Stateless Cryptographic Protocols

Vipul Goyal
Microsoft Research, India.
Email: vipul@microsoft.com

Hemanta K. Maji *
Department of Computer Science,
University of Illinois at Urbana-Champaign.
Email: hmaji2@uiuc.edu

Abstract— Secure computation protocols inherently involve multiple rounds of interaction among the parties where, typically a party has to keep a state about what has happened in the protocol so far and then *wait* for the other party to respond. We study if this is inherent. In particular, we study the possibility of designing cryptographic protocols where the parties can be completely stateless and compute the outgoing message by applying a single fixed function to the incoming message (independent of any state). The problem of designing stateless secure computation protocols can be reduced to the problem of designing protocols satisfying the notion of resettable computation introduced by Canetti, Goldreich, Goldwasser and Micali (FOCS'01) and widely studied thereafter.

The current start of art in resettable computation allows for construction of protocols which provide security only when a *single predetermined* party is resettable [13]. An exception is for the case of the zero-knowledge functionality for which a protocol in which both parties are resettable was recently obtained by Deng, Goyal and Sahai (FOCS'09). The fundamental question left open in this sequence of works is, whether fully-resettable computation is possible, when:

1) An adversary can corrupt any number of parties, and
2) The adversary can reset any party to its original state during the execution of the protocol and can restart the protocol.

In this paper, we resolve the above problem by constructing secure protocols for *most* efficiently computable multi-party functionalities in the plain model under standard cryptographic assumptions. More precisely, we are able to realize any functionality except the ones which behave like a worst-case pseudorandom function. First, we construct a Fully-Resettable Simulation Sound Zero-Knowledge (ss-rs-rZK) protocol. Next, based on these ss-rs-rZK protocols, we show how to compile a semi-honest secure protocol for such functionalities into a protocol secure against fully resetting adversaries.

Next, we study a seemingly unrelated open question: "Does there exist a functionality which, in the concurrent setting, is impossible to securely realize using BB simulation but can be realized using NBB simulation?". We resolve the above question in the affirmative by giving an example of such a (reactive) functionality. Somewhat surprisingly, this is done by making a connection to the existence of a fully resettable simulation sound zero-knowledge protocol.

1. Introduction

General feasibility results for secure computation were obtained more than two decades ago [24], [11]. Secure computation protocols inherently involve multiple rounds of interaction among the parties where, typically a party has to keep a state about what has happened in the protocol so far and then *wait* for the other party to respond. We study if this is inherent. In particular, we study the possibility of designing cryptographic protocols where the parties can be completely stateless and compute the outgoing message by applying a single fixed function to the incoming message. Being stateless is a highly desirable property in the design of network protocols since it allows for the construction of more robust and easy to maintain systems. SYN flooding and related attacks are well known as a tool to launch a *denial of service attack* against a system running a stateful protocol. It has been argued that internet has been going in the direction of using stateless protocol (from using stateful ones) [14]. In this work, we study of question of obtaining stateless cryptographic secure computation protocols.

Question of designing stateless secure computation protocols is intimately connected to the notion of resettably secure protocols. The study of resettable protocols was initiated in the seminal work of Canetti, Goldreich, Goldwasser, and Micali [5] who introduced the notion of resettable zero-knowledge (ZK). In resettable zero knowledge, the zero knowledge property is required to hold even if a malicious verifier can reset the prover to the initial state and start a new interaction where the prover uses the same random tape. Canetti et al. [5] proposed constructions of resettable zero knowledge protocols based on standard cryptographic assumptions. Barak, Goldreich, Goldwasser, and Lindell [3] showed how to construct zero knowledge protocols for opposite setting (where soundness is required to hold even if the verifier can be reset and restarted with the same random tape), which following Micali and Reyzin [15][1] they call resettably sound (rS) zero-knowledge. Goyal and Sahai [13] extended the study of resettable protocol from zero-knowledge to general functionalities and presented a construction where only a *single predetermined* party can be reset.

In a cryptographic protocol, if a set of parties is resettable, there is a simple compiler to construct another protocol in which those set of parties can be made completely stateless [13]. Thus, the task of obtaining stateless secure computation protocols can be reduced to the task of obtaining resettably secure protocols. The existence of resettably secure protocols also has interesting implications for the fundamental

*Work done in part while interning at Microsoft Research, India.

[1]Micali and Reyzin defined resettable soundness (and other soundness notions) in a public-key model, but did not consider the plain model, which is the focus of the present work.

question of reusing randomness in cryptographic protocols.

Subsequent to the works of Canetti et al. [5] and Barak et al. [3], there have been a number of works studying protocols in the setting where only a single predetermined party is resettable (see the related work subsection for more details). Barak et al. [3] conjectured the existence of zero-knowledge protocols where both the parties may be resettable. This conjecture remained open despite partial progress over several years and was resolved only recently by Deng, Goyal and Sahai [8]. Deng et al. constructed a protocol for the zero-knowledge functionality where the security holds if either of the party maybe resettable.[2] In contrast, the construction of Goyal and Sahai [13] is for all functionalities but the security holds when only a single predetermined party may be reset. In the work of Goyal and Sahai, if any party other than the predetermined one is reset by the adversary, the protocol completely breaks and there is an explicit attack that the adversary can launch. This state of affairs leads to the following natural question:

"Do there exist protocols for functionalities other than zero-knowledge where more than one party may be reset?"

The above can be considered as the last step in the understanding of the feasibility (as opposed to efficiency) of resettable protocols.

1.1. Our Contributions

We resolve the above question in the affirmative. We provide a construction for most (PPT computable) functionalities in the multi-party setting where, in fact, all the parties may be fully resettable. To be more precisely, we are able to realize all functionalities except the ones where the pseudoentropy of the output could be much large than that of the input (for at least one input). We call such functionalities as *worst-case pseudoentropy generators* (w-PEG) (and they can also be used to construct worst-case pseudorandom functions by using a strong extractor). The above results subsume most known feasibility results in the area of resettable protocols. The key ingredient in our construction is a *non-malleable* fully resettable zero-knowledge protocol which we believe to be of independent interest.[3]

Fully Stateless Protocols: A key implication of our result is the first construction of *truly stateless protocols* for a large class of (PPT computable) functionalities. In other words, for any functionality (which is not a w-PEG) in question, we now have a cryptographic protocol where all the parties can be fully stateless and compute the next outgoing message by applying a fixed (next message) function to the incoming message (independent of any state).

[2]This is called a fully resettable zero-knowledge protocol. Note that this implies the existence of stateless zero-knowledge protocols.

[3]The proceedings version of this paper incorrectly claimed to realize *all* PPT computable functionalities in the fully resettable settting. This is the corrected version.

Concurrently Secure Computation: Separating the Power of Non-black-box and Black-box Simulation: There have been a number of tasks which are shown to be impossible using black-box (BB) simulation but become possible once we resort to non-black-box (NBB) simulation. Examples are constant round bounded concurrent zero-knowledge [1], public coin concurrent zero-knowledge [22], constant round covert computation [12], etc. However so far, we don't know of any example of a functionalities which in the concurrent setting: (a) is possible to realize using a NBB simulator, but, (b) is impossible to realize using a BB simulator. Indeed, all such separations between the power of BB and NBB simulation are known only if we place additional constraints on the design of the real world protocol (e.g., it should be public coin, or constant rounds, or should have the covertness property, etc). Thus, we have the following natural question:

"Does there exist a functionality which, in the concurrent setting, is impossible to realize using BB simulation (in any polynomial number of rounds) but can be realized using NBB simulation?"

We answer the above question in the affirmative by giving an example of such a (reactive) functionality. Somewhat surprisingly, this is done by making a connection of the above question to the existence of non-malleable (nm) fully resettable (rs-rZK) zero-knowledge (which is the key ingredient behind our stateless computation protocols as mentioned above). We consider the ideal world functionality where the verifier runs the NM-SR-ZK protocol with the trusted party (who gets the witness from the prover). To rule out the existence of a real world protocol realizing this functionality with a BB simulator, we rely on the resettable soundness and non-malleability of the protocol being run in the ideal world. To show the existence of a real world protocol realizing this functionality with a NBB simulator, we rely on the ideal world protocol being a concurrent zero-knowledge protocol.

1.2. Related Works.

The only other work to consider the issue of non-malleability in the setting of resettable protocols is the recent work by Ostrovsky, Pandey, Sahai and Visconti [17]. However [17] is only able to construct protocols where only a single predetermined party is resettable. Hence the protocols in [17] don't have any direct application towards obtaining new results for resettable / stateless secure computation protocols. The techniques used in our work are quite different from the ones used in [17].

