UNIVERSITY OF CALIFORNIA

Los Angeles

Constant-round protocols of stronger security via relaxed set-up assumptions

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Chongwon Cho

2013

ABSTRACT OF THE DISSERTATION

Constant-round protocols of stronger security via relaxed set-up assumptions

by

Chongwon Cho

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2013

Professor Rafail Ostrovsky, Chair

The main aim of cryptography is to provide the frameworks and solutions for information security. The fundamental weapons to protect information are interactions and private randomness. Since the breakthrough result, zero-knowledge proof system, by Goldwasser, Micali, and Rackoff in mid 80s, the cryptography community has endeavored to propose the new notions of information security and the relative solutions which more closely reflect the modern computing environment.

Concurrent security first introduced by Dwork, Naor, and Sahai was suggested to capture the information security in the modern internet environment. That is, the adversary may interact with a honest parties in many concurrent executions of a protocol where the messages are scheduled in any adversarial way.

Resettable security was first introduced by Canetti, Goldreich, Goldwasser, and Micali, which models the security issues in which parties have a limited source of private randomness. In other words, an adversary might interact with honest parties in many executions of a protocol while the honest parties are only allowed to use the polynomially bounded number of (hard-wired) randomness.

An important question is: "How much efficient protocol can we construct to achieve the above security in terms of round complexity?" Unfortunately, it has been showed that the best possible round complexity for such protocols is poly-logarithmic in the security parameter based on *black-*

box simulation without help of external set-up assumptions. Thus, the question is now to minimize the round complexity of protocols with a minimal set-up assumption.

In this thesis, we positively answer the above question with help of set-up assumptions. Specifically, we consider two set-up assumptions, the Bare Public Key (BPK) model and the Cross-Domain (CD) model. The BPK model was first introduced by Canetti, Goldreich, Goldwasser, and Micali. In the BPK model, each party is required to register their public keys before the start of interacting with each other. The CD model is a newly proposed model in this work. In the CD model, we have domains which models key certification authorities in the real world and each party belongs to one of the domains.

In the BPK model, we show a constructions of constant-round simultaneously resettable zero-knowledge argument of knowledge with a standard cryptographic assumptions. As a main building block for this result, we show a construction of constant-round simultaneously resettable witness-indistinguishable argument of knowledge.

In the CD model, we show a construction of constant-round concurrently secure multi-party computation protocol with the fixed number of domains. On the other hand, we prove that if the number of domains is not fixed, such a constant-round protocol does not exist.

The dissertation of Chongwon Cho is approved.

Alexander Sherstov

Amit Sahai

Igor Pak

Rafail Ostrovsky, Committee Chair

University of California, Los Angeles

2013

To my wife Julia and my son Brent.

Most specially to my parents, Kang-jin and Kwang-won.

TABLE OF CONTENTS

LIST OF FIGURES

Acknowledgments

First of all, I would like to deeply thank my advisor, Professor Rafail Ostrovsky. Rafi has been supporting me not only in my research but also in general life of me and my family. He got me into the interesting world of cryptography and gave me countless advices in the course of my research. It would have been impossible for me to complete my Ph.D. without his mentor-ship.

Furthermore, I would like to thank Professor Amit Sahai very much as well. Throughout my Ph.D. study at UCLA, his insightful seminar classes broadened my vision in cryptography. His explanations on subjects are always crisp and perfectly structured, which makes it extremely helpful to those people starting to work on new subjects.

I also thank all of my co-authors, Professor Ivan Visconti, Dr. Alessandra Scafuro, and my friends Sanjam Garg and Chen-Kuei Lee. Ivan and Alessandra gladly spent their time and energy to advise me when we worked on a project. Sanjam always inspires me with his sharp intelligence. Chen-Kuei and I shared the experience of starting Ph.D. research and overcame the difficulties together during my first research project. I am in debt to my co-authors as they graciously gave me their permission for me to use the research works in this dissertation. I clearly acknowledge that this dissertation is a result of combining two previous works in the almost verbatim manner with minor editions in terms of format, orders of contents, addition of figures, etc.: One is *Simultaneously Resettable Arguments of Knowledge* with Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. The other is *Cross-Domain Secure Protocols* with Sanjam Garg and Rafail Ostrovsky.

I would like to thank all my friends in the cryptography group at UCLA. Specially, I am in debt to Ran Gelles. Ran has been my great friend during my Ph.D. study at UCLA. We had shared a lot of fun and various experiences together as lab-mates during this tough period of time.

