authenticated, secure or synchronous communication, an ideal functionality is added to the basic model to capture the respective properties.

Authenticated communication assumes the adversary cannot alter content of messages without being detected. The synchronous communication model captures the property that messages are all delivered and without delay from the moment they were generated. The ideally secure communication model assumes the adversary receives all messages, but it has neither access to the content of communication, nor possibility to modify any message without breaking authenticity. In this model, the adversarial capabilities are limited to either delaying or not delivering some or all messages between the uncorrupted PPITMs.

When the protocol execution ends, $\mathcal{Z}$ outputs its view of that execution. This view contains messages that $\mathcal{Z}$ has received from the adversary $\mathcal{A}$ and outputs of all other PPITMs. Formally, $EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k, z)$ denote the output of $\mathcal{Z}$ in an execution of the protocol ρ with adversary $\mathcal{A}$ and environment $\mathcal{Z}$, where k is the security parameter and z is the auxiliary input to the environment $\mathcal{Z}$. We denote by $EXEC_{\rho,\mathcal{A},\mathcal{Z}}$ the family of random variables $\{EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k, z)\}_{k\in\mathbb{N}}$.

Ideal Process and Ideal Functionalities In order to formalize the ideal process, we do not want to define a different model, but we rather need to adapt to the one above. In the same way as in the real-world, the environment $\mathcal{Z}$ is the only PPITM that can send inputs at any moment to the ideal process parties and to the ideal adversary. In the case of the ideal process, the adversary is called the ideal simulator and is commonly denoted by $\mathcal{S}$. Moreover, $\mathcal{Z}$ receives all the outputs generated by the parties, as well the possible outputs of $\mathcal{S}$.

The first difference is that in the ideal model there exists a trusted party, the ideal functionality, that cannot be directly accessed by the environment. This works as follows: PPITMs involved in the ideal process give their inputs to the ideal functionality which computes outputs for each party and sends these values to them. Hence, the role of the ideal functionality is to receive inputs, perform computations and send results to the ideal PPITMs. As these PPITMs do not take an active role in the computation and just send inputs to and receive outputs from the ideal functionality, they are called dummy parties of the ideal functionality.

The second difference with the real-world model is that messages delivered by the adversary to dummy parties are ignored. In the ideal protocol the adversary sends corruption messages directly to the ideal functionality. The ideal functionality then determines the effect of corrupting a party. A typical response is to let the adversary know all the inputs received and outputs sent by the party so far.

The environment $\mathcal{Z}$ and the simulator $\mathcal{S}$ can communicate freely during the execution of the ideal process. Additionally, the ideal functionality informs the simulator every time it wants to output a message. If the simulator agrees, then the respective output is made. This is required by the UC ideal model in order to allow $\mathcal{S}$ to simulate the behavior of a UC real world adversary delaying messages or not sending some or all of the communication among real-world protocol PPITMs.

Similar to the real-world model, the environment $\mathcal{Z}$ outputs its view in the end of the ideal process execution. The view contains all the messages received from the simulator as well as all the messages that the dummy parties output to $\mathcal{Z}$. More formally, by $EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)$ we denote the output of $\mathcal{Z}$ in an execution of the ideal process with the trusted party $\mathcal{F}$, simulator $\mathcal{S}$ and environment $\mathcal{Z}$, where k is the security parameter and z is the auxiliary input to the environment $\mathcal{Z}$. We denote by $EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ the family of random variables $\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}}$.

Protocol Emulation We now define what it means that a real-world protocol ρ *emulates* with respect to UC security an ideal functionality $\mathcal{F}$. The environment $\mathcal{Z}$ is the PPITM deciding whether the interaction with the protocols and their respective adversaries can be distinguished.

All the PPITMs used in either of the protocol executions for ρ or $\mathcal{F}$, including the environment $\mathcal{Z}$, are computationally bounded. Thus, it is sufficient if we formalize the notion of emulation in terms of computational indistinguishability. The environment $\mathcal{Z}$ will act as a distinguisher for the two protocol executions. Since all the information $\mathcal{Z}$ gains throughout its interaction is contained within the view $\mathcal{Z}$ outputs in the end, it is sufficient to compare the two views. Essentially, protocol ρ emulates $\mathcal{F}$ if for every adversary $\mathcal{A}$ there is an ideal simulator $\mathcal{S}$ such that for every environment $\mathcal{Z}$ the views of the two interactions are computationally indistinguishable.

Definition 5 (UC Security). *Let ρ be a protocol and $\mathcal{F}$ an ideal functionality. We say that ρ UC securely emulates $\mathcal{F}$ if for every PPITM adversary $\mathcal{A}$ there is a PPITM simulator $\mathcal{S}$ such that for every PPITM distinguisher $\mathcal{Z}$ and for every input $z \in \{0,1\}^*$, the two families of random variables $\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}}$ and $\{EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}}$ are computationally indistinguishable.*

In the following we also use a relaxed version of this definition, where the order of quantifiers between the environment and the ideal-world simulator is reversed [25].

Definition 6 (Specialized Simulator UC Security). *Let ρ be a protocol and $\mathcal{F}$ an ideal functionality. We say that ρ emulates $\mathcal{F}$ under specialized simulator UC security if for every probabilistic polynomial time adversary $\mathcal{A}$ and for every environment $\mathcal{Z}$, there exists a simulator $\mathcal{S}$ such that for every distribution of auxiliary input $z \in \{0,1\}^*$, we have:*

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}} \equiv \{EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}}$$

It had been shown [22] that the two notions defined above are not equivalent. In the above definition, the output of the environment is considered to be a string of arbitrary length. If the only change we make to the above definition is to consider environments that have a 1-bit output, we obtain the notion of *1*-bit specialized simulator UC security. It has been an open problem [25] whether considering only environments with one bit output would produce an equivalent definition. In this work we show how to separate the notions of specialized simulator UC security and 1-bit specialized simulator UC security.

Definition 7 (1-bit Specialized Simulator UC Security). *Let ρ be a protocol and $\mathcal{F}$ an ideal functionality. We say that ρ emulates $\mathcal{F}$ under 1-bit specialized simulator UC security if for every probabilistic polynomial time adversary $\mathcal{A}$ and for every 1-bit output environment $\mathcal{Z}$, there exists a simulator $\mathcal{S}$ such that for every input $z \in \{0,1\}^*$, we have:*

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}} \equiv \{EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}}.$$

If in the specialized simulator UC definition we let the simulator also depend on the distinguisher who is the only PPITM to establish whether the output of the executions in the real UC world and ideal UC world cannot be told apart, then we obtain the notion of *weak specialized simulator UC security.*

Definition 8 (Weak Specialized Simulator UC Security).
Let ρ be a protocol and $\mathcal{F}$ an ideal functionality. We say that ρ emulates $\mathcal{F}$ under weak specialized simulator UC security if for every PPITM adversary $\mathcal{A}$, for every PPITM environment $\mathcal{Z}$ and for every PPITM distinguisher $\mathcal{D}$, there exists a PPITM simulator $\mathcal{S}$ such that for every distribution of input $z \in \{0,1\}^*$, we have:

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}} \stackrel{\mathcal{D}}{\equiv} \{EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}}.$$

In the revised version of [5] there is an extension of the UC model we reviewed above. This extension mainly considers that PPT machines run in time polynomial in both the security parameter and the length of the input. While the extended model is seemingly more expressive in terms of adversarial attacks, it does not allow for fine grained separation between security notions (e.g., the separation result from [22] does not hold in the extended UC model). Another reason for choosing the original model is that, as it will be detailed in section 5, most of the UC results have been obtained in this model.

2.2 Weak Security under 1- bounded Concurrent General Composition

Given a security notion there are two approaches to ensure the security properties of a protocol under composition. One way is to prove that the security property defined for the stand-alone case is preserved under composition. The other way is to define the security notion for the protocol directly under composition. The latter approach has the benefit that it captures the security property without having the drawback of a possible very strong and thus very restrictive stand-alone definition. Due to this reason we will focus on the second approach.

The concurrent general composition has been introduced in [25]. In this security model, a protocol ρ that is being investigated is run concurrently, possibly multiple times, with an arbitrary protocol π. The protocol π can be any arbitrary protocol and intuitively, it represents the network activity around ρ. There is another way to look at this: one can consider protocol π to be the external protocol that gives inputs and reads the outputs of the internal protocol ρ. As π is arbitrary, it can call multiple instances of ρ. However, we consider that different instances run independently from one another. The only correlation between them are the inputs and outputs, in the following way: the inputs for a certain run of ρ that are provided by π might depend on the previous inputs and outputs given and collected by π. Also, the messages of π may be sent concurrently to the execution of ρ. This composition of π with ρ is denoted as in the original notation by π^ρ.

As in the case of universal composability, in order to give the definition of security for ρ under concurrent general composition, we need to compare the execution of ρ with that of an ideal functionality so we have to define the real and the ideal world.

The computation in the ideal world is performed among the parties of π and a trusted party, playing the role of ideal functionality $\mathcal{F}$. Thus the messages considered in the ideal world are standard messages between parties of π and ideal messages between π and $\mathcal{F}$. The protocol π is providing $\mathcal{F}$ with inputs and after performing necessary computations, $\mathcal{F}$ sends the results to parties of π. The ideal adversary is called a simulator, and as in the UC model, is denoted by $\mathcal{S}$. In addition to having full control over the parties it corrupts (see also the case of real world adversary), the simulator controls the scheduling of the messages between the parties of π and if not otherwise mentioned, it can also arbitrarily read and change messages. An exception is represented by the messages between π and $\mathcal{F}$: they are ideally secure, so the simulator can neither read nor change them. This comes in contrast with the standard definition of UC ideal protocol execution, where it is not enforced that the channels between the trusted parties and the rest of the participants are ideally secure.

During the computation, the honest parties follow the instructions given by π and in the end they output on their outgoing communication tape whatever value is prescribed by π. The corrupted parties output a special corrupted symbol and additionally the adversary may output an arbitrary image of its view. Let z be the auxiliary input for the ideal-world adversary $\mathcal{S}$ and let the inputs vector be $\bar{x} = (x_1, ..., x_m)$. Then the outcome of the computation of π with $\mathcal{F}$ in the ideal world (which we may also call $\mathcal{F}$-hybrid world) is defined by the output of all parties and $\mathcal{S}$ and is denoted by $\{HYBRID^{\mathcal{F}}_{\pi,\mathcal{S}}(k, \bar{x}, z)\}_{k\in\mathbb{N}}$.

The computation in the real world follows the same rules as the computation in the ideal world, only that this time there is no trusted party. Instead, each party of π has an PPITM that works as the specification of ρ for that party. Thus, all messages that a party of π sends to the ideal functionality in the ideal world are now written on the input tape of its designated PPITM. These PPITMs communicate with each other in the same manner as specified for the parties of ρ. After the computation is performed, the results are output by these PPITMs and the corresponding parties of π copy them on their incoming communication tapes. These messages are used by the parties of π in the same way as the messages output by $\mathcal{F}$ in the ideal-world. Similarly as above, in the real-world the adversary has full control over message delivery. There is one exception: any uncorrupted party of π can write and read directly to and from the input and respectively output tape of its designated PPITM without any interference from the adversary. Actually, the ideal adversary is not even aware of this taking place. This is similar to the UC communication between the environment and the real-world or ideal-world parties. Moreover, when we say that a real-world party is corrupted, we mean that a party of π and its corresponding PPITM are corrupted. This is not a restriction as an adversary that corrupts both a party of π and its PPITM can just fully control only one of them and let the other one follow its prescribed protocol.

Similarly to the ideal world, during the computation, the honest parties follow the instructions of π and their corresponding PPITM and in the end they output on their outgoing communication tape whatever value is prescribed by π. The corrupted parties output a special corrupted symbol and additionally the real-world adversary $\mathcal{A}$ may output an arbitrary image of its view. Let z be the auxiliary input for $\mathcal{A}$ and let the inputs vector be $\bar{x} = (x_1, ..., x_m)$. Then the outcome of the computation of π with ρ in the real world is defined by the output of all parties and $\mathcal{A}$ and is denoted by $\{REAL_{\pi^\rho,\mathcal{A}}(k, \bar{x}, z)\}_{k\in\mathbb{N}}$.

Independent of the world where the corruption takes place, the adversary could be static or adaptive. If the adversary is static, then the parties that are under the control of the adversary are fixed and do not depend on its auxiliary input or random tape. This is a restrictive definition of static corruption. However, the definition of adaptive corruption and the corresponding proof include the proof for a standard static corruption case. In the case of adaptive corruption, the adversary may decide during the protocol to arbitrarily corrupt a party, depending on the messages received so far. In both cases, once the adversary has corrupted a party then it learns all previous inputs and messages that the party received. From the moment of the corruption further on, the adversary has full control over the messages that the party sends. Moreover, we consider that the adversary fully controls the message scheduling: he decides if and when to deliver the messages between output tape of one party (or, more general, machine) to the input tape of another. As mentioned above, there is one exception: the adversary does not have any control over the messages that an uncorrupted party sends to its corresponding PPITM.

We are now ready to state the definition of security under concurrent general composition as in [25]. There are two notions of security under concurrent general composition: one for unbounded or polynomial calls that π may make to $\mathcal{F}$ and the second one, when π utilizes a fixed number of calls to $\mathcal{F}$.

Definition 9 (Security under Concurrent General Composition). *Let ρ be a protocol and $\mathcal{F}$ a functionality. Then, ρ* securely computes $\mathcal{F}$ under concurrent general composition *if for every probabilistic polynomial-time protocol π in the $\mathcal{F}$-hybrid model that utilizes ideals calls to $\mathcal{F}$ and every PPITM real-model adversary $\mathcal{A}$ for π^ρ, there exists a PPITM hybrid-model adversary $\mathcal{S}$ such that for every $\bar{x}, z \in \{0,1\}^*$:*

$$\{HYBRID^{\mathcal{F}}_{\pi,\mathcal{S}}(k, \bar{x}, z)\}_{k\in\mathbb{N}} \equiv \{REAL_{\pi^\rho,\mathcal{A}}(k, \bar{x}, z)\}_{k\in\mathbb{N}}.$$

If we restrict the protocols π to those that utilize at most ℓ ideal calls to $\mathcal{F}$, then ρ is said to securely compute $\mathcal{F}$ under ℓ-bounded concurrent general composition.

We also use a weak version of the security definition from above.

Definition 10 (Weak Security under Concurrent General Composition). *Let ρ be a protocol and $\mathcal{F}$ a functionality. Then, ρ* computes $\mathcal{F}$ under concurrent general composition with weak security *if for every probabilistic polynomial-time protocol π in the $\mathcal{F}$-hybrid model that utilizes ideals calls to $\mathcal{F}$, for every PPITM real-model adversary $\mathcal{A}$ for π^ρ and for every PPITM distinguisher $\mathcal{D}$, there exists a PPITM hybrid-model adversary $\mathcal{S}$ such that for every $\bar{x}, z \in \{0,1\}^*$:*

$$\{HYBRID^{\mathcal{F}}_{\pi,\mathcal{S}}(k, \bar{x}, z)\}_{k\in\mathbb{N}} \stackrel{\mathcal{D}}{\equiv} \{REAL_{\pi^\rho,\mathcal{A}}(k, \bar{x}, z)\}_{k\in\mathbb{N}}.$$

If we restrict the protocols π to those that utilize at most ℓ ideal calls to $\mathcal{F}$, then ρ is said to compute $\mathcal{F}$ under ℓ-bounded concurrent general composition with weak security.