Subsequent to the works of Canetti et al. [5] and Barak et al. [3] described above, a number of works have investigated the problem of security against resetting attacks for zero-knowledge protocols in the plain model. Barak, Lindell, and Vadhan [4] constructed the first constant-round public-coin argument that is *bounded* resettable zero-knowledge. Deng

and Lin [9] showed a zero-knowledge argument system that is bounded resettable zero-knowledge and satisfies a weak form of resettable soundness.

A larger body of work has investigated the same problems in a relaxed setting, called the "bare public key" (BPK) model, introduced by [5], which assumes that parties must register (arbitrarily chosen) public keys prior to any attack taking place. See [7], [25] and the references therein.

Note that the problem of constructing resettable protocols (for functionalities other than zero-knowledge) where more than one party can be reset was open even in the BPK model.

There have been other works using the notion of resettable protocols towards solving unrelated questions (c.f., [19], [6]).

2. Preliminaries

In this section, we introduce some basic definitions pertaining to our simulation based definition of Resettable computation and zero-knowledge protocols.

2.1. ZK Background

In this section, we introduce some definitions related to Zero-Knowledge and Simulation Soundness. We first recall the definition of a relaxed concurrent adversary from [8]. Informally speaking, a relaxed concurrent adversary $\boldsymbol{\mathcal{A}}$, in the beginning of a particular session, chooses a random r (using any adversarial strategy) and then behaves as an honest verifier using that tape r in that session. Hence, messages of $\boldsymbol{\mathcal{A}}$ are independent of the prover messages sent in other sessions during the execution of this session.

Definition 1 (Relaxed Concurrent Adversaries [8]): An adversary $\boldsymbol{\mathcal{A}}$ interacting with the prover $\boldsymbol{P}$ concurrently in several sessions is a relaxed concurrent adversary, if at the beginning of every session, it writes a string s to a special tape such that, there exists a fixed function f (not necessarily efficiently computable) such that $r = f(s)$ and $\boldsymbol{\mathcal{A}}$ interacts in the session as the honest verifier $\boldsymbol{V}$ with random tape r.

Note that any concurrent ZK protocol secure against relaxed concurrent adversaries can be made secure against general concurrent adversaries by asking the verifier to commit to its randomness in the beginning and then with every outgoing message, requiring it to prove in (standalone) ZK that it has behaved honestly using that randomness.

Definition 2 (Resettable Soundness (rs) [3]): A *resetting attack* of a cheating prover $\boldsymbol{P}^*$ on a *resettable verifier* $\boldsymbol{V}$ is defined by the following two-step random process, indexed by a security parameter n.

1) Uniformly select and fix $t = \mathrm{poly(n)}$ random-tapes, denoted $r_1, \ldots, r_t$, for $\boldsymbol{V}$, resulting in deterministic strategies $\boldsymbol{V}^{(j)}(x) = \boldsymbol{V}_{x,r_j}$ defined by $\boldsymbol{V}_{x,r_j}(\alpha) = \boldsymbol{V}(x, r_j, \alpha)$, where $x \in \{0,1\}^n$ and $j \in [t]$. Each $\boldsymbol{V}^{(j)}(x)$ is called an *incarnation* of $\boldsymbol{V}$.
2) On input 1^n, machine P^* is allowed to initiate $\mathrm{poly}(n)$-many interactions with the $\boldsymbol{V}^{(j)}(x)$'s. The activity of $\boldsymbol{P}^*$ proceeds in rounds. In each round $\boldsymbol{P}^*$ chooses $x \in \{0,1\}n$ and $j \in [t]$, thus defining $\boldsymbol{V}^{(j)}(x)$, and conducts a complete session with it.

Let $\boldsymbol{P}$ and $\boldsymbol{V}$ be some pair of interactive machines, and suppose that $\boldsymbol{V}$ is implementable in probabilistic polynomial-time. We say that $(\boldsymbol{P}, \boldsymbol{V})$ is a *resettably-sound proof system for L (resp., resettably-sound argument system for L)* if the following two conditions hold:

- *Resettable-completeness:* Consider an arbitrary resetting attack (resp., polynomial-size resetting attack), and suppose that in some session after selecting an incarnation $V^j(x)$, the attacker follows the strategy $\boldsymbol{P}$. The, if $x \in L$ then $\boldsymbol{V}^{(j)}(x)$ accepts with negligible probability.
- *Resettable-soundness:* For every resetting attack (resp., polynomial size resetting attack), the probability that in some session the corresponding $\boldsymbol{V}^{(j)}(x)$ has accepted and $x \notin L$ is negligible.

Verifier Admissible Proof Systems and Hybrid-Soundness: Instead of achieving full resettable-soundness, it is sometimes easier to achieve a weaker form of guarantee called "hybrid-soundness." In this section we will introduce the definitions of Hybrid-Soundness and Verifier Admissible Proof Systems. Informally speaking, in such a system, the first message of the prover determines all its subsequent "important" messages (for the given verifier).

Definition 3 (Verifier-admissible proof-system): A proof-system $(\boldsymbol{P}, \boldsymbol{V})$ is called "verifier-admissible" if the following requirements hold:

1) The verifier $\boldsymbol{V}$ consists of two parts $\boldsymbol{V}_1, \boldsymbol{V}_2$. Similarly, the verifier's random input ω is partitioned into two disjoint parts, $\omega^{(1)}, \omega^{(2)}$, where $\omega^{(i)}$ is given to $\boldsymbol{V}_i$
2) A message sent by the prover may either be labeled as "main" message or an "authenticator" message. The first main message sent by the prover in the protocol is called the "determining" message. Each prover message is first received by $\boldsymbol{V}_1$. In each round of interaction, the prover and $\boldsymbol{V}_1$ exchange a number of messages in which exactly one of the messages is a main message while the rest are authenticator messages. At the end of an interaction round, $\boldsymbol{V}_1$ decides whether to accept the (only) main message received based on the transcript of interaction round itself and the transcript of the prover main messages and the corresponding replies of $\boldsymbol{V}_2$ so far. If $\boldsymbol{V}_1$ accepts, it forwards the main message to $\boldsymbol{V}_2$ who generates the reply.
3) Let $\boldsymbol{P}^*$ be an arbitrary (deterministic) polynomial-size circuit, where $\boldsymbol{P}^*$ may execute a resetting-attack on $\boldsymbol{V}$ (see Definition 2). Let $\boldsymbol{P}^*$ interact with some

incarnation of $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2)$. Then, except with negligible probability, $\boldsymbol{P}^*$ is unable to generate two different main messages, both accepted by $\boldsymbol{V}_1$, for some round ℓ in two different interactions with $\boldsymbol{V}$ with the same determining message.

Hybrid Model: In our hybrid model, the prover is given the ability to "partially reset" the verifier (while otherwise interacting in the concurrent setting). More precisely, in the verifier admissible proof system, each incarnation of the verifier is identified by three indices: $\boldsymbol{V}^{i,j,k} = \boldsymbol{V}_{x_i,y_i,\omega_{j,k}}$, where $\omega_{j,k} = (\omega_j^{(1)}, \omega_k^{(2)})$. The string $\omega_j^{(1)}$ represents the random input to $\boldsymbol{V}_1$ while $\omega_k^{(2)}$ represents the random input to $\boldsymbol{V}_2$. The prover is allowed to interact concurrently with these incarnations with at most one session of every incarnation active at any time. However, the prover is not allowed to interact with two incarnations (i, j, k) and (i', j', k') such that $k = k'$. Moreover, the prover can restart $\boldsymbol{V}_1$ in an incarnation from the beginning although it leaves $\boldsymbol{V}_2$ in its current state. After being restarted, $\boldsymbol{V}_1$ operates as usual using the same random tape $\omega^{(1)}$ expect the following. $\boldsymbol{V}_1$ aborts if the prover sends a different determining message in the first round of the interaction after $\boldsymbol{V}_1$ restarts. Furthermore, in a round of interaction with the prover, $\boldsymbol{V}_1$ does not forward the received main message to $\boldsymbol{V}_2$ (even if it is accepted) in case a main message in that round had been forwarded earlier. Instead $\boldsymbol{V}_1$ simply sends `accept` or `reject` to the verifier. $\boldsymbol{V}_1$ then waits for the verifier messages for the next round as if $\boldsymbol{V}_2$ had sent the same reply it sent earlier in that round before $\boldsymbol{V}_1$ restarted the interaction. Intuitively, such a setting ensures that $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ do not go out of sync even though only $\boldsymbol{V}_1$ restarts the interaction (with the same randomness).

Definition 4 (Hybrid Soundness (hs)): A "hybrid cheating prover" $\boldsymbol{P}^*$ works against verifier-admissible proof systems in the hybrid model as described above. A proof system is hs if it is verifier admissible and satisfies Definition 2 with respect to hybrid cheating provers.

