Composable Security in the Tamper-Proof Hardware Model under Minimal Complexity

Carmit Hazay[1], Antigoni Polychroniadou[2], and Muthuramakrishnan Venkitasubramaniam[3]

[1] Bar-Ilan University, Israel
[2] Aarhus University, Denmark
[3] University of Rochester, Rochester, New York

Abstract. We put forth a new formulation of tamper-proof hardware in the Global Universal Composable (GUC) framework introduced by Canetti et al. in TCC 2007. Almost all of the previous works rely on the formulation by Katz in Eurocrypt 2007 and this formulation does not fully capture tokens in a concurrent setting. We address these shortcomings by relying on the GUC framework where we make the following contributions:

1. We construct secure Two-Party Computation (2PC) protocols for general functionalities with optimal round complexity and computational assumptions using stateless tokens. More precisely, we show how to realize arbitrary functionalities in the two-party setting with GUC security in two rounds under the minimal assumption of One-Way Functions (OWFs). Moreover, our construction relies on the underlying function in a black-box way. As a corollary, we obtain feasibility of Multi-Party Computation (MPC) with GUC-security under the minimal assumption of OWFs. As an independent contribution, we identify an issue with a claim in a previous work by Goyal, Ishai, Sahai, Venkatesan and Wadia in TCC 2010 regarding the feasibility of UC-secure computation with stateless tokens assuming collision-resistant hash-functions (and the extension based only on one-way functions).
2. We then construct a 3-round MPC protocol to securely realize arbitrary functionalities with GUC-security starting from any semi-honest secure MPC protocol. For this construction, we require the so-called one-many commit-and-prove primitive introduced in the original work of Canetti, Lindell, Ostrovsky and Sahai in STOC 2002 that is round-efficient and black-box in the underlying commitment. Using specially designed "input-delayed" protocols we realize this primitive (with a 3-round protocol in our framework) using stateless tokens and one-way functions (where the underlying one-way function is used in a black-box way).

Keywords: Secure Computation, Tamper-Proof Hardware, Round Complexity, Minimal Assumptions

1 Introduction

Secure Multi-Party Computation (MPC) enables a set of parties to mutually run a protocol that computes some function f on their private inputs, while preserving two important properties: *privacy* and *correctness*. The former implies data confidentiality, namely, nothing leaks by the protocol execution but the computed output, while, the later requirement implies that no corrupted party or parties can cause the output to deviate from the specified function. It is by now well known how to securely compute any efficient functionality [58, 29, 50, 4] under the stringent simulation-based definitions (following the ideal/real paradigm). These traditional results prove security in the stand-alone model, where a *single* set of parties run a *single* execution of the protocol. However, the security of most cryptographic protocols proven in the stand-alone setting does not remain intact if many instances of the protocol are executed concurrently [6, 8, 48]. The strongest (but also the most realistic) setting for concurrent security, known as *Universally Composable* (UC) security [6] considers the execution of an unbounded number of concurrent protocols in an arbitrary and adversarially controlled network environment. Unfortunately, stand-alone secure protocols typically fail to remain secure in the UC setting. In fact, without assuming some *trusted help*, UC-security is impossible to achieve for most tasks [8, 10, 48]. Consequently, UC-secure protocols have been constructed under various *trusted setup* assumptions in a long series of works; see [2, 7, 42, 41, 13, 46, 22] for few examples.

One such setup assumption and the focus of this work is the use of tamper-proof hardware tokens. The first work to model tokens in the UC framework was by Katz in [42] who introduced the $\mathcal{F}_{\text{WRAP}}$-functionality to capture such tokens and demonstrated feasibility of realizing general functionalities with UC-security. Most of the previous works in the tamper proof hardware [42, 15, 46, 30, 22, 17, 25] rely on this formulation. As we explain next, this formulation does not provide adequate composability guarantees. We begin by mentioning that any notion of composable security in an interactive setting should allow for multiple protocols to co-exist in the same system and interact with each other. We revisit the following desiderata put forth by Canetti, Lin and Pass [11] for any notion of composable security:

Concurrent multi-instance security: The security properties relating to local objects (including data and tokens) of the analyzed protocol itself should remain valid even when multiple instances of the protocol are executed concurrently and are susceptible to coordinated attacks against multiple instances. Almost all prior works in the tamper proof model do not specifically analyze their security in a concurrent setting. In other words, they only discuss UC-security of a single instance of the protocol. In particular, when executing protocols in the concurrent setting with tokens, an adversary could in fact transfer a token received from one execution to another and none of the previous works that are based on the $\mathcal{F}_{\text{WRAP}}$-functionality accommodate transfers.

Modular analysis: Security of the larger overall protocols must be deducible from the security properties of its components. In other words, composing protocols should preserve security in a modular way. One of the main motivations and features in the UC-framework is the ability to analyze a protocol locally in isolation while guaranteeing global security. This does not only enable easier design but identifies the required security properties. The current framework proposed by Katz [42] does not allow for such a mechanism.

Environmental friendliness: Unknown protocols in the system should not adversely affect the security of the analyzed protocol. Prior UC-formulation of tamper proof tokens are not "fully" environment friendly as tokens cannot be transferred to other unknown protocols. Furthermore, prior works in the $\mathcal{F}_{\text{WRAP}}$-hybrid do not explicitly prove multi-instance security in the presence of an environment (i.e., they do not realize the multi-versions of the corresponding complete functionality).

The state-of-affairs regarding tamper-proof tokens leads us to ask the following question.

Does there exist a UC-formulation of tamper-proof hardware tokens that guarantee strong composability guarantees and allows for modular design?

Since the work of [42], the power of hardware tokens has been explored extensively in a long series of works, especially in the context of achieving UC-security (for example, [15, 51, 30, 23, 26, 25, 17]). While the work of Katz [42] assumed the stronger stateful tokens, the work of Chandran, Goyal and Sahai [15] was the first to achieve UC-security using only stateless tokens. In this work we will focus only on the weaker stateless token model. In the tamper-proof model with stateless tokens, as we argue below, the issue of minimal assumptions and round-complexity have been largely unaddressed. The work of Chandran et al. [15] gives an $O(\kappa)$-round protocol (where κ is the security parameter) based on enhanced trapdoor permutations. Following that, Goyal et al. [30] provided an (incorrect) $O(1)$-round construction based on Collision-Resistant Hash Functions (CRHFs). The work of Choi et al. [17], extending the techniques of [30] and [23], establishes the same result and provide a five-round construction based on CRHFs.

All previous constructions require assumptions stronger than one-way functions (OWFs), namely either trapdoor permutations or CRHFs. Thus as a first question, we investigate the minimal assumptions required for token-based secure computation protocols. The works of [30] and [17] rely on CRHFs for realizing statistically-hiding commitment schemes. Towards minimizing assumptions, both these works, consider the variant of their protocol where they replace the construction of the statistically-hiding commitment scheme based on CRHFs to the one based on one-way functions [33] to obtain UC-secure protocols under minimal assumptions (See Theorem 3 in [30] and Footnote 7 in [17]). While analyzing the proof of this variant in the work of [30], we found a flaw[4] in the original construction based on CRHFs. We present a concrete attack that breaks the security of their construction in Section 3. More recently, the authors of [17] have conveyed in private communication that the variant that naively replaces the commitment in their protocol is in fact vulnerable to covert attacks. They have since retracted this result (see the updated eprint version [16]). Given the state of affairs, our starting point is to address the following fundamental question regarding tokens that remains open.

Can we construct tamper-proof UC-secure protocols using stateless tokens assuming only one-way functions?

[4] In private communication, the authors have acknowledged this flaw and are in the process of updating their result. We remark that we point out a flaw *only* in one particular result, namely, realizing the UC-secure oblivious transfer functionality based on CRHFs and stateless tokens.

A second important question that we address here is:

What is the round complexity of UC-secure two-party protocols using stateless tokens assuming only one-way functions?

We remark here that relying on black-box techniques, it would be impossible to achieve non-interactive secure computation even in the tamper proof model as any such approach would be vulnerable to a residual function attack.[5] This holds even if we allow an initial token exchange phase, where the two parties exchange tokens (that are independent of their inputs). Hence, the best we could hope for is two rounds.[6]

(G)UC-secure protocols in the multi-party setting. In the UC framework, it is possible to obtain UC-secure protocols in the MPC setting by first realizing the UC-secure oblivious transfer functionality (UC OT) in the two-party setting and then combining it with general compilation techniques (e.g., [44, 12, 40, 47] to obtain UC-secure multi-party computation protocols. First, we remark that specifically in the stateless tamper-proof tokens model, prior works fail to consider multi-versions of the OT-functionality while allowing transferrability of tokens which is important in an MPC setting.[7] As such, none of the previous works explicitly study the round complexity of multi-party protocols in the tamper proof model (with stateless tokens), we thus initiate this study in this work and address the following question.

Can we obtain round-optimal multi-party computation protocols with GUC-security in the tamper proof model?

Unidirectional token exchange. Consider the scenario where companies such as Amazon or Google wish to provide an *email spam-detection* service and users of this service want to keep their emails private (so as to not have unwanted advertisements posted based on the content of their emails). In such a scenario, it is quite reasonable to assume that Amazon or Google have the infrastructure to create tamper-proof hardware tokens in large scale while the clients cannot be expected to create tokens on their own. Most of the prior works assume (require) that both parties have the capability of constructing tokens. When relying on non-black-box techniques, the work of [17] shows how to construct UC-OT using a single stateless token and consequently requires only one of the parties to create the token. The work of Moran and Segev in [51] on the other hand shows how to construct UC-secure two-party computation via a black-box construction where tokens are required to be passed only in one direction, however, they require

[5] Intuitively, this attack allows the recipient of the (only) message to repeatedly evaluate the function on different inputs for a fixed sender's input.

[6] Note that in the plain model, without trusted setup, Katz and Ostrovsky [43] showed that five rounds are necessary and sufficient for general 2PC functionalities. Garg et al. [28] revisit the lower bound of [43] and showed that four rounds are necessary and sufficient for realizing general 2PC functionalities in the simultaneous message exchange model where both parties can simultaneously exchange messages in each round.

[7] We remark that the work of [17] considers multiple sessions of OT between a single pair of parties. However, they do not consider multiple sessions between multiple pairs of parties which is required to realize UC-security in the multiparty setting.

the stronger model of stateful tokens. It is desirable to obtain a black-box construction when relying on stateless tokens. Unfortunately, the work of [17] shows that this is impossible in the fully concurrent setting. More precisely, they show that UC-security is impossible to achieve for general functionalities via a black-box construction using stateless tokens if only one of the parties is expected to create tokens. In this work, we therefore wish to address the following question:

> *Is there a meaningful security notion that can be realized in a client-server setting relying on black-box techniques using stateless tokens where tokens are created only by the server?*

1.1 Our Results

As our first contribution, we put forth a formulation of the tamper-proof hardware as a "global" functionality that provides strong composability guarantees. Towards addressing the various shortcomings of the composability guarantees of the UC-framework, Canetti et al. [7] introduced the Global Universal Composability (GUC) framework which among other things allows to consider global setup functionalities such as the common reference string model, and more recently the global random oracle model [9]. In this work, we put forth a new formulation of tokens in the GUC-framework that will satisfy all our desiderata for composition. Furthermore, in our formulation, we will be able to invoke the GUC composition theorem of [7] in a modular way. A formal description of the $\mathcal{F}_{\mathrm{gWRAP}}$-functionality can be found in Figure 2 and more detailed discussion is presented in the next section.

In the two-party setting we resolve both the round complexity and computational complexity required to realize GUC-secure protocols in the stronger $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid stated in the following theorem:

Theorem 1.1 (Informal) *Assuming the existence of OWFs, there exists a two-round protocol that GUC realizes any (well-formed) two-party functionality in the global tamper proof model assuming stateless tokens. Moreover it only makes black-box use of the underlying OWF.*

As mentioned earlier, any (black-box) non-interactive secure computation protocol is vulnerable to a residual function attack assuming stateless tokens. Therefore, the best round complexity we can hope for assuming (stateless) tamper-proof tokens is two which our results shows is optimal. In concurrent work [24], Dottling et al. show how to obtain UC-secure two-party computation protocol relying on one-way functions via non-black-box techniques.