3 Game-theoretic Definitions

In this section we define some game-theoretic concepts that we further need for establishing the equivalence between our notion of weak security and the strong univeral implementation notion given in [19] and redefined below.

A *Bayesian game* $\Gamma = (\{T_i\}_{i=1}^n, \{A_i\}_{i=1}^n, Pr, \{u_i\}_{i=1}^n)$, also called *a game with incomplete information*, consists of *players* $1, \ldots, n$. The incomplete information is captured by the fact that the *type* for each player i (i.e., its private information) is chosen externally, from a set T_i, prior to the beginning of the game. Pr is a publicly known distribution over the types. Each player has a set A_i of possible *actions* to play and individual *utility functions* u_i. Actions are played either simultaneously or sequentially; afterwards, every player i receives a *payoff* that is determined by applying its utility function u_i to the vector of types received in the game, i.e., *profile types*, and the actions played, i.e., *action profile*.

Recent work has extended the traditional notion of a game to the requirements of cryptographic settings with their probabilistically generated actions and computationally-bounded running times. The resulting definition – called *computational game* [23] – allows each player i to decide on a probabilistic polynomial-time, in the security parameter, interactive Turing machine M_i (short PPITM). The machine M_i is called the *strategy* for player i. The output of M_i in the joint execution of these interactive Turing machines denotes the actions played by participant i.

Definition 11 (Computational Game). *Let k be the security parameter and let $\Gamma = (\{T_i\}_{i=1}^n, \{A_i\}_{i=1}^n, Pr, \{u_i\}_{i=1}^n)$ be a Bayesian game. Then Γ is a* computational game *if the played action A_i of each participant i is computed by a PPITM M_i and if the utility u_i of each player i is polynomial-time computable.*

Because of the probabilistic strategies, the utility functions u_i now correspond to the expected payoffs. Thus, when there is no possibility for confusion, we overload the notation for u_i. However, when the utility we employ is not clear from the context, we denote by U_i the expected utility for party i.

Rationally behaving players aim to maximize these payoffs. In particular, if a player knew which strategies the remaining players intend to choose, he would hence pick the strategy that induces the most benefit for him. As this simultaneously holds for every player, we are looking for a so-called *Nash equilibrium*, i.e., a strategy vector where each player has no incentive to deviate from, provided that the remaining strategies do not change. Similar to the notion of a game, we consider a computational variant of a Nash equilibrium.

Definition 12 (Computational Nash Equilibrium). *Let Γ be a computational game, where $\Gamma = (\{T_i\}_{i=1}^n, \{A_i\}_{i=1}^n, Pr, \{u_i\}_{i=1}^n)$ and let k be the security parameter. A* strategy vector *(or* machine profile) consisting of PPITMs $\overrightarrow{M} = (M_1, \ldots, M_n)$ is a *computational Nash equilibrium* if for all i and any PPITM M_i' there exists a negligible function ϵ such that

$$u_i(k, M_i', \overrightarrow{M_{-i}}) - u_i(k, \overrightarrow{M}) \leq \epsilon(k)$$

holds.

Here $u_i(k, M_i', \overrightarrow{M_{-i}})$ denotes the function u_i applied to the setting where every player $j \neq i$ sticks to its designated strategy M_j and only player i deviates by choosing the strategy M_i'. In the definition above, we call M_i a *computational best response* to $\overrightarrow{M_{-i}}$.

Next, we define the *outcome* of a computational game as the transcript of all players' inputs and the actions each has taken. In contrast to strategy vectors, an outcome thus constitutes a finished game where every player can determine its payoff directly. A utility function is thus naturally defined on the outcome of a computational game: When applied to a strategy vector with its probabilistic choices, it describes the vector's expected payoff; when applied to an outcome of the game, it describes the exact payoff for this outcome.

We are now able to extend the definition of a computational game such that it takes into account which are the utilities of a group of players participating in the prescribed protocol, or deviating from it. In the rest of the paper we denote by Z the set of players participating in such a coalition and we denote by u_Z and U_Z respectively, the utility and the expected utility for such a coalition. We also denote for example by M_Z the vector of strategies (or the PPT ITMs) that the parties in Z run (or are controlled by).

The definition of computational Nash equilibrium can be extended to the notion of *computational Nash equilibrium with immunity with respect to coalitions.* In our current scenario, we require that the property in the definition of computational Nash equilibrium is fulfilled for all subsets Z of players, i.e., for all Z and all PPITM M_Z' controlling the parties in Z there exists a negligible function ϵ_Z such that $U_Z(k, M_Z', \overrightarrow{M_{-Z}}) - U_i(k, \overrightarrow{M}) \leq \epsilon_Z(k)$ holds.

So far we have assumed that players communicate only among each other. We extend a computational game to a *computational game with mediator*. The mediator is modeled by a PPITM denoted $\mathcal{F}$. Without loss of generality, we assume all communication passes between players and the trusted mediator (that can also forward messages among players).

Next we follow the approach from [20] to formalize the intuition that the machine profile $\overrightarrow{M} = (M_1, \ldots, M_n)$ implements a mediator $\mathcal{F}$ whenever a set of players want to truthfully provide a value (e.g., their input or type) to the mediator $\mathcal{F}$, they also want to run $\overrightarrow{M}$ using the same values. For each player i, let its type be $t_i = (x_i, z_i)$, where x_i is player's input and z_i is some auxiliary information about the state of the world.

Let $\Lambda^{\mathcal{F}}$ denote the machine that, given the type $t_i = (x_i, z_i)$ of player i, it sends x_i to the mediator $\mathcal{F}$, outputs as action the string it receives from $\mathcal{F}$ and halts. So $\Lambda^{\mathcal{F}}$ uses only input[4] x_i and ignores auxiliary information z_i. By $\overrightarrow{\Lambda^{\mathcal{F}}}$ we denote the machine profile where each player uses only $\Lambda^{\mathcal{F}}$. We ensure that whenever the players want to use mediator $\mathcal{F}$, they also want to run $\overrightarrow{M}$ if every time $\overrightarrow{\Lambda^{\mathcal{F}}}$ is a computational Nash equilibrium for the game $(G, \mathcal{F})$, then running $\overrightarrow{M}$ using the intended input is a computational Nash equilibrium as well.

Finally, we provide our definition for game-theoretic protocols implementing trusted mediators. We call our notion game universal implementation. A closely related notion, called strong universal implementation, has been previously defined [19]. On an intuitive level, the main difference between the existing notion and the new notion is that for strong universal implementation, parties consider computation to be costly (i.e., time or memory used for computation may incur additional costs in the utility of the users), while our notion basically regards computation as "for free". The naive intuition suggests that game universal implementation is a weaker notion than strong universal implementation. However, as we will see in Sect. 5.1, this intuition does not hold.

Definition 13 (Game Universal Implementation). *Let $\perp_i$ be the PPITM ran by party i that sends no message (to the other parties or to the mediator) and outputs nothing. Let Games be a set of m-player games, $\mathcal{F}$ and $\mathcal{F}'$ be mediators and let $M_1, \ldots, M_m$ be PPITMs. We call $((M_1, \ldots, M_m), \mathcal{F}')$ a game universal implementation of $\mathcal{F}$ with respect to Games if for all $n \in \mathbb{N}$ and all games $G \in$ Games with input length n*

[4] As in [19], the games considered are canonical games of fixed input n. Any game where there are only finitely many possible types can be represented (by corresponding padding of the input) as a canonical game for some length n.

if $\overrightarrow{\Lambda}^{\mathcal{F}}$ is a computational Nash equilibrium in the mediated game $(G, \mathcal{F})$ with immunity with respect to coalitions, then the following two properties hold:

– *(Preserving Equilibrium) $(M_1, \ldots, M_m)$ is a computational Nash equilibrium in the mediated machine game $(G, \mathcal{F}')$ with immunity with respect to coalitions;*
– *(Preserving Action Distributions) For each type profile $(t_1, \ldots, t_m)$, the output distribution induced by $\overrightarrow{\Lambda}^{\mathcal{F}}$ in $(G, \mathcal{F})$ is statistically close to the output distribution induced by $(M_1, \ldots, M_m)$ in $(G, \mathcal{F}')$;*
– *(Preservation of Best Response $\perp_i$) Additionally, for all $n \in \mathbb{N}$, all games $G \in$ Games with input length n and all $i \in \{1, \ldots, m\}$, if $\perp_i$ is a computational best response to $\overrightarrow{\Lambda}^{\mathcal{F}}_{-i}$ in $(G, \mathcal{F})$, then $\perp_i$ is a computational best response to $\overrightarrow{M}_{-i}$ in $(G, \mathcal{F}')$.*

4 Specialized Simulator UC Variants

Our main result in this section shows the separation between the notions of specialized simulator UC and 1-bit specialized simulator UC. This answers an existing open problem from [25] and furthermore clarifies the relations among different (weak) security notions.

4.1 On 1-bit Specialized Simulator UC

We start by showing that 1-bit specialized simulator UC (1-bit SSUC) is equivalent to weak specialized simulator UC (weak SSUC). This will give us a simpler alternative security notion that we can further work with.

Lemma 1 (Equivalence between 1-bit SSUC and weak SSUC). *A protocol fulfills the 1-bit specialized simulator UC security if and only if it fulfills the weak specialized simulator UC security.*

Proof. Let protocol ρ and ideal functionality $\mathcal{F}$ be such that ρ is as secure as $\mathcal{F}$ with respect to 1-bit specialized simulator UC. We show this implies ρ as secure as $\mathcal{F}$ with respect to weak specialized simulator UC security. Given a triple $(\mathcal{A}, \mathcal{Z}, \mathcal{D}^*)$ consisting of adversary, environment and distinguisher we have to provide a simulator $\mathcal{S}$ such that for every auxiliary input[5] z the following holds:

[5] Here and in the following "for every auxiliary input z" should be read as "for every distribution of auxiliary input z for $\mathcal{Z}$".

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}} \overset{\mathcal{D}^*}{\equiv} \{EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}}. \quad (2)$$

Given $\mathcal{Z}$ and $\mathcal{D}^*$, we can construct a 1-bit output environment $\mathcal{Z}^{\mathcal{D}^*}$ in the following way: $\mathcal{Z}^{\mathcal{D}^*}$ internally runs a copy of $\mathcal{Z}$. When internal $\mathcal{Z}$ writes on its output tape, this is forwarded by $\mathcal{Z}^{\mathcal{D}^*}$ to an internal copy of $\mathcal{D}^*$. The output of $\mathcal{D}^*$ becomes the output of $\mathcal{Z}^{\mathcal{D}^*}$. Due to the hypothesis, there exist $\mathcal{S}$ such that for every auxiliary input z and for every distinguisher $\mathcal{D}$ we have:

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}^{\mathcal{D}^*}}(k,z)\}_{k\in\mathbb{N}} \overset{\mathcal{D}}{\equiv} \{EXEC_{\rho,\mathcal{A},\mathcal{Z}^{\mathcal{D}^*}}(k,z)\}_{k\in\mathbb{N}}.$$

In particular:

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}^{\mathcal{D}^*}}(k,z)\}_{k\in\mathbb{N}} \overset{\mathcal{D}_{ind}}{\equiv} \{EXEC_{\rho,\mathcal{A},\mathcal{Z}^{\mathcal{D}^*}}(k,z)\}_{k\in\mathbb{N}},$$

where $\mathcal{D}_{ind}$ is the distinguisher that outputs whatever $\mathcal{D}^*$ outputs. As the simulator $\mathcal{S}$ can be used without modification in an interaction with $\mathcal{F}$ and the environment[6] $\mathcal{Z}$, the last relation is equivalent to (2). We conclude that ρ is as secure as $\mathcal{F}$ with respect to weak specialized simulator UC security.

The implication in the opposite direction is proven as follows. Given a pair $(\mathcal{A}, \mathcal{Z}_{1-bit})$ consisting of adversary and 1-bit output environment, we need to construct a simulator $\mathcal{S}$ such that for every auxiliary input z and for every distinguisher $\mathcal{D}$, we have:

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}_{1-bit}}(k,z)\}_{k\in\mathbb{N}} \overset{\mathcal{D}}{\equiv} \{EXEC_{\rho,\mathcal{A},\mathcal{Z}_{1-bit}}(k,z)\}_{k\in\mathbb{N}}.$$

Given a 1-bit output environment $\mathcal{Z}_{1-bit}$, we can uniquely decompose it into an environment $\mathcal{Z}$ and a distinguisher $\mathcal{D}^*$ (that given the view of $\mathcal{Z}$ outputs what $\mathcal{Z}_{1-bit}$ outputs).

Indeed, to each 1-bit environment $\mathcal{Z}_{1-bit}$ we can uniquely associate the environment $\mathcal{Z}$ that internally runs $\mathcal{Z}_{1-bit}$: when a party or adversary sends a message to $\mathcal{Z}$, the environment forwards it internally and replies back with the messages that the copy of $\mathcal{Z}_{1-bit}$ would reply. Analogously, when the internal copy of $\mathcal{Z}_{1-bit}$ wants to send a message to a party or to the adversary, the environment $\mathcal{Z}$ forwards this message to the corresponding party or adversary. Finally, $\mathcal{Z}$ gives as output the entire view of the interaction, i.e., all the inputs and messages it sent to the

[6] Indeed, by construction $\mathcal{Z}^{\mathcal{D}^*}$ does not interact with an adversarial party (i.e., $\mathcal{S}$ or $\mathcal{A}$) after the simulation of internal $\mathcal{Z}$ is over.

parties and to the adversary, all the outputs and messages it received from the other entities as well as the random bits used.

Similarly, for each environment $\mathcal{Z}_{1-bit}$ we uniquely associate the distinguisher $\mathcal{D}^*$: after receiving the input, $\mathcal{D}^*$ internally simulates the environment $\mathcal{Z}_{1-bit}$ and emulates the rest of the entities in the protocol, including the adversary; $\mathcal{D}^*$ treats its input as the entire view of the simulated copy of $\mathcal{Z}_{1-bit}$ so $\mathcal{D}^*$ can use it to send all inputs, reply messages and random bits required by the simulated copy. The output bit of the simulated $\mathcal{Z}_{1-bit}$ becomes the output of the distinguisher $\mathcal{D}^*$.