Finally, I would like to express my thankfulness to my family. Since my wife Julia and my son Brent have come into my life, they have supported me as a family. At uttermost, even though I realize that nothing would not be enough to express my thankfulness, I here would like to thank my parents for their sacrificial love and support.

VITA

1996–2000	B.S. (Mathematics), Dongguk University.
2000–2003	Army officer (Lieutenant), Korean Army.
2004–2007	B.S. (Computer Science), with Magna Cum Laude, Iowa State University, Ames, Iowa.
2007–2009	M.S. (Computer Science), UCLA, Los Angeles, California.
Spring 2011	Teaching Assistant, Computer Science Department, UCLA. Taught discussion sections for CS180.
Spring 2012	Teaching Assistant, Computer Science Department, UCLA. Taught discussion sections for CS180.
2010 – present	Graduate Student Researcher, Computer Science Department, UCLA.

PUBLICATIONS

Equivalence of Uniform Key Agreement and Composition Insecurity, CRYPTO, 2010, Santa Barbara, California, USA, with Chen-kuei Lee, Rafail Ostrovsky.

Simultaneously Resettable Arguments of Knowledge, TCC, 2012, Taormina, Italy, with Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti.

Cross-Domain Secure Protocols, Manuscript (in submission), 2013, with Sanjam Garg, Rafail Ostrovsky.

CHAPTER 1

Introduction

1.1 The cryptographic problem concerned in this thesis

The main aim of cryptography is to provide the frameworks and solutions for information security. In the modern cryptography, the fundamental weapons to protect information are interactions and private randomness. The breakthrough result, zero-knowledge proof system, by Goldwasser, Micali, and Rackoff [GMR85] brought the state of art in cryptography one step further as it introduced the security definition based on a mental experiment, so called *simulation*. To see the very high-level intuition of simulation-based security, consider that two parties interact with each other to compute some functionality. Then, according to the simulation-based security, the protocol run by two parties is secure if there exists an imaginary adversary (called *simulator*) to by itself generate an interaction record (almost) identical to the real interaction record. Since the introduction of usage of mental experiment to prove the security of cryptographic protocol, the researchers in the cryptography community have endeavored to propose the new notions of security which more closely reflect the fast evolving modern computing environment.

The emergence of internet aroused a new challenge. The computing systems no longer interact with each other in the one to one fashion but one to many or possibly many to many. The notion of concurrent security introduced by Dwork, Naor, and Sahai [DNS98] was suggested to capture the new requirement of security in the modern internet environment. That is, a adversary, called concurrent adversary, may interact with honest parties in many concurrent executions of a protocol where the exchange of messages are scheduled by the adversary in any adversarial way. A protocol is said to be concurrently secure if the protocol remains secure against such a concurrent adversary.

The use of stateless devices also evoked a new cryptographic problem. Unlike a usual computing system, stateless devices may use only the handful amount of private randomness which is hard-wired to the devices. One of such exemplary devices is a smart card system. In the example of smart card systems, a smart card (a prover) is required to provide a credential to a card reader (a verifier) where the credential contains a private information for the smart card to protect. Hence, so-called resettable security was introduced by Canetti, Goldreich, Goldwasser, and Micali [CGGM00] in order to reflect the above security issue where the prover has a limited source of private randomness to protect its private information. On the other hand, the opposite direction of the above security was introduced by Barak, Goldreich, Goldwasser and Lindell [BGGL01] where the verifier only has a limited source of randomness.[1] Finally, in [BGGL01], simultaneous resettable security was proposed where both parties have the limited amount of randomness.

A common important question related to the above topics is “How much efficient protocol can we construct to achieve the above security in terms of round complexity?” Unfortunately, it has been showed that the best possible round complexity of concurrently secure protocol is at least poly-logarithmic in the security parameter based on *black-box* simulation in the standard model. It is well-known that the source of such round-inefficiency is due to the complexity of simulation. Thus, the main question of the thesis is “Can we construct more round-complexity efficient protocol with the help of set-up assumptions?”

1.2 Set-up assumptions considered in the thesis

In this thesis, we positively answer the above question with help of set-up assumptions. Specifically, we consider two set-up assumptions, the Bare Public Key (BPK) model and the Cross-Domain (CD) model.

The BPK model was first introduced by Canetti, Goldreich, Goldwasser, and Micali [CGGM00]. In the BPK model, each party is required to register their public keys with the key registration functionality before beginning interaction with each other. The key registration functionality is not

[1] See Section 3.1 for more details on these notions.

necessarily trusted by the participating parties. Only guarantees by the key registration functionality are to store the (possibly maliciously formed) public key on the registration request and to retrieve the registered public key on the retrieval request.