As mentioned before, we also identify a flaw in a prior construction that attempted to construct a UC-secure protocols in the stateless tamper-proof model from OWFs. We describe a concrete attack on this protocol in Section 3. On a high-level, the result of Goyal et al. first constructs a "Quasi oblivious transfer" protocol based on tokens that admits one-sided simulation and one-sided indistinguishability. Next, they provide a transformation from Quasi-OT to full OT. We demonstrate that the transformation in the second step is insecure by constructing an adversary that breaks its security. The purpose of presenting the flaw is to illustrate a subtlety that arises when arguing security in the token model. While there are mechanisms that could potentially facilitate compiling a Quasi-OT to full OT in the token model, we do not pursue this approach for two reasons. First, fixing this issue will still result in a protocol that requires statistically-hiding commitments and in light of the vulnerability of the [17] protocol, it is unclear if we can simply rely on one-way functions for the statistically-hiding commitment scheme to obtain a construction under minimal assumptions. Second, even if this construction is secure, it would yield only a $O(\kappa)$-round protocol [33]. Instead, we directly construct a round-optimal construction based on OWFs using a more modular, and in our opinion, simpler construction.

In the multi-party setting, our first theorem follows as a corollary of our results from the two-party setting.

Theorem 1.2 *Assuming the existence of OWFs, there exists a $O(d_f)$-round protocol that GUC realizes any multi-party (well formed) functionality f in the global tamper proof model assuming stateless tokens, where d_f is the depth of any circuit implementing f.*

Next, we improve the round-complexity of our construction to obtain the following theorem:

Theorem 1.3 *Assuming the existence of OWFs and stand-alone semi-honest MPC in the OT-hybrid, there exists a three-round protocol that GUC realizes any multi-party (well formed) functionality in the global tamper proof model assuming stateless tokens.*

We remark that our construction is black-box in the underlying one-way function but relies on the code of the MPC protocol in a non-black-box way. It is conceivable that one can obtain a round-optimal construction if we do not require it to be black-box in the underlying primitives and leave it as future work.

Finally, in the client-server setting, we prove the following theorem in the full version [34]:

Theorem 1.4 (Informal) *Assuming the existence of one-way functions, there exists a two-round protocol that securely realizes any two-party functionality assuming stateless tokens in a client-server setting, where the tokens are created only by the server. We also provide an extension where we achieve UC-security against malicious clients and sequential and parallel composition security against malicious servers.*

In more detail, we provide straight-line (UC) simulation of malicious clients and standard rewinding-based simulation against malicious servers. Our protocols guarantee security of the servers against arbitrary malicious coordinating clients and protects every individual client executing sequentially or in parallel against a corrupted server. We believe that this is a reasonable model in comparison to the Common Reference String (CRS) model where both parties require a trusted entity to sample the CRS. Furthermore, it guarantees meaningful concurrent security that is otherwise not achievable in the plain model in two rounds.

1.2 Our Techniques

Our starting point for our round optimal secure two-party computation is the following technique from [30] for an extractable commitment scheme.

Roughly speaking, in order to extract the receiver's input, the sender chooses a function F from a pseudorandom function family that maps $\{0,1\}^m$ to $\{0,1\}^n$ bits where $m >> n$, and incorporates it into a token that it sends to the receiver. Next, the receiver commits to its input b by first sampling a random string $u \in \{0,1\}^m$ and querying the PRF token on u to receive the value v. It sends as its commitment the string $\mathsf{com}_b = (\mathsf{Ext}(u;r) \oplus b, r, v)$ where $\mathsf{Ext}(\cdot,\cdot)$ is a strong randomness extractor. Now, since the PRF is highly compressing, it holds with high probability that conditioned on v, u has very high min-entropy and therefore $\mathsf{Ext}(u;r) \oplus b, r$ statistically hides b. Furthermore, it allows for extraction as the simulator can observe the queries made by the sender to the token and observe that queries that yields v to retrieve u. This commitment scheme is based on one-way functions but is only extractable. To obtain a full-fledged UC-commitment from an extractable commitment we can rely on standard techniques (See [56, 35] for a few examples). Instead, in order to obtain round-optimal constructions for secure two-party computation, we extend this protocol directly to realize the UC oblivious transfer functionality. A first incorrect approach is the following protocol. The parties exchange two sets of PRF tokens. Next, the receiver commits to its bit com_b using the approach described above, followed by the sender committing to its input $(\mathsf{com}_{s_0}, \mathsf{com}_{s_1})$ along with an OT token that implements the one-out-of-two string OT functionality. More specifically, it stores two strings s_0 and s_1, and given a single bit b outputs s_b. Specifically, the code of that token behaves as follows:

- On input b^*, u^*, the token outputs $(s_b, \mathsf{decom}_{s_b})$ only if $\mathsf{com}_b = (\mathsf{Ext}(u^*;r) \oplus b^*, r, v)$ and $\mathsf{PRF}(u^*) = v$. Otherwise, the token aborts.

The receiver then runs the token to obtain s_b and verifies if decom_{s_b} correctly decommits com_{s_b} to s_b. This simple idea is vulnerable to an input-dependent abort attack, where the token aborts depending on the value b^*. The work of [30] provides a combiner to handle this particular attack which we demonstrate is flawed. We describe the attack in Section 3. We instead will rely on a combiner from the recent work of Ostrovsky, Richelson and Scafuro [54] to obtain a two-round GUC-OT protocol.

GUC-secure multi-party computation protocols. In order to demonstrate feasibility, we simply rely on the work of [40] who show how to achieve GUC-secure MPC protocols in the OT-hybrid. By instantiating the OT with our GUC-OT protocol, we obtain MPC protocols in the tamper proof model assuming only one-way functions. While this protocol minimizes the complexity assumptions, the round complexity would be high. In this work, we show how to construct a 3-round MPC protocol. Our starting point is to take any semi-honest MPC protocol in the stand-alone model and compile it into a malicious one using tokens following the paradigm in the original work of Canetti et al. [12] and subsequent works [55, 48]. Roughly, the approach is to define a commit-and-prove GUC-functionality $\mathcal{F}_{\text{CP}}$ and compile the semi-honest protocol using this functionality following a GMW-style compilation.

We will follow an analogous approach where we directly construct a full-fledged $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-functionality that allows a single prover to commit to a string and then prove multiple statements on the commitment simultaneously to several parties. In the token model, realizing this primitive turns out to be non-trivial. This is because we need the commitment in this protocol to be straight-line extractable and the proof to be about the value committed. Recall that, the extractable commitment is based on a PRF token supplied by the receiver of the commitment (and the verifier in the zero-knowledge proof). The prover cannot attest the validity of its commitment (via an NP-statement) since it does not know the code (i.e. key) of the PRF. Therefore, any commit and prove scheme in the token model necessarily must rely on a zero-knowledge proof that is black-box in the underlying commitment scheme. In fact, in the seminal work of Ishai et al. [39] they showed how to construct such protocols that have been extensively used in several works where the goal is to obtain constructions that are black-box in the underlying primitives. Following this approach and solving its difficulties that appear in the tamper-proof hardwire model, we can compile a T-round semi-honest secure MPC protocol to a $O(T)$-round protocol. Next, to reduce the rounds of the computation we consider the approach of Garg et al. [27] who show how to compress the round complexity of any MPC protocol to a two-round GUC-secure MPC protocol in the CRS model using obfuscation primitives.

In more detail, in the first round of the protocol in [27], every party commits to its input along with its randomness. The key idea is the following compiler used in the second round: it takes any (interactive) underlying MPC protocol, and has each party obfuscate their "next-message" function in that protocol, providing one obfuscation for each round. To ensure correctness, zero-knowledge proofs are used to validate the actions of each party w.r.t the commitments made in the first step. Such a mechanism is also referred to as a commit-and-prove strategy. This enables each party to independently evaluate the obfuscation one by one, generating messages of the underlying MPC protocol and finally obtain the output. The observation here is that party P_i's next-message circuit for round j in the underlying MPC protocol depends on its private input x_i and randomness r_i (which are hard-coded in the obfuscation) and on input the transcript of the communication in the first $j-1$ rounds outputs its message for the next round.

To incorporate this approach in the token model, we can simply replace the obfuscation primitives with tokens. Next, to employ zero-knowledge proofs via a black-box construction, we require a zero-knowledge protocol that allows commitment of a witness via tokens at the beginning of the protocol and then in a later step prove a statement about this witness where the commitment scheme is used in a "black-box" way. A first idea here would be to compile using the zero-knowledge protocol of [39] that facilitate such a commit-and-prove paradigm. However, as we explain later this would cost us in round-complexity. Instead we will rely on so-called input-delayed proofs [45] that have recently received much attention [20, 21, 36]. In particular, we will rely on the recent work of [36] who shows how to construct the so-called "input-delay" commit-and-prove protocols which allow a prover to commit a string in an initial commit phase and then prove a statement regarding this string at a later stage where the input statement is determined later. However, their construction only allows for proving one statement regarding the commitment. One of our technical contributions is to extend this idea to allow multiple theorems and further extend it so that a single prover can prove several theorems to multiple parties simultaneously. This protocol will be 4-round and we show how to use this protocol in conjunction with the Garg et al.'s round collapsing technique.

1.3 Related Work

In recent and independent work, using the approach of [9], Nilges [53, 49] consider a GUC-like formulation of the tokens for the two-party setting where the parties have fixed roles. The focus in [53, 49] was to obtain a formulation that accommodates reusability of a single token for several independent protocols in the UC-setting for the specific two-party case. In contrast to our work, they do not explicitly model or discuss adversarial transferability of the tokens. In particular they do not discuss in the multi-party case, which is the main motivation behind our work.

Another recent work by Boureanu, Ohkubo and Vaudenay [5] studies the limit of composition when relying on tokens. In this work, they prove that EUC (or GUC)-security is impossible to achieve for most functionalities if tokens can be transferred in a restricted framework. More precisely, their impossibility holds, if the tokens themselves do not "encode" the session identifier in any way. Our work, circumvents this impossibility result by precisely allowing the tokens generated (by honest parties) to encode the session identifier in which they have to be used.

2 Modeling Tamper-Proof Hardware in the GUC Framework

In this section we describe our model and give our rationale for our approach. We provide a brief discussion on the Universal Composability (UC) framework [6], UC with *joint state* [14] (JUC) and Generalized UC [7] (GUC). For more details, we refer the reader to the original works and the discussion in [9].

Basic UC. Introduced by Canetti in [6], the Universal Composability (UC) framework provides a framework to analyse security of protocols in complex network environments in a modular way. One of the fundamental contributions of this work was to give a definition that will allow to design protocols and demonstrate security by "locally" analyzing a protocol but guaranteeing security in a *concurrent* setting where security of the protocol needs to be intact even when it is run concurrently with many instances of arbitrary protocols. Slightly more technically, in the UC-framework, to demonstrate that a protocol Π securely realizes an ideal functionality $\mathcal{F}$, we need to show that for any adversary $\mathcal{A}$ in the real world interacting with protocol Π in the presence of arbitrary environments $\mathcal{Z}$, there exists an ideal adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$ the view of an interaction with $\mathcal{A}$ is indistinguishable from the view of an interaction with the ideal functionality $\mathcal{F}$ and $\mathcal{S}$.