According to the definition of weak specialized simulator UC security, for $\mathcal{A}$, $\mathcal{Z}$, $\mathcal{D}^*$ there exists a simulator $\mathcal{S}$ such that for every auxiliary input z we have:

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}} \stackrel{\mathcal{D}^*}{\equiv} \{EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}}.$$

As $\mathcal{D}^*$ has binary output (i.e., thus finite output), the above equation implies the two random variables $\{\mathcal{D}^*(EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z))\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ and $\{\mathcal{D}^*(EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k,z))\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ are statistically close. Hence, for any computationally bounded distinguisher $\mathcal{D}$ and for any auxiliary input z the random variables $\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}_{1-bit}}(k,z)\}_{k\in\mathbb{N}}$ and $\{EXEC_{\rho,\mathcal{A},\mathcal{Z}_{1-bit}}(k,z)\}_{k\in\mathbb{N}}$ are indistinguishable and this concludes the proof.

4.2 Separation Result

Next we separate the notions of weak specialized simulator UC and specialized simulator UC. For this we use a cryptographic tool called time-lock puzzles, originally introduced in [36].

Definition 14 (Time-lock puzzles). *A probabilistic polynomial time algorithm $\mathcal{G}$ (problem generator) together with a probabilistic polynomial time algorithm $\mathcal{V}$ (solution verifier) represent a* time-lock puzzle *if the following holds:*
-sufficiently hard puzzles: for every probabilistic polynomial time algorithm B and for every $e \in \mathbb{N}$, there is some $f \in \mathbb{N}$ such that

$$up^s_{t\geq k^f,|h|\leq k^e} Pr[(q,a) \leftarrow \mathcal{G}(1^k,t) : \mathcal{V}(1^k,a,B(1^k,q,h)) = 1] \quad (3)$$

is negligible in k.
-sufficiently good solvers: there is some $m \in \mathbb{N}$ such that for every $d \in \mathbb{N}$ there is a PPT algorithm C such that

$$\min_{t\leq k^d} Pr[(q,a) \leftarrow \mathcal{G}(1^k,t); v \leftarrow C(1^k,q) : \mathcal{V}(1^k,a,v) = 1 \wedge |v| \leq k^m] \quad (4)$$

is overwhelming in k.

Intuitively, a time-lock puzzle is a cryptographic tool used for proving the computational power of a PPITM. $\mathcal{G}(1^k, t)$ generates puzzles of hardness t and $\mathcal{V}(1^k, a, v)$ verifies that v is a valid solution as specified by a. The first requirement is that B cannot solve any puzzle of hardness t, with $t \geq k^f$, for some f depending on B, with more than negligible probability. The algorithm B may have an auxiliary input. This ensures that even puzzles generated using hardness t chosen by B together with a trap-door like auxiliary information (of polynomial length), do not provide B with more help in solving the puzzle.

The second requirement is that for any polynomial hardness value there exist an algorithm that can solve any puzzle of that hardness. It is important that the solution for any puzzle can be expressed as a string of length bounded above by a fixed polynomial.

As promoted by [36] and later by [22], a candidate family for time-lock puzzles which is secure if the RSA assumption holds, is presented next. A puzzle of hardness t consists of the task to compute $2^{2^{t'}} \mod n$ where $t' := \min(t, 2^k)$ and $n = p_1 \cdot p_2$ is a randomly chosen Blum integer. Thus, $\mathcal{G}(1^k, t) = ((n, \min\{t, 2^k\}), (p_1, p_2, \min\{t, 2^k\}))$, where n is a k-bit Blum integer with factorization $n = p_1 \cdot p_2$, and $\mathcal{V}(1^k, (p_1, p_2, t'), v) = 1$ if and only if $(v = v_1, v_2)$ [7] and $v_1 \equiv 2^{2^{t'}} \, mod \, n$ and $v_2 = n$. Both solving the puzzle and verifying the solution can be efficiently done if p_1 and p_2 are known. From this point further we call these puzzles the Blum integer puzzles. An important property that we use in the following is that any Blum integer puzzle has a unique solution.

Before we state and prove our main separation result in theorem 1, we give as reminder the definition of hard-core predicates and then we state two properties related to them.

Definition 15 (Hard-Core Predicate). *A hard-core predicate of a collection of functions* $g_{k,t} : \{0,1\}^* \rightarrow \{0,1\}^*$ *is a boolean predicate* $HC : \{0,1\}^* \rightarrow \{0,1\}$ *such that:*

- *there exists a probabilistic polynomial time algorithm* E *with* $HC(x) = E(x)$*, for every* x*;*

[7] Without loosing any security of the initial definition of time-lock puzzles [36,22], in addition to the value $2^{2^t} \, mod \, n$, our solution for the puzzle $q = (t, n)$ contains also the value n. The full use of defining solutions in such a way, will become more clear when we define the one-way function based on time-lock puzzles: There is a one-to-one correspondence between the pair of values $(v = (2^{2^{t'}} \, mod \, n, n), t)$ and $q = (t, n)$.

– *for every probabilistic polynomial time algorithm A and for every polynomial p, there exists k_p and t_p such that for every $k > k_p$ and $t > t_p$, we have $Pr[A(1^k, t, g_{k,t}(x)) = HC(x)] < \frac{1}{2} + \frac{1}{p(k)}$.*

Now we are ready to state the two lemmas related hard-core predicates. The first result shows that from a Blum integer time-lock puzzle we can construct a one-way function and a hard-core predicate.

Lemma 2 (One-Way Function and Hard-Core Predicate from Blum Integer Time-Lock Puzzles). *Let $(\mathcal{G}, \mathcal{V})$ be a Blum integer time-lock puzzle and let t be an integer. Let $S_{k,t}$ be the set of all correctly generated solutions $v = (2^{2^t} \mod n, n)$ for puzzles q, where $q = (t, n)$ is the output of algorithm $\mathcal{G}$ when invoked with parameters 1^k and t. Then the collection of functions $\{f_{k,t} : S_{k,t} \to \{0,1\}^*\}_{(k \in \{0,1\}^*, t \in \{0,1\}^k)}$ and $\{g_{k,t} : S_{k,t} \times \{0,1\}^* \to \{0,1\}^*\}_{(k \in \{0,1\}^*, t \in \{0,1\}^k)}$ defined below are collections of one-way functions and the predicate $HC : \{0,1\}^* \to \{0,1\}^*$ defined below is a hard-core predicate for $\{g_{k,t}\}_{(k \in \{0,1\}^*, t \in \{0,1\}^k)}$. We would alternatively call HC the hard-core predicate for $(\mathcal{G}, \mathcal{V})$. We define $f_{k,t}(2^{2^t} \mod n, n) = (t, n)$ and for $v, r \in \{0,1\}^*$ such that $|v| = |r|$, let $g_{k,t}(v, r) = (f_{k,t}(v), r)$ and $HC(v, r) = \sum_{i=1}^{|v|} v_i \cdot r_i \mod 2$.*

Proof. First we prove that $\{f_{k,t}\}_{(k \in \{0,1\}^*, t \in \{0,1\}^k)}$ defined above is a collection of one-way functions. For every security parameter k, let $m(k)$ be the maximum number of bits that machine $\mathcal{G}$ can read from its randomness tape when invoked with security parameter k. Assume by contradiction that there exist adversary A and polynomial p such that for every integers k_p and t_p there exist $k \geq k_p$ and $t \geq t_p$ with:

$$Pr[A(1^k, t, (t, n)) = v : \mathcal{G}(1^k, t) = ((t, n), a), V(1^k, a, v) = 1] \geq \frac{1}{p(k)}.$$

If in the definition of the first property of time-lock puzzles, we take $e = 0$ and we use algorithm A for solving the puzzles, then we immediately obtain a contradiction so our assumption is false.

It is also clear that the property we have shown to hold for f, can be shown in a similar way to hold for g.

By following exactly the steps of the well known proof by Goldreich and Levin [12], which gives a hard-core predicate construction for any one-way function, it follows that $HC(v, r) = \sum_{i=1}^{k} v_i \cdot r_i \mod 2$ is a hard-core predicate for g and this concludes the proof.

The second result is a straight forward consequence of the definition of hard-core predicates.

Lemma 3 (Distribution of Hard-Core Predicates). *Let k be a security parameter. Then, for any given integer t, let $g_{k,t} : D_{k,t} \to \{0,1\}^*$ be a function such that $HC : \{0,1\}^* \to \{0,1\}$ is a hard-core predicate for the collection of functions $\{g_{k,t}\}_{k\in\{0,1\}^*, t\in\{0,1\}^k}$. Let $X(k,t)$ be the distribution of $(g_{k,t}(x), HC(x))$ and let $Y(k,t)$ be the distribution of $(g_{k,t}(x), U(x))$ with x taken from the domain $D_{k,t}$ and $U(x)$ being the uniform distribution on $\{0,1\}$. Then the ensembles $\{X(k,t)\}_{(k\in\{0,1\}^*, t\in\{0,1\}^k)}$ and $\{Y(k,t)\}_{(k\in\{0,1\}^*, t\in\{0,1\}^k)}$ are computationally indistinguishable. x, the random variable $U(x)$ has the uniform distribution on $\{0,1\}$.*

Proof. In definition 15 we choose an adversary A such that its output is independent of its input. More precisely, we take A that outputs 1 with constant probability c. This implies that the output distributions of A and HC are also independent. If we denote by $w_{k,t}(x)$ the probability that HC outputs 1 given x from a distribution $Output(1^k, t)$, then we obtain:

$$
\begin{aligned}
&Pr[A(1^k, t, g_{k,t}(x)) = HC(x) : x \leftarrow Output(1^k, t)] = \\
&= (Pr[A(1^k, t, g_{k,t}(x))] = 0) \cdot (Pr[HC(x)] = 0) + \\
&\qquad\qquad + (Pr[A(1^k, t, g_{k,t}(x))] = 1) \cdot (Pr[HC(x)] = 1) = \\
&= w_{k,t}(x)(2 \cdot c - 1) + 1 - c.
\end{aligned}
$$

Substituting this in the definition of hard-core predicate, we have that for every polynomial p, for all sufficiently large k and all sufficiently large t: $w_t(x)(2 \cdot c - 1) + 1 - c < \frac{1}{2} + \frac{1}{p(k)}$, which is equivalent to $w_{k,t}(x) < \frac{1}{2} + \frac{1}{p(k)\cdot(2c-1)}$ for large enough t and k. Since c is a constant, this implies that for large enough t and k, the probability $w_{k,t}(x)$, (where $x \leftarrow Output(1^k, t)$), is negligibly close to $\frac{1}{2}$ and this concludes our proof.

Using lemmas 2 and 3, the following statement can be shown:

Lemma 4 (Weak SSUC Does Not Imply SSUC).
Assume Blum integer time-lock puzzles exist. Then there are protocols that fulfill weak specialized simulator UC security but do not fulfill specialized simulator UC security.

Proof. Let $(\pi, \mathcal{F})$ be a pair of protocol and ideal functionality as defined below. The only input the ideal functionality $\mathcal{F}$ requires is the security parameter 1^k. Then $\mathcal{F}$ sends a message to the adversary (i.e. ideal simulator $\mathcal{S}$) asking for its computational hardness. Using the reply value t' from

$\mathcal{S}$ (which is truncated by $\mathcal{F}$ to maximum k bits), the ideal functionality invokes $Gen(1^k, t') \rightarrow (q', a')$ to generate a time-lock puzzle q' of hardness t', whose solution should verify the property a'. The puzzle q' is sent to $\mathcal{S}$ which replies with v'. Finally, $\mathcal{F}$ checks whether v' verifies the property a'. In case a' does not hold, $\mathcal{F}$ stops without outputting any message to the environment. Otherwise, for every value $i \in \{1, \ldots, k\}$, $\mathcal{F}$ generates a puzzle q_i of hardness $t_i = 2^i$. Let j be such that $2^j \leq t' < 2^{j+1}$, so $j \in \{1, \ldots, k\}$.

For the puzzle q_j, $\mathcal{F}$ computes the solution v_j. $\mathcal{F}$ can efficiently compute this solution as it knows the additional information a_j. Additionally, $\mathcal{F}$ chooses r uniformly at random from $\{0,1\}^{2k}$. Without loss of generality, we can assume the solution v of each puzzle q generated using the parameters 1^k and t has length $2 \cdot k$. Indeed, we can prepend with 0's to the string v such that its length reaches $2 \cdot k$. It is easy to see that after this operation, the properties stated in lemma 2 still hold. The output of $\mathcal{F}$ to the environment is the tuple $(q_1, \ldots, q_k, r, HC(v_j, r))$, where HC is the hard-core predicate of $(\mathcal{G}, \mathcal{V})$ as given by lemma 2.

For each hardness t', we call $P(t')$ the distribution of the view of $\mathcal{Z}$ when interacting in the ideal world.

The real world protocol π, is defined similarly to $\mathcal{F}$, the only difference is the final output: π outputs to $\mathcal{Z}$ a tuple $(q_1, \ldots, q_k, r, b)$, with r randomly chosen from $\{0,1\}^{2k}$ and b randomly chosen from $\{0,1\}$. For each hardness t used by the adversary $\mathcal{A}$ when interacting with $\mathcal{Z}$, we call $R(t)$ the distribution of the view of $\mathcal{Z}$ when interacting in the real world.

The proof has two steps. First, we show that π is as secure as $\mathcal{F}$ with respect to weak specialized simulator UC security. Let $\mathcal{D}$ be a distinguisher of hardness $t_{\mathcal{D}}$ (i.e., it can solve puzzles of hardness less or equal to $t_{\mathcal{D}}$ with overwhelming probability but it cannot solve puzzles of hardness greater than $t_{\mathcal{D}}$ with more than negligible probability) and an adversary $\mathcal{A}$ of hardness $t_{\mathcal{A}}$. Let l be the minimum value such that $2^l > \max(t_{\mathcal{D}}, t_{\mathcal{A}})$. We now require that the simulator $\mathcal{S}$ has hardness t' such that $t' \geq 2^l$. As we will see next, this is one of the constraints necessary for making the two distributions $R(t')$ and $P(t)$ indistinguishable to $\mathcal{D}$.

The intuition is that in the ideal world $\mathcal{D}$ would have to solve a puzzle with hardness larger than $t_{\mathcal{D}}$ and learn the hard-core bit for such a puzzle. According to lemma 3, this hard-core bit is indistinguishable from a random bit, which is actually what the protocol π outputs to the environment.

More formally, let $(\mathcal{A}, \mathcal{Z}, \mathcal{D})$ be a triple of real world adversary, environment and distinguisher and let 1^k be the security parameter. Then, let e be such that the length of the messages sent by $\mathcal{Z}$ to $\mathcal{D}$ is bounded above by k^e. From (3), there exists $f_e^{\mathcal{D}}$ such that for every polynomial p there exists k_p^0 such that:

$$up^s_{t \geq k^{f_e^{\mathcal{D}}}, |h| \leq k^e} Pr[(q', a') \leftarrow \mathcal{G}(1^k, t') : \mathcal{V}(1^k, a', \mathcal{D}(1^k, q', h)) = 1] < \frac{1}{p(k)}$$

for every $k > k_p^0$.This intuitively means that $\mathcal{D}$ can solve puzzles of hardness larger than $k^{f_e^{\mathcal{D}}}$ only with negligible probability. Given $\mathcal{A}$, in an analogue way we define $k^{f_e^{\mathcal{A}}}$ and k_p^1. With the notation used in the description of π and $\mathcal{F}$, it now becomes clear that we can take $t_{\mathcal{D}} = k^{f_e^{\mathcal{D}}}$ and $t_{\mathcal{A}} = k^{f_e^{\mathcal{A}}}$.