Simulation Soundness: Simulation Soundness, intuitively, refers to the robustness of proof systems where a man-in-middle adversary is not able to violate the soundness of the verifier in the right interaction, even though it participates as a verifier in other simulated proofs in the left interaction.

Definition 5 ($(1-\epsilon)$ Simulation Sound): Consider any arbitrary man-in-middle adversary $\boldsymbol{\mathcal{A}}$ which is acting as a verifier of Π ZK-protocols in the left interaction and is trying to prove a statement "$\tilde{x} \in L$" in its right interaction by acting as the prover for a Σ protocol. We will use n to represent the security parameter. The length of the adversarial code is at most a polynomial in n, i.e. $|\boldsymbol{\mathcal{A}}| \leq n^{\Theta(1)}$, and we will assume that $\boldsymbol{\mathcal{A}}$ terminates within $n^{\Theta(1)}$ time.

In the left interaction, $\boldsymbol{\mathcal{A}}$ could possible interact with several concurrent sessions of Π proving statements "$x_i \in L$." If $\boldsymbol{\mathcal{A}}$ violates the soundness of Σ with probability ϵ, when $\boldsymbol{\mathcal{A}}$ is provided with simulated Π proofs, then we say: Σ is $(1-\epsilon)$ Simulation Sound with respect to the protocol Π and man-in-middle adversary $\boldsymbol{\mathcal{A}}$.

When we consider man-in-middle adversaries $\boldsymbol{\mathcal{A}}$, then the permissible actions of $\boldsymbol{\mathcal{A}}$ in the left and the right interactions will be clear from the context. For example, consider the case when Π is a relaxed-concurrent-ZK protocol and Σ is a resettably-sound protocol. In this case, $\boldsymbol{\mathcal{A}}$ can interact with several concurrent sessions of Π in the left interaction, scheduled according to its strategy, but in each session $\boldsymbol{\mathcal{A}}$ has to act like a relaxed adversary. Moreover, in the right interaction $\boldsymbol{\mathcal{A}}$ can reset the protocol Σ.

For any adversary $\boldsymbol{\mathcal{A}}$, if there exists a negligible function ν such that Σ is $(1-\epsilon)$ Simulation Sound with respect to the protocol Π and man-in-middle adversary $\boldsymbol{\mathcal{A}}$, and $\epsilon \leq \nu(n)$, then Σ is Simulation Sound with respect to Π. If Π and Σ are identical, then we say: Σ is Simulation Sound. This definition of Simulation Soundness is consistent with the one introduced by Sahai in [23] for Simulation Sound Non-Interactive Zero-Knowledge.

2.2. Fully Resettable Computation

We consider n parties indexed by $[n]$ trying to compute a function of their local inputs using an interactive protocol. Party i has input $x^{(i)}$ and random tape $\omega^{(i)}$. Let $X^{(i)}$ be the pair of input $x^{(i)}$ and random tape $\omega^{(i)}$; and X represent the vector $(X^{(1)}, \ldots, X^{(n)})$. All parties have their random tape independently chosen but when an adversary resets a party, it reuses the same random tape (and the same input). The adversary $\boldsymbol{\mathcal{S}}$ controls the parties indexed by $[m]$ and at any point during the execution of the protocol it can reset any of the honest parties. We shall consider computational security against parties which have been statically corrupted by the adversary $\boldsymbol{\mathcal{S}}$.

Ideal Model: In the ideal world there is a mutually trusted party which can aggregate the inputs provided to it by the various parties, perform the computation $f(\cdot)$ on their behalf and provide them their respective outputs. The execution in the Ideal world proceeds as follows:

Inputs. Each party i has input $x^{(i)}$. Honest parties send their inputs to the trusted party; but corrupted parties may decide to send modified inputs to the trusted party.

Computation. The trusted party computes the function f on all the inputs provided to it by various parties and, say, z represents the outcome of the function.

Output to Malicious parties. The trusted party sends z to every malicious party $[m]$.

Output to Honest parties. The adversary prepares a list of honest parties which should receive the outcome. The trusted party forwards z to all parties whom the adversary intends to receive their output. Remaining honest parties receive $\perp$.

Reset. The adversary can reset the ideal world at any point of time. When the adversary decides to reset the ideal world,

it requests the trusted party to reset all honest parties and the trusted party sends a reset signal to all honest parties. At this point, the ideal world returns to the first stage where honest parties feed their inputs to the ideal functionality.

Output. Any honest party i always reports the outcome z it receives from the trusted party. By convention, all malicious parties report $\perp$. The adversary outputs an arbitrary function of its entire view and the view of all malicious parties $[m]$.

The output of the Ideal world interaction with adversary $\boldsymbol{\mathcal{S}}$ is represented by $\text{IDEAL}_{f,\boldsymbol{\mathcal{S}}}(X)$.

Real Model: For a n-party protocol Σ, honest parties indexed by $[n] \setminus [m]$ follow the protocol honestly while all the malicious parties, indexed by $[m]$, launch an attack coordinated by an adversary $\boldsymbol{\mathcal{A}}$. The adversary can reset any honest party at any point of time during the execution of the protocol (and potentially even bring them out of sync). At the end of the protocol, honest parties report their outputs as instructed by the protocol Σ and all malicious parties report $\perp$. The adversary outputs its whole view. The output of the real world execution of Σ on X according to the procedure mentioned above is represented by $\text{REAL}_{\Sigma,\boldsymbol{\mathcal{A}}}(X)$.

Definition 6 (Fully Resettable Secure Computation): Let f be a multi-party function and Σ be a protocol in the real world. The protocol Σ is a secure protocol for computing f if for every PPT adversary $\boldsymbol{\mathcal{A}}$ in the real world, there exists an EPPT adversary $\boldsymbol{\mathcal{S}}$ in the ideal world such that:

$$\{\text{IDEAL}_{f,\boldsymbol{\mathcal{S}}}(X)\}_{X\in(\{0,1\}^*)^n} \approx_c \{\text{REAL}_{\Sigma,\boldsymbol{\mathcal{A}}}(X)\}_{X\in(\{0,1\}^*)^n}$$

The simulator may need to reset the functionality in the Ideal World several times and the number of times the Ideal functionality is reset by the simulator could possibly be more than the resets performed by the adversary in the Real World, though this number will be bounded by a polynomial in the security parameter.

2.3. Resettable to Stateless

Any protocol which is resettably secure can be transformed to a stateless protocol using relatively standard techniques. In other words, the parties which were allowed to be resettable in the original protocol need not maintain any state at all in the transformed protocol. By a stateless device we mean that the device only supports a "request-reply" interaction (i.e., the device just outputs $f(x)$ when fed with x for some fixed f). We describe the case of two party first assuming both the parties are resettable. Let parties $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ participating in the original resettably secure protocol Σ. Now we define a protocol Σ' having parties $\boldsymbol{P}'_1$ and $\boldsymbol{P}'_2$. Each of these parties will have a secret key of a CCA-2 secure encryption scheme and a secret MAC key. The party $\boldsymbol{P}'_1$ computes the first message to be sent in the protocol Σ' by running $\boldsymbol{P}_1$ internally. However it then sends to $\boldsymbol{P}'_2$ not only the computed message but also an encryption of the current state of $\boldsymbol{P}_1$ and a MAC on it. Party $\boldsymbol{P}'_2$ similarly computes the reply by feeding the received message to $\boldsymbol{P}_2$ and sends to $\boldsymbol{P}'_1$ not only the computed reply but also (a) the received encrypted state of $\boldsymbol{P}_1$ and the MAC, and, (b) an encryption of the current state of $\boldsymbol{P}_2$ and a MAC on it using its own keys. Thus for the next round, $\boldsymbol{P}'_1$ can decrypt, verify and load the received state into $\boldsymbol{P}_1$, feed it the incoming reply and then compute the next outgoing message. $\boldsymbol{P}_2$ can similarly continue with the protocol. The case of multi-party protocols can also be handled by first transforming the given protocol into one where only one party sends a message in any given round and then applying ideas similar to the one for the two party case to this resulting protocol.

3. Simulation Sound Fully Resettable ZK

3.1. Technical Overview

Our starting point in this section is the fully resettable ZK protocol provided in [8]. The major step in constructing such a protocol is the construction of a *relaxed concurrent hybrid sound ZK* protocol (called the DGS protocol from this point onwards). Once a relaxed concurrent hybrid sound ZK protocol is obtained, it can then be compiled into a fully resettable ZK protocol using known techniques [8]. Following the modular approach of [8], we first construct a simulation sound relaxed concurrent hybrid sound ZK. Once we obtain such a protocol, we present a compiler to convert it into a simulation sound fully resettable ZK protocol.