Unfortunately, soon after its inception, a series of impossibility results [8, 10, 48] demonstrated that most non-trivial functionalities cannot be realized in the UC-framework. Most feasibility results in the UC-framework relied on some sort of trusted setup such as the common reference string (CRS) model [8], tamper-proof model [42] or relaxed security requirements such as super-polynomial simulation [55, 57, 3]. When modeling trusted setup such as the CRS model, an extension of the UC-framework considers the $\mathcal{G}$-hybrid model where "all" real-world parties are given access to an ideal setup functionality $\mathcal{G}$. In order for the basic composition theorem to hold in such a $\mathcal{G}$-hybrid model, two restrictions have to be made. First, the environment $\mathcal{Z}$ cannot access the ideal setup functionality *directly*; it can only do so indirectly via the adversary. In some sense, the setup $\mathcal{G}$ is treated as "local" to a protocol instance. Second, two protocol instances of the same or different protocol cannot share "state" for the UC-composition theorem to hold. Therefore, a setup model such as the CRS in the UC-framework necessitates that each protocol uses its own local setup. In other words, an independently sampled reference string for every protocol instance. An alternative approach that was pursued in a later work was to realize a multi-version of a functionality and proved security of the multi-version using a single setup. For example, the original feasibility result of Canetti, Lindell, Ostrovsky and Sahai [12] realized the $\mathcal{F}_{\rm MCOM}$-functionality which is the multi-version of the basic commitment functionality$\mathcal{F}_{\rm COM}$ in the CRS model.

JUC. Towards accommodating a global setup such as the CRS for multiple protocol instances, Canetti and Rabin [14] introduced the Universal Composition with Joint State (JUC) framework. Suppose we want to analyze several instances of protocol Π with an instance $\mathcal{G}$ as common setup, then at the least, each instance of the protocol must share some state information regarding $\mathcal{G}$ (e.g., the reference string in the CRS model). The JUC-framework precisely accommodates such a scenario, where a new composition theorem is proven, that allows for composition of protocols that share some state. However, the JUC-model for the CRS setup would only allow the CRS to be accessible to a pre-determined set of protocols and in particular still does not allow the environment to directly access the CRS.

GUC. For most feasibility results in the (plain) CRS model both in the UC and JUC framework, the simulator $\mathcal{S}$ in the ideal world needed the ability to "program" the CRS. In particular, it is infeasible to allow the environment to access the setup reference string. As a consequence, we can prove security only if the reference string is privately transmitted to the protocols that we demand security of and cannot be made *publicly* accessible. The work of Canetti, Pass, Dodis and Walfish [7] introduced the Generalized UC-framework to overcome this shortcoming in order to model the CRS as a global setup that is publicly available. More formally, in the GUC-framework, a global setup $\mathcal{G}$ is accessible by any protocol running in the system and in particular allows direct access by the environment. This, in effect, renders all previous protocols constructed in the CRS model not secure in the GUC framework as the simulator loses the programmability of the CRS. In fact, it was shown in [7] that the CRS setup is insufficient to securely realize the ideal commitment functionality in the GUC-framework. More generally, they show that any setup that simply provides only "public" information is not sufficient to realize GUC-security for most non-trivial functionalities. They further demonstrated a feasibility in the Augmented CRS model, where the CRS contains signature keys, one for each party and a secret signing key that is not revealed to the parties, except if it is corrupt, in which case the secret signing key for that party is revealed.

As mentioned before, the popular framework to capture the tamper-proof hardware is the one due to [42] who defined the $\mathcal{F}_{\text{WRAP}}$-functionality in the UC-framework. In general, in the token model, the two basic advantages that the simulator has over the adversary is "observability" and "programmability". Observability refers to the ability of the simulator to monitor all queries made by an adversary to the token and programmability refers to the ability to program responses to the queries in an online manner. In the context of tokens, both these assumptions are realistic as tamper-proof tokens do provide both these abilities in a real-world. However, when modeling tamper proof hardware tokens in the UC-setting, both these properties can raise issues as we discuss next.

Apriori, it is not clear why one should model the tamper proof hardware as a global functionality. In fact, the tokens are local to the parties and it makes the case for it *not* to be globally accessible. Let us begin with the formulation by Katz [42] who introduced the $\mathcal{F}_{\text{WRAP}}$-functionality (see Figure 1 for the stateless variant). In the real world the creator or sender of a token specifies the code to be incorporated in a token by sending the description of a Turing machine M to the ideal functionality. The ideal functionality then emulates the code of M to the receiver of the token, only allowing black-box access to the input and output tapes of M. In the case of stateful tokens, M is modeled as an interactive Turing machine while for stateless tokens, standard Turing machines would suffice. Slightly more technically, in the UC-model, parties are assigned unique identifiers PID and sessions are assigned identifiers sid. In the tamper proof model, to distinguish tokens, the functionality accepts an identifier mid when a token is created. More formally, when one party PID_i creates a token with program M with token identifier mid and sends it to another party PID_j in session sid, then the $\mathcal{F}_{\text{WRAP}}$ records the tuple $(\mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, M)$. Then whenever a party with identifier PID_j sends a query $(\mathsf{Run}, \mathsf{sid}, \mathsf{PID}_i, \mathsf{mid}, x)$ to the $\mathcal{F}_{\text{WRAP}}$-functionality, it first checks whether there is a tuple of the form $(\cdot, \mathsf{PID}_j, \mathsf{mid}, \cdot)$ and then runs the machine M in this tuple if one exists.

Functionality $\mathcal{F}_{\text{WRAP}}^{\text{Stateless}}$

Functionality $\mathcal{F}_{\text{WRAP}}^{\text{Stateless}}$ is parameterized by a polynomial $p(\cdot)$ and an implicit security parameter κ.

Create. Upon receiving $(\mathsf{Create}, \mathsf{sid}, \mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, M)$ from S, where M is a Turing machine, do:

1. Send $(\mathsf{Create}, \mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid})$ to R.
2. Store $(\mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, M)$.

Execute. Upon receiving $(\mathsf{Run}, \mathsf{sid}, \mathsf{PID}_i, \mathsf{mid}, x)$ from R, find the unique stored tuple $(\mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, M)$. If no such tuple exists, do nothing. Run $M(x)$ for at most $p(\kappa)$ steps, and let out be the response ($\mathsf{out} = \perp$ if M does not halt in $p(k)$ steps). Send $(\mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, \mathsf{out})$ to R.

Fig. 1. The ideal functionality for stateless tokens [42].

In the UC-setting (or JUC), to achieve any composability guarantees, we need to realize the multi-use variants of the specified functionality and then analyze the designed protocol in a concurrent man-in-the-middle setting. In such a multi-instance setting, it is reasonable to assume that an adversary that receives a token from one honest party in a left interaction can forward the token to another party in a right interaction. Unfortunately, the $\mathcal{F}_{\text{WRAP}}$-functionality does not facilitate such a transfer.

Let us modify $\mathcal{F}_{\text{WRAP}}$ to accommodate transfer of tokens by adding a special "transfer" query that allows a token in the possession of one party to be transferred to another party. Since protocols designed in most works do not explicitly prove security in a concurrent man-in-the-middle setting, such a modification renders the previous protocols designed in $\mathcal{F}_{\text{WRAP}}$ insecure. For instance, consider the commitment scheme discussed in the introduction based on PRF tokens. Such a scheme would be insecure as an adversary can simply forward the token from the receiver in a right interaction to the sender in a left interaction leading to a malleable commitment.

In order to achieve security while allowing transferability we need to modify the tokens themselves in such a way to be not useful in an execution different from where it is supposed to be used. If every honestly generated token admits only queries that are prefixed with the correct session identifier then transferring the tokens created by one honest party to another honest party will be useless as honest parties will prefix their queries with the right session and the honestly generated tokens will fail to answer on incorrect session prefixes. This is inspired by an idea in [9], where they design GUC-secure protocols in the Global Random Oracle model [9]. As such, introducing transferrability naturally requires protocols to address the issue of non-malleability.

While this modification allows us to model transferrability, it still requires us to analyze protocols in a concurrent man-in-the-middle setting. In order to obtain a more modular definition, where each protocol instance can be analyzed in isolation we need to allow the token to be transferred from the adversary to the environment. In essence, we require the token to be somewhat "globally" accessible and this is the approach we take.

2.1 The Global Tamper-Proof Model

A natural first approach would be to consider the same functionality in the GUC-framework and let the environment to access the $\mathcal{F}_{\text{WRAP}}$-functionality. This is reasonable as an environment can have access to the tokens via auxiliary parties to whom the tokens were transferred to. However, naively incorporating this idea would deny "observability" and "programmability" to the simulator as all adversaries can simply transfer away their tokens the moment they receive them and let other parties make queries on their behalf. Indeed, one can show that the impossibility result of [17] extends to this formulation of the tokens (at least if the code of the token is treated in a black-box manner).[8] A second approach would be to reveal to the simulator all queries

[8] Informally, the only advantage that remains for the simulator is to see the code of the tokens created by the adversary. This essentially reduces to the case where tokens are sent only in one direction and is impossible due to a result of [17] when the code is treated as a black-box.

made to the token received by the adversary even if transferred out to any party. However, such a formulation would be vulnerable to the following transferring attack. If an adversary received a token from one session, it can send it as its token to an honest party in another session and now observe all queries made by the honest party to the token. Therefore such a formulation of tokens is incorrect.

Our formulation will accommodate transferrability while still guaranteeing observability to the simulator. In more detail, we will modify the definition of $\mathcal{F}_{\text{WRAP}}$ so that it will reveal to the simulator all "illegitimate" queries made to the token by any other party. This approach is analogous to the one taken by Canetti, Jain and Scafuro [9] where they model the Global Random Oracle Model and are confronted by a similar issue; here queries made to a globally accessible random oracle via auxiliary parties by the environment must be made available to the simulator while protecting the queries made by the honest party. In order to define "legitimate" queries we will require that all tokens created by an honest party, by default, will accept an input of the form (sid, x) and will respond with the evaluation of the embedded program M on input x, only if $\mathsf{sid} = \overline{\mathsf{sid}}$, where $\overline{\mathsf{sid}}$ corresponds to the session where the token is supposed to be used, i.e. the session where the honest party created the token. Furthermore, whenever an honest party in session $\overline{\mathsf{sid}}$ queries a token it received on input x, it will prefix the query with the correct session identifier, namely issue the query $(\overline{\mathsf{sid}}, x)$. An *illegitimate query* is one where the sid prefix in a query differs from the session identifier from which the party is querying from. Every illegitimate query will be recorded by our functionality and will be disclosed to the party whose session identifier is actually sid.

More formally, the $\mathcal{F}_{\text{gWRAP}}$-functionality is parameterized by a polynomial $p(\cdot)$ which is the time bound that the functionality will exercise whenever it runs any program. The functionality admits the following queries:

Creation Query: This query allows one party S to create and send a token to another party R by sending the query $(\mathsf{Create}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid}, M)$ where M is the description of the machine to be embedded in the token, mid is a unique identifier for the token and sid is the session identifier. The functionality records $(\mathrm{R}, \mathsf{sid}, \mathsf{mid}, M)$.[9]

Transfer Query: We explicitly provide the ability for parties to transfer tokens to other parties that were not created by them (eg, received from another session). Such a query will only be used by the adversary in our protocols as honest parties will always create their own tokens. When a transfer query of the form $(\mathsf{transfer}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid})$ is issued, the tuple $(\mathrm{S}, \overline{\mathsf{sid}}, \mathsf{mid}, M)$ is erased and a new tuple $(\mathrm{R}, \mathsf{sid}, \mathsf{mid}, M)$ is created where $\overline{\mathsf{sid}}$ is the identifier of the session where it was previously used.

Execute Query: To run a token the party needs to provide an input in a particular format. All honest parties will provide the input as $x = (\mathsf{sid}, x')$ and the functionality will run M on input x and supply the answer. In order to achieve non-malleability, we will make sure in all our constructions that tokens generated by honest parties will respond to a query only if it contains the correct sid.

Retrieve Query: This is the important addition to our functionality following the approach taken by [9]. $\mathcal{F}_{\text{gWRAP}}$-functionality will record all illegitimate queries made

[9] We remark here that the functionality does not explicitly store the PID of the creator of the token. We made this choice since the simulator in the ideal world will create tokens for itself which will serve as a token created on behalf of an honest party.

to a token. Namely for a token recorded as the tuple $(\mathrm{R}, \overline{\mathsf{sid}}, \mathsf{mid}, M)$ an illegitimate query is of the form (sid, x) where $\mathsf{sid} \neq \overline{\mathsf{sid}}$ and such a query will be recorded in a set $\mathcal{Q}_{\mathsf{sid}}$ that will be made accessible to the receiving party corresponding to sid.