We construct $\mathcal{S}$ such that there exists a negligible function ϵ and k_2 such that for every $k \geq k_2$ and for every distribution of auxiliary input z we have:

$$|(Pr(\mathcal{D}(EXEC_{\mathcal{A},\pi,\mathcal{Z}}(k, z)) = 1) - (Pr(\mathcal{D}(EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k, z)) = 1)| < \epsilon(k). \quad (5)$$

We take k_2 such that for every $k \geq k_2$, it holds that $\max(t_{\mathcal{A}}, t_{\mathcal{D}}) < 2^k$.

For a given $t_{\mathcal{A}}$ and $t_{\mathcal{D}}$ and for l defined as above, let f' be such that for sufficiently large k, $2^l \leq k^{f'} \leq 2^k$. Let $\mathcal{S}$ be the simulator of hardness $k^{f'}$ that as first reply to $\mathcal{F}$ sends $t' := k^{f'}$. According to (4), there exists m such that for $d := f'$ there exists $C_{f'}$ such that

$$Pr[(q', a') \leftarrow \mathcal{G}(1^k, k^{f'}); v' \leftarrow C_{f'}(1^k, q') : \mathcal{V}(1^k, a', v') = 1 \wedge |v'| \leq k^m]$$

is overwhelming in k. When $\mathcal{F}$ sends a puzzle q' to $\mathcal{S}$, the simulator invokes $C_{f'}$ for $(1^k, q')$ and sends to $\mathcal{F}$ the output v' of $C_{f'}$. Internally, $\mathcal{S}$ simulates the adversary $\mathcal{A}$ and emulates the messages that the adversary would receive from $\mathcal{Z}$ and π as follows: When $\mathcal{F}$ requires the value of the computational hardness from $\mathcal{S}$, then $\mathcal{S}$ acts as π and requires the computational hardness from simulated $\mathcal{A}$. When $\mathcal{S}$ receives t from $\mathcal{A}$, then it invokes $Gen(1^k, t)$, obtaining output (q, a) and forwards to simulated $\mathcal{A}$ the puzzle q. Moreover, any message that internal $\mathcal{A}$ wants to send to the environment, $\mathcal{S}$ forwards it to $\mathcal{Z}$. Any message for $\mathcal{A}$ coming from $\mathcal{Z}$ is immediately forwarded by $\mathcal{S}$ to the internally simulated adversary. This completes the construction of $\mathcal{S}$.

By construction, $\mathcal{S}$ solves the puzzle sent by $\mathcal{F}$ with overwhelming probability and hence the output of $\mathcal{F}$ to $\mathcal{Z}$ is $(q_1, \ldots, q_k, r, HC(v_j, r))$

with the same probability. The view of $\mathcal{Z}$ in the real world is $(1^k, t, q, v, (q_1, \ldots, q_k, r, b))$ and the view of $\mathcal{Z}$ in the ideal world is $(1^k, t, q, v, (q_1, \ldots, q_k, r, HC(v_j, r)))$. One may argue of course that the view of $\mathcal{Z}$ may or may not contain the values t, q, v, depending on the adversary $\mathcal{A}$. Also, additionally to the view(s) stated above, the environment could output the interaction that it has with $\mathcal{A}$ besides messages t, q, v. However, for the analysis of this proof, the views considered above are the worst case scenario that would allow a distinguisher to tell apart the two worlds.

By applying lemma 3 for the distinguisher $\mathcal{D}$ and polynomial p, there exists k_p and t_p, such that for every $k > k_p$ and $t > t_p$, the advantage of $\mathcal{D}$ for distinguishing between the distributions of $((q, r), b)$ and $((q, r), HC(v, r))$ (with $\mathcal{G}(1^k, t) \leftarrow (q, a)$, v the solution to q, b the random bit and r the uniformly distributed string of k bits) is less than $\frac{1}{p(k)}$. Hence, additionally to the previous constraints on k and t', we take k such that $k > k_p$ and $\max\{t_{\mathcal{A}}, t_{\mathcal{D}}, t_p\} < 2^k$ and t' such that $t' > \max\{t_{\mathcal{A}}, t_{\mathcal{D}}, t_p\}$. With this we can conclude that the real and the ideal world views are indistinguishable to $\mathcal{D}$.

Second, we prove that π is not as secure as $\mathcal{F}$ with respect to specialized simulator UC security. Intuitively, for every hardness $t_{\mathcal{S}}$ (polynomial in the security parameter k) of a simulator machine $\mathcal{S}$, there exists a distinguisher $\mathcal{D}_{\mathcal{S}}$ such that for every $t \leq t_{\mathcal{S}}$, $\mathcal{D}_{\mathcal{S}}$ can solve puzzles of hardness t. As we will see next, $\mathcal{D}_{\mathcal{S}}$ uses this property to distinguish with non-negligible probability between the environment's output distribution in the real and in the ideal world.

Formally, let $\mathcal{A}$ be the real world adversary that can solve puzzles of hardness $t_{\mathcal{A}}$ such that when receiving its input from the environment, it replies to π with $t_{\mathcal{A}}$ and the corresponding correct solution for the puzzle received. Let $\mathcal{Z}$ be the environment that just sends the security parameter to all parties (i.e., including the adversarial parties), receives their outputs and then outputs as view the messages received from the honest parties (i.e., protocol π in the real world or $\mathcal{F}$ in the ideal world). For every simulator $\mathcal{S}$, we show that there exists a distinguisher $\mathcal{D}_{\mathcal{S}}$ and a distribution for the auxiliary input z such that:

$$\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k, z)\}_{k \in \mathbb{N}} \overset{\mathcal{D}_{\mathcal{S}}}{\not\equiv} \{EXEC_{\pi,\mathcal{A},\mathcal{Z}}(k, z)\}_{k \in \mathbb{N}}.$$

Given $\mathcal{S}$ of hardness $t_{\mathcal{S}}$, we choose $\mathcal{D}_{\mathcal{S}}$ such that it can solve puzzles of hardness at least $t_{\mathcal{D}} = max(t_{\mathcal{S}}, t_{\mathcal{A}})$ with overwhelming probability in k. Such a $\mathcal{D}_{\mathcal{S}}$ exists according to (4). Additionally, after receiving the view of $\mathcal{Z}$, $\mathcal{D}_\mathcal{S}$ solves one by one each puzzle q_i included in that view that has associated hardness $t_i \leq t_\mathcal{D}$ and it obtains each time the corresponding correct and unique solution v_i with overwhelming probability. Then $\mathcal{D}_\mathcal{S}$ evaluates $HC(v_i, r)$. Lets call m the last bit in the output of the honest party (i.e., $\mathcal{F}$ or π) to $\mathcal{Z}$[8]. Next, $\mathcal{D}_\mathcal{S}$ checks if $m \neq HC(v_i, r)$ for all i as defined above. If this holds, then $\mathcal{D}$ outputs 1, otherwise it outputs 0.

If m is part of the view of the real world, then according to the definition of π, m is a random bit in $\{0, 1\}$ so it is different than a given bit $HC(v_i, r)$ with probability $\frac{1}{2}$. This is equivalent to $\mathcal{D}_\mathcal{S}$ outputting 1 with probability $\frac{1}{2^{\log 2t_\mathcal{D}}} = \frac{1}{t_\mathcal{D}}$ when the view of $\mathcal{Z}$ is from the real world. Similarly, if m is part of the view of $\mathcal{Z}$ in the ideal world, then there exists at least an index i such that $HC(v_i, r)$ can be computed by $\mathcal{D}_\mathcal{S}$ and $m = HC(v_i, r)$; so $\mathcal{D}_\mathcal{S}$ outputs 1 with probability 0. This implies $\mathcal{D}_\mathcal{S}$ can distinguish at least with the non-negligible probability $\frac{1}{t_\mathcal{D}}$ [9] between the output distributions from the two worlds and this concludes the proof.

We are now ready to conclude that 1-bit specialized simulator UC security and specialized simulator UC security are not equivalent notions. By putting together the results from lemma 1 and from lemma 4 we obtain:

Theorem 1 (1-bit SSUC and SSUC Not Equivalent). *Assume Blum integer time-lock puzzles exist. Then there are protocols secure with respect to 1-bit specialized simulator UC security which are not secure with respect to specialized simulator UC security.*

4.3 Discussion

The separation result presented in theorem 1 is conditioned on the existence of Blum integer time-lock puzzles, which in turn is based on the RSA assumption. To the best of our knowledge, the only other known time-lock puzzle constructions are possible in the random oracle model [27,26]. However, these constructions cannot replace the Blum integer time-lock puzzle in our proof method for the separation lemma 4.

On one hand, the construction from [27] allows only a fixed linear gap between the time needed for generating and the time needed for solving a puzzle; with the Blum integer time-lock puzzles we can control

[8] Due to the definition of $\mathcal{Z}$, the string m is also a part of the output of the environment.

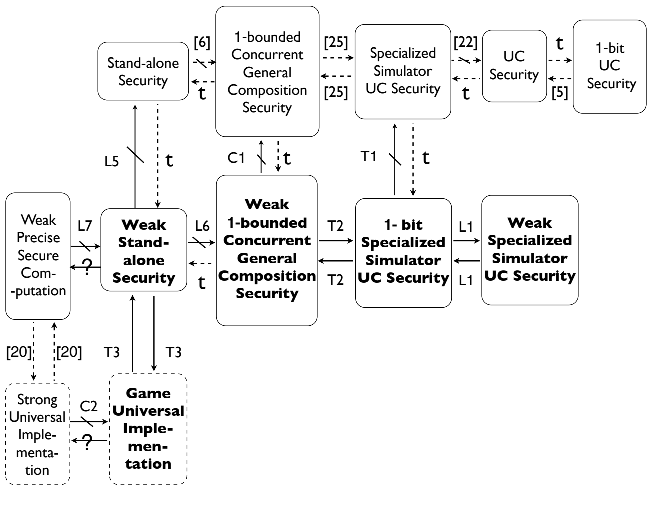
[9] Since $\mathcal{D}$ is a polynomial time machine, its hardness $t_\mathcal{D}$ is also a polynomial in the security parameter k, so the function $\frac{1}{t_\mathcal{D}}$ is non-negligible.

the hardness of the puzzle. On the other hand, the puzzles from [26] allow for more fine tuning of the time gap, but it is not clear how to use them to construct the one-way functions that allowed us to conclude the separation lemma 4. This is the case since the constructions from [26] do not allow for generation of hard enough solutions in efficient time. We did not encounter this impediment in the case of Blum integer time-lock puzzles, since those puzzles were generated together with a trap-door which allowed for efficient computation of the solution.

It is an open question how to construct a time lock puzzle based on general cryptographic assumption (e.g., the existence of one-way functions) or to show that such a construction cannot be used for our separation result.

5 Equivalence of Security Notions

Fig. 1. Implication Relations among Computational Security Concepts



Implication relations among various security notions with respect to computational security are depicted in Fig. 1. Previously existing notions are written in a regular font, while the notions defined in this work are written in a boldface font. The continuos line arrows depict relations we prove in this paper[10]; all other relations have been previously known (the

[10] A letter and number next to an arrow represent the theorem T or the lemma L or the corollary C where the respective result is shown in this paper.

number in the square brackets denotes the reference) or can be trivially derived (i.e., denoted by letter t). Continuous line frames highlight security notions, while dotted frames highlight game-theoretic concepts. Finally, open questions are marked by question marks.

It is a well-known result that UC security and 1-bit UC security are equivalent [5]. It has been also shown [22] that specialized simulator UC security does not imply UC security. Moreover, specialized simulator UC security is equivalent to security under 1-bounded concurrent general composition [25]. It has been shown [6] that stand-alone security does not imply specialized simulator UC security.[11] The implication in the opposite direction holds trivially. Similarly, it is trivial to see that universal composability implies specialized simulator UC security.

Our first result in this section proves that weak security under 1-bounded concurrent general composition is equivalent to 1-bit specialized simulator UC security. A similar proof technique has been used in [25], however, our proof requires more technicalities.

Theorem 2 (Equivalence between Weak 1-bounded CGC Security and 1-bit SS UC Security). *Let ρ be a protocol and $\mathcal{F}$ an ideal functionality. We have that ρ implements $\mathcal{F}$ under weak 1-bounded concurrent general composition security, if and only if ρ securely computes $\mathcal{F}$ under 1-bit specialized simulator UC security.*

Proof. In the following we need *one-time information-theoretic message authentication codes* so we include the definition below.

Definition 16 (One-Time Information-Theoretic Message Authentication Code). *A one-time information-theoretic message authentication code is a triple $(Gen, Mac, Verify)$ where $Gen(1^n)$ outputs a key k, $Mac(k,x)$ outputs a tag t (obtained using k) for the message x of length n and $Verify(k,m,t)$ outputs 0 or 1. The correctness property requires that $\forall n, \forall k$ in the range of $Gen(1^n)$ and $\forall x \in \{0,1\}^n$ we have $Verify(k,x,Mac(k,x)) = 1$.*

Moreover, the following security property is fulfilled. For every adversary $\mathcal{A}$ such that

$$Pr[(x',t') \leftarrow A(x,t) \wedge x' \neq x \wedge Verify(k,x',t') = 1 : \\ : x \leftarrow A(1^n), k \leftarrow Gen(1^n), t \leftarrow Mac(k,x)],$$

[11] In order to preserve the symmetry and clarity of our picture, we have indicated that the result in [6] is that stand-alone security does not imply 1-bounded concurrent general composition. This is indeed a immediate consequence of combining the results from [6] and [25].

is negligible in n.

Next, we prove an important lemma which will allow us to conclude theorem 2.

Lemma 5 (Equivalence between Weak Security under 1-bounded Concurrent General Composition and Weak Specialized Simulator UC Security). *Let ρ be a protocol and $\mathcal{F}$ an ideal functionality. Then ρ securely computes $\mathcal{F}$ under 1-bounded concurrent general composition with weak security if and only if ρ securely implements $\mathcal{F}$ under weak specialized simulator UC security.*

Indeed, putting together lemma 5 and lemma 1 we can conclude the theorem.

And now we give in full detail the proof for lemma 5.

Proof. As expected, the more involved part of the proof is the implication from weak security under concurrent general composition to specialized simulator 1-bit UC security. The reverse direction can be shown analogously to the proof existing in the initial version of [25].

Let $R_1, \ldots, R_m$ be the parties for ρ. Let $(\mathcal{A}, \mathcal{Z}, \mathcal{D})$ be a triple consisting of UC real world adversary (possibly adaptive), environment and distinguisher. We need to show there exists an UC ideal world simulator $\mathcal{S}$ such that the views of the environment in real world and in the ideal world cannot be distinguished by $\mathcal{D}$. The adversary $\mathcal{A}$ may not corrupt any party, in which case $\mathcal{A}$ is still capable of scheduling messages in the network. Additionally, remember that in the UC model the only messages that $\mathcal{A}$ has no control of, even by scheduling, are the input messages that the environment $\mathcal{Z}$ writes directly on the input tapes of the parties and the output messages that $\mathcal{Z}$ reads directly from the parties output tapes.