The CD model is a newly proposed model in this work. In the CD model, we have domains which models key certification authorities in the real world and each party belongs to one of these domains. Before joining the execution of protocol, each party is required to obtain a certification on their public key from its own domain. In the CD model, each player needs to trust only its own domain but the others.

1.3 Our contributions

In the BPK model, we show a constructions of constant-round simultaneously resettable zero-knowledge ($\mathcal{ZK}$) argument of knowledge with a standard cryptographic assumptions. As a main building block for this result, we show a construction of constant-round simultaneously resettable witness-indistinguishable argument of knowledge.

In the CD model, we show a construction of constant-round concurrently secure multi-party computation (MPC) protocol with the fixed number of domains. On the other hand, we prove an impossibility result that if the number of domains is not fixed, a constant-round concurrently secure multi-party computation protocol cannot exist.

1.4 The organization of the dissertation

The thesis consists of two main part: Part I and Part II. In Part I, we present our results in the BPK model, which is a construction of constant-round simultaneously resettable $\mathcal{ZK}$ argument of knowledge. In Part II, we introduce our new cross-domain model and show a construction of concurrently secure MPC protocol in the CD model. For reader's convenience, each part separately has introduction and preliminary sections.

Part I

Constant-round simultaneously resettable $\mathcal{ZK}$ argument of knowledge in the BPK model

CHAPTER 2

Introduction

2.1 History of resettable security

Interaction and private randomness are the two fundamental ingredients in the foundations of Cryptography since they have been proved to be necessary for achieving zero-knowledge proofs [GMR85] and several other security definitions. Interestingly, in [CGGM00] Canetti et al. showed that when private randomness is limited and re-used in several instances of a proof system, it is still possible to preserve the zero-knowledge requirement. The setting proposed by [CGGM00] is that a malicious verifier resets the prover, therefore forcing the prover to run several protocol executions using the same randomness. This setting applies also to protocols where the player is implemented by a stateless device, therefore can only count on the limited hardwired randomness while it can be adaptively run any polynomial number of times. The resulting security notion against such powerful verifiers is referred to as *resettable zero knowledge* ($\mathsf{r}\mathcal{ZK}$, in short) and it turns out to be very non-trivial since it is proven to be even harder to achieve than concurrent zero knowledge [DNS98, KPR98]. Feasibility results have been achieved in [CGGM00, KP01] in the standard model with polylogarithmic round complexity for the case of $\mathsf{r}\mathcal{ZK}$ and in a constant number of rounds for the case of resettable witness indistinguishability ($\mathsf{r}\mathcal{WI}$, in short). Since then, a line of studies has shown how to achieve resettable zero knowledge in the Bare Public-Key Model (introduced by Canetti, Goldreich, Goldwasser and Micali in [CGGM00]) with improved round complexity and assumptions [MR01, DPV04, APV05, YZ07, SV12].

The reverse of the above question was considered by Barak et al. in [BGGL01] where a malicious prover resets a verifier, called *resettable soundness*. In [BGGL01], it was shown how to obtain resettable soundness along with $\mathcal{ZK}$ in a constant number of rounds. In light of the feasibility of $\mathsf{r}\mathcal{ZK}$, zero knowledge with resettable soundness and the existence of ZAPs [DN00] (i.e., resettably sound $\mathsf{r}\mathcal{WI}$ proof systems), Barak et al. in [BGGL01] proposed the challenging *simultaneous resettability conjecture*, where one would like to prove that a protocol is secure against both a resetting malicious prover and a resetting malicious verifier.

The above machinery turned out to be insufficient to easily solve the simultaneous resettability conjecture and indeed a definitive answer required almost a decade until a breakthrough by Deng, Goyal and Sahai [DGS09] introduced new non-black-box techniques that in synergy with previous machinery, produced a resettably sound $\mathsf{r}\mathcal{ZK}$ argument for np with polynomial round complexity. Very recently, a similar claim has been obtained in [DFG+11] and in [Ari11] by only requiring a constant number of rounds in the BPK model.

The missing piece: arguments of knowledge under simultaneous resettability. Argument systems are often used with a different goal than proving membership of an instance in a language. Indeed, it is commonly required to prove knowledge (possession) of a witness instead of the truthfulness of a statement. As arguments of knowledge are major building blocks in Cryptography (e.g., in identification schemes), it is a legitimate question whether the previous results for arguments of membership extend to arguments of knowledge.