A formal description of the ideal functionality $\mathcal{F}_{\mathrm{gWRAP}}$ is presented in Figure 2. We emphasize that our formulation of the tamper-proof model will now have the following benefits:

1. It overcomes the shortcomings of the $\mathcal{F}_{\mathrm{WRAP}}$-functionality as defined in [42] and used in subsequent works. In particular, it allows for transferring tokens from one session to another while retaining "observability".
2. Our model allows for designing protocols in the UC-framework and enjoys the composition theorem as it allows the environment to access the token either directly or via other parties.
3. Our model explicitly rules out "programmability" of tokens. We remark that it is (potentially) possible to explicitly provide a mechanism for programmability in the $\mathcal{F}_{\mathrm{gWRAP}}$-functionality. We chose to not provide such a mechanism so as to provide stronger composability guarantees.
4. In our framework, we can analyze the security of a protocol in isolation and guarantee concurrent multi-instance security directly using the GUC-composition theorem. Moreover, it suffices to consider a "dummy" adversary that simply forwards the environment everything (including the token).

An immediate consequence of our formulation is that it renders prior works such as [42, 15, 23, 24] that rely on the programmability of the token insecure in our model. The works of [30, 17] on the other hand can be modified and proven secure in the $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid as they do not require the tokens to be programmed.

We now provide the formal definition of UC-security in the Global Tamper-Proof model.

Definition 2.1 (GUC security in the global tamper-proof model) *Let $\mathcal{F}$ be an ideal functionality and let π be a multi-party protocol. Then protocol π GUC realizes $\mathcal{F}$ in $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid model, if for every uniform PPT hybrid-model adversary $\mathcal{A}$, there exists a uniform PPT simulator $\mathcal{S}$, such that for every non-uniform PPT environment $\mathcal{Z}$, the following two ensembles are computationally indistinguishable,*

$$\left\{\mathbf{View}_{\pi,\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathrm{gWRAP}}}(\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{\mathrm{c}}{\approx} \left\{\mathbf{View}_{\mathcal{F},\mathcal{S},\mathcal{Z}}^{\mathcal{F}_{\mathrm{gWRAP}}}(\kappa)\right\}_{\kappa\in\mathbb{N}}.$$

3 Issue with *Over Extraction* in Oblivious Transfer Combiners [30]

In the following we identify an issue that affects one of the feasibility results in [30, Section 5]. More precisely, this result establishes that UC security for general functionalities is feasible in the tamper-proof hardware model in $O(\kappa)$-round assuming only OWFs (or $O(1)$-round based on CRHFs) based on stateless tokens. The issue arises as a result of *over extraction* where a fully-secure OT protocol is constructed from a weaker variant and the simulation extracts values for sender's inputs even on certain executions

Functionality $\mathcal{F}_{\text{gWRAP}}$

Parameters: Polynomial $p(\cdot)$.

Create. Upon receiving $(\mathsf{Create}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid}, M)$ from S, where M is a Turing machine, do:

1. Send $(\mathsf{Receipt}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid})$ to R.
2. Store $(\mathrm{R}, \mathsf{sid}, \mathsf{mid}, M)$.

Execute. Upon receiving $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}, x)$ from R, find the unique stored tuple $(\mathrm{R}, \mathsf{sid}, \mathsf{mid}, M)$. If such a tuple does not exist, do nothing. Otherwise, interpret $x = (\overline{\mathsf{sid}}, x')$ and run $M(x)$ for at most $p(\kappa)$ steps, and let out be the response ($\mathsf{out} = \perp$ if M does not halt in $p(k)$ steps). Send $(\mathsf{sid}, \mathrm{R}, \mathsf{mid}, \mathsf{out})$ to R.
Handling Illegal Queries: If $\mathsf{sid} \neq \overline{\mathsf{sid}}$, then add $(x', \mathsf{out}, \mathsf{mid})$ to the list $\mathcal{Q}_{\overline{\mathsf{sid}}}$ that is initialized to be empty.

Transfer. Upon receiving $(\mathsf{transfer}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid})$ from S, find the unique stored tuple $(\mathrm{S}, \overline{\mathsf{sid}}, \mathsf{mid}, M)$. If no such tuple exists, do nothing. Otherwise,

1. Send $(\mathsf{Receipt}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid})$ to R.
2. Store $(\mathrm{R}, \mathsf{sid}, \mathsf{mid}, M)$. Erase $(\mathrm{S}, \overline{\mathsf{sid}}, \mathsf{mid}, M)$.

Retrieve Queries: Upon receiving a request $(\mathsf{retreive}, \mathsf{sid})$ from a party R, return the list $\mathcal{Q}_{\mathsf{sid}}$ of illegitimate queries.

Fig. 2. The global stateless token functionality.

where the receiver aborts. The term over extraction has been studied before in the context of commitment schemes where a scheme with over extraction is constructed as an intermediate step towards achieving full security [56, 31].

On a high-level, in the work of [30], they first construct an OT protocol with milder security guarantees. More precisely, a QuasiOT protocol achieves UC-security against a malicious receiver and straight-line extraction against malicious sender. However, the scheme is not fully secure as a malicious sender can cause an input-dependent abort for an honest receiver. Towards amplifying the security, [30] consider the following protocol:

1. The sender with input (s_0, s_1) and receiver with input b interact in n executions of QuasiOTs. The sender picks $z_1, \ldots, z_n$ and Δ at random and sets the inputs to the i^{th} QuasiOT instance as $(z_i, z_i + \Delta)$. The receiver on the other hand chooses bits $b_1, \ldots, b_n$ at random subject to the sum being its input b.
2. If the first step completes, the sender sends $(s'_0 = s_0 + \sum_i z_i, s'_1 = s_1 + \sum_i z_i + \Delta)$ to the receiver. The receiver computes its output as $s'_b + \sum_i w_i$ where w_i is the output of the receiver in the i^{th} QuasiOT.

This protocol remains secure against a malicious receiver. However, an issue arises with a malicious sender. To simulate a malicious sender in this protocol, [30] rely on the straight-line extractor of the n QuasiOTs by sampling two sets of random $(b_1, \ldots, b_n)$, one set summing up to 0 and another set summing up to 1 and computing what the receiver outputs in the two cases. As we demonstrate below such a strategy leads to failure in the simulation. More precisely, consider the following malicious sender strategy.

- Pick $z_1, z_2, ..., z_{n-1}$ and $\varDelta$ at random.
- The inputs of the first $n-1$ tokens are set to $z_1, z_1+\varDelta, \ldots, z_{n-1}, z_{n-1}+\varDelta$.
- Let $z_1+\ldots+z_{n-1}=a$ and $z_1+\ldots+z_{n-1}+\varDelta=b$.
- The inputs to the n-th token are some fixed values c (when $b_n=0$) and d (when $b_n=1$), where $c+d\neq\varDelta$.

Next, the sender modifies the code of the tokens used in the QuasiOT protocol so that the first $n-1$ QuasiOTs never abort. The n-th instantiations however is made to abort whenever the input b_n, the receiver's input is 1. Let $s_0=0$ and $s_1=1$ (we remark that we are not concerned about the actual inputs of the sender, but focus on what the receiver learns). We next examine the honest receiver's output in both the real and ideal worlds. First, in the real world the honest receiver learns an output only if $b_n=0$ (since the n-th token aborts whenever $b_n=1$). We consider two cases:

Case 1: The receiver's input is $b=0$. Then $b_n=0$ with probability $1/2$, and $b_n=1$ with probability $1/2$. Moreover, when $b_n=0$, the sum of the outputs obtained by the receiver is $a+c$. This is because when $b_n=0$, then, $b_1+\ldots+b_{n-1}=0$, and the receiver learns a as the sum of the outputs in the the first $n-1$ QuasiOTs and c from the n-th QuasiOT. On the other hand, if $b_n=1$ then the receiver aborts in the n-th QuasiOT and therefore aborts.

Case 2: The receiver's input is $b=1$. Similarly, in this case the receiver will learn $b+c$ with probability $1/2$ and aborts with probability $1/2$.

In the ideal world, the simulator runs first with a random bit-vector and extracts its inputs in the QuasiOTs by monitoring the queries to the corresponding PRF tokens. Next, it generates two bit-vectors b_i's and b'_i's that add up to 0 and 1, respectively, and computes the sums of the sender's input that correspond to these bits. Then the distribution of these sums can be computed as follows:

Case 1: In case that $\sum b_i=0$, then $b_n=0$ with probability $1/2$, and $b_n=1$ with probability $1/2$. In the former case the receiver learns $a+c$, whereas in the latter case it learns $b+d$.

Case 2: In case that $\sum b'_i=1$, then with probability $1/2$, $b'_n=0$ and with probability $1/2$, $b'_n=1$. In the former case the receiver learns $b+c$, whereas in the latter it learns $a+d$.

Note that this distribution is different from the real distribution, where the receiver never learns $b+d$ or $a+d$ since the token will always abort and not reveal d. We remark that in our example the abort probability of the receiver is independent of its input as proven in Claim 17 in [30], yet the distribution of what it learns is different.

On a more general note, our attack presents the subtleties that need to be addressed with the "selective" abort strategy. Recent works by Ciampi et al. [18, 19] have identified subtleties in recent construction of non-malleable commitments [32] where selective aborts were not completely addressed.

4 Two-Round Token-Based GUC Oblivious Transfer

In this section we present our main protocol that implements GUC OT in two rounds. We first construct a three-round protocol and then show in [34] how to obtain a two-round protocol by exchanging tokens just once in a setup phase. Recall that the counter example to the [30] protocol shows that directly extracting the sender's inputs does not necessarily allow us to extract the sender's inputs correctly, as the tokens can behave maliciously. Inspired by the recently developed protocol from [54] we consider a new approach here for which the sender's inputs are extracted directly by monitoring the queries it makes to the PRF tokens and using additional checks to ensure that the sender's inputs can be verified.

Protocol intuition. As a warmup consider the following sender's algorithm that first chooses two random strings x_0 and x_1 and computes their shares $[x_b] = (x_b^1, \dots, x_b^{2\kappa})$ for $b \in \{0,1\}$ using the $\kappa+1$-out-of-2κ Shamir secret-sharing scheme. Next, for each $b \in \{0,1\}$, the sender commits to $[x_b]$ by first generating two vectors α_b and β_b such that $\alpha_b \oplus \beta_b = [x_b]$, and then committing to these vectors. Finally, the parties engage in 2κ parallel OT executions where the sender's input to the jth instance are the decommitments to $(\alpha_0[j], \beta_0[j])$ and $(\alpha_1[j], \beta_1[j])$. The sender further sends $(s_0 \oplus x_0, s_1 \oplus x_1)$. Thus, to learn s_b, the receiver needs to learn x_b. For this, it enters the bit b for $\kappa+1$ or more OT executions and then reconstructs the shares for x_b, followed by reconstructing s_b using these shares. Nevertheless, this reconstruction procedure works only if there is a mechanism that verifies whether the shares are consistent.