The intuition behind the proof is as follows: We use the fact that ρ composed with an instance of any protocol (i.e., even one that has more parties than ρ) is secure[12]. We construct a protocol π for $m+2$ parties that besides the m parties of ρ has $P_{\mathcal{Z}}$ and $P_{\mathcal{A}}$ playing the role of $\mathcal{Z}$ and $\mathcal{A}$ respectively. In this way, we reduce the proof of weak specialized simulator UC security of ρ to weak security under concurrent general composition. As mentioned above, the adaptive adversary $\mathcal{A}$ could corrupt everyone or could corrupt no party and act as a network adversary. Thus, the motivation behind using the two extra parties in the protocol π is to ensure

[12] The security is of course in the sense of definition 10.

there is always an honest entity and also a corrupted entity, same as in the UC world. In order to model the ideally secure channels that the specialized simulator UC (real/ideal) setting ensures by definition between $\mathcal{Z}$ and the parties of ρ, we use one-time pads and one-time authentication MACs in the concurrent general composition world between $P_{\mathcal{Z}}$ and the parties of ρ.

However, it is important to know how long should the keys be. They should suffice for all necessary encrypted and authenticated communication. Let q be a polynomial such that for every security parameter n and for every i the value $q(n)$ bounds above the length of encryption and authentication keys needed between each pair $P_{\mathcal{Z}}$ and P_i with $i \in \{1, \ldots, m\}$. We postpone until after the description of π why such polynomial q exists and how it is computed.

Formally, protocol π is described below and it can be used for both the real and the ideal concurrent general composability worlds.

1. *Inputs:* Each party P_i with $i \in \{1, \ldots, m\}$ receives a pair (k^i_{mac}, k^i_{enc}) of keys[13]. Party $P_{\mathcal{A}}$ receives the empty string λ as input. Party P_{m+1} receives an input z and also the tuples $((k^1_{mac}, k^1_{enc}), \ldots, (k^m_{mac}, k^m_{enc}))$[14];
2. *Outputs:* The protocol outputs whatever $P_{\mathcal{Z}}$ outputs. The rest of the parties of π output an empty string λ;
3. *Instructions for P_i, with $i \in \{1, \ldots, m\}$:* When P_i receives $(input, x_i, t_i)$ from $P_{\mathcal{A}}$, it verifies the correctness of the tag. If verification succeeds, it computes $m_i = x_i \oplus k^i_{enc}$ and sends m_i either to its corresponding ITM that emulates R_i of ρ or to the functionality $\mathcal{F}$. (This depends on whether π is part of the composed protocol π^ρ or $\pi^{\mathcal{F}}$. Remember that independent of the channels model, an adversary in the concurrent general composability world cannot interfere in any way with the messages that an uncorrupted party of π wants to send to its associated ITM for ρ.) If verification fails, then P_i halts. When the ITM emulating R_i or when $\mathcal{F}$ respectively sends the output value

[13] For ease of notation, we use one encryption key and one MAC key per party P_i, as they can be considered long enough to encrypt and authenticate the entire communication between P_i and $P_{\mathcal{Z}}$. However, for each different encryption (authentication) that needs to be performed, a new part of the string k^i_{enc} (and k^i_{mac}, respectively) is used.

[14] The input strings to π may have any distribution and the indistinguishability between the real and the ideal concurrent general composability worlds would still be preserved. However, for this proof we restrict the inputs to encryption keys (i.e., they are uniformly distributed in $\{0,1\}^{q(k)}$) and MAC keys (i.e., they are generated with the *Gen* key generation algorithm).

y_i to P_i, then P_i computes $e_i = y_i \oplus k^i_{enc}$ and $v_i = MAC(k^i_{mac}, e_i)$ and sends the message $(output, e_i, v_i)$ to party $P_{\mathcal{Z}}$;

4. *Instructions for $P_{\mathcal{Z}}$:* Upon receiving an input value z, it uses it for internally invoking $\mathcal{Z}$. When internal $\mathcal{Z}$ wants to send a message $(input, m_i)$ to party i, then $P_{\mathcal{Z}}$ computes $x_i = m_i \oplus k^i_{enc}$ and $t_i = MAC(k^i_{mac}, x_i)$ and sends $(input, x_i, t_i)$ to P_i. When $P_{\mathcal{Z}}$ receives a message $(output, y_i, v_i)$ from party P_i, it first checks the correctness of the tag v_i. If verification succeeds, then $P_{\mathcal{Z}}$ computes $m_i = y_i \oplus k^i_{enc}$ and stores m_i. Otherwise, it halts. When internal $\mathcal{Z}$ wants to read the output tape of party i, then $P_{\mathcal{Z}}$ looks up if there is a message m_i stored from party P_i. If so, it writes m_i to corresponding tape of $\mathcal{Z}$, otherwise it just writes λ to $\mathcal{Z}$. Regarding the communication with its adversary, when $P_{\mathcal{Z}}$ receives a message from $\mathcal{Z}$ of the form $(\mathcal{Z}, \mathcal{A}, m)$, it forwards it to $P_{\mathcal{A}}$. Similarly when $P_{\mathcal{Z}}$ receives a message of the form $(\mathcal{A}, \mathcal{Z}, m)$ from $P_{\mathcal{A}}$, it forwards it internally to $\mathcal{Z}$.
5. *Instructions for $P_{\mathcal{A}}$:* This party has no predefined instructions. $P_{\mathcal{A}}$ is needed in order to provide a means of communication for the adversary of the general concurrent composition setting which in this model can only send messages through a corrupted party[15].

We now explain how the polynomial q is chosen. Since the communication between $P_{\mathcal{Z}}$ and each of the parties P_i with $i \in \{1, \ldots, m\}$ has to be secure and authenticated, the length of the secret keys for the one-time pad and and for the one-time MAC should be long enough. The intuition is that the length of the encryption keys shared by $P_{\mathcal{Z}}$ and P_i is bounded above by the length of the longest string that machine $\mathcal{Z}$ can write plus the longest string that R_i can write. Since both machines are polynomially bounded and they are fixed before the protocol π is constructed, there exist a polynomial q_i such that $q_i(n)$ bounds from above the length of the common encryption keys for every security parameter n. Moreover, the length of the secret key needed for the authenticated messages between $P_{\mathcal{Z}}$ and P_i is at most as long as the one-time pad secret keys. Putting the above arguments together we conclude there exists a polynomial q such that $q(n) \geq max\{q_1(n), \ldots, q_m(n)\}$.

For the protocol π given above we construct an adversary $\mathcal{A}_\pi$ interacting with the composed protocol π^ρ. Intuitively, the task of $\mathcal{A}_\pi$ is to enable the communication among $\mathcal{Z}$ (invoked by $P_{\mathcal{A}}$), $\mathcal{A}$ (invoked by the adversary $\mathcal{A}_\pi$) and the ITMs implementing ρ, in the same way as it happens in

[15] This is in contrast to the UC model where even if none of the protocol parties is corrupted, the adversary can interact with the environment $\mathcal{Z}$.

the UC real world. In order to make this work and for reasons explained above, the adversary A_π corrupts $P_{\mathcal{A}}$. We construct the adversary $\mathcal{A}_\pi$ as follows: It internally runs the code of the UC real world adversary $\mathcal{A}$ and if $\mathcal{A}$ corrupts a party R_i, then $\mathcal{A}_\pi$ corrupts the party P_i together with its corresponding ITM for computing ρ. The intuition is that $\mathcal{A}_\pi$ instructs the corrupted parties of π to run the protocol as before, while their corresponding corrupted ITMs follow the instructions of $\mathcal{A}$. The handling of messages by $\mathcal{A}_\pi$ is as follows:

1. Input messages $(input, x_i, t_i)$ sent by $P_{\mathcal{Z}}$ are forwarded *immediately* by $\mathcal{A}_\pi$ to P_i; Output messages $(output, e_i, v_i)$ sent by P_i are *immediately* forwarded to $P_{\mathcal{Z}}$.
   Moreover, as soon as party P_i is corrupted, its current state and all its previously received messages are sent to $\mathcal{A}$. The information that $\mathcal{Z}$ expects to receive upon corruption is sent by $\mathcal{A}_\pi$ to $P_{\mathcal{Z}}$. All messages received from this point on by P_i are forwarded by A_π to $\mathcal{A}$.
2. When $P_{\mathcal{Z}}$ sends a message $(\mathcal{Z}, \mathcal{A}, m)$ to party $P_{\mathcal{A}}$, then $\mathcal{A}_\pi$ forwards it to its internal run of $\mathcal{A}$ as if coming from $\mathcal{Z}$. The messages $(\mathcal{A}, \mathcal{Z}, m)$ that $\mathcal{A}$ wants to send to $\mathcal{Z}$ are forwarded by $\mathcal{A}_\pi$ to $P_{\mathcal{Z}}$;
3. All messages that $\mathcal{A}$ instructs a corrupted party R_i to send to an uncorrupted party R_j will be forwarded by $\mathcal{A}_\pi$ to the corresponding ITM of P_j as if coming from the corresponding ITM of P_i; However $\mathcal{A}$ schedules messages among parties R_i with $i \in \{1, \ldots, m\}$, $\mathcal{A}_\pi$ does the same for the messages between the corresponding ITMs of parties P_i with $i \in \{1, \ldots, m\}$.
4. The adversary $\mathcal{A}_\pi$ has no control over the messages between an uncorrupted P_i and its corresponding ITM for computing ρ.

After having defined protocol π and adversary $\mathcal{A}_\pi$, we prove the output of $P_{\mathcal{Z}}$ in the execution of π^ρ (which we denote by $\{REAL_{\pi^\rho, \mathcal{A}_\pi}(k, \bar{z}) | P_{\mathcal{Z}}\}_{k \in \mathbb{N}}$) and the output of $\mathcal{Z}$ in the UC real world are identically distributed. For every $z \in \{0,1\}^*$, $\bar{z} = (z, k_{enc}^1, k_{mac}^1, \ldots, k_{enc}^m, k_{mac}^m), \lambda, (k_{enc}^1, k_{mac}^1), \ldots, (k_{enc}^m, k_{mac}^m)$ is the vector where the first component is the input to $P_{\mathcal{Z}}$, the second component is the input to $P_{\mathcal{A}}$, and each of the other components is the input to a party P_i, with $i \in \{1, \ldots, m\}$.

We prove that for every $z \in \{0,1\}^*$, for every k_{enc}^i randomly chosen from $\{0,1\}^{q(n)}$ and for every k_{mac}^i generated by $Gen(1^{q(n)})$ we have:

$$\{EXEC_{\rho, \mathcal{A}, \mathcal{Z}}(k, z)\}_{k \in \mathbb{N}} \equiv \{REAL_{\pi^\rho, \mathcal{A}_\pi}(k, \bar{z}) | P_{\mathcal{Z}}\}_{k \in \mathbb{N}} \tag{6}$$

which as a special case, of course implies:

$$\{EXEC_{\rho,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}} \stackrel{\mathcal{D}}{\equiv} \{REAL_{\pi^{\rho},\mathcal{A}_{\pi}}(k,\bar{z})|P_{\mathcal{Z}}\}_{k\in\mathbb{N}} \tag{7}$$

Our claim is based on the following facts: First, the inputs to parties are provided by $\mathcal{Z}$ in both models, as in the composed protocol π^{ρ} the party $P_{\mathcal{Z}}$ distributing the inputs is internally running $\mathcal{Z}$. Thus the input messages in both worlds are identically distributed. By construction, $\mathcal{A}_{\pi}$ follows the instructions of $\mathcal{A}$ (i.e., for network scheduling and for the corrupted messages among the corresponding ITMs for $P_1,\ldots,P_m$) and it also provides an internal perfect emulation for the view of $\mathcal{A}$. Once an honest party P_i receives an input, it immediately writes it on the input tape of its associated ITM for ρ. This implies that such a party with its ITM follows the same protocol as the corresponding party of ρ. We can now conclude that the view of $\mathcal{Z}$ in the UC real world for ρ and the view of $P_{\mathcal{A}}$ in the composed protocol π^{ρ} are identically distributed, so equation (6) follows.

According to the definition of weak security under 1-bounded general concurrent composition, we know that for the triple π, $\mathcal{A}_{\pi}$ and $\mathcal{D}$, there exists a polynomially bounded hybrid simulator $\mathcal{S}_{\pi}$ such that for every $\bar{z}$ defined as above we have:

$$\{HYBRID^{\mathcal{F}}_{\pi,\mathcal{S}_{\pi}}(k,\bar{z})\}_{k\in\mathbb{N}} \stackrel{\mathcal{D}}{\equiv} \{REAL_{\pi^{\rho},\mathcal{A}_{\pi}}(k,\bar{z})\}_{k\in\mathbb{N}}. \tag{8}$$

We are now ready to construct a simulator $\mathcal{S}$ for the UC ideal world by using $\mathcal{S}_{\pi}$. We have to observe that in the hybrid world of concurrent general composition and in the UC real world the messages going over the network are the same. Intuitively, the new simulator $\mathcal{S}$ has to have a scheduling indistinguishable from that of $\mathcal{S}_{\pi}$ so the constructed simulator $\mathcal{S}$ internally invokes $\mathcal{S}_{\pi}$. As a short summary of the messages that have to be defined for $\mathcal{S}$: communication from $\mathcal{S}$ to $\mathcal{F}$, communication from $\mathcal{S}$ to $\mathcal{Z}$ and network scheduling (between parties of π and $\mathcal{F}$). As $\mathcal{S}$ internally runs $\mathcal{S}_{\pi}$, the constructed adversary has to provide an emulation for the entities that $\mathcal{S}_{\pi}$ is interacting with: the parties of $\pi^{\mathcal{F}}$[16]. Such an emulation of $\pi^{\mathcal{F}}$ consists of defining the input/output messages of the parties, the messages among $P_1,\ldots,P_m,P_{\mathcal{A}},P_{\mathcal{Z}}$ and the messages from $P_1,\ldots,P_m$ to $\mathcal{F}$.

[16] Observe that it is actually sufficient to simulate the parties of π without the messages sent by $\mathcal{F}$ as they can be forwarded by $\mathcal{S}$ from its communication with the ideal functionality.