In [8], the key ingredient in obtaining a relaxed concurrent hybrid sound ZK was a novel NBB simulation technique. In the DGS protocol, in any given session, the prover makes use of several NBB ZK executions proving different statements. In each NBB ZK execution, as in Barak's protocol [1], the idea is to have the prover commit in advance to a program that claims to predict (using an input of small length) a string that is later randomly chosen by the verifier. The prover then must prove that either its committed program really can predict the verifiers string, or that the statement is true. However since the setting in [8] is more demanding compared to [1], to make the simulation go though, the committed program is additionally allowed *access to an oracle*. The prover then must prove that there exists an oracle such that its committed program (on given the small input and queries to that oracle) can really predict the verifier's string, or, that the statement is true.

Our key modification in order to make the DGS protocol [8] simulation sound is to introduce non-malleability features in the NBB ZK protocol described above. Towards that end, we rely on "two-slot" simulation technique of Pass [18] (see also [21]). We introduce two slots for simulation in our NBB ZK protocol (as in [18]) while still keeping oracle access to the committed program (as in [8]) in both the slots. The full protocol nm-ZK_{TAG} and the description of the language is given in Figure 1. Our NBB ZK protocol nm-ZK_{TAG} can still be used in the DGS construction (since essentially the same proof of security goes through). Furthermore, following

[18], the protocol nm-ZK_{TAG} is also simulation sound in the standalone (as well as in the bounded concurrent) setting.

With this modification, however, it is still far from clear that the final resulting construction rel-conc-hs-ZK_{TAG} (obtained by using the nm-ZK_{TAG} in the DGS construction) is simulation sound. This is because our setting is significantly more demanding than that of [18], [21]. Firstly, there is no apriori fixed bound on the number of left executions. Hence, a man-in-the-middle gets to see any unbounded (polynomial) number of concurrent executions of nm-ZK_{TAG} (and of rel-conc-hs-ZK_{TAG} in general of which nm-ZK_{TAG} is a component). Secondly, the man-in-the-middle is additionally allowed to reset the verifier on the right. Thus, it seems like we need the protocol nm-ZK_{TAG} to be simulation sound in the fully concurrent setting in the presence of reset attacks against the verifier. However, this is precisely the goal that we started with!

The key observation which allows us to still move forward is a specific property of the DGS rewinding strategy. Note that the DGS rewinding strategy can be seen as giving rise to a main thread and several "look ahead threads". In the DGS protocol, in any given thread, only a constant number of NBB ZK arguments are replaced (even though over the course of the entire simulation, every NBB ZK argument will be simulated). Thus, in the left interaction in any given thread, the man-in-the-middle gets to see an unbounded (polynomial) number of executions of nm-ZK_{TAG} out of which only a constant number are being simulated. This, for a standalone verifier, allows us to argue that the soundness of the right execution of nm-ZK_{TAG} is still maintained. However, in our setting, the verifier on the right is resettable. To deal with such a scenario and complete the argument, we first observe that fortunately, the protocol nm-ZK_{TAG} is a constant round public coin protocol (since NBB ZK arguments in both [18] as well as [8] are public coin). We then rely on the ideas behind the BGGL transformation [3] to obtain resettable soundness from constant round public coin protocols.

Once we show that in the right interaction, the protocol nm-ZK_{TAG} remains sound even if there are several concurrent executions of the protocol rel-conc-hs-ZK_{TAG} on the left, we finally move forward to show that protocol rel-conc-hs-ZK_{TAG} is simulation sound. To do this, we convert any attack on protocol rel-conc-hs-ZK_{TAG} into an attack on either nm-ZK_{TAG} or the soundness the ZAP system. More details are given in the next sections.

3.2. Our NBB Technique: Constructing Non-malleable ZK

In this section we define a tag based zero-knowledge argument system ZK_{TAG}, where the parameter $\text{TAG} \in [M]$, where M is polynomial in the security parameter. We assume $\texttt{Com}(\cdot;\cdot)$ is a perfectly binding commitment scheme. The description of the protocol nm-ZK_{TAG} is summarized in Figure 1.

In this protocol, there are two slots 1 and 2. In each slot, the prover commits to a challenge and the verifier reveals a random string of appropriate length. The length of the response string of the verifier in slot 1 and 2 are respectively, $\text{TAG} \cdot n^4$ and $(M - \text{TAG} + 1) \cdot n^4$. It has been shown in [20] that the the proposed scheme is standalone sound.

Lemma 1 ([20]): For any $\text{TAG} \in [M]$, the ZK_{TAG} zero-knowledge argument Protocol nm-ZK_{TAG} (refer Figure 1) is standalone sound.

We re-iterate that Protocol nm-ZK_{TAG} is a constant round public coin zero-knowledge argument system. By compiling Protocol nm-ZK_{TAG} using the BGGL compiler proposed in [3] we can obtain a zero-knowledge argument system which is resettably-sound. We call the compiled protocol as Protocol nm-rs-ZK_{TAG} and will use Protocol nm-rs-ZK_{TAG} in our later constructions.

3.3. Relaxed-concurrent Hybrid-sound ZK

In this section we will use nm-rs-ZK_{TAG}, which is a BGGL compilation of nm-ZK_{TAG}, to construct a Relaxed-Concurrent Hybrid-Sound ZK (and will further prove its simulation soundness in the next subsection). This construction is inspired by the construction provided in [8] and is described in Figure 2.

This protocol is identical to the protocol provided in [8] except that the non-black box ZK it uses is slightly modified. Instead of the non-black box scheme inspired by Barak's construction [1] we use Protocol nm-rs-ZK_{TAG}. We state without proof the following result which follows immediately from the results in [20], [8]:

Lemma 2 ([20], [8]): Protocol rel-conc-hs-ZK_{TAG} is a Relaxed-concurrent Hybrid-sound (tag based) ZK protocol. We emphasize that the "relaxed-concurrent" property of the adversary is utilized by the Simulator to provide simulated proofs for Protocol nm-rs-ZK_{TAG} to the adversary using the simulator of Protocol nm-rs-ZK_{TAG} which is not guaranteed to work in the general concurrent setting.

3.4. Simulation Soundness

In this section, we intend to show that rel-conc-hs-ZK_{TAG} is in fact simulation sound as well. We start by proving two fundamental results:

Lemma 3 ([8]): The rZAP protocol is Simulation-Sound with respect to protocol rel-conc-hs-ZK_{TAG}.

Any man-in-middle adversary $\mathcal{A}$ in the above experiment can perform the following actions:

1) $\mathcal{A}$ acts as relaxed-adversary for concurrent sessions of protocol rel-conc-hs-ZK_{TAG} in the left interaction, and
2) $\mathcal{A}$ can reset the rZAP protocol in the right interaction.

In fact, we can claim that the rZAP protocol is simulation sound with respect to any concurrent zero-knowledge protocol π.

Proof Sketch. Interested readers may refer to the full version for the complete proof. We consider a new

Common Input to $\boldsymbol{P}$ and $\boldsymbol{V}$. x' supposedly in the language $L' \in \mathbf{NP}$.
Auxiliary input to $\boldsymbol{P}$. A $\mathbf{NP}$-witness w for $x' \in L'$.
Protocol. The zero-knowledge argument ZK_{TAG} proceeds as follows:

1) The verifier $\boldsymbol{V}$ chooses a hash function h uniformly at random from a family of collision resistant hash functions $\mathcal{H}$ and sends it to $\boldsymbol{P}$.
2) The prover $\boldsymbol{P}$ sends the commitment $z_1 = \texttt{Com}(h(0))$ to the verifier $\boldsymbol{V}$.
3) The verifier selects a string $r_1 \xleftarrow{\$} \{0,1\}^{\text{TAG}\cdot n^4}$ and sends it to the prover $\boldsymbol{P}$.
4) The prover $\boldsymbol{P}$ sends the commitment $z_2 = \texttt{Com}(h(0))$ to the verifier $\boldsymbol{V}$.
5) The verifier selects a string $r_2 \xleftarrow{\$} \{0,1\}^{(M-\text{TAG}+1)\cdot n^4}$ and sends it to the prover $\boldsymbol{P}$.
6) The prover $\boldsymbol{P}$ and the verifier $\boldsymbol{V}$ execute a witness indistinguishable constant round public coin universal argument [a] [2], where $\boldsymbol{P}$ proves to $\boldsymbol{V}$:
   - $x' \in L'$, or
   - The transcript $\tau = (h, z_1, r_1, z_2, r_2)$ generated above belongs to the language Λ_{TAG} described below.