To resolve this issue, Ostrovsky et al. made the observation that the Shamir secret-sharing scheme has the property for which there exists a linear function ϕ such that any vector of shares $[x_b]$ is valid if and only if $\phi(x_b) = 0$. Moreover, since the function ϕ is linear, it suffices to check whether $\phi(\alpha_b) + \phi(\beta_b) = 0$. Nevertheless, this check requires from the receiver to know the entire vectors α_b and β_b for its input b. This means it would have to use b as the input to all the 2κ OT executions, which may lead to an input-dependent abort attack. Instead, Ostrovsky et al. introduced a mechanism for checking consistency indirectly via a *cut-and-choose* mechanism. More formally, the sender chooses κ pairs of vectors that add up to $[x_b]$. It is instructive to view them as matrices $A_0, B_0, A_1, B_1 \in \mathbb{Z}_p^{\kappa \times 2\kappa}$ where for every row $i \in [\kappa]$ and $b \in \{0,1\}$, it holds that $A_b[i,\cdot] \oplus B_b[i,\cdot] = [x_b]$. Next, the sender commits to each entry of each matrix separately and sets as input to the jth OT the decommitment information of the entire column $((A_0[\cdot,j], B_0[\cdot,j]), (A_1[\cdot,j], B_1[\cdot,j]))$. Upon receiving the information for a particular column j, the receiver checks if for all i, $A_b[i,j] \oplus B_b[i,j]$ agree on the same value. We refer to this as the *shares consistency check.*

Next, to check the validity of the shares, the sender additionally sends vectors $[z_1^b], \dots, [z_\kappa^b]$ in the clear along with the sender's message where it commits to the entries of A_0, A_1, B_0 and B_1 such that $[z_i^b]$ is set to $\phi(A_0[i,\cdot])$. Depending on the challenge message, the sender decommits to $A_0[i,\cdot]$ and $A_1[i,\cdot]$ if $c_i = 0$ and $B_0[i,\cdot]$ and $B_1[i,\cdot]$ if $c_i = 1$. If $c_i = 0$, then the receiver checks whether $\phi(A_b[i,\cdot]) = [z_i^b]$, and if $c_i = 1$ it checks whether $\phi(B_b[i,\cdot]) + z_i^b = 0$. This check ensures that except for at most $s \in \omega(\log \kappa)$ of the rows $(A_b[i,\cdot], B_b[i,\cdot])$ satisfy the condition that $\phi(A_b[i,\cdot]) + \phi(B_b[i,\cdot]) = 0$ and for each such row i, $A_b[i,\cdot] + B_b[i,\cdot]$ represents a valid set of shares for both $b = 0$ and $b = 1$. This check is denoted by the *shares validity check*. In the final protocol, the sender sets as input in the jth parallel OT, the decommitment to the entire jth columns of A_0 and B_0 corresponding to the receiver's input 0 and A_1 and B_1 for input 1. Upon receiving the decommitment information on input b_j, the receiver considers a column "good" only if $A_{b_j}[i,j] + B_{b_j}[i,j]$ add up to the same value for every i. Using another cut-and-choose mechanism, the receiver ensures that there are sufficiently many good columns which consequently prevents any input-independent behavior. We refer this to the shares-validity check.

Our oblivious transfer protocol. We obtain a two-round oblivious transfer protocol as follows. The receiver commits to its input bits $b_1, \ldots, b_{2\kappa}$ and the challenge bits for the share consistency check $c_1, \ldots, c_\kappa$ using the PRF tokens. Then, the sender sends all the commitments *a la* [54] and $2\kappa + \kappa$ tokens, where the first 2κ tokens provide the decommitments to the columns, and the second set of κ tokens give the decommitments of the rows for the shares consistency check. The simulator now extracts the sender's inputs by retrieving its queries and we are able to show that there cannot be any input dependent behavior of the token if it passes both the shares consistency check and the shares validity check.

We now describe our protocol $\Pi_{\mathrm{OT}}^{\mathrm{GUC}}$ with sender S and receiver R using the following building blocks: let (1) Com be a non-interactive perfectly binding commitment scheme, (2) let $\mathcal{SS} = (\mathsf{Share}, \mathsf{Recon})$ be a $(\kappa+1)$-out-of-2κ Shamir secret-sharing scheme over $\mathbb{Z}_p$, together with a linear map $\phi : \mathbb{Z}_p^{2\kappa} \to \mathbb{Z}_p^{\kappa-1}$ such that $\phi(v) = 0$ iff v is a valid sharing of some secret, (3) F, F' be two families of pseudorandom functions that map $\{0,1\}^{5\kappa} \to \{0,1\}^\kappa$ and $\{0,1\}^\kappa \to \{0,1\}^{p(\kappa)}$, respectively (4) H denote a hardcore bit function and (5) $\mathsf{Ext} : \{0,1\}^{5\kappa} \times \{0,1\}^d \to \{0,1\}$ denote a randomness extractor where the source has length 5κ and the seed has length d. See Protocol 1 for the complete description.

Protocol 1 *Protocol $\Pi_{\mathrm{GUC}}^{\mathrm{OT}}$ - GUC OT with stateless tokens.*

- **Inputs:** S *holds two strings* $s_0, s_1 \in \{0,1\}^\kappa$ *and* R *holds a bit* b. *The common input is* sid.
- **The protocol:**
  1. $\mathbf{S} \to \mathbf{R}$*:* S *chooses* 3κ *random PRF keys* $\{\gamma_l\}_{[l \in 3\kappa]}$ *for family* F. *Let* $\mathsf{PRF}_{\gamma_l} : \{0,1\}^{5\kappa} \to \{0,1\}^\kappa$ *denote the pseudorandom function.* S *creates token* $\mathsf{TK}_{\mathrm{S}}^{\mathsf{PRF},l}$ *sending* $(\mathsf{Create}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid}_l, M_1)$ *to* $\mathcal{F}_{\mathrm{gWRAP}}$ *where* M_1 *is the functionality of the token that on input* $(\overline{\mathsf{sid}}, x)$ *outputs* $\mathsf{PRF}_{\gamma_l}(x)$ *for all* $l \in [3\kappa]$*; For the case where* $\overline{\mathsf{sid}} \neq \mathsf{sid}$ *the token aborts;*
  2. $\mathbf{R} \to \mathbf{S}$*:* R *selects a random subset* $T_{1-b} \subset [2\kappa]$ *of size* $\kappa/2$ *and defines* $T_b = [2\kappa]/T_{1-b}$. *For every* $j \in [2\kappa]$, R *sets* $b_j = \beta$ *if* $j \in T_\beta$. R *samples uniformly at random* $c_1, \ldots, c_\kappa \leftarrow \{0,1\}$. *Finally,* R *sends*
     (a) $(\{\mathsf{com}_{b_j}\}_{j \in [2\kappa]}, \{\mathsf{com}_{c_i}\}_{i \in [\kappa]})$ *to* S *where*
     $$\forall\, j \in [2\kappa], i \in [\kappa] \quad \mathsf{com}_{b_j} = (\mathsf{Ext}(u_j) \oplus b_j, v_j) \quad \textit{and} \quad \mathsf{com}_{c_i} = (\mathsf{Ext}(u'_i) \oplus c_i, v'_i)$$
     $u_j, u'_i \leftarrow \{0,1\}^{5\kappa}$ *and* v_j, v'_i *are obtained by sending respectively* $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_j, u_j)$ *and* $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{2\kappa+i}, u'_i)$.
     (b) R *generates the tokens* $\{\mathsf{TK}_{\mathrm{R}}^{\mathsf{PRF},l'}\}_{l' \in [8\kappa^2]}$ *which are analogous to the PRF tokens* $\{\mathsf{TK}_{\mathrm{S}}^{\mathsf{PRF},l}\}_{l \in [3\kappa]}$ *by sending* $(\mathsf{Create}, \mathsf{sid}, \mathrm{R}, \mathrm{S}, \mathsf{mid}_{l'}, M_2)$ *to* $\mathcal{F}_{\mathrm{gWRAP}}$ *for all* $l' \in [8\kappa^2]$.

3. $\mathbf{S} \rightarrow \mathbf{R}$*:* S *picks two random strings* $x_0, x_1 \leftarrow \mathbb{Z}_p$ *and secret shares them using* $\mathcal{SS}$. *In particular,* S *computes* $[x_b] = (x_b^1, \ldots, x_b^{2\kappa}) \leftarrow \mathsf{Share}(x_b)$ *for* $b \in \{0,1\}$. S *commits to the shares* $[x_0], [x_1]$ *as follows. It picks random matrices* $A_0, B_0 \leftarrow \mathbb{Z}_p^{\kappa \times 2\kappa}$ *and* $A_1, B_1 \leftarrow \mathbb{Z}_p^{\kappa \times 2\kappa}$ *such that* $\forall i \in [\kappa]$*:*

$$A_0[i,\cdot] + B_0[i,\cdot] = [x_0], \;\; A_1[i,\cdot] + B_1[i,\cdot] = [x_1].$$

S *computes two matrices* $Z_0, Z_1 \in \mathbb{Z}_p^{\kappa \times \kappa - 1}$ *and sends them in the clear such that:*

$$Z_0[i,\cdot] = \phi(A_0[i,\cdot]), Z_1[i,\cdot] = \phi(A_1[i,\cdot]).$$

S *sends:*

(a) *Matrices* $(\mathsf{com}_{A_0}, \mathsf{com}_{B_0}, \mathsf{com}_{A_1}, \mathsf{com}_{B_1})$ *to* R*, where,*

$$\forall\, i \in [\kappa],\ j \in [2\kappa], \beta \in \{0,1\} \quad \mathsf{com}_{A_\beta[i,j]} = (\mathsf{Ext}(u^{A_\beta[i,j]} \oplus A_\beta[i,j], v^{A_\beta[i,j]})$$
$$\mathsf{com}_{B_\beta[i,j]} = (\mathsf{Ext}(u^{B_\beta[i,j]} \oplus B_\beta[i,j], v^{B_\beta[i,j]})$$

where $(u^{A_\beta[i,j]}, u^{B_\beta[i,j]}) \leftarrow \{0,1\}^{5\kappa}$ *and* $(v^{A_\beta[i,j]}, v^{B_\beta[i,j]})$ *are obtained by sending* $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{[i,j,\beta]}, u^{A_\beta[i,j]})$ *and* $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{2\kappa^2+[i,j,\beta]}, u^{B_\beta[i,j]})$*, respectively, to the token* $\mathsf{TK}_{\mathrm{R}}^{\mathsf{PRF},[i,j,\beta]}$ *where* $[i,j,\beta]$ *is an encoding of the indices* i, j, β *into an integer in* $[2\kappa^2]$.

(b) $C_0 = s_0 \oplus x_0$ *and* $C_1 = s_1 \oplus x_1$ *to* R.

(c) *For all* $j \in [2\kappa]$, S *creates a token* TK_j *sending* $(\mathsf{Create}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid}_{3\kappa+j}, M_3)$ *to* $\mathcal{F}_{\mathrm{gWRAP}}$ *where* M_3 *is the functionality that on input* $(\overline{\mathsf{sid}}, b_j, \mathsf{decom}_{b_j})$*, aborts if* $\overline{\mathsf{sid}} \neq \mathsf{sid}$ *or if* decom_{b_j} *is not verified correctly. Otherwise it outputs* $(A_{b_j}[\cdot,j], \mathsf{decom}_{A_{b_j}[\cdot,j]}, B_{b_j}[\cdot,j], \mathsf{decom}_{B_{b_j}[\cdot,j]})$.