Unfortunately, the answer is negative. Indeed, arguments of knowledge have been achieved so far when only one party can reset the other party. Therefore, we have $\mathsf{r}\mathcal{ZK}$ arguments of knowledge [CGGM00] and resettably sound $\mathcal{ZK}$ arguments of knowledge [BGGL01]. Instead, when reset attacks are possible in both directions, no result is known even when only $\mathsf{r}\mathcal{WI}$ with resettable argument of knowledge is desired. The above results ([DN00, DGS09, DFG+11, Ari11]) achieving simultaneous resettability for arguments of membership can not be simply adjusted in order to provide also a argument of knowledge property under reset attacks. Resettable security for ZAPs come almost for free because of the minimal round complexity (1 or 2 rounds) that however does not give a reasonable hope for any knowledge extraction[1]. For the case of resettably sound

[1]In this work, we only consider standard assumptions and standard security only, therefore we do not take into account controversial non-black-box assumptions such as the Knowledge of Exponent Assumption.

r$\mathcal{ZK}$, all the above results [DGS09, DFG$^+$11, Ari11] critically use an instance-dependent technique along with ZAPs: when the statement is true (i.e., when proving r$\mathcal{ZK}$), the prover/simulator can run ZAPs which allow the use of multiple witnesses. Such use of multiple witnesses gives some flexibility that turns out to be very useful to prove resettable zero knowledge. Instead when the statement is false (i.e., when proving resettable soundness), there is only one valid witness in the ZAPs. Consequently, the adversarial malicious prover is stuck with some fixed messages only to be played during the execution of protocol. Therefore, rewinding capabilities do not help the resetting malicious prover. This is critically used in the proofs of resettable soundness in order to reach a contradiction when a prover proves a false statement. It is easy to see that the above approach falls down when arguments of knowledge are considered. Indeed, when the malicious resetting prover proves a true statement, the same freedom that allows one to prove r$\mathcal{ZK}$/r$\mathcal{WI}$, also gives extra power to the malicious prover and does not seem to give much hope to design an extractor for those protocols. New techniques are therefore needed so that the simultaneous resettability conjecture is still unresolved when we consider knowledge extraction.

2.2 Overview of our results

Our main result is the first construction of a constant-round simultaneously resettable witness-indistinguishable argument of knowledge (simres$\mathcal{WI}$AoK, for short) for any **NP** language. Our protocol is based on the novel use of ZAPs and resettably sound zero-knowledge arguments, which improved over the techniques previously used in [DGS09, DFG$^+$11] that achieved only resettable soundness. Therefore, our result gives the first positive answer to the open problem discussed before.

In addition, we show that our main result also turns out to be a very useful ingredient. Indeed, by combining two executions of our protocol for simres$\mathcal{WI}$AoK, we obtain a constant-round simultaneously resettable zero-knowledge argument of knowledge in the BPK model. This improves the results of [DFG$^+$11, Ari11] which do not enjoy witness extraction with respect to adversarial resetting provers.

As another application of our main protocol, we also consider the challenging question of secure identification under simultaneous resettability and show how to use the above $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ to obtain the first simultaneously resettable identification scheme which follows the knowledge extraction paradigm. We describe it by extending the work of Bellare, Fischlin, Goldwasser and Micali [BFGM01].

Finally, we also show how to obtain a constant-round resettably sound concurrent zero knowledge argument of knowledge in the BPK model by relying only on collision-resistant hash functions (CRHFs, for short) (i.e., we do not require ZAPs, and thus trapdoor permutations). This result was not achieved in literature. Indeed, the previous constructions in [DFG$^+$11, Ari11] use ZAPs to obtain such a result.

Summing up, this work provides better understanding of the complexity of simultaneously resettability. It includes tools that can be very useful when stateless or resettable devices are considered. We finally stress that the issues of reset attacks have been considered in several other contexts [Yil10, DNW08, GS09] and our techniques can also be of a broader impact.

Comparison with the recent and independent work of [GM11]. In a very recent and independent work [GM11], Goyal and Maji used different techniques to achieve simultaneously resettable secure computation. Their results use some additional tools such as simulation-sound zero knowledge and lossy public-key encryption. Considering the argument of knowledge functionality, their results also imply arguments of knowledge under simultaneous resettability, in particular with simulation based security (i.e., zero-knowledge). However, beyond the fact that their independent work is based on different techniques, they obtain simultaneous resettability with polynomial round complexity (while we show constant-round protocols only) and assume the existence of lossy trapdoor encryption (while we stick with CRHFs and standard trapdoor permutations). Hence, our results are incomparable and independent to theirs.