   We require the communication complexity of this universal argument to be at most $O(n^2)$.

Language Λ_{TAG}. A string $(h, z_1, r_1, z_2, r_2) \in \Lambda_{\text{TAG}}$ if there exists $\langle \mathcal{M}, y_1, y_2, r' \rangle$ such that $(\langle h, z_1, r_1 \rangle, \langle \mathcal{M}, y_1, y_2, r' \rangle) \in \mathcal{R}(\text{TAG} \cdot n^4)$ or $(\langle h, z_2, r_2 \rangle, \langle \mathcal{M}, y_1, y_2, r' \rangle) \in \mathcal{R}((M - \text{TAG} + 1) \cdot n^4)$.

A string $(\langle h, z, r \rangle, \langle \mathcal{M}, y_1, y_2, r' \rangle)$ satisfies the relation $\mathcal{R}(i)$ if the following conditions hold:

1) $|y_2| \leq n^{\log\log n}$, $|r| = i$, $|y_1| \leq i - n$,
2) $z = \texttt{Com}(h(\mathcal{M}); r')$
3) The oracle machine $\mathcal{M}$ makes calls to y_2 with a query q expecting a reply $(s, \tilde{r})$ such that $q = \texttt{Com}(s; \tilde{r})$. It expects the entry $(q, s, \tilde{r})$ corresponding to this query to lie in y_2; otherwise, if such an entry does not exist in y_2 then $\mathcal{M}$ aborts. The machine $\mathcal{M}$, with oracle access to y_2 on input y_1, outputs r in at most $n^{\log\log n}$ steps.

[a] This protocol has a weak proof of knowledge guarantee. There exists an extractor such that, if the prover succeeds to prove a statement with probability ρ then, it extracts a witness for the statement with probability ρ^k, for some constant $k \geq 1$.

Figure 1. Protocol nm-ZK_{TAG} for Non Black-box zero-knowledge argument system ZK_{TAG}.

Common input to $\boldsymbol{P}$ and $\boldsymbol{V}$. x supposedly in $L \in \mathbf{NP}$.
Auxiliary input to $\boldsymbol{P}$. A $\mathbf{NP}$-witness w for $x \in L$.
Protocol. The Relaxed Concurrent Hybrid Sound ZK_{TAG} proceeds as follows:

1) The prover $\boldsymbol{P}$ chooses $2n^2$ challenge strings $ch_i \xleftarrow{\$} \{0,1\}^n$ and sends the verifier $\boldsymbol{V}$ the commitments to each of them, i.e. it sends $z_i = \texttt{Com}(ch_i)$, for all $i \in [2n^2]$.
2) The verifier $\boldsymbol{V}$ prepares a trapdoor trap$= \texttt{Com}(1)$ and sends it to the prover $\boldsymbol{P}$. For the last step of this protocol, the verifier $\boldsymbol{V}$ sends the first message σ of a rZAP to the prover $\boldsymbol{P}$. Now, the verifier proceeds to prepare the first message of $2n^3$ instances of the Blum's 3-round protocol for the statement "trap is a commitment to 1". And sends all these messages to the prover $\boldsymbol{P}$.
3) For each $i \in [2n^2]$ the following is performed:
   a) The prover $\boldsymbol{P}$ sends the string ch_i to the verifier $\boldsymbol{V}$. Using the rs-ZK_{TAG} protocol nm-rs-ZK_{TAG}, $\boldsymbol{P}$ proves to $\boldsymbol{V}$ that: "ch_i is the correct opening of z_i, or $x \in L$." If the verifier accepts the proof then the bits of the string ch_i are interpreted as n challenges corresponding to the second message of the 3-round Blum protocol.
   b) For each $j \in [n]$, the verifier $\boldsymbol{V}$ interprets the j-th bit of ch_i, represented by $ch_i[j\ ]$, as the challenge bit for the $((i-1)n+j)$-th execution of 3-round Blum-protocol. The verifier responds suitably as required by the challenge bit $ch_i[j\ ]$.
4) The prover $\boldsymbol{P}$ provides a rZAP to the verifier for the statement: "trap is a commitment to 1, or $x \in L$."

Figure 2. Protocol rel-conc-hs-ZK_{TAG} realizing Relaxed Concurrent Hybrid-Sound ZK_{TAG}.

man-in-middle adversary which learns the first message sent by the rZAP protocol in the right interaction and then resets it. Then it simulates all the rZAP interaction in the right hand interaction on its own; except for one, the reply is forwarded to the external rZAP verifier. The left side interactions can be easily simulated by $\boldsymbol{\mathcal{S}}_\pi$.

The next lemma, shows that a man-in-middle adversary cannot violate the resettable-soundness of Protocol nm-rs-ZK_{TAG} be interacting with several rel-conc-hs-ZK_{TAG} provers in the left:

Lemma 4 ([3], [20], [8]): The nm-rs-ZK_{TAG} protocol is Simulation-Sound with respect to protocol rel-conc-hs-ZK_{TAG}.

Any man-in-middle adversary which tries to violate the soundness of nm-rs-ZK_{TAG} after seeing rel-conc-hs-ZK_{TAG} proofs can perform the following actions:

1) $\boldsymbol{\mathcal{A}}$ acts as relaxed-concurrent adversary of rel-conc-hs-ZK_{TAG} in the left interaction, and
2) $\boldsymbol{\mathcal{A}}$ can reset the verifier of nm-rs-ZK_{TAG} in the right interaction.

Proof Sketch. Here we explain the two key ideas used to show this result and defer the complete proof to the

full version. If there exists a man-in-middle adversary $\mathcal{A}$ which violates the soundness of nm-rs-ZK$_{\text{TAG}}$ in the right interaction with non-negligibly probability, then we can construct another non-resetting man-in-middle $\mathcal{A}'$ which can violate the soundness of nm-ZK$_{\text{TAG}}$ in the right interaction with non-negligble probability. This new adversary simulates all incarnations of the verifier $\boldsymbol{V}_{\text{nm-rs-ZK}_{\text{TAG}}}$ using a random tape R; except one randomly chosen incarnation, for which it uses an external $\boldsymbol{V}_{\text{nm-ZK}_{\text{TAG}}}$ verifier. The second part of the reduction shows that if such a man-in-middle adversary $\mathcal{A}'$ exists then it can be used to construct a standalone adversary $\mathcal{A}''$ which violates the soundness of nm-ZK$_{\text{TAG}}$ with non-negligible. The main hurdle is to simulate the left interactions of $\mathcal{A}'$ when the adversary gets additional input from the external $\boldsymbol{V}_{\text{nm-ZK}_{\text{TAG}}}$. To show that we can efficiently simulate the left interactions of $\mathcal{A}'$ we need to combine the techniques introduced in [8] and [20].

Simulation Soundness of rel-conc-hs-ZK$_{\text{TAG}}$: The main component of our proof is to show that Protocol rel-conc-hs-ZK$_{\text{TAG}}$ is Simulation-Sound. Consider a man-in-middle adversary which participates as relaxed concurrent verifier in rel-conc-hs-ZK$_{\text{TAG}}$ incarnations in the left and as a prover of rel-conc-hs-ZK$_{\text{TAG}}$ in the right interaction. We prove the result by showing that any soundness violation of rel-conc-hs-ZK$_{\text{TAG}}$ in the right interaction can be attributed to soundness violation of nm-rs-ZK$_{\text{TAG}}$ or rZAP protocol. Suppose, there exists a man-in-middle adversary $\mathcal{A}$ which violates the soundness of Protocol rel-conc-hs-ZK$_{\text{TAG}}$ in the right interaction. We consider the event that the adversary $\mathcal{A}$ violates the soundness of Protocol nm-rs-ZK$_{\text{TAG}}$ associated with one of the challenges in the right interaction with significant probability or not. If this event occurs then we can construct a man-in-middle adversary which violates the simulation soundness of Protocol nm-rs-ZK$_{\text{TAG}}$ with respect to Protocol rel-conc-hs-ZK$_{\text{TAG}}$. Otherwise, i.e. the adversary *never*, except with negligible probability, violates the soundness of any nm-rs-ZK$_{\text{TAG}}$ associated with any challenge, then we can construct a man-in-middle adversary which violates the simulation soundness of rZAP protocol with respect to Protocol rel-conc-hs-ZK$_{\text{TAG}}$.