(d) *For all* $i \in [\kappa]$, S *creates a token* $\widehat{\mathsf{TK}}_i$ *sending* $(\mathsf{Create}, \mathsf{sid}, \mathrm{S}, \mathrm{R}, \mathsf{mid}_{5\kappa+i}, M_4)$ *to* $\mathcal{F}_{\mathrm{gWRAP}}$ *where* M_4 *is the functionality that on input* $(\overline{\mathsf{sid}}, c_i, \mathsf{decom}_{c_i})$ *aborts if* $\overline{\mathsf{sid}} \neq \mathsf{sid}$ *or if* decom_{c_i} *is not verified correctly. Otherwise it outputs,*

$$(A_0[i,\cdot], \mathsf{decom}_{A_0[i,\cdot]}, A_1[i,\cdot], \mathsf{decom}_{A_1[i,\cdot]}), \text{ if } c = 0$$
$$(B_0[i,\cdot], \mathsf{decom}_{B_0[i,\cdot]}, B_1[i,\cdot], \mathsf{decom}_{B_1[i,\cdot]}), \text{ if } c = 1$$

4. **Output Phase:**
For all $j \in [2\kappa]$, R *sends* $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{3\kappa+j}, (b_j, \mathsf{decom}_{b_j}))$ *and receives*

$$(A_{b_j}[\cdot,j], \mathsf{decom}_{A_{b_j}[\cdot,j]}, B_{b_j}[\cdot,j], \mathsf{decom}_{B_{b_j}[\cdot,j]}).$$

For all $i \in [\kappa]$, R *sends* $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{5\kappa+i}, (c_i, \mathsf{decom}_{c_i}))$ *and receives*

$$(A_0[\cdot,i], A_1[\cdot,i]) \text{ or } (B_0[\cdot,i], B_1[\cdot,i]).$$

(a) SHARES VALIDITY CHECK PHASE: *For all* $i \in [\kappa]$*, if* $c_i = 0$ *check that* $Z_0[i,\cdot] = \phi(A_0[i,\cdot])$ *and* $Z_1[i,\cdot] = \phi(A_1[i,\cdot])$. *Otherwise, if* $c_i = 1$ *check that* $\phi(B_0[i,\cdot]) + Z_0[i,\cdot] = 0$ *and* $\phi(B_1[i,\cdot]) + Z_1[i,\cdot] = 0$. *If the tokens do not abort and all the checks pass, the receiver proceeds to the next phase.*

(b) SHARES CONSISTENCY CHECK PHASE: *For each* $b \in \{0,1\}$, R *randomly chooses a set* T_b *for which* $b_j = b$ *of* $\kappa/2$ *coordinates. For each* $j \in T_b$, R *checks that there exists a unique* x_b^j *such that* $A_b[i,j] + B_b[i,j] = x_b^j$ *for all* $i \in [\kappa]$. *If so,* x_b^j *is marked as consistent. If the tokens do not abort and all the shares obtained in this phase are consistent,* R *proceeds to the reconstruction phase. Else it abort.*

(c) OUTPUT RECONSTRUCTION: *For $j \in [2\kappa]/T_{1-b}$, if there exists a unique x_b^j such that $A_b[i,j] + B_b[i,j] = x_b^j$, mark share j as a* good *column. If R obtains less than $\kappa + 1$* good *shares, it aborts. Otherwise, let $x_b^{j_1}, \ldots, x_b^{j_{\kappa+1}}$ be any set of $\kappa + 1$ consistent shares.* R *computes $x_b \leftarrow \mathsf{Recon}(x_b^{j_1}, \ldots, x_b^{j_{\kappa+1}})$ and outputs $s_b = C_b \oplus x_b$.*

Next, we state the following theorem, the proof can be found in [34].

Theorem 4.1 *Assume the existence of one-way functions, then protocol $\Pi_{\text{GUC}}^{\text{OT}}$ GUC realizes $\mathcal{F}_{\text{OT}}$ in the $\mathcal{F}_{\text{gWRAP}}$-hybrid.*

Proof overview. On a high-level, when the sender is corrupted our simulation proceeds analogously to the simulation from [54] where the simulator generates the view of the malicious sender by honestly generating the receiver's messages and then extracting all the values committed to by the sender. Nevertheless, while in [54] the authors rely on extractable commitments and extract the sender's inputs via rewinding, we directly extract its inputs by retrieving the queries made by the malicious sender to the $\{\mathsf{TK}_{\mathrm{R}}^{\mathsf{PRF},i}\}_i$ tokens. The proof of correctness follows analogously. More explicitly, the share consistency check ensures that for any particular column that the receiver obtains, if the sum of the values agree on the same bit, then the receiver extracts the correct share of $[x_b]$ with high probability. Note that it suffices for the receiver to obtain $\kappa + 1$ good columns for its input b to extract enough shares to reconstruct x_b since the shares can be checked for validity. Namely, the receiver chooses $\kappa/2$ indices T_b and sets its input for these OT executions as b. For the rest of the OT executions, the receiver sets its input as $1 - b$. Denote this set of indices by T_{1-b}. Then, upon receiving the sender's response to its challenge and the OT responses, the receiver first performs the shares consistency check. If this check passes, it performs the shares validity check for all columns, both with indices in T_{1-b} and for the indices in a random subset of size $\kappa/2$ within T_b. If one of these checks do not pass, the receiver aborts. If both checks pass, it holds with high probability that the decommitment information for $b = 0$ and $b = 1$ are correct in all but $s \in \omega(\log n)$ indices. Therefore, the receiver will extract $[x_b]$ successfully both when its input $b = 0$ and $b = 1$. Furthermore, it is ensured that if the two checks performed by the receiver pass, then a simulator can extract both x_0 and x_1 correctly by simply extracting the sender's input to the OT protocol and following the receiver's strategy to extract.

On the other hand, when the receiver is corrupted, our simulation proceeds analogous to the simulation in [54] where the simulator generates the view of the malicious receiver by first extracting the receiver's input b and then obtaining s_b from the ideal functionality. It then completes the execution following the honest sender's code with (s_0, s_1), where s_{1-b} is set to random. Moreover, while in [54] the authors rely on a special type of interactive commitment that allows the extraction of the receiver's input via rewinding, we instead extract this input directly by retrieving the queries made by the malicious receiver to the $\{\mathsf{TK}_{\mathrm{S}}^{\mathsf{PRF},l}\}_{l\in[3\kappa]}$ tokens. The proof of correctness follows analogously. Informally, the idea is to show that the receiver can learn $\kappa + 1$ or more shares for either x_0 or x_1 but not both. In other words there exists a bit b for which a corrupted receiver can learn at most κ shares relative to s_{1-b}. Thus, by replacing s_{1-b}

with a random string, it follows from the secret-sharing property that obtaining at most κ shares keeps s_{1-b} information theoretically hidden.

On relying on one-way functions. In this protocol the only place where one-way permutations are used is in the commitments made by the sender in the second round of the protocol via a non-interactive perfectly-binding commitment. This protocol can be easily modified to rely on statistically-binding commitments which have two-round constructions based on one-way functions [52]. Specifically, since the sender commits to its messages only in the second-round, the receiver can provide the first message of the two-round commitment scheme along with the first message of the protocol.

5 Three-Round Token-Based GUC Secure Multi-Party Computation

In this section, we show how to compile an arbitrary round semi-honest protocol Π to a three-round protocol using stateless tokens. As discussed in the introduction, the high-level of our approach is borrowing the compressing round idea from [27] which proceeds in three steps. In the first step, all parties commit to their inputs via an extractable commitment and then in the second step, each party provides a token to emulate their actions with respect to Π given the commitments. Finally, each party runs the protocol Π locally and obtains the result of the computation. For such an approach to work, it is crucial that an adversary, upon receiving the tokens, is not be able to "rewind" the computation and launch a resetting attack. This is ensured via zero-knowledge proofs that are provided in each round. In essence, the zero-knowledge proofs validates the actions of each party with respect to the commitments made in the first step. Such a mechanism is also referred to as a commit-and-prove strategy. In Section 5.1, we will present a construction of a commit-and-prove protocol in the $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid and then design our MPC protocol using this protocol. We then take a modular approach by describing our MPC protocol in an idealized version of the commit-and-prove functionality analogous to [12] and then show how to realize this functionality. As we mentioned before we then rely on the approach of [27] to compress the rounds of our MPC protocol compiled with our commit and prove protocol in 3 rounds. This is presented in the full version [34].

5.1 One-Many Commit-and-Prove Functionality

The commit and prove functionality $\mathcal{F}_{\mathrm{CP}}$ introduced in [12] is a generalization of the commitment functionality and is core to constructing protocols in the GUC-setting. The functionality parameterized by an NP-relation $\mathcal{R}$ proceeds in two stages: The first stage is a commit phase where the receiver obtains a commitment to some value w. The second phase is a prove phase where the functionality upon receiving a statement x from the committer sends x to the receiver along with the value $\mathcal{R}(x, w)$. We will generalize the $\mathcal{F}_{\mathrm{CP}}$-functionality in two ways. First, we will allow for asserting multiple statements on a single committed value w in the $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid. Second, we will allow a single party to assert the statement to many parties. In an MPC setting this will be useful as each party will assert the correctness of its message to all parties in each step.

Our generalized functionality can be found in Figure 3 and is parameterized by an NP relation $\mathcal{R}$ and integer $m \in \mathbb{N}$ denoting the number of statements to be proved.

Functionality $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$

Functionality $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$ is parameterized by an NP-relation $\mathcal{R}$, an integer m and an implicit security parameter κ, and runs with set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$.

Commit Phase: Upon receiving a message $(\mathsf{commit}, \mathsf{sid}, \mathcal{P}, w)$ from P_i, where $w \in \{0,1\}^\kappa$, record the tuple $(\mathsf{sid}, P_i, \mathcal{P}, w, 0)$ and send $(\mathsf{receipt}, P_i, \mathcal{P}, \mathsf{sid})$ to all parties in $\mathcal{P}$.

Prove Phase: Upon receiving a message $(\mathsf{prove}, \mathsf{sid}, \mathcal{P}, x)$ from P_i, where $w \in \{0,1\}^{poly(\kappa)}$, find the record $(\mathsf{sid}, P_i, \mathcal{P}, w, ctr_{\mathsf{sid}})$. If no such record is found or $ctr_{\mathsf{sid}} \geq m$ then ignore. Otherwise, send $(\mathsf{proof}, \mathsf{sid}, \mathcal{P}, (x, \mathcal{R}(x, w)))$ to all parties in $\mathcal{P}$. Replace the tuple $(\mathsf{sid}, P_i, \mathcal{P}, w, ctr_{\mathsf{sid}})$ with $(\mathsf{sid}, P_i, \mathcal{P}, w, ctr_{\mathsf{sid}} + 1)$.

Fig. 3. The one-many multi-theorem commit and prove functionality [12].

To realize this functionality, we will rely on the so-called input-delayed proofs [45, 20, 21, 36]. In particular, we rely on the recent work of Hazay and Venkitasubramaniam [36], who showed how to obtain a 4-round commit-and-prove protocol where the underlying commitment scheme and one-way permutation are used in a black-box way, and requires the statement only in the last round. Below, we extend their construction and design a protocol Π_{CP} that securely realizes functionality $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$, and then prove the following theorem.

Theorem 5.1 *Assuming the existence of one-way functions, then protocol Π_{CP} securely realizes the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-functionality in the $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid.*

Realizing $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$ in the $\mathcal{F}_{\mathrm{gWRAP}}$-Hybrid In the following section we extend ideas from [36] in order to obtain a one-many commit-and-prove protocol with negligible soundness using a specialized randomized encodings (RE) [38, 1], where the statement is only known at the last round. Loosely speaking, RE allows to represent a "complex" function by a "simpler" randomized function. Given a string $w_0 \in \{0,1\}^n$, the [36] protocol considers a randomized encoding of the following function:

$$f_{w_0}(x, w_1) = (\mathcal{R}(x, w_0 \oplus w_1), x, w_1)$$

where $\mathcal{R}$ is the underlying NP relation and the function has the value w_0 hardwired in it. The RE we consider needs to be secure against *adaptively chosen inputs* and *robust*. Loosely speaking, an RE is secure against adaptive chosen inputs if both the encoding and the simulation can be decomposed into offline and online algorithms and security should hold even if the input is chosen adaptively after seeing the offline part of the encoding. Moreover, an offline/online RE is said to be robust if no adversary can produce an offline part following the honest encoding algorithm and a (maliciously generated) online part that evaluates to a value outside the range of the function. Then the ZK proof follows by having the prover generate the offline phase of the randomized encoding for this functionality together with commitments to the randomness r used for this generation and w_1. Next, upon receiving a challenge bit ch from the verifier, the prover completes the proof as follows. In case $ch = 0$, then the prover reveals r and w_1 for which the verifier checks the validity of the offline phase. Otherwise, the prover sends the online part of the encoding and a decommitment of w_1 for which the verifier runs the decoder and checks that the outcome is $(1, x, w_1)$.

A concrete example based on garbled circuits [58] implies that the offline part of the randomized encoding is associated with the garbled circuit, where the randomness r can be associated with the input key labels for the garbling. Moreover, the online part can be associated with the corresponding input labels that enable to evaluate the garbled circuit on input x, w_1. Clearly, a dishonest prover cannot provide both a valid garbling and a set of input labels that evaluates the circuits to 1 in case x is a false statement. Finally, adaptive security is achieved by employing the construction from [37] (see [36] for a discussion regarding the robustness of this scheme).