CHAPTER 3

Preliminaries

3.1 Notations

We denote by $n \in \mathbb{N}$ the security parameter and by PPT the property of an algorithm of running in probabilistic polynomial-time. A function ϵ is negligible in n (or just negligible) if for every polynomial $p(\cdot)$ there exists a value $n_0 \in \mathbb{N}$ such that for all $n > n_0$ it holds that $\epsilon(n) < 1/p(n)$. We denote by $x \leftarrow \mathcal{D}$ the sampling of an element x from the distribution $\mathcal{D}$. We also use $x \overset{\$}{\leftarrow} \mathsf{A}$ to indicate that the element x is sampled from set A according to the uniform distribution. Let $\mathcal{P}, \mathcal{V}$ be interactive Turing machines, we denote by $\langle \mathcal{P}(\cdot), \mathcal{V}(\cdot) \rangle(x)$ the random variable representing the local output of $\mathcal{V}$ when interacting with $\mathcal{P}$ where x is the common input and y and z are the private auxiliary information of $\mathcal{P}$ and $\mathcal{V}$ respectively and the randomness of each machine is uniformly and independently chosen.

3.2 Fundamental Definitions

Definition 1 (Indistinguishability [GM84]). *Let X and Y be countable sets. Two ensembles $\{A_x\}_{x \in X}$ and $\{B_x\}_{x \in X}$ are computationally (statistically) indistinguishable over X if for every probabilistic polynomial-time (resp. unbounded) "distinguishing" machine D there exists a negligible function $\epsilon(\cdot)$ so that for every $x \in X, y \in Y$:*

$$|\mathsf{Pr}[D(x, A_x) = 1] - \mathsf{Pr}[D(x, , B_x) = 1]| < \epsilon(|x|).$$

Definition 2 (Interactive Proof System [GMR89]). *A tuple of interactive algorithms $(\mathcal{P}, \mathcal{V})$ is an* *interactive proof system for a language L, if $\mathcal{V}$ is PPT and there exists a negligible function ϵ such that the following conditions hold:*

- *Completeness.* $\forall x \in L$, $\mathsf{Pr}[\langle \mathcal{P}, \mathcal{V}\rangle(x) = 1] > 1 - \epsilon(|x|)$.
- *Soundness.* $\forall x \notin L$, $\forall$ *interactive machine* $\mathcal{P}^*$, $\mathsf{Pr}[\langle \mathcal{P}^*, \mathcal{V}\rangle(x) = 1] < \epsilon(|x|)$.

In case the soundness condition is required to hold only against computationally bounded prover, the pair $(\mathcal{P}, \mathcal{V})$ is called an interactive argument system and $\mathcal{P}$ gets the witness as auxiliary input.

3.3 Resettable Zero Knowledge and Witness Indistinguishability

We briefly recall the definition of resettable zero knowledge and witness indistinguishability introduced in [CGGM00]. Very roughly, a resetting verifier is a PPT adversary that is able to interact with the prover polynomially many times upon (possibly) distinct theorems forcing the prover to execute the protocol using the same randomness several times. Namely, the malicious verifier invokes the prover by two indexes (i, j): the theorem to be proved x_i and the randomness to be used ω_j. The formal definition is provided below and is taken almost verbatim from [CGGM00].

Definition 3 (r$\mathcal{ZK}$ and r$\mathcal{WI}$ [CGGM00])**.** *An interactive proof system $(\mathcal{P}, \mathcal{V})$ for a language L is said to be resettable zero knowledge (r$\mathcal{ZK}$) if for every PPT adversary $\mathcal{V}^*$ there exists a probabilistic polynomial time simulator* M *so that distribution ensembles D_1 and D_2 described below are computationally indistinguishable. Let each distribution be indexed by a sequence of distinct common inputs $\bar{x} = x_1, \ldots, x_{\mathsf{poly}(n)} \in L \cap \{0,1\}^n$ and the corresponding prover's auxiliary-inputs $\bar{y} = y_1, \ldots, y_{\mathsf{poly}(n)}$.*

Distribution D_1 is defined by the following random process which depends on $\mathcal{P}$ and $\mathcal{V}^*$.

Randomly select and fix $t = \mathsf{poly}(n)$ random tapes $\omega_1, \ldots, \omega_t$ for $\mathcal{P}$, resulting in deterministic strategies $\mathcal{P}^{(i,j)} = \mathcal{P}_{x_i, y_i, \omega_j}$ defined by $\mathcal{P}_{x_i, y_i, \omega_j}(\alpha) = \mathcal{P}(x_i, y_i, \omega_j, \alpha)$ for $j \in 1, \ldots, t$, $i \in 1, \ldots, \mathsf{poly}(n)$. Each $\mathcal{P}^{(i,j)}$ is called incarnation of $\mathcal{P}$.