Lemma 5: The rel-conc-hs-ZK$_{\text{TAG}}$ protocol is Simulation Sound. The man-in-middle adversary $\mathcal{A}$ can perform the following operations:

1) $\mathcal{A}$ is a relaxed-concurrent verifier for polynomially many sessions of Protocol rel-conc-hs-ZK$_{\text{TAG}}$ in the left interaction, and
2) $\mathcal{A}$ acts as a hybrid-prover for Protocol rel-conc-hs-ZK$_{\text{TAG}}$ running in the right interaction, i.e. it can partially reset the verifier. It can run several concurrent sessions with various hybrid-verifier incarnations for the protocol and restart the authenticator part of the hybrid-verifier, although leaving the main part of the hybrid-verifier in the same state.

Proof Sketch. Here we provide the key ideas necessary to prove the statement and the complete proof is deferred to the full version. There are two main cases to consider. First, suppose a man-in-middle adversary $\mathcal{A}$ is able to prove a false statement "$\tilde{x} \in L$" in some slot of the right interaction by launching a hybrid attack on the verifier $\boldsymbol{V}_{\text{rel-conc-hs-ZK}_{\text{TAG}}}$ of rel-conc-hs-ZK$_{\text{TAG}}$ after seeing simulated proofs in the left interaction with non-negligible probability. Then we can convert it into a man-in-middle adversary who can violate the resettable soundness of a nm-rs-ZK$_{\text{TAG}}$ protocol after seeing simulated rel-conc-hs-ZK$_{\text{TAG}}$ proofs in the left interaction. The second case considers the event that the adversary does not, except with negligible probability, break the resettable soundness of nm-rs-ZK$_{\text{TAG}}$ in any slot in the right interaction. For this case, we can show that violation of the soundness on rel-conc-hs-ZK$_{\text{TAG}}$ in the right interaction implies that the soundness of the rZAP protocol was violated. Furthermore, we can construct a standalone simulator violating the soundness guarantee of the rZAP protocol which, due to Lemma 3, is impossible.

Theorem 1 (Main Theorem): Protocol rs-rZK$_{\text{TAG}}$ is Simulation Sound Resettably-sound Resettable-ZK.

4. Fully Resettable Computation

Suppose $\mathcal{F}$ is a general n-party functionality where parties have N bit inputs. For every partition of the party indices into malicious ($\mathcal{S}$) and honest ($\bar{\mathcal{S}} = [n] \setminus \mathcal{S}$) sets, we can consider a two-party version of $\mathcal{F}_{\mathcal{S}}$ which takes two inputs: the first input is the input of all honest parties and the second input is the input of all the malicious parties; and outputs the concatenation of honest party outputs to the first party and the concatenation of malicious party outputs to the second party. In this section, we will present a secure protocol to perform fully resettable computation of functionalities which satisfy the technical property given below. The multi-party functionality $\mathcal{F}$ satisfies the GM-technical property if for every partition of input indices into honest and malicious parties, the resulting two-party version of the $\mathcal{F}$ satisfies Definition 7.

Definition 7 (GM-Technical Property): A two-party functionality $\mathcal{F}$ (defined for inputs of a certain length) is said to satisfy the GM-Technical Property if there exists an efficient *compression algorithm* $\mathcal{C}$, an efficient *decompression* algorithm $\mathcal{D}$ and a polynomial bound B such that, for all honest inputs x, polynomial μ and adversarial inputs $x_1^*, \ldots, x_\mu^*$ the following holds: Define the *suggestion string* $s := \mathcal{C}(x_1^*, \ldots, x_\mu^*, \theta_1^*, \ldots, \theta_\mu^*)$, then

1) Either, s is the abort symbol with negligible probability,
2) Or,

$$|s| \leq B, \text{ and}$$
$$\text{For all } i \in [\mu],\ \mathcal{D}(s; x_1^*, \ldots, x_i^*, \theta_1^*, \ldots, \theta_{i-1}^*) = \theta_i^*$$

Here θ_i^* is the output received by the adversaries when the functionality $\mathcal{F}$ in invoked with honest parties' input set to x and adversarial parties' input set to x_i^*.

Intuitively, the definition says that there exists a good compression and a good decompression algorithm such that, except with negligible probability, the compression algorithm generates a small suggestion string s which helps the decompression algorithm correctly predict the i-th output received by the adversary given its prior input-output pairs and the current input.

We emphasize that B depends on the size of the honest party input x. Note that if we allow the compression algorithm $\mathcal{C}$ to run in super-polynomial time, the above condition is trivially satisfied for all functionalities. This is because otherwise $\mathcal{C}$ can exhaustively find an honest input x consistent with all the input-output behavior and send it as the suggestion string s.

We will show that any functionality which does *not* satisfy the GM-Technical Property is, intuitively, a (worst case) pseudo-entropy source according to the following definition:

Definition 8 (Worst-Case Pseudo-entropy Generation Source). A keyed-function f is called a Worst-Case Pseudo-entropy Generation Source, if for all prediction algorithms $\mathcal{P}$, there exists a key k, polynomial μ, inputs $x_1^*, \ldots, x_\mu^*$ and $u \geq 1$ such that the size of the following set is greater than $100u\tilde{N}$:

$$\left\{ i \;\middle|\; i \in [m] \text{ and } \Pr\left(\mathcal{P}(x_1^*, \ldots, x_i^*, \theta_1^*, \ldots, \theta_{i-1}^*) \neq \theta_i^*\right) \geq \frac{1}{u} \right\}$$

Here $\theta_i^* = f(k; x_i^*)$ and $\tilde{N}$ is the size of key, i.e. $|x|$.

In other words, "unpredictability" of the outputs is much higher than warranted by the length of the unknown key.

Lemma 6: Let $\mathcal{S}$ be the set of malicious parties and $\mathcal{F}_\mathcal{S}(x, x^*)$ represent the output of the adversaries when concatenation of honest party inputs is x and concatenation of adversary inputs is x^*. Define its corresponding keyed function $f_\mathcal{S}(k; x) = \mathcal{F}_\mathcal{S}(k, x)$. If the functionality $\mathcal{F}$ does not satisfy the GM-Technical Property then there exists S such that $f_\mathcal{S}$ is a w-PEG.

4.1. Simulation-soundness in presence of Output Oracle

To use simulation soundness guarantees while constructing protocols for secure fully resettable computations, we need to extend the simulation soundness result of Theorem 1 to stronger man-in-middle adversaries. We consider adversaries which have access to an output oracle, i.e. the adversary can query the oracle on input x^* and receive its output $\mathcal{F}(x, x^*)$, where the honest party's input to the functionality is x. Since, such an adversary can query the output oracle any polynomial number of times, it is evident that the simulation strategy used to prove Lemma 4 is not applicable in straight-forward fashion. If the functionality $\mathcal{F}$ satisfies the GM-Technical Property, then we extend the simulation soundness guarantee of Theorem 1 to man-in-middle adversaries with access to output oracles.

4.1.1. Modifying Language Λ_{TAG}: To motivate the change in description of language Λ_{TAG}, consider the following example. Suppose a man-in-middle adversary $\boldsymbol{\mathcal{A}}$ interacts as a relaxed concurrent verifier in several instances of rel-conc-hs-ZK$_{\text{TAG}}$ in the left; while it interacts as a prover for nm-ZK$_{\text{TAG}}$ in the right interaction. Recall, that $\boldsymbol{\mathcal{A}}$ also has access to an output oracle. Now, assuming that $\boldsymbol{\mathcal{A}}$ can violate the soundness of the instance of nm-ZK$_{\text{TAG}}$ running in the right interaction, we cannot construct a standalone adversary which violates the soundness of nm-ZK$_{\text{TAG}}$ using the simulation strategy of [8], [21]. The simulation of the left interactions will fail because, in a look ahead thread between the point where the simulator commits the code and the point where it receives the challenge string from the adversary, the adversary could query the output oracle on polynomially many inputs. To predict the random challenge, the code committed by the simulator might need to simulate the answers to all the queries made to the output oracle by $\boldsymbol{\mathcal{A}}$.

Here we plan to leverage the fact that $\mathcal{F}$ satisfies the GM-Technical Property (Definition 7). We will use the fact that the compression algorithm $\mathcal{C}$ can produce a bounded length string s after receiving $x_1^*, \ldots, x_\mu^*, \theta_1^*, \ldots, \theta_\mu^*$, where $\theta_i^* = \mathcal{F}(x, x_i^*)$. This suggestion string s can help the decompression algorithm $\mathcal{D}$ predict θ_i^* given $x_1^*, \ldots, x_i^*, \theta_1^*, \ldots, \theta_{i-1}^*$. So, the code committed by the simulator could use the suggestion string s as input and internally use the decompression algorithm $\mathcal{D}$ to predict the answers to the queries made by $\boldsymbol{\mathcal{A}}$.