We discuss next how to extend Theorem 5.5 from [36] by adding the one-many multi-theorem features. In order to improve the soundness parameter of their ZK proof Hazay and Venkitasubramaniam repeated their basic proof sufficiently many times in parallel, using fresh witness shares each time embedding the [39] approach in order to add a mechanism that verifies the consistency of the shares. Consider a parameter N to be the number of repetitions and let m denote the number of proven theorems. Our protocol employs two types of commitments schemes: (1) Naor's commitment scheme [52] denoted by Com. (2) Token based extractable commitment scheme in the $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid denoted by $\mathsf{Com}_{\mathrm{gWRAP}}$ and defined as follows. First, the receiver R in the commitment scheme will prepare a token that computes a PRF under a randomly chosen key k and send it to the committer in an initial setup phase, incorporated with the session identifier sid. Such that on input (x, sid) the token outputs PRF evaluated on the input x. More, precisely, the receiver on input sid creates a token $\mathsf{TK}^{\mathsf{PRF}_k}$ with the following code:

- On input $(x, \widetilde{\mathsf{sid}})$: If $\widetilde{\mathsf{sid}} = \mathsf{sid}$ output $\mathsf{PRF}_k(x)$. Otherwise, output $\perp$.

Then, to commit to a bit b, the committer C first queries the token $\mathsf{TK}^{\mathsf{PRF}_k}$ on input (u, sid) where $u \in \{0,1\}^{5\kappa}$ is chosen at random and sid is the session identifier. Upon receiving the output v from the token, it sends $(\mathsf{Ext}(u) \oplus b, v)$ where Ext is a randomness extractor as used in Section 4. We remark here that if the tokens are exchanged initially in a token exchange phase, then the commitment scheme is non-interactive.

Protocol 2 *Protocol Π_{CP} - one-many commit-and-prove protocol.*

- **Input:** *The prover holds a witness w, where the prover is a designated party P_τ for some $\tau \in [n]$.*
- **The Protocol:**
  1. *Each party P_k for $k \neq \tau$ plays the role of the verifier and picks random m t-subsets I_j^k of $[N]$ for each $j \in [m]$ and $k \in [n-1]$ where m is the number of proven statements. It also picks t random challenge bits $\{ch_{i,j}^k\}_{i \in I_j^k}$ and commits to them using $\mathsf{Com}_{\mathrm{gWRAP}}^k$. It further sends the first message of the Naor's commitment scheme.*

2. *The prover then continues as follows:*
   (a) *It first generates $N \times m \times (n-1)$ independent XOR sharings of the witness w, say*
   $$\{w^0_{i,j,k}, w^1_{i,j,k}\}_{(i \times j \times k) \in [N \times m \times (n-1)]}.$$
   (b) *Next, for each $j \in [m]$ and $k \in [n-1]$, it generates the views of $2N$ parties $P^0_{i,j,k}$ and $P^1_{i,j,k}$ for all $i \in [N]$ executing a t-robust t-private MPC protocol, where $P^b_{i,j,k}$ has input $w^b_{i,j,k}$, that realizes the functionality that checks if $w^0_{i,j,k} \oplus w^1_{i,j,k}$ are all equal. Let $V^b_{i,j,k}$ be the view of party $P^b_{i,j,k}$.*
   (c) *Next, for each $j \in [m]$ and $k \in [n-1]$, it computes N offline encodings of the following set of functions:*
   $$f_{w^0_{i,j,k}, V^0_{i,j,k}}(x_j, w^1_{i,j,k}, V^1_{i,j,k}) = (b, x_j, w^1_{i,j,k}, V^1_{i,j,k})$$
   where $b = 1$ if and only if $\mathcal{R}(x_j, w^0_{i,j,k} \oplus w^1_{i,j,k})$ holds and the views $V^0_{i,j,k}$ and $V^1_{i,j,k}$ are consistent with each other.
   (d) *Finally, the prover broadcasts to all parties the set containing*
   $$\big\{(f^{\mathsf{off}}_{w^0_{i,j,k}, V^0_{i,j,k}}(r_{i,j,k}), \mathsf{Com}(r_{i,j,k}), \mathsf{Com}(w^0_{i,j,k}), \mathsf{Com}(w^1_{i,j,k}), \mathsf{Com}(V^0_{i,j,k}), \mathsf{Com}(V^1_{i,j,k}))\big\}_{(i \times j \times k) \in [N \times m \times (n-1)]}.$$
   Moreover, let $\mathsf{decom}_{r_{i,j,k}}, \mathsf{decom}_{w^0_{i,j,k}}, \mathsf{decom}_{w^1_{i,j,k}}, \mathsf{decom}_{V^0_{i,j,k}}, \mathsf{decom}_{V^1_{i,j,k}}$ be the respective decommitment information of the above commitments. Then for every $k \in [n-1]$, P_i commits to the above decommitment information with respect to party P_k and all $(i \times j) \in [N] \times [m]$, using $\mathsf{Com}_{\text{gWRAP}}$.
3. *The verifier decommits to all its challenges.*
4. *For every index (i, j) in the t subset the prover replies as follows:*
   - *If $ch^i_{j,k} = 0$ then it decommits to $r_{i,j,k}$, $w^0_{i,j,k}$ and $V^0_{i,j,k}$. The verifier then checks if the offline part was constructed correctly.*
   - *If $ch^i_{j,k} = 1$ then it sends $f^{\mathsf{on}}_{w^0_{i,j,k}, V^0_{i,j,k}}(r_{i,j,k}, x_j, w^1_{i,j,k}, V^1_{i,j,k})$ and decommits $w^1_{i,j,k}$ and $V^1_{i,j,k}$. The verifier then runs the decoder and checks if it obtains $(1, x_j, w^1_{i,j,k}, V^1_{i,j,k})$.*

   Furthermore, from the decommitted views $V^{ch^i_{j,k}}_{i,j,k}$ for every index (i, j) that the prover sends, the verifier checks if the MPC-in-the-head protocol was executed correctly and that the views are consistent.

Theorem 5.2 *Assuming the existence of one-way functions, then protocol Π_{CP} GUC realizes $\mathcal{F}^{1:\text{M}}_{\text{CP}}$ in the $\mathcal{F}_{\text{gWRAP}}$-hybrid.*

Proof: Let $\mathcal{A}$ be a malicious PPT real adversary attacking protocol Π_{CP} in the $\mathcal{F}_{\text{gWRAP}}$-hybrid model. We construct an ideal adversary $\mathcal{S}$ with access to $\mathcal{F}^{1:\text{M}}_{\text{CP}}$ which simulates a real execution of Π_{CP} with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal process with $\mathcal{S}$ and $\mathcal{F}_{\text{gWRAP}}$-hybrid from a real execution of Π_{CP} with $\mathcal{A}$ in the $\mathcal{F}_{\text{gWRAP}}$-hybrid. $\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with environment $\mathcal{Z}$, emulating the honest party. We describe the actions of $\mathcal{S}$ for every corruption case.

Simulating the communication with $\mathcal{Z}$: Every message that $\mathcal{S}$ receives from $\mathcal{Z}$ it internally feeds to $\mathcal{A}$ and every output written by $\mathcal{A}$ is relayed back to $\mathcal{Z}$. In case the adversary $\mathcal{A}$ issues a transfer query on any token $(\mathsf{transfer}, \cdot)$, $\mathcal{S}$ relays the query to the $\mathcal{F}_{\text{gWRAP}}$.

Party $P_\mathcal{T}$ is not corrupted. In this scenario the adversary only corrupts a subset of parties $\mathcal{I}$ playing the role of the verifiers in our protocol. The simulator proceeds as follows.

1. Upon receiving a commitment $\mathsf{Com}^k_{\mathrm{gWRAP}}$ from a corrupted party P_k, the simulator extracts the m committed t-subsets I^k_j and the challenge bits $\{ch^k_{i,j}\}_{i \in I^k_j}$ for all $j \in [m]$, by retrieving the queries made to the tokens.
2. For each $j \in [m]$ and $k \in [\mathcal{I}]$, the simulator generates the views of $2N$ parties $P^0_{i,j,k}$ and $P^1_{i,j,k}$ for all $i \in [N]$ emulating the simulator of the t-robust t-private MPC protocol underlying in the real proof, where the set of corrupted parties for the $(j,k)^{th}$ execution is fixed to be I^k_j extracted above. Let $V^b_{i,j,k}$ be the view of party $P^b_{i,j,k}$.
3. Next, for each $j \in [m]$ and $k \in [\mathcal{I}]$, the simulator computes N offline encodings as follows.
   - For every index i in the t subset I^k_j the simulator replies as follows:
     - If $ch^k_{i,j} = 0$, then the simulator broadcasts the following honestly generated message: $f^{\mathsf{off}}_{w^0_{i,j,k},V^0_{i,j,k}}(r_{i,j,k}), \mathsf{Com}(r_{i,j,k}), \mathsf{Com}(w^0_{i,j,k}), \mathsf{Com}(0),$ $\mathsf{Com}(V'^0_{i,j,k}), \mathsf{Com}(V'^1_{i,j,k})$. where $V'^0_{i,j,k} = 0$ and $V'^1_{i,j,k} = V^1_{i,j,k}$ if the matched challenge bit equals one, and vice versa.
     - Else, if $ch^k_{i,j} = 1$, then the simulator invokes the simulator for the randomized encoding and broadcasts the following message:
$$\big\{\mathcal{S}^{\mathsf{off}}_{w^0_{i,j,k},V^0_{i,j,k}}(r_{i,j,k}), \mathsf{Com}(0), \mathsf{Com}(0), \mathsf{Com}(w^1_{i,j,k}), \mathsf{Com}(V'^0_{i,j,k}), \mathsf{Com}(V'^1_{i,j,k})\big\}_{(i \times j \times k) \in [N \times m \times (n-1)]}$$
       where $w^1_{i,j,k}$ is a random string and $V'^0_{i,j,k} = 0$ and $V'^1_{i,j,k} = V^1_{i,j,k}$ if the matched challenge bit equals one, and vice versa.
   - For every index i not in the t subset I^k_j the simulator broadcasts
$$f^{\mathsf{off}}_{w^0_{i,j,k},V^0_{i,j,k}}(r_{i,j,k}), \mathsf{Com}(r_{i,j,k}), \mathsf{Com}(w^0_{i,j,k}), \mathsf{Com}(0), \mathsf{Com}(0), \mathsf{Com}(0).$$

   The simulator correctly commits to the decommitments information with respect to the honestly generated commitments (namely, as the honest prover would have done) using $\mathsf{Com}_{\mathrm{gWRAP}}$. Else, it commits to the zero string.
4. Upon receiving the decommitment information from the adversary, the simulator aborts if the adversary decommits correctly to a different set of messages than the one extracted above by the simulator.
5. Else, $\mathcal{S}$ completes the protocol by replying to the adversary as the honest prover would do.

Note that the adversary's view is modified with respect to the views it obtains with respect to the underlying MPC and both types of commitments. Indistinguishability follows by first replacing the simulated views of the MPC execution with a real execution. Namely the simulator for this hybrid game commits to the real views. Indistinguishability follows from the privacy of the protocol. Next, we modify the fake commitments into real commitments computed as in the real proof. The reduction for this proof follows easily as the simulator is not required to open these commitments.

Party $P_\mathcal{T}$ is corrupted. In this scenario the adversary corrupts a subset of parties $\mathcal{I}$ playing the role of the verifiers in our protocol as well as the prover. The simulator for this case follows the honest verifier's strategy $\{P_k\}_{k \notin [\mathcal{I}]}$, with the exception that it extracts the prover's witness by extracting one of the witness' pairs. Recall that only the decommitment information is committed via the extractable commitment scheme $\mathsf{Com}_{\mathrm{gWRAP}}$. Since a commitment is made using tokens from every other party and there is at least one honest party, the simulator can extract the decommitment information and from that extract the real value. We point out that in general extracting out shares from only one-pair could cause the problem of "over-extraction" where the adversary does not necessarily commit to shares of the same string in each pair. In our protocol this is not an issue because in conjunction with committing to these shares, it also commits to the views of an MPC-in-the-head protocol which verifies that all shares are correct. Essentially, the soundness argument follows by showing that if an adversary deviates, then with high-probability the set $\mathcal{I}$ will include a party with an "inconsistent view". This involves a careful argument relying on the so-called t-robustness of the underlying MPC-in-the-head protocol. Such an argument is presented in [36] to get negligible soundness from constant soundness and this proof can be naturally extended to our setting (our protocol simply involves more repetitions but the MPC-in-the-head views still ensure correctness of all repetition simultaneously).