1. Messages sent by $\mathcal{F}$ to $\mathcal{S}$ are forwarded to the internally emulated $\mathcal{S}_\pi$. The messages that internally emulated $\mathcal{S}_\pi$ wants to send to $\mathcal{F}$ are forwarded by $\mathcal{S}$ to $\mathcal{F}$. Similarly, the messages that internally emulated $\mathcal{S}_\pi$ sends to the internally simulated $P_\mathcal{Z}$ are forwarded by $\mathcal{S}$ to $\mathcal{Z}$. The messages that $\mathcal{Z}$ sends to $\mathcal{S}$ are forwarded internally to $\mathcal{S}_\pi$ as coming from $\mathcal{P}_\mathcal{Z}$.
2. Simulation of $P_\mathcal{Z}$: When $\mathcal{S}$ receives a message $(\mathcal{Z}, \mathcal{A}, m)$ from $\mathcal{Z}$, it sends it to the internally emulated $P_\mathcal{A}$ as if coming from the emulated $P_\mathcal{Z}$. When $\mathcal{S}_\pi$ instructs emulated $\mathcal{P}_\mathcal{A}$ (which is a corrupted party) to send a message $(\mathcal{A}, \mathcal{Z}, m)$ to $P_\mathcal{Z}$, the simulator $\mathcal{S}$ forwards the same message to $\mathcal{Z}$.
3. Simulation of $P_\mathcal{A}$: As an uncorrupted party, $\mathcal{P}_\mathcal{A}$ does not do anything, just receives messages from $\mathcal{P}_\mathcal{Z}$. These messages were actually sent by $\mathcal{Z}$ to $\mathcal{S}$. When internal $\mathcal{S}_\pi$ wants to corrupt emulated $P_\mathcal{A}$ (and this is actually the first party of π that $\mathcal{S}_\pi$ corrupts), then all that $\mathcal{S}$ needs to do is to send $\mathcal{S}_\pi$ all the messages it received from $\mathcal{Z}$.
4. In the UC ideal world, when an uncorrupted dummy party $\mathcal{D}_i$ receives an $(input, m_i)$ from the environment $\mathcal{Z}$, it immediately forwards the input value to $\mathcal{F}$. When $\mathcal{S}$ receives over the network such a message[17], it generates x_i randomly in the length of the received input and a MAC key k^i_{mac} with the corresponding generation algorithm, computes $t_i = MAC(k^i_{mac}, x_i)$ and internally sends the message $(input, x_i, t_i)$ to P_i as if coming from $P_\mathcal{Z}$. When $\mathcal{F}$ wants to send an output message (i.e., same discussion as above) to $\mathcal{D}_i$, the simulator $\mathcal{S}$ internally randomly generates y_i in the length of the output received over the network, then computes $v_i = MAC(k^i_{mac}, y_i)$ and sends message $(output, y_i, v_i)$ to $\mathcal{S}_\pi$ as if coming from the ideal functionality in $\pi^\mathcal{F}$.
   Whenever $\mathcal{S}_\pi$ corrupts a party P_i, we have one of the following 3 cases:
   -For a corrupted party P_i, that $\mathcal{S}_\pi$ wants to corrupt before a certain input is sent to it by $P_\mathcal{Z}$, the simulator $\mathcal{S}$ corrupts the corresponding dummy party $\mathcal{D}_i$, informs $\mathcal{Z}$ about it and generates a correct key pair (k^i_{enc}, k^i_{mac}) for encryption and authentication and gives them to $\mathcal{S}_\pi$[18]. When input value(s) x_i for $\mathcal{D}_i$ are received by $\mathcal{S}$ over the network[19], then $\mathcal{S}$ computes $y_i = x_i \oplus k^i_{enc}$ and $v_i = MAC(k^i_{mac}, y_i)$. Next, $\mathcal{S}$ sends y_i, v_i to $\mathcal{S}_\pi$ as coming from $P_\mathcal{Z}$. When an output o_i is sent by

[17] If the channels between the dummy parties and the ideal functionality are ideally secure, then the value received could also be encrypted, so what is forwarded should not depend on what is received.

[18] This simulates the information that $\mathcal{S}_\pi$ should learn from the newly corrupted (simulated) party.

[19] As $\mathcal{D}_i$ is corrupted, they are received from $\mathcal{Z}$ unencrypted.

$\mathcal{F}$ to $\mathcal{D}_i$, then $\mathcal{S}$ computes $c_i = x_i \oplus k^i_{enc}$ and $t_i = MAC(k^i_{mac}, y_i)$ and sends c_i, t_i to simulated P_i as if coming from $P_{\mathcal{Z}}$.

-For a corrupted party P_i, that $\mathcal{S}_\pi$ corrupts after a certain input is sent to P_i, but before the corresponding output is received, first the emulation from the case of uncorrupted input takes place. Thus, a message (y_i, v_i) has been already sent from $\mathcal{P}_{\mathcal{Z}}$ to P_i. When the corruption takes place, the simulator $\mathcal{S}$ corrupts the corresponding dummy party $\mathcal{D}_i$, informs $\mathcal{Z}$ about it and generates a correct key pair (k^i_{enc}, k^i_{mac}) for encryption and authentication. Then it sends the pair to $\mathcal{S}_\pi$, together with the correct input x_i in plain. When an output o_i is sent by $\mathcal{F}$ to $\mathcal{D}_i$, then $\mathcal{S}$ computes $c_i = x_i \oplus k^i_{enc}$ and $t_i = MAC(k^i_{mac}, y_i)$ and sends c_i, t_i to simulated P_i as if coming from $P_{\mathcal{Z}}$.

-For a corrupted party P_i that $\mathcal{S}_\pi$ corrupts after a certain input is sent to it and after the corresponding output is received, the simulator $\mathcal{S}$ corrupts the corresponding dummy party $\mathcal{D}_i$ and informs $\mathcal{Z}$ about the corruption[20]. Then $\mathcal{S}$ reads in plain the input and output values received by $\mathcal{D}_i$ and, using the simulated encrypted messages, computes the corresponding encryption keys which are sent to S_π as P_i input.

5. The following is valid only for honest parties: When $\mathcal{S}_\pi$ delivers a message from P_i to the ideal functionality in $\pi^{\mathcal{F}}$, then $\mathcal{S}$ delivers the same message from $\mathcal{D}_i$ to $\mathcal{F}$[21]. When $\mathcal{S}_\pi$ delivers an output from P_i to $P_{\mathcal{Z}}$, then $\mathcal{S}$ delivers the output from $\mathcal{F}$ to $\mathcal{D}_i$[22].

In order to conclude the proof we have to show that the output of the executions in both hybrid composition world and UC ideal world can be distinguished only with negligible probability. For this we detail the following three steps: a proof that the view of internally emulated $\mathcal{S}_\pi$ is identical with $\pi^{\mathcal{F}}$, a proof that the messages in the two worlds (hybrid composition and the UC ideal world) are identically distributed and finally, a proof that the delivery of output messages happens in the same time in both worlds.

We start by analyzing $\mathcal{S}$ internal emulation for $\mathcal{S}_\pi$. It is easy to see that by construction $\mathcal{S}_\pi$, internally invoked by $\mathcal{S}$, gets and delivers the same messages as $\mathcal{S}_\pi$ does in the concurrent general composition world.

[20] Note that the simulation done by $\mathcal{S}$ for uncorrupted P_i receiving encrypted and authenticated input and output from $\mathcal{P}_{\mathcal{Z}}$ already took place.

[21] Actually, the simulator $\mathcal{S}_\pi$ has to make two deliveries (from $P_{\mathcal{Z}}$ to P_i and from P_i to the ideal functionality in $\pi^{\mathcal{F}}$), before $\mathcal{S}$ does the delivery of message from $\mathcal{D}_i$ to $\mathcal{F}$.

[22] Similarly as above, the simulator $\mathcal{S}_\pi$ has to make two deliveries (the output of $\mathcal{F}$ to P_i and from P_i to $P_{\mathcal{Z}}$), before $\mathcal{S}$ does its delivery from $\mathcal{F}$ to $\mathcal{D}_i$.

Next, we look at the messages sent between entities in both worlds. In the ideal UC world, the inputs are sent by $\mathcal{Z}$ and in the hybrid world with $\pi^{\mathcal{F}}$, the inputs are sent by $P_{\mathcal{Z}}$ who runs $\mathcal{Z}$. The messages that are sent between $P_{\mathcal{Z}}$ (running $\mathcal{Z}$) and $P_{\mathcal{A}}$ (corrupted and controlled by $\mathcal{S}_\pi$), are the same as the messages sent in the UC ideal world between $\mathcal{Z}$ and $\mathcal{S}$ who runs $\mathcal{S}_\pi$. In both worlds, the messages sent by parties to the ideal functionality are the same: the honest parties just forward their inputs and the corrupted parties are instructed by $\mathcal{S}_\pi$ and respectively by $\mathcal{S}$ running $\mathcal{S}_\pi$. We only need to show that the delivery of messages is the same in both worlds. Combining this claim with the proof above, we obtain that the outputs of both worlds are computationally indistinguishable.

Finally, we compare message delivery in both worlds. It is clear that the messages between adversary and the environment $\mathcal{Z}$ or party $P_{\mathcal{Z}}$ running $\mathcal{Z}$ are identically delivered. The same hods for messages between the parties and the ideal functionality. We treat in more detail the case of inputs and outputs delivery. By definition, in the UC world, the input messages are written by $\mathcal{Z}$ directly on the input tapes of the protocol parties and for the honest parties, the adversary has no control over this step[23]. In the execution of $\pi^{\mathcal{F}}$, $P_{\mathcal{Z}}$ is distributing the inputs to the rest of the parties, but they are scheduled by $\mathcal{S}_\pi$, so we cannot know when they are delivered. However, we ensure that in both worlds an input of an honest party reaches the ideal functionality in the same time. Indeed, this holds as an honest dummy party $\mathcal{D}_i$ once it receives its input, it immediately sends it to the ideal functionality. As simulator $\mathcal{S}$ delivers this message only after $\mathcal{S}_\pi$ has delivered the same message to $\mathcal{F}$, we have shown the claim.

Similarly, we show that an output message is delivered to $\mathcal{Z}$ and to the party $P_{\mathcal{Z}}$ in the same time. Both entities have basically the same instructions. We assume the machine environment $\mathcal{Z}$ reads all output tapes whenever it is activated. This gives the most power to the environment to distinguish between the delivery of messages. By construction, $\mathcal{S}$ sends an output of $\mathcal{F}$ to an honest dummy party $\mathcal{D}_i$ only when $\mathcal{S}_\pi$ sends the same output to $P_{\mathcal{Z}}$. Once it receives its output, the honest $\mathcal{D}_i$ immediately writes this value on its output tape (and this can be read by $\mathcal{Z}$ at any time). Analogously, $\mathcal{Z}$, (which is internally run by $P_{\mathcal{Z}}$), can read at

[23] However, in the UC ideal world, immediately after receiving inputs, the honest dummy parties are activated and they write their inputs on the communication tape for the ideal functionality. As the simulator is responsible for the delivery of messages, in this way it will learn that inputs have been sent to the ideal functionality.

any time the tape with output messages sent for it. So we have that also the outputs from the ideal functionality are delivered simultaneously in both worlds. This implies that for every $\bar{z}$ defined as before we have:

$$\{HYBRID^{\mathcal{F}}_{\pi,\mathcal{S}_\pi}(k,\bar{z})|P_{\mathcal{Z}}\}_{k\in\mathbb{N}} \equiv \{EXEC_{\mathcal{F},\mathcal{S}_\pi,\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}} \tag{9}$$

Thus, it holds that:

$$\{HYBRID^{\mathcal{F}}_{\pi,\mathcal{S}_\pi}(k,\bar{z})|P_{\mathcal{Z}}\}_{k\in\mathbb{N}} \stackrel{\mathcal{D}}{\equiv} \{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N}} \tag{10}$$

By combining relations (7),(8) and (10), we can conclude the proof.

As a consequence of theorem 1 and of theorem 2, we are now also able to compare the notion of 1-bounded concurrent general composition security [25] with our variant, i.e., weak 1-bounded concurrent general composition security.

Corollary 1 (Weak 1-bounded CGC and 1-bounded CGC Not Equivalent). *Assume Blum integer time-lock puzzles exist. Then there are protocols secure with respect to weak 1-bounded concurrent general composition which are not secure with respect to 1-bounded concurrent general composition.*

Next we show that the approach taken in Theorem 2 is not an overkill. Indeed, there are protocols that are secure with respect to weak stand-alone security but they are not secure anymore in the standard stand-alone model.

Lemma 6 (Weak Security Does Not Imply Stand-alone Security). *If Blum integer time-lock puzzles exist, then there are protocols that fulfill the weak security notion, but do not fulfill the stand-alone security notion.*

Proof. From theorem 2, weak security under 1-bounded concurrent general composition is equivalent to 1-bit specialized simulator UC. As shown in [25], stand-alone security under 1-bounded concurrent general composition is equivalent to specialized simulator UC. According to theorem 1, the two UC variants are not equivalent. This implies weak security and stand-alone security are also not equivalent. One may wonder if the equivalence result between UC security and specialized simulator UC security that is known to hold in the extended UC model does not hinder the correctness of this result. However, this is not the case. On one hand, in the extended UC model, specialized simulator UC security and UC security are equivalent. Combining this with the well known result of equivalence between UC security and 1-bit UC security, we obtain that in the extended UC model, specialized simulator UC security and 1-bit specialized UC security are equivalent. This equivalence should not look surprising, as it is obtained in a more "permissive" adversarial UC model. On the other hand, the results obtained in this work show that there is at least one composition operation under which weak security and stand-alone security are not equivalent.

The results of Lemma 7 and Lemma 8 complete Fig. 1.

Lemma 7 (Weak Stand-alone Security Does Not Imply Weak 1-bounded CGC Security). *There exists a protocol π which is secure with respect to weak stand-alone model, but is not secure with respect to weak 1-bounded concurrent general composition security.*

Proof. The proof is based on the same idea presented in [6].

As shown in Sect. 5.1, the next security result is essential for establishing the relation between the existing game-theoretic notion of strong universal implementation [20] and our notion of game universal implementation. As a preamble, we first give the intuition for weak precise secure computation. While the traditional notion of secure computation [14] requires only the worst case running time complexity of the ideal world simulator to match the running time of the real world adversary, weak precise secure computation [29] requires the complexity of the simulator to match the complexity of the real world adversary, for each arbitrary distinguisher and input.

Definition 17 (Weak Precise Secure Computation). *Let π be a protocol, $\mathcal{F}$ an ideal functionality and let $\mathcal{C}$ be the function that given a security parameter k, a polynomially bounded party Q and the view v of Q in the protocol π, it computes the complexity of Q running with k and v. We say that π is a weak precise secure computation of $\mathcal{F}$ if there exists a polynomial p such that for every real world adversary $\mathcal{A}$, for every distinguisher $\mathcal{D}$ and for every input z, there exists an ideal simulator $\mathcal{S}$, with $\mathcal{C}(k, \mathcal{S}, v) \leq p(k, \mathcal{C}(k, \mathcal{A}, \mathcal{S}(v)))$ such that :*

$$\{IDEAL(k, z, \mathcal{S}, \mathcal{F})\}_{k\in\mathbb{N}} \stackrel{D}{\equiv} \{REAL(k, z, A, \overrightarrow{M}\}_{k\in\mathbb{N}}.$$

Lemma 8 (Weak Precise Secure Computation Does Not Imply Weak Stand-alone Security). *If Blum integer time-lock puzzles exist,* *then there exists a protocol* π *which is secure with respect to weak precise secure computation, but is not secure with respect to weak stand-alone security.*

Proof. The proof follows the general lines of the constructions that we have used for our main separation result in lemma 4, however, as expected, the details are much more straight forward.