Language Definition Modification.: Suppose $\mathcal{F}$ is a functionality which satisfies the GM-Technical Property (Definition 7). A string $(h, z_1, r_1, z_2, r_2) \in \Lambda_{\text{TAG}}$ if there exists $\langle \mathcal{M}, y_1, y_2, r' \rangle$ such that $(\langle h, z_1, r_1 \rangle, \langle \mathcal{M}, y_1, y_2, r' \rangle) \in \mathcal{R}(\text{TAG} \cdot n^4 + B)$ or $(\langle h, z_2, r_2 \rangle, \langle \mathcal{M}, y_1, y_2, r' \rangle) \in \mathcal{R}((M - \text{TAG} + 1) \cdot n^4 + B)$. Recall that a string $(\langle h, z, r \rangle, \langle \mathcal{M}, y_1, y_2, r' \rangle)$ satisfies the relation $\mathcal{R}(i)$ if the following conditions hold:

1) $|y_2| \leq n^{\log \log n}$, $|r| = i$, $|y_1| \leq i - n$,
2) $z = \texttt{Com}(h(\mathcal{M}); r')$
3) The oracle machine $\mathcal{M}$ makes calls to y_2 with a query q expecting a reply $(s, \tilde{r})$ such that $q = \texttt{Com}(s; \tilde{r})$. It expects the entry $(q, s, \tilde{r})$ corresponding to this query to lie in y_2; otherwise, if such an entry does not exist in y_2 then $\mathcal{M}$ aborts. The machine $\mathcal{M}$, with oracle access to y_2 on input y_1, outputs r in at most $n^{\log \log n}$ steps.

4.1.2. Non-Black Box Simulation Modification: Consider a man-in-middle adversary $\boldsymbol{\mathcal{A}}$ which interacts as relaxed concurrent verifier of rel-conc-hs-ZK$_{\text{TAG}}$ in its left interactions and also interacts as a prover of nm-ZK$_{\text{TAG}}$ in its right interaction. We will show that if $\boldsymbol{\mathcal{A}}$ violates the soundness of nm-ZK$_{\text{TAG}}$ in the right interaction, then we can construct a standalone simulator which violates the soundness of nm-ZK$_{\text{TAG}}$.

As indicated above, we will need to modify the non-black box simulator strategy of [8] suitably so that the standalone simulator can simulate the left interactions even when the adversary has access to an output oracle and interacts with an external $\boldsymbol{V}_{\text{nm-ZK}_{\text{TAG}}}$. Consider a look ahead thread generated by the simulator and let j be a simulated slot in session i in the left interaction. Consider the transcript generated between the point where the simulator commits the machine and receives the challenge string from $\boldsymbol{\mathcal{A}}$. Let $x_1^*, \ldots, x_\mu^*$ be the inputs for which $\boldsymbol{\mathcal{A}}$ queries the output oracle during this transcript generation; and let $\theta_1^*, \ldots, \theta_\mu^*$ be the respective answers received by $\boldsymbol{\mathcal{A}}$ from the output oracle. We run $\mathcal{C}$ with input $x_1^*, \ldots, x_\mu^*, \theta_1^*, \ldots, \theta_\mu^*$ and, except with negligible probability of abort, we receive a suggestion string s of length at most B.

We will provide the string s as an input to the code to provide a non-black box proof in the simulated slot. Using the proof technique of [8], [21], we can generate the first query x_1^* to the output oracle. Next, we generate the answer θ_1^* for this query using $\mathcal{D}(s; x_1^*)$. Thereafter, given (x_1^*, θ_1^*), we can generate the next query x_2^* using the technique of [8], [21] and its answer $\theta_2^* = \mathcal{D}(s; x_1^*, x_2^*, \theta_1^*)$. Inductively, the simulator will be able to regenerate all the query-answer pairs for all μ using an additional length B input to the committed code.

Using the modified language Λ_{TAG} and modifying the non-black box proof technique of [8], we can extend Lemma 4 to man-in-middle adversaries which have access to output oracles. Thus, we can conclude the following generalization of Theorem 1:

Theorem 2 (Main Theorem in Output Oracle World): Let $\mathcal{F}$ be a functionality which satisfies the GM-Technical Property. Protocol rs-rZK$_{\text{TAG}}$ is Simulation Sound Resettably-sound Resettable-ZK even when adversaries are permitted to invoke the $\mathcal{F}$ output oracle.

4.2. Protocol Description

Suppose the parties participating in the protocol are indexed by $[n]$ and they wish to securely compute a function f of their local inputs $x^{(1)}, \ldots, x^{(n)}$. Given a semi-honest secure protocol Π realizing this function, we will provide a general compiler to transform it into a protocol Σ which is fully resettable secure. The compiler is described in Figure 3 Without loss of generality, we will assume that parties $[m]$ are corrupt and $\{m+1, \ldots, n\}$ are honest. As a convention for our notation, we will always use the superscript (i) to represent a variable of party i, for example the input of party i is denoted by $x^{(i)}$. To prove a statement, party i will always use Protocol rs-rZK$_{\text{TAG}}$ with tag i and $M \geq n$. Simulation soundness of rs-rZK$_{\text{TAG}}$ will imply that any adversarial party $i \in [m]$ cannot violate the simulation soundness of Protocol rs-rZK$_{\text{TAG}}$ with tag i even if it gets to see several Protocol rs-rZK$_{\text{TAG}}$ proofs, possibly simulated ones, with tag $j \neq i$. We restrict our attention to deterministic functionalities (for a treatment of randomized functionalities, please see the full version).

Figure 3 summaries our fully resettable secure computation protocol for any deterministic functionality. It runs in four phases. At the end of every phase, every party proves to other parties that it has followed the protocol honestly using rs-rZK$_{\text{TAG}}$. In the first phase, parties generate a determining phase where each party commits to its input $x^{(i)}$, local random tape $R^{(i)}$, a pseudorandom function $G^{(i)}$ and seed $s^{(i)}$. Each party also generates a lossy function index $u^{(i)}$ and, finally, broadcasts the determining message $D^{(i)}$ is defined as the concatenation of these commitments and the lossy function index. In the next phase, every party leaks $x^{(i)} \parallel R^{(i)} \parallel G^{(i)} \parallel s^{(i)}$ using a semantically-secure encryption scheme with key $u^{(j)}$ to party j, such that it satisfies the following property: If $u^{(j)}$ is an injective function index then the message can be recovered from the cipher text; otherwise, if it is a lossy function index, then it cannot be recovered. In the third phase, a random pad is generated which is uniform if any one of the parties is honest. Each party applies the pseudorandom function $G^{(i)}$ to the concatenation of all determining messages $D = D^{(1)} \parallel D^{(2)} \parallel \cdots \parallel D^{(n)}$. The XOR of these respective random tapes forms a global random mask and every party uses an exclusive portion of this pad in further computations. Finally, parties simulate the semi-honest protocol using a random tape generated from their local random tape and their portion of the global mask generated in the previous phase.

The simulator, on the other hand, sends an injective function key $u^{(i)}$ in the first phase and simulates a proof for the correctness of that phase. In every session, once the deterministic message generation phase has been completed, the simulator uses the witness "$u^{(i)}$ is an injective function index" to complete the proofs in subsequent phases. Observe that this can be done in a straightforward fashion after the first phase's proof has been successfully simulated in a session. Interested readers can refer to the full version for details of the simulator and the proof of security. Finally, we state our main theorem statement for Fully Resettable Computation:

Theorem 3: Protocol in Figure 3 is a Fully Resettable Computation protocol for $\mathcal{F}$ when $\mathcal{F}$ satisfies the GM-Technical Property (Definition 7) or $\mathcal{F}$ is a randomized functionality without any input.

5. Concurrent Secure Computation: Separating the Power of NBB Simulation from BB Simulation

Let Σ be a non-malleable tag-based rs-ZK protocol. Consider a functionality $\mathcal{F}$ running between a prover $\boldsymbol{P}$ and a verifier $\boldsymbol{V}$. Let $\mathcal{F}$ be the following functionality:

1) The prover $\boldsymbol{P}$ sends a witness w and the statement "$x \in L$" to the trusted party.

Algorithm for Party i: Local input $x^{(i)}$ for Π, local random tape $R^{(i)}$, a pseudorandom function $G^{(i)}$ chosen uniformly at random from the family $\mathcal{G}$ and a random seed $s^{(i)}$ for lossy function generation. A $(2n', n')$-lossy trapdoor function is defined by the four tuple of PPT algorithms $(S_{\text{inj}}, S_{\text{loss}}, F_{\text{ltdf}}, F_{\text{ltdf}}^{-1})$. Let $\mathcal{H}_{2-\text{univ}}$ be a 2-universal hash function family from $2n'$ bits to n' bits. Given a function index u, the encryption of x using u is defined as $\text{Enc}(x; u) = (x \oplus h(r)) \parallel h \parallel F_{\text{ltdf}}(u, r)$ for a randomly chosen r and $h \xleftarrow{\$} \mathcal{H}_{2-\text{univ}}$.