As for straight-line extraction, the argument follows as for the simpler protocol. Namely, when simulating the verifier's role the simulator extracts the committed values within the forth message of the prover. That is, following a similar procedure of extracting the committed message via obtaining the queries to the token, it is sufficient to obtain two shares of the witness as the robustness of the MPC protocol ensures that all the pairs correspond to the same witness. ■

5.2 Warmup: Simple MPC Protocol in the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-Hybrid

We next describe our MPC protocol in the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-hybrid. On a high-level, we follow GMW-style compilation [29] of a semi-honest secure protocol Π to achieve malicious security using the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-functionality. Without loss of generality, we assume that in each round of the semi-honest MPC protocol Π, each party broadcasts a single message that depends on its input and randomness and on the messages that it received from all parties in all previous rounds. We let $m_{i,j}$ denote the message sent by the i^{th} party in the j^{th} round in the protocol Π. We define the function π_i such that $m_{i,t} = \pi_i(x_i, r_i, (M_1, \ldots, M_{t-1}))$ where $m_{i,t}$ is the t^{th} message generated by party P_i in protocol Π with input x_i, randomness r_i and where M_r is the message sent by all parties in round i of Π. We leave the complete construction to the full version [34].

Protocol description. Our protocol Π_{MPC} proceeds as follows:

Round 1. In the first round, the parties commit to their inputs and randomness. More precisely, on input x_i, party P_i samples random strings $r_{i,1}, r_{i,2}, \ldots, r_{i,n}$ and sends $(\mathsf{commit}, \mathsf{sid}, \mathcal{P}, \overline{w})$ to $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$ and $\overline{w} = (x, R_i)$ where $R_i = (r_{i,1}, r_{i,2}, \ldots, \underline{r_{i,n}})$.

Round 2. $\underline{P_i}$ broadcasts shares $\overline{R}_i = R_i - \{r_{i,i}\}$ and sends $(\mathsf{prove}, P_i, \mathcal{P}, \overline{R}_i)$. Let $M_0 = (\overline{R}_1, \ldots, \overline{R}_n)$.

Round $2+\delta$**.** Let $M_{\delta-1}$ be the messages broadcast by all parties in rounds $3, 4, \ldots, 2+(\delta-1)$ and let $m_{i,\delta} = \pi_i(x_i, r_i, (M_1, \ldots, M_{\delta-1}))$ where $r_i = \oplus_j r_{j,i}$. P_i broadcasts $m_{i,\delta}$ and sends to $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$ the message $(\mathsf{prove}, P_i, \mathcal{P}, \overline{M}_{t-1} : m_{i,\delta})$ where $\overline{M}_{\delta-1} = (M_0, M_1, \ldots, M_{\delta-1})$.

The NP-relation $\mathcal{R}$ used to instantiate the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$ functionality will include:

1. (M_0, R_i) : if M_0 contains $\overline{R}_i$ as its i^{th} component where $\overline{R}_i = R_i - \{r_{i,i}\}$ and $R_i = \{r_{i,1}, \ldots, r_{i,n}\}$.
2. $((\overline{M}_{\delta-1}, m_{i,\delta}), (x_i, R_i))$: if $(M_0, R_i) \in \mathcal{R}$ and $m_{i,\delta} = \pi_i(x_i, r_i, (M_1, \ldots, M_{\delta-1}))$ where $r_i = \oplus_{j\in[n]} r_{j,i}$, $\overline{M}_{\delta-1} = (M_0, M_1 \ldots, M_{\delta-1})$ and $R_i = \{r_{i,1}, \ldots, r_{i,n}\}$.

Theorem 5.3 *Let f be any deterministic polynomial-time function with n inputs and a single output. Assume the existence of one-way functions and an n-party semi-honest MPC protocol Π. Then the protocol Π_{MPC} GUC realizes $\mathcal{F}_f$ in the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-hybrid.*

Proof: Let $\mathcal{A}$ be a malicious PPT real adversary attacking protocol Π_{MPC} in the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-hybrid model. We construct an ideal adversary $\mathcal{S}$ with access to $\mathcal{F}_f$ which simulates a real execution of Π_{MPC} with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal process with $\mathcal{S}$ interacting with $\mathcal{F}_f$ from a real execution of Π_{MPC} with $\mathcal{A}$ in the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-hybrid. $\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with environment $\mathcal{Z}$, emulating the honest party. We describe the actions of $\mathcal{S}$ for every corruption case.

Simulating honest parties: Let $\mathcal{I}$ be the set of parties corrupted by the adversary $\mathcal{A}$. This means $\mathcal{S}$ needs to simulate all messages from parties in $\mathcal{P}/\mathcal{I}$. $\mathcal{S}$ emulates the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$ functionality for $\mathcal{A}$ as follows. For every $P_j \in \mathcal{P}/\mathcal{I}$ it sends the commitment message $(\mathsf{receipt}, P_j, \mathcal{P}, \mathsf{sid})$ to all parties $P_i \in \mathcal{I}$. Next, for every message $(\mathsf{commit}, \mathsf{sid}, P_i, \mathcal{P}, \overline{w}_i)$ received from $\mathcal{A}$, it records $\overline{w}_i = (x_i, r_{i,1}, \ldots, r_{i,n})$. Upon receiving this message on behalf of every $P_i \in \mathcal{I}$, the simulator $\mathcal{S}$ sends x_i on behalf of every $P_i \in \mathcal{I}$ to $\mathcal{F}_f$ and obtains the result of the computation output. Then using the simulator of the semi-honest protocol Π, it generates random tapes r_i for every $P_i \in \mathcal{I}$ and messages $m_{j,\delta}$ for all honest parties $P_j \in \mathcal{P}/\mathcal{I}$ and all rounds δ. Next, it sends $\overline{R}_j$ on behalf of the honest parties $P_j \in \mathcal{P}/\mathcal{I}$ so that for every $P_i \in \mathcal{I}$, $r_i = \oplus r_{j,i}$. This is possible since there is at least one party P_j outside $\mathcal{I}$ and $\mathcal{S}$ can set $r_{j,i}$ so that it adds to r_i. Next, in round $2+\delta$, it receives the messages from $P_i \in \mathcal{I}$ and supplies messages from the honest parties according to the simulation of P_i. Along with each message it receives the prove message that the parties in $\mathcal{I}$ send to $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$. $\mathcal{S}$ simply honestly emulates $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$ for these messages. For messages that the honest parties send to $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$, $\mathcal{S}$ simply sends the receipt message to all parties in $\mathcal{I}$.

Indistinguishability of the simulation follows from the following two facts:

- Given an input x_i and random tape r_i for every $P_i \in \mathcal{I}$ and the messages from the honest parties, there is a unique emulation of the semi-honest protocol Π where all the messages from parties P_i if honestly generated are deterministic.
- Since the simulation is emulating the $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$ functionality, the computation immediately aborts if a corrupted party P_i deviates from the deterministic strategy.

■

6 Acknowledgements

We thank Yuval Ishai, Amit Sahai, and Vipul Goyal for fruitful discussions regarding token-based cryptography. The first author acknowledges support from the Israel Ministry of Science and Technology (grant No. 3-10883) and support by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. The second author was also supported by the Danish National Research Foundation; the National Science Foundation of China (grant no. 61061130540) for the Sino-Danish CTIC; the CFEM supported by the Danish Strategic Research Council. In addition, this work was done in part while visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467. The third author was supported by Google Faculty Research Grant and NSF Award CNS-1526377.

References

1. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *FOCS*, pages 166–175, 2004.
2. Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, pages 186–195, 2004.
3. Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552, 2005.
4. Donald Beaver. Foundations of secure interactive computing. In *CRYPTO*, pages 377–391, 1991.
5. Ioana Boureanu, Miyako Ohkubo, and Serge Vaudenay. The limits of composable crypto with transferable setup devices. In *CCS*, pages 381–392, 2015.
6. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.
7. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In *TCC*, pages 61–85, 2007.
8. Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO*, pages 19–40, 2001.
9. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In *CCS*, pages 597–608, 2014.
10. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.
11. Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *FOCS*, pages 541–550, 2010.
12. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, 2002.
13. Ran Canetti, Rafael Pass, and Abhi Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *FOCS*, pages 249–259, 2007.
14. Ran Canetti and Tal Rabin. Universal composition with joint state. In *CRYPTO*, pages 265–281, 2003.
15. Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In *EUROCRYPT*, pages 545–562, 2008.

16. Seung Geol Choi, Jonathan Katz, Dominique Schröder, Arkady Yerukhimovich, and Hong-Sheng Zhou. (efficient) universally composable oblivious transfer using a minimal number of stateless tokens. *IACR Cryptology ePrint Archive*, 2013:840, 2013.
17. Seung Geol Choi, Jonathan Katz, Dominique Schröder, Arkady Yerukhimovich, and Hong-Sheng Zhou. (efficient) universally composable oblivious transfer using a minimal number of stateless tokens. In *TCC*, pages 638–662, 2014.
18. Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. *To appear at CRYPTO*, 2016.
19. Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. On round-efficient non-malleable protocols. *IACR Cryptology ePrint Archive*, 2016:621, 2016.
20. Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved or-composition of sigma-protocols. In *TCC*, pages 112–141, 2016.
21. Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In *EUROCRYPT*, pages 63–92, 2016.
22. Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Muthuramakrishnan Venkitasubramaniam. Adaptive and concurrent secure computation from new adaptive, non-malleable commitments. In *ASIACRYPT*, pages 316–336, 2013.
23. Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In *TCC*, pages 164–181, 2011.
24. Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade, and Tobias Nilges. From stateful hardware to resettable hardware using symmetric assumptions. In *ProvSec*, pages 23–42, 2015.
25. Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade, and Tobias Nilges. General statistically secure computation with bounded-resettable hardware tokens. In *TCC*, pages 319–344, 2015.
26. Nico Döttling, Thilo Mie, Jörn Müller-Quade, and Tobias Nilges. Implementing resettable uc-functionalities with untrusted tamper-proof hardware-tokens. In *TCC*, pages 642–661, 2013.
27. Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *TCC*, pages 74–94, 2014.
28. Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 448–476, 2016.
29. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.
30. Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.
31. Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *FOCS*, pages 51–60, 2012.
32. Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. An algebraic approach to non-malleability. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 41–50, 2014.
33. Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM J. Comput.*, 44(1):193–242, 2015.
34. Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Composable security in the tamper proof hardware model under minimal complexity. *IACR Cryptology ePrint Archive*, 2015:887, 2015.

35. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On black-box complexity of universally composable security in the CRS model. In *ASIACRYPT*, pages 183–209, 2015.
36. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. *To appear at CRYPTO*, 2016.
37. Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. *IACR Cryptology ePrint Archive*, 2015:1250, 2015.
38. Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.
39. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.
40. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.
41. Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent composition of secure protocols in the timing model. *J. Cryptology*, 20(4):431–492, 2007.
42. Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, pages 115–128, 2007.
43. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.
44. Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
45. Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, pages 353–365, 1990.
46. Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In *STOC*, pages 179–188, 2009.
47. Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for UC from only OT. In *ASIACRYPT*, pages 699–717, 2012.
48. Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *FOCS*, pages 394–403, 2003.
49. Jeremias Mechler, Jörn Müller-Quade, and Tobias Nilges. Universally composable (non-interactive) two-party computation from untrusted reusable hardware tokens. *IACR Cryptology ePrint Archive*, 2016:615, 2016.
50. Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *CRYPTO*, pages 392–404, 1991.
51. Tal Moran and Gil Segev. David and goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In *EUROCRYPT*, pages 527–544, 2008.
52. Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.
53. Tobias Nilges. *The Cryptographic Strength of Tamper-Proof Hardware*. PhD thesis, Karlsruhe Institute of Technology, 2015.
54. Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In *CRYPTO*, pages 339–358, 2015.
55. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.
56. Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.
57. Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251, 2004.
58. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.