Let π be such that on an input pair (k, t), where k is the security parameter, it truncates t to the first k bits obtaining t' and generates $Gen(k, t') = (q', a')$. Then it sends q' to A and regardless of the reply received from the adversary, it outputs 1.

In the ideal world, on an input pair (k, t), the ideal functionality $\mathcal{F}$ behaves exactly like π with the only exception that it outputs 1 if and only if it receives from the adversary it interacts with, i.e., from $\mathcal{S}$, the correct solution v' from the puzzle. Otherwise $\mathcal{F}$ outputs 0.

Given π and $\mathcal{F}$ as defined above, first we show that π is not as secure as $\mathcal{F}$ with respect to weak stand-alone security. Assume by contradiction that weak stand-alone security property holds. Since in the real world π outputs 1, in the ideal world the ideal functionality should output 1 with overwhelming probability. This in turn means that $\mathcal{S}$ should produce the correct solution for the puzzle sent by $\mathcal{F}$ with overwhelming probability. However, this should be the case independent of the input t'. For a fixed polynomially bounded simulator $\mathcal{S}$, there is a polynomially bounded hardness $t_{\mathcal{S}}$ for the puzzles that is can solve. However, by the definition of the time-lock puzzles, if the input $t' > t_{\mathcal{S}}$ then the simulator fails to reply correctly to the challenge sent by $\mathcal{F}$ with overwhelming probability. In conclusion, for a given simulator there is always an input that the real and the ideal world are distinguishable with non-negligible probability, thus our assumption is false.

Second, we prove that π is as secure as $\mathcal{F}$ with respect weak precise secure computation. For ease of presentation, we assume that $\mathcal{C}$ represents the run time complexity function. In order to show this claim we make the following observation: From relation (4) we deduce that for every integer d there exists an integer p_d and a polynomial time solver C_d with run time at most p_d when solving puzzles of hardness k^d. By induction, it is easy to see that there is a polynomial *poly* such that for every d there is a polynomial solver C'_d such that the run time of C'_d is at most $poly(d)$ when solving puzzles of hardness k^d. This polynomial *poly* we can use as polynomial p in the definition of weak precise secure computation. Given an adversary $\mathcal{A}$, a distinguisher $\mathcal{D}$ and an input t, it is easy to see that if we take simulator $\mathcal{S}$ such that it has hardness at least t, then the real and the ideal world will be indistinguishable for $\mathcal{D}$. And this concludes our proof.

5.1 Relation Between 1-bit Specialized Simulator UC and Game Universal Implementation

In the following we prove an equivalence result between game universal implementation and our definition of weak security. A similar result exists in connection with strong universal implementation [20], but that notion considers a refined version for computational games, where the utility of the players may have strong correlations with the complexity of the computation they perform (e.g., time complexity, memory complexity, communication complexity or complexity of operations like reading inputs or copying messages). The general idea of our proof is related to the one used in [20].

Theorem 3 (Equivalence Between Game Universal Implementation and Weak Stand-alone Security). *Let comm be the communication mediator represented by the cryptographic notion of ideally secure channels. Let f be an m-ary function with the property that outputs the empty string to a party if and only if it had received the empty string from that party. Let $\mathcal{F}$ be a mediator that computes f* [24] *and let $\overrightarrow{M}$ be an abort-preserving computation of f*[25]*. Then $\overrightarrow{M}$ is a weak secure computation of f*[26] *with respect to statistical security if and only if $(\overrightarrow{M}, comm)$ is a game universal implementation of $\mathcal{F}$ with respect to Games, where Games is*

---

[24] The ideal machine profile $\overrightarrow{\Lambda^{\mathcal{F}}}$ *computes* f if for all $n \in \mathbb{N}$, all inputs $\overrightarrow{x} \in (\{0,1\}^n)^m$, the output vector of the players after an execution of $\overrightarrow{\Lambda^{\mathcal{F}}}$ on input $\overrightarrow{x}$ is identically distributed to $f(\overrightarrow{x})$.

[25] $\overrightarrow{M}$ is an abort-preserving computation of f if for all $n \in \mathbb{N}$ and for all inputs $\bar{x} \in (\{0,1\}^n)^m$, the output vector of the players after an execution of $(\perp, \overrightarrow{M_{-Z}})$ on input $\bar{x}$ is identically distributed to $f(\lambda, \bar{x}_{-Z})$, where Z is a subset of all parties and λ is the empty string.

[26] We call $\overrightarrow{M}$ a weak secure computation of f if the following two properties are fulfilled:

- For all $n \in \mathbb{N}$, all inputs $\overrightarrow{x} \in (\{0,1\}^n)^m$, the output vector of the players after an execution of $\overrightarrow{M}$ on input $\overrightarrow{x}$ is distributed statistically close to $f(\overrightarrow{x})$;
- For every adversary A and for every distinguisher D, there exists a simulator $\mathcal{S}$ such that for every input z, the following relation is fulfilled :

$$\{IDEAL(k, z, \mathcal{S}, \mathcal{F})\}_{k\in\mathbb{N}} \stackrel{D}{\equiv} \{REAL(k, z, A, \overrightarrow{M}\}_{k\in\mathbb{N}}.$$

In the second property, the indistinguishability relation can be further detailed with respect to perfect, statistical or computational security.

the class of games for which the utility functions of the players depend only on players types and on the output values.

Proof. In this proof, without loss of generality, we assume that the ideal functionality $\mathcal{F}$ outputs to each of the parties in the ideal world the output of the computation for each of their input together with their input value. More formally, if party i sends $\mathcal{F}$ as input the value x_i, it receives from $\mathcal{F}$ as output $f_i(x_1, \ldots, x_m); x_i$, where by ";" we denote the concatenation of strings.

First we prove that if $\overrightarrow{M}$ is a weak secure computation of f with statistical security, then $(\overrightarrow{M}, comm)$ is a strong *Games* universal implementation of $\mathcal{F}$. Since both $\overrightarrow{M}$ and $\overrightarrow{\Lambda^{\mathcal{F}}}$ compute f, according to the definition, it means they have statistically close output distributions. So the second property contained in the definition of game universal implementation has been proven. Next we prove that $\forall i \in \{1, \ldots, n\}$, there exists a negligible function ϵ_i and an integer k_i such that for all $k \geq k_i$:

$$U_i(k, \overrightarrow{M}) = U_i(k, \overrightarrow{\Lambda^{\mathcal{F}}}) + \epsilon_i(k). \tag{11}$$

We prove that below. Let $\overrightarrow{t}$ be the vector of inputs and $\overrightarrow{o}$ be the vector of outputs. Then we have:

$$\begin{aligned}
&|U_i(k, \overrightarrow{M}) - U_i(k, \overrightarrow{\Lambda^{\mathcal{F}}})| = \\
&= |\sum_{\overrightarrow{t}, \overrightarrow{o}} [Pr(REAL(k, \overrightarrow{M}) = \overrightarrow{o}) - Pr(IDEAL(k, \overrightarrow{\Lambda^{\mathcal{F}}}) = \overrightarrow{o})] \cdot u_i(k, \overrightarrow{t}, \overrightarrow{o})| \leq \\
&\leq [\sum_{\overrightarrow{t}, \overrightarrow{o}} |Pr(REAL(k, \overrightarrow{M}) = \overrightarrow{o}) - Pr(IDEAL(k, \overrightarrow{\Lambda^{\mathcal{F}}}) = \overrightarrow{o})|] \cdot p_i(k) \leq \\
&\leq \epsilon(k) \cdot p_i(k) = \\
&= \epsilon_i(k).
\end{aligned}$$

In the inequalities above, we have used the following facts: The output distributions in the real and in the ideal world are finite and statistically close and the utility function u_i is bounded above by a polynomial p_i. Hence, equation (11) holds.

Also, in an analogous manner, the following equation

$$U_Z(k, \overrightarrow{M}) = U_Z(k, \overrightarrow{\Lambda^{\mathcal{F}}}) + \epsilon_Z(k).$$

trivially holds, where Z can be any subset of players.

We are now ready to show the left to right implication of the theorem in two steps, by following the remaining properties in the definition of game universal implementation. If $\overrightarrow{\Lambda^{\mathcal{F}}}$ is a computational Nash equilibrium in the mediated game $(G, \mathcal{F})$, assume by contradiction that $\overrightarrow{M}$ is not a Nash equilibrium in $(\overrightarrow{M}, comm)$. Then there exists a player i, a deviating strategy A_i, a polynomial p_i such that for every k_0 there exists $k \geq k_0$ with the property:

$$U_i(k, A_i, \overrightarrow{M_{-i}}) > U_i(k, \overrightarrow{M}) + \frac{1}{p_i(k)}. \tag{12}$$

Due to equation (11) and also due to the hypothesis that $\overrightarrow{\Lambda^{\mathcal{F}}}$ is a computational Nash equilibrium (this is where the following negligible function ϵ comes from) we additionally have:

$$\begin{aligned} U_i(k, \overrightarrow{M}) + \frac{1}{p_i(k)} + \epsilon(k) &= U_i(k, \overrightarrow{\Lambda^{\mathcal{F}}}) + \frac{1}{p_i(k)} + \epsilon_i(k) + \epsilon(k) \\ &\geq U_i(k, \mathcal{S}_i, \overrightarrow{\Lambda^{\mathcal{F}}_{-i}}) + \frac{1}{p_i(k)}, \end{aligned} \tag{13}$$

for every simulator $\mathcal{S}_i$. Thus we obtain $U_i(k, A_i, \overrightarrow{M_{-i}}) > U_i(k, \mathcal{S}_i, \overrightarrow{\Lambda^{\mathcal{F}}_{-i}}) + \frac{1}{p'_i(k)}$, for every $\mathcal{S}_i$. Given the hypothesis on polynomially bounded utility functions, for every simulator $\mathcal{S}_i$, the output distributions of $REAL(k, A_i, \overrightarrow{M_{-i}})$ and $IDEAL(k, \mathcal{S}_i, \mathcal{F})$ are not statistically close (This can be shown in an analogous way as relation (11)). So for every $\mathcal{S}_i$, there exists a distinguisher $\mathcal{D}_i$ such that

$$\{IDEAL(k, \mathcal{S}_i, \mathcal{F})\}_{k\in\mathbb{N}} \overset{\mathcal{D}_i}{\not\equiv} \{REAL(k, A_i, \overrightarrow{M_{-i}}\}_{k\in\mathbb{N}}, \tag{14}$$

i.e., $\mathcal{D}_i$ (that can be also computationally unbounded) has non-negligible probability to distinguish between the ensembles above.

Let *Sim* be the set of all simulators $\mathcal{S}_i$ and let *Dist* be the set of all distinguishers $\mathcal{D}_i$ as described above. Let $\mathcal{D}$ be the distinguisher that runs every distinguisher in the set *Dist* and outputs 1 if and only if at least one of the distinguishers in the set *Dist* outputs 1. Such a $\mathcal{D}$ is obviously computationally unbounded, but is a viable distinguisher in relation with the definition of weak security with statistical security.

By the definition of weak security, for adversary A_i and for distinguisher $\mathcal{D}$, there exists a simulator $\mathcal{S}$ such that the following ensembles are statistically close distributed:

$$\{IDEAL(k, \mathcal{S}, \mathcal{F})\}_{k\in\mathbb{N}} \overset{\mathcal{D}}{\equiv} \{REAL(k, A_i, \overrightarrow{M_{-i}}\}_{k\in\mathbb{N}}$$

But since $\mathcal{D}$ also runs the distinguisher that can tell apart between the world with $\mathcal{S}$ and the world with A_i, the above relation is a contradiction with the definition of $\mathcal{D}$. Thus if $\overrightarrow{\Lambda^{\mathcal{F}}}$ is a computational Nash equilibrium, then so is $\overrightarrow{M}$. It can be shown in an analogous way, mainly by substituting the deviating player i with any set Z of deviating players, that if $\overrightarrow{\Lambda^{\mathcal{F}}}$ is immune to the coalition in Z, then $\overrightarrow{M}$ is also immune to the coalition in Z.

Finally, we look at the preservation of $\perp_i$ as a best response for a party i in $\overrightarrow{M}$. As $\overrightarrow{M}$ is abort preserving and f has the empty string property, $U_i(\perp_i, \overrightarrow{M}_{-i}) = U_i(\perp_i, \overrightarrow{\Lambda^{\mathcal{F}}_{-i}})$. On the other hand, if playing $\perp_i$ is the best response to $\overrightarrow{\Lambda^{\mathcal{F}}_{-i}}$, (i.e., $\perp_i$ gives the highest utility for player i in the world with $\overrightarrow{\Lambda^{\mathcal{F}}_{-i}}$ up to a negligible value), due to the weak security property and by using the same technique based on distinguishers' properties as in the preservation of computational Nash equilibrium above, the highest utility for party i in the real world is $U_i(\perp_i, \overrightarrow{M}_{-i})$ up to a negligible value. This concludes the implication from left to right.

For the implication from right to left, we follow the case-separation idea of the proof for theorem 4.2 (Information Theoretic Case) in [19] and we specify below the details.

Assume by contradiction that $\overrightarrow{M}$ is not a weak secure computation of f with statistical security. Thus, there exists a set Z of corrupted parties, there exists an adversary A_Z corrupting the parties in Z and a distinguisher D (possibly unbounded) such that for every simulator $\mathcal{S}$ there exists a polynomial $p_{\mathcal{S},Z}$ such that for every integer k_0 there exists $k \geq k_0$:

$$\begin{aligned} &Pr(D(k, REAL(k, A_Z, \overrightarrow{M}_{-Z})) = 1) - \\ &\qquad - Pr(D(k, IDEAL(k, \mathcal{S}, \mathcal{F})) = 1) > \frac{1}{p_{\mathcal{S},Z}(k)} \end{aligned} \tag{15}$$

As in [19], we distinguish between two cases:

Case 1: $A_Z = \perp_Z$

The proof idea in this case is to design a game in the class *Games*, with utilities depending on $\mathcal{D}$ such that $\overrightarrow{\Lambda^{\mathcal{F}}}$ is a computational Nash equilibrium (with immunity with respect to coalitions). By hypothesis, this implies that $\overrightarrow{M}$ is a computational Nash equilibrium (with immunity with respect to coalitions). However, for the constructed game, we obtain that $\perp_Z$ is the best response to $\overrightarrow{M_{-Z}}$, which represents a contradiction.

Let $d = Pr(D(k, IDEAL(k, \perp_Z, \mathcal{F})) = 1)$. We denote by $\overrightarrow{t}$ the inputs of the parties, which in game-theoretic terms correspond to the secret types of the players; and we denote by $\overrightarrow{o}$ the outputs of the parties, which in game-theoretic terms correspond to the actions taken by the players. In the following, by o_Z and by λ_Z respectively, we denote the input for parties in Z and the empty string corresponding to the output of the parties in Z.