1. *Machine $\mathcal{V}^*$ is allowed to run polynomially-many sessions with $\mathcal{P}^{(i,j)}$. Throughout these sessions, $\mathcal{V}^*$ is required to complete its current interaction (an interaction is complete if is either terminated or aborted) with the current copy of $\mathcal{P}^{(i,j)}$ before starting a new interaction with any $\mathcal{P}^{(i',j')}$, regardless if $(i,j) = (i',j')$ or not. Thus, the activity of $\mathcal{V}^*$ proceeds in rounds. In each round it selects a $\mathcal{P}^{(i,j)}$ and conducts a complete executions with it.*
2. *Once $\mathcal{V}^*$ decides it is done interacting with all $\mathcal{P}^{(i,j)}$, it produces an output based on its view of these interactions. This output is denoted by $\langle \mathcal{P}(\bar{y}), \mathcal{V}^* \rangle(\bar{x})$ and is the output of the process.*

Distribution D_2: The output of $\mathsf{M}(\bar{x})$.

An interactive proof system $(\mathcal{P}, \mathcal{V})$ for a language L is said to be resettable witness indistinguishable ($\mathsf{r}\mathcal{WI}$) if every two distribution ensembles of type D_1 that are indexed by the same sequence of distinct inputs $\bar{x} = x_1, \ldots, x_{\mathsf{poly}(n)} \in L \cap \{0,1\}^n$ but possibly different sequences of prover's auxiliary inputs: $\bar{y}^{(0)}(\bar{x}) = y^0_1, \ldots, y^0_{\mathsf{poly}(n)}$ and $\bar{y}^{(1)}(\bar{x}) = y^1_1, \ldots, y^1_{\mathsf{poly}(n)}$ are computationally indistinguishable. That is, we require that ensembles $\{\langle \mathcal{P}(\bar{y}^{(0)}), \mathcal{V}^* \rangle(\bar{x})\}_{\bar{x}}$ and $\{\langle \mathcal{P}(\bar{y}^{(1)}), \mathcal{V}^* \rangle(\bar{x})\}_{\bar{x}}$ are computationally indistinguishable.

Resettable Soundness. In the following definition we consider only computationally bounded malicious provers.

Definition 4 (resettable soundness rs [BGGL01])**.** *A resetting attack of a cheating prover $\mathcal{P}^*$ on a resettable verifier $\mathcal{V}$ is defined by the following two-step random process, indexed by a security parameter n.*

1. *Uniformly select and fix $t = \mathsf{poly}(n)$ random tapes, denoted $r_1, \ldots, r_t$, for $\mathcal{V}$, resulting in deterministic strategies $\mathcal{V}^{(j)}(x) = \mathcal{V}_{x,r_j}$ defined by $\mathcal{V}_{x,r_j}(\alpha) = \mathcal{V}(x, r_j, \alpha)$, where $x \in \{0,1\}^n$ and $j \in 1, \ldots, t$. Each $\mathcal{V}^{(j)}(x)$ is called an incarnation of $\mathcal{V}$.*
2. *On input 1^n, machine $\mathcal{P}^*$ is allowed to initiate $\mathsf{poly}(n)$-many interactions with $\mathcal{V}$. The activity of $\mathcal{P}^*$ proceeds in rounds. In each round $\mathcal{P}^*$ chooses $x \in \{0,1\}^n$ and $j \in 1, \ldots, t$,*

thus defining $\mathcal{V}^{(j)}(x)$, and conducts a complete session (again, a session is complete if is either terminated or aborted) with it.

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

- *Resettable-completeness: Consider a polynomial-size resetting attack and suppose that in some session after selecting an incarnation $\mathcal{V}^{(j)}(x)$, the attacker follows the strategy $\mathcal{P}$ (for those sessions will also be given the witness). Then, if $x \in L$ then $\mathcal{V}^{(j)}(x)$ rejects with negligible probability.*
- *Resettable-soundness: For every polynomial-size resetting attack, the probability that in some session the corresponding $\mathcal{V}^{(j)}(x)$ has accepted and $x \notin L$ is negligible.*

Arguments of knowledge in the simultaneous resettable setting. Proving the argument of knowledge property of an argument system usually requires to show an expected polynomial-time Turing Machine called Extractor, that having oracle access to the prover, is able to extract the witness of any accepting proof. Such extractor is called black-box extractor. In the setting of resetting verifier, where $\mathcal{V}^*$ is allowed to rewind the honest prover, the verifier has the same power of the extractor, therefore a protocol cannot be resettable witness indistinguishable (or r$\mathcal{ZK}$) and proof (or argument) of knowledge at the same time, unless we provide the extractor with more power then the malicious verifier. Thus as a natural relaxation of the standard notion of proof of knowledge [BG92] in the simultaneous resettable setting consists in allowing non-black box extraction, that is, the extractor gets the code of the (possible malicious) prover. We are now ready to provide the definition of resettably-sound argument of knowledge for NP relations.