1) **Determining Message Generation:**
   a) Compute the commitments $\alpha_1^{(i)} = \texttt{Com}(x^{(i)})$, $\alpha_2^{(i)} = \texttt{Com}(R^{(i)})$, $\alpha_3^{(i)} = \texttt{Com}(G^{(i)})$, $\alpha_4^{(i)} = \texttt{Com}(s^{(i)})$.
   b) Let $u^{(i)}$ be the index of the lossy function obtained by running $(u^{(i)}, \perp) = S_{\text{loss}}(s^{(i)})$.
   c) Interpret $R^{(i)} = r_1^{(i)} \parallel r_2^{(i)}$. Use the random tape $r_2^{(i)}$ to obtain randomness for all prover/verifier algorithms to run Protocol rs-rZK$_{\text{TAG}}$.
   d) Broadcast the determining message $D^{(i)} = \alpha_1^{(i)} \parallel \alpha_2^{(i)} \parallel \alpha_3^{(i)} \parallel \alpha_4^{(i)} \parallel u^{(i)}$. Using protocol rs-rZK$_{\text{TAG}}$ with tag i, prove to party $j \in [n] \setminus \{i\}$ that: "$\exists\ x, r, g, s$ such that, (i) $\alpha_1^{(i)}$, $\alpha_2^{(i)}$, $\alpha_3^{(i)}$ and $\alpha_4^{(i)}$ are commitments to x, r, g and s respectively, and (ii) $(u^{(i)}, \perp) = S_{\text{loss}}(s)$.

2) **Simulation Extraction Phase:**
   a) After receiving the determining message $D^{(k)}$ for every $k \in [n] \setminus \{i\}$ and verifying their associated proof, calculate $\beta_k^{(i)} = \text{Enc}(x^{(i)} \parallel R^{(i)} \parallel G^{(i)}; u^{(k)})$.
   b) Broadcast $\beta^{(i)} = \beta_1^{(i)} \parallel \beta_2^{(i)} \parallel \ \dots\ \parallel \beta_n^{(i)}$, where $\beta_i^{(i)}$ is an all 0 string. Using Protocol rs-rZK$_{\text{TAG}}$ with tag i, prove to party $j \in [n] \setminus \{i\}$ that: "$u^{(i)}$ is an injective function index", or "$\exists\ x, r, g, s$ such that, (i) $\alpha_1^{(i)}$, $\alpha_2^{(i)}$, $\alpha_3^{(i)}$ and $\alpha_4^{(i)}$ are commitments to x, r, g and s respectively, (ii) $(u^{(i)}, \perp) = S_{\text{loss}}(s)$, and (iii) For all $k \in [n] \setminus \{i\}$, $\beta_k^{(i)} = \text{Enc}(x \parallel r \parallel g; u^{(k)})$."

3) **Coin Tossing Phase:**
   a) Let $D = D^{(1)} \parallel D^{(2)} \parallel \cdots \parallel D^{(n)})$ and evaluate $r_3^{(i)} = G^{(i)}(D)$.
   b) Broadcast $r_3^{(i)}$ and use Protocol rs-rZK$_{\text{TAG}}$ with tag i to prove to every other party $j \in [n] \setminus \{i\}$ that: "$u^{(i)}$ is an injective function index", or "$\exists\ g$ such that $\alpha_3^{(i)} = \texttt{Com}(g)$ and $r_3^{(i)} = g(D)$."
   c) After receiving $r_3^{(j)}$ for all $j \in [n] \setminus \{i\}$, compute $a^{(1)} \parallel a^{(2)} \parallel \cdots \parallel a^{(n)} = a = \oplus_{k \in [n]}\ r_3^{(k)}$. Define $r^{(i)} = a^{(i)} \oplus r_1^{(i)}$.

4) **Computation Phase:** Before broadcasting the message for round t of the protocol Π, let the party's current view be $V_t^{(i)} = (\tau_{t-1}, x^{(i)}, r^{(i)})$.
   a) Let $msg_t^{(i)}$ be the next message obtained by applying the next message function of Π on the view $V_t^{(i)}$.
   b) Broadcast $msg_t^{(i)}$ and prove to every party $j \in [n] \setminus \{i\}$ the statement: "$u^{(i)}$ is an injective function index", or "$\exists\ v$ such that $msg_t^{(i)}$ is the next message for view v in protocol Π," using Protocol rs-rZK$_{\text{TAG}}$ with tag i.

Figure 3. Fully Resettable Secure Computation.

2) The trusted party verifies whether w is a witness for "$x \in L$" or not. If it is not a witness of "$x \in L$" then it sends $\perp$ to the verifier $\boldsymbol{V}$. Otherwise, the trusted party picks a secret key sk, sends $pk = Gen(sk)$ to the verifier $\boldsymbol{V}$. Next, the trusted party acts as a prover of the protocol Σ with tag pk and engages in an interactive protocol with the verifier $\boldsymbol{V}$ (who acts as a verifier of Σ). The protocol Σ with tag $pk = pk_1 pk_2 \dots pk_l$ is a parallel repetition of protocol Σ' with tags (i, pk_i) for all $i \in [l]$. Finally, the trusted party signs the transcript τ generated by the protocol Σ with the secret key sk and sends the signature $\sigma = Sign_{sk}(\tau)$ to the verifier. The proof is accepted if and only if the verifier of Σ accepts all the Σ' proofs and σ verifies as a valid signature of τ using the public key pk, i.e. $Ver_{pk}(\tau, \sigma) = 1$.

We crucially rely on the non-malleability of the zero-knowledge protocol and the security of the digital signature scheme in our argument, details of which can be found in the full version.

ACKNOWLEDGEMENT

We would like to thank Abhishek Jain, Rafail Ostrovsky, and Ivan Visconti for pointing out an error in the proceedings version of this work. The current version is the corrected one.

REFERENCES

[1] Boaz Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115. IEEE, 2001. Preliminary full version available on http://www.math.ias.edu/~boaz.

[2] Boaz Barak and Oded Goldreich. Universal arguments and their applications. Cryptology ePrint Archive, Report 2001/105, 2001. Extended abstract appeared in CCC' 2002.

[3] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. Record 2001/063, Cryptology ePrint Archive, August 2001. Preliminary version appeared in FOCS' 01.

[4] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *FOCS*, pages 384–393, 2003.

[5] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge. In *Proc. 32th STOC*, pages 235–244. ACM, 2000.

[6] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for uc secure computation using tamper-proof hardware. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 545–562. Springer, 2008.

[7] Yi Deng, Dengguo Feng, Vipul Goyal, Dongdai Lin, Amit Sahai, and Moti Yung. Constant-round simultaneously resettable zero-knowledge in the bpk model. In *Manuscript*, 2011.

[8] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009.

[9] Yi Deng and Dongdai Lin. Instance-dependent verifiable random functions and their application to simultaneous resettability. In Naor [16], pages 148–168.

[10] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[11] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In ACM, editor, *Proc. 19th STOC*, pages 218–229. ACM, 1987. See [10, Chap. 7] for more details.

[12] Vipul Goyal and Abhishek Jain 0002. On the round complexity of covert computation. In Leonard J. Schulman, editor, *STOC*, pages 191–200. ACM, 2010.

[13] Vipul Goyal and Amit Sahai. Resettably secure computation. In *EUROCRYPT*, pages 54–71, 2009.

[14] S. Huang, D. MacCallum, and D.Z. Du. *Network Security*. Springer, 2010.

[15] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In *CRYPTO*, pages 542–565, 2001.

[16] Moni Naor, editor. *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*. Springer, 2007.

[17] Rafail Ostrovsky, Omkant Pandey, Amit Sahai, and Ivan Visconti. Non-malleability under reset attacks. In *Manuscript*, 2011.

[18] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th STOC*, pages 232–241. ACM, 2004.

[19] Rafael Pass. Limits of security reductions from standard assumptions. In *STOC*, 2011.

[20] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *FOCS*, pages 563–572, 2005.

[21] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *Proc. 37th STOC*. ACM, 2005.

[22] Rafael Pass, Wei-Lung Dustin Tseng, and Douglas Wikström. On the composition of public-coin zero-knowledge protocols. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2009.

[23] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

[24] Andrew Chi-Chih Yao. Protocols for secure computation. In *Proc. 23rd FOCS*, pages 160–164. IEEE, 1982.

[25] Moti Yung and Yunlei Zhao. Generic and practical resettable zero-knowledge in the bare public-key model. In Naor [16], pages 129–147.