Next, we define a game G such that for any subset of players $Z' \neq Z$, we have $u_{Z'} = 0$ and for the set Z we have:

$$u_Z(k, \overrightarrow{t}, \overrightarrow{o}) = \begin{cases} Pr(D(k, \overrightarrow{t}, \overrightarrow{o}) = 1) & \text{if } \overrightarrow{o_Z} = \lambda_Z \\ \\ d & \text{otherwise} \end{cases}$$

We show for the game G the strategy $\overrightarrow{\Lambda^{\mathcal{F}}}$ is a computational Nash equilibrium in the ideal world. Indeed, for any subset of players $Z' \neq Z$, we have that $U_{Z'}(k, \overrightarrow{\Lambda^{\mathcal{F}}}) = 0 = U_{Z'}(k, \mathcal{S}_{Z'}, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z'}})$, for any simulator $\mathcal{S}$. For the set Z, on one hand we have $U_Z(k, \overrightarrow{\Lambda^{\mathcal{F}}}) = d$, as following the strategy $\overrightarrow{\Lambda^{\mathcal{F}}}$ does result in an "empty" output for the parties in Z only with negligible probability (i.e., if and only if the inputs to all the parties in Z are also the empty string). On the other hand, we have that:

$$U_Z(k, \mathcal{S}_Z, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}) = \begin{cases} Pr(D(k, IDEAL(k, \perp_Z, \mathcal{F})) = 1) & \text{if } \mathcal{S}_Z = \perp_Z \\ \\ d + \epsilon_{\mathcal{S},Z}(k) & \text{otherwise,} \end{cases}$$

where for every $\mathcal{S}_Z$, $\epsilon_{\mathcal{S},Z}$ is a negligible function.

Hence $U_Z(k, \overrightarrow{\Lambda^{\mathcal{F}}}) + \epsilon_{\mathcal{S},Z}(k) \geq U_Z(k, \mathcal{S}_Z, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z}})$, for every $\mathcal{S}_Z$. To summarize, $\overrightarrow{\Lambda^{\mathcal{F}}}$ is a Nash equilibrium with immunity with respect to coalitions. Adding the hypothesis of $(\overrightarrow{M}, comm)$ being a game universal implementation of $\mathcal{F}$, we obtain that $\overrightarrow{M}$ is a Nash equilibrium with immunity with respect to coalitions. But $\overrightarrow{M}$ and $\overrightarrow{\Lambda^{\mathcal{F}}}$ have statistically close output distributions, so similar to (11) we conclude $U_Z(k, \overrightarrow{M}) = d + \epsilon_Z(k)$. However, $U_Z(k, \perp_Z, \overrightarrow{M}_{-Z}) = Pr(D(k, REAL(k, \perp_Z, \overrightarrow{M}_{-Z})) = 1)$. By assumption (15), $Pr(D(k, REAL(k, A_Z, \overrightarrow{M}_{-Z})) = 1) >$

$Pr(D(k, IDEAL(k, \mathcal{S}, \mathcal{F})) = 1) + \frac{1}{p_{\mathcal{S},Z}(k)}$, for every simulator $\mathcal{S}$. Thus, $U_Z(k, \bot_Z, \overrightarrow{M_{-Z}}) > d + \frac{1}{p_{\mathcal{S},Z}(k)} - \epsilon(k) = d + \frac{1}{p'_{\mathcal{S},Z}(k)}$.

As this contradicts the equilibrium property of $\overrightarrow{M}$, we conclude the first case.

Case 2: $A_Z \neq \bot_Z$

Without loss of generality, we assume that A_Z lets one of the players in Z output the entire view of the adversary A_Z. Indeed, we can construct A'_Z from A_Z such that besides the output for each of the parties in Z, the first player in Z also outputs $v' = A_Z(v)$. If we define the distinguisher D' such that

$$D'(k, REAL(k, A_Z(v), \overrightarrow{M}_{-Z}); v') = D(k, REAL(k, A_Z(v), \overrightarrow{M}_{-Z}))$$

and

$$D'(k, IDEAL(k, \mathcal{S}(v),)\overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}; v') = D'(k, IDEAL(k, \mathcal{S}(v),)\overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}),$$

then the property (15) fulfilled by D is also fulfilled by D'. So we can assume A_Z lets one of the players in Z output the entire view of A_Z.

Let $d = up^{s}_{\mathcal{S}_Z} Pr(D(k, IDEAL(k, \mathcal{S}_Z, \mathcal{F})) = 1)$. We construct a game H in the following way. For any subset of players $Z' \neq Z$, the utility corresponding to the coalition Z' is 0, independent of the parties inputs and outputs. Then we define:

$$u_Z(k, \overrightarrow{t}, \overrightarrow{o}) = \begin{cases} d & \text{if } \overrightarrow{o_Z} = \lambda_Z \\ Pr(D(k, \overrightarrow{t}, \overrightarrow{o}, v) = 1) & \text{if } \exists\, o_{i_Z} = o'_{i_Z}; v \text{ and } \overrightarrow{o'_Z} \neq \lambda_Z \\ 0 & \textit{otherwise} \end{cases}$$

where for every $j_Z \neq i_Z$, $o'_{j_Z} = o_{j_Z}$.

We prove that for the game H defined above, $\bot_Z$ is the best response to $\overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}$. Assume by contradiction this does not hold. Let the simulator $\mathcal{S}^{best}_Z$ be such that the strategy it implements for the parties in Z is the best response to $\overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}$. This implies $\mathcal{S}^{best}_Z \neq \bot_Z$ and $U_Z(k, \mathcal{S}^{best}_Z, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}) > U_Z(k, \bot_Z, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}) + \frac{1}{p_Z(k)} = d + \frac{1}{p_Z(k)}$. From the last relation we can conclude that $IDEAL(k, \mathcal{S}^{best}_Z, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z}})$ and $IDEAL(k, \bot_Z, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z}})$ are not statistically close distributions.

Hence, the expected utility $U_Z(k, \mathcal{S}_Z^{best}, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z}})$ can be computed using the second or the third branch of the definition of the utility function u_Z. Thus,

$$U_Z(k, \mathcal{S}_Z^{best}, \overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}) \leq Pr(D(k, IDEAL(k, \mathcal{S}_Z^{best}, \mathcal{F})) = 1).$$

So $d < Pr(D(k, IDEAL(\mathcal{S}_Z^{best}, \mathcal{F})) = 1)$, which is a contradiction with the definition of d, so $\perp_Z$ is the best response to $\overrightarrow{\Lambda^{\mathcal{F}}_{-Z}}$.

By hypothesis of the current implication, we conclude $\perp_Z$ is the best response to $\overrightarrow{M_{-Z}}$. However, we show that $U_Z(k, A_Z, \overrightarrow{M_{-Z}}) > U_Z(k, \perp_Z, \overrightarrow{M_{-Z}}) + \frac{1}{p(k)}$, which is an obvious contradiction.

Indeed, on one hand:

$$\begin{aligned} &U_Z(k, A_Z, \overrightarrow{M_{-Z}}) = Pr(D(k, REAL(k, A_Z, \overrightarrow{M_{-Z}}))) > \\ &> up^s_{\mathcal{S}_Z} Pr(D(k, IDEAL(k, \mathcal{S}_Z, \mathcal{F})) = 1) + \frac{1}{p_{\mathcal{S},Z}(k)} \\ &= d + \frac{1}{p_{\mathcal{S},Z}(k)}. \end{aligned}$$

On the other hand, $U_Z(k, \perp_Z, \overrightarrow{M_{-Z}}) = d$ due to the definition of the utility function u_Z, so the contradiction is obvious.

Our proof needs to clarify only one last point. What happens in the case that Z is the empty set, or to put it equivalently, A_Z is the empty adversary $\perp$. Then the assumption in equation (15) becomes: $Pr(D(k, REAL(k, \perp, \overrightarrow{M})) = 1) - Pr(D(k, IDEAL(k, \mathcal{S}, \mathcal{F})) = 1) > \frac{1}{p_S(k)}$, for every simulator $\mathcal{S}$. But this directly contradicts the fact that $\overrightarrow{M}$ and $\overrightarrow{\Lambda^{\mathcal{F}}}$ have statistically close output distributions.

Hence theorem 3 has been proven.

So we have shown that by restricting the class of games to those for which the computation cost for parties during protocol execution is free, our variant of universal implementation becomes equivalent to more standard notions of security (i.e., where the simulator depends only on the distinguisher and not anymore on both the distinguisher and input). Finding such equivalences was stated as an open question in [20] and to the best of our knowledge, our work makes the first step towards answering it.

One may ask if our new notion of game universal implementation is a subcase of the existing notion of strong universal implementation [20].

Using Lemma 8, Theorem 3 and the equivalence result between strong universal implementation and weak precise secure computation [20], we obtain:

Corollary 2 (Non-Equivalence of Universal Implementation Variants). *The notion of strong universal implementation does not imply the notion of game universal implementation.*

6 Conclusions

In this work we have established an equivalence result between a security notion (i.e., weak stand-alone security) and a game-theoretic notion (i.e., game universal implementation). In the process of deducing this result, we had a closer look at the implication relations among a wider set of security notions, including variants of the UC security definition. Along this path, an important result was the proof that two variants of the UC security definition where the order of quantifiers is reversed, namely 1-bit specialized simulator UC security and specialized simulator UC security, are not equivalent. This comes in contrast with the well known result that UC security and 1-bit UC security are equivalent [5] and solves the open question raised by Lindell [25] about the implication relation between the two above mentioned UC variants.

Based on the results mentioned above, as future work it is worth investigating whether one can add "composability properties" to game universal implementation in order to derive a game-theoretic notion equivalent to 1-bit specialized simulator UC.

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: 25th Annual ACM Symposium on Principles of Distributed Computing (PODC'06). pp. 53–62. ACM (2006)
2. Backes, M., Pfitzmann, B., Waidner, M.: A general composition theorem for secure reactive systems. In: First Theory of Cryptography Conference (TCC'04). pp. 336–354. Springer (2004)
3. Beaver, D.: Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. Journal of Cryptology 4(2) (1991)
4. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (2000)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd IEEE symposium on Foundations of Computer Science (FOCS '01). pp. 136–145. IEEE Computer Society (2001)

6. Canetti, R., Fischlin, M.: Universally composable commitments. In: 21st Annual International Cryptology Conference (CRYPTO'01). pp. 19–40. Springer (2001)
7. Dodis, Y., Halevi, S., Rabin, T.: A cryptographic solution to a game theoretic problem. In: 20th Annual International Cryptology Conference (CRYPTO'00). pp. 112–130. Springer (2000)
8. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. Journal of the ACM 50(6), 852–921 (2003)
9. Feige, U.: Alternative Models For Zero Knowledge Interactive Proofs (PhD Thesis). Ph.D. thesis, Weizmann Institute of Science (1992)
10. Feigenbaum, J., Shenker, S.: Distributed algorithmic mechanism design: recent results and future directions. In: 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M'02). pp. 1–13. ACM (2002)
11. Fuchsbauer, G., Katz, J., Naccache, D.: Efficient rational secret sharing in standard communication networks. In: 7th Theory of Cryptography Conference (TCC'10). pp. 419–436. Springer (2010)
12. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st Anual ACM Symposium on Theory of Computing (STOC'89). pp. 25–32. ACM (1989)
13. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science (FOCS'86). pp. 174–187. IEEE Computer Society (1986)
14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: 19th Annual ACM Symposium on Theory of Computing (STOC'87). pp. 218–229. ACM (1987)
15. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7(1), 1–32 (1994)
16. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
17. Gordon, D., Katz, J.: Rational secret sharing, revisited. In: 5th Conference on Security and Cryptography for Networks (SCN'06). pp. 229–241. Springer (2006)
18. Halpern, J., Teague, V.: Rational secret sharing and multiparty computation: extended abstract. In: 36th Annual ACM Symposium on Theory of Computing (STOC'04). pp. 623–632. ACM (2004)
19. Halpern, J.Y., Pass, R.: A computational game-theoretic framework for cryptography. unpublished manuscript (2010)
20. Halpern, J.Y., Pass, R.: Game theory with costly computation: Formulation and application to protocol security. In: Innovations in Computer Science (ICS'10). pp. 120–142. Tsinghua University Press (2010)
21. Hirt, M., Maurer, U.: Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In: 6th Annual ACM Symposium on Principles of Distributed Computing (PODC '97). pp. 25–34. ACM (1997)
22. Hofheinz, D., Unruh, D.: Comparing two notions of simulatability. In: 2nd Theory of Cryptography Conference (TCC'05). pp. 89–103. Springer (2005)
23. Katz, J.: Bridging game theory and cryptography: Recent results and future directions. In: 5th Theory of Cryptography Conference (TCC'08). pp. 251–272. Springer (2008)
24. Kol, G., Naor, M.: Cryptography and game theory: Designing protocols for exchanging information. In: 5th Theory of Cryptography Conference (TCC'08). pp. 320–339. Springer (2008)

25. Lindell, Y.: General composition and universal composability in secure multiparty computation. In: 44th Symposium on Foundations of Computer Science (FOCS'03). pp. 394–403. IEEE Computer Society (2003)
26. Mahmoody, M., Moran, T., Vadhan, S.P.: Non-interactive time-stamping and proofs of work in the random oracle model. IACR Cryptology ePrint Archive 2011, 553 (2011)
27. Mahmoody, M., Moran, T., Vadhan, S.P.: Time-lock puzzles in the random oracle model. In: 31st Annual Cryptology Conference(CRYPTO'11). pp. 39–50. Springer (2011)
28. Megiddo, N., Wigderson, A.: On play by means of computing machines: preliminary version. In: Proceedings of the 1986 conference on Theoretical aspects of reasoning about knowledge (TARK '86). pp. 259–274. Morgan Kaufmann Publishers Inc. (1986)
29. Micali, S., Pass, R.: Local zero knowledge. In: 38th Anual ACM Symposium on Theory of Computing (STOC'06). pp. 306–315. ACM (2006)
30. Micali, S., Rogaway, P.: Secure computation (abstract). In: 1th Annual International Cryptology Conference (CRYPTO'91). pp. 392–404. Springer (1991)
31. Neyman, A.: Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma. Economics Letters 19(3), 227–229 (1985)
32. Pfitzmann, B., Schunter, M., Waidner, M.: Cryptographic security of reactive systems. Electr. Notes Theor. Comput. Sci. 32 (2000)
33. Pfitzmann, B., Waidner, M.: A general framework for formal notions of secure systems. http://www.semper.org/sirene/lit. (1994)
34. Pfitzmann, B., Waidner, M.: Composition and integrity preservation of secure reactive systems. In: 7th ACM conference on Computer and Communications Security (CCS'00). pp. 245–254. ACM (2000)
35. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: IEEE Symposium on Security and Privacy. pp. 184– (2001)
36. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Tech. rep., Massachusetts Institute of Technology (1996)
37. Rubinstein, A.: Finite automata play the repeated prisoner's dilemma. Journal of Economic Theory 20(1-3), 83–96 (1986)
38. Urbano, A., Vila, J.: Computationally restricted unmediated talk under incomplete information. Economic Theory 23(2), 283–320 (2004)
39. Yao, A.C.: Theory and application of trapdoor functions. In: 23rd Annual Symposium on Foundations of Computer Science (FOCS '82). pp. 80–91. IEEE Computer Society (1982)