Definition 5 (resettably-sound argument of knowledge (adapted from [BGGL01])). *Let $\mathcal{R}_L :\subseteq \{0,1\}^* \times \{0,1\}^*$ be an NP-relation for a language $L = \{x : \exists y\ (x, y) \in \mathcal{R}_L\}$. We say that $(\mathcal{P}, \mathcal{V})$ is a resettably-sound argument of knowledge for $\mathcal{R}_L$ if:*

- *$(\mathcal{P}, \mathcal{V})$ is a resettably-sound argument for L;*

- *there exists an expected polynomial time extractor **E** such that for every PPT resetting machine $\mathcal{P}^*$, there exists a negligible function ϵ for which the following condition holds:*

$$|\mathsf{Pr}[\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = 1] - \mathsf{Pr}[\boldsymbol{E}(desc(\mathcal{P}^*), x) \in \mathcal{R}_L(x)]| < \epsilon(|x|)$$

  where $desc(\mathcal{P}^*)$ denotes the description of $\mathcal{P}^*$'s strategy and $\mathcal{R}_L(x)$ denotes the set of witnesses for x for the NP language L.

Remark 1 (Our Extractor)**.** *The extractor that we provide in our main construction (the simultaneous resettable WI argument of knowledge) follows the typical two-phase extraction paradigm: in the first phase the extractor runs the honest verifier procedure, and if it obtains an accepting transcript, proceeds to the second phase. In the second phase, it rewinds $\mathcal{P}^*$ several times (activating her with the same randomness) and uses fresh randomness trying to reconstruct a new accepting transcript.*

In order to use the extractor in a larger protocol, we define our extractor with three inputs. The first two inputs are the random tape used for the activation of the prover and the random tape for the execution of the first phase; the length of these tapes is polynomially bounded by the size of the prover. The last input is a random tape used to perform the second phase (the length of this last tape can not be established in advance since the number of attempts required to obtain a distinct accepting transcript is not dependent of the size of $\mathcal{P}^*$).

3.4 Bare Public-Key (BPK) Model in Resettable Settings

The *Bare Public key* model was first introduced in [CGGM00]. Informally, in the BPK model, a file (called a public file) of polynomially many public keys is selected and published by a (possibly malicious) verifier before any protocol execution begins. Such public keys in the public file represent the identities of verifiers in the interactions with provers. More formally, in the BPK model, any interactive argument system consists of two phases, key generation (or registration) phase and main protocol execution phase described as follows.

Interactive argument systems in the BPK model.

Key Generation Phase. A verifier $\mathcal{V}$, running on security parameter n and random tape R, generates a pair of public and secret keys (pk, sk). Then, $\mathcal{V}$ stores public key pk in the public file F and keeps the secret key sk as its secret trapdoor information. The size of the public file F is polynomial in the security parameter.

Main protocol execution phase. Once the key generation phase is over, F can not be changed anymore, and therefore players can start the actual interactions by exploiting the BPK model to obtain more efficient protocols.

- On inputs security parameter n, random tape $r_{\mathcal{V}}$, theorem x, public key pk, and secret key sk, $\mathcal{V}$ interacts with $\mathcal{P}$.
- On inputs security parameter n and random tape $r_{\mathcal{P}}$, theorem x, witness w, and public key pk, $\mathcal{P}$ interacts with $\mathcal{V}$ to prove the validity of x.

Definition of malicious resetting provers in the BPK model: Let $\mathcal{P}^*$ be a PPT malicious prover which takes security parameter n and runs in at most time $s(n)$ for some polynomial s. Without loss of generality, as a common input, $\overline{x}$ is a vector of theorems which contains $s(n)$ theorems x_i for $i \in [s(n)]$ such that $|x_i| = n$. Then, the $s(n)$-resetting malicious prover $\mathcal{P}^*$ in the BPK model is defined as follows:

- $\mathcal{P}^*$ chooses a vector of random strings $\vec{\mathcal{R}}_{\mathcal{V}}$ for $\mathcal{V}$, which contains $s(n)$ random strings r_j for $j \in [s(n)]$.
- $\mathcal{P}^*$ runs $\mathcal{V}$ to obtain public file F which contains at most $s(n)$ public keys pk^k for $k \in [s(n)]$. The corresponding secret keys sk^k for $k \in [s(n)]$ are privately stored by $\mathcal{V}$.
- $\mathcal{P}^*$ initiates at most $s(n)$ sessions indexed by a tuple (i, j, k) for $i, j, k \in [s(n)]$. That is, $\mathcal{P}^*$ invokes sessions at most $s(n)$ times by choosing a tuple (i, j, k) and in each session (i, j, k), $\mathcal{P}^*$ attempts to prove theorem x_i to the incarnation of $\mathcal{V}_{jk}$ which uses random tape $r_j \in \vec{\mathcal{R}}_{\mathcal{V}}$, $pk^k \in F$, and sk^k as its secret key for the session. We call index k the identity of $\mathcal{V}$ for the session.

Definition of malicious resetting verifiers in the BPK model: Let $\mathcal{V}^*$ be a PPT malicious verifier which takes security parameter n and runs in at most time $s(n)$ for some polynomial s. $\mathcal{V}^*$ takes, as a common input $\overline{x}$ which is a vector of theorems containing $s(n)$ theorems x_i for $i \in [s(n)]$ such that $|x_i| = n$. Let $\vec{\mathcal{R}}_{\mathcal{P}}$ be a vector of $s(n)$ random strings r_j for $j \in [s(n)]$ for $\mathcal{P}$. An $s(n)$-resetting malicious verifier $\mathcal{V}^*$ in BPK model works as follows. First, $\mathcal{V}^*$ generated the public file F. Then $\mathcal{V}^*$ initiates at most $s(n)$ sessions indexed by a tuple (i, j, k) by choosing (i, j, k) for $i, j, k \in [s(n)]$. In particular, $\mathcal{V}^*$ invokes sessions at most $s(n)$ times and in each session (i, j, k), $\mathcal{V}^*$ interacts with the incarnation of $\mathcal{P}_{ij}$ which by using random tape $r_j \in \vec{\mathcal{R}}_{\mathcal{P}}$ is supposed to prove theorem x_i to $\mathcal{V}^*$ using $pk^k \in F$ for some $k \in [s(n)]$. $\mathcal{V}^*$ outputs the view which contains the entire interaction transcript.

3.5 Overview of Building Blocks

Blum's protocol. In Fig. 3.1 we describe the 3-round $\mathcal{WI}\mathsf{PoK}$ protocol for the NP-complete language Graph Hamiltonicity (HC). We use this proof system as sub-protocol in our construction and we refer to it as BL protocol. Notice that by getting the answer for both $b = 0$ and $b = 1$ allows the extraction of the cycle. The reason is the following. For $b = 0$ one gets the permutation of the original graph G. Then for $b = 1$ one gets the of the Hamiltonian cycle of the permuted graph.

Resettably sound Statistical Zero Knowledge. In our construction we will use the constant-round public-coin zero-knowledge argument of Barak [Bar01, BGGL01] as modified in [PR05b]. In [PR05b] the original Barak's protocol is tweaked in order to obtain statistical zero knowledge (instead of computational ZK) for NP languages along with argument of knowledge (instead of weak proof of knowledge). Finally by applying the transformation of [BGGL01] to the construction of [PR05b] one can obtain a resettably-sound statistical zero knowledge argument of knowledge.

The BL Protocol [Blu86] for HC.

Inputs: $\mathcal{V}$, $\mathcal{P}$ have as input a graph G. $\mathcal{P}$ has as auxiliary input a witness $y \in \mathcal{R}_{\mathsf{HC}}(G)$. Let n be the number of vertexes of G. G is represented by a $n \times n$ adjacency matrix M where $\mathsf{M}[i][j] = 1$ if there exists an edge between vertexes i and j in G, and $\mathsf{M}[i][j] = 0$ otherwise. Each of the following step is repeated **in parallel** n **times** using independent randomness.

$\mathsf{BL1}$ ($\mathcal{P} \to \mathcal{V}$): $\mathcal{P}$ picks a random permutation π of the graph G and commits bit-by-bit to the corresponding adjacency matrix using a statistically binding commitment scheme.

$\mathsf{BL2}$ ($\mathcal{V} \to \mathcal{P}$): $\mathcal{V}$ responds with a randomly chosen bit b.

$\mathsf{BL3}$ ($\mathcal{P} \to \mathcal{V}$):

- if $b = 0$, $\mathcal{P}$ opens all the commitments, and send the permutation π showing that the matrix committed in step $\mathsf{BL1}$ is actually the instance G.
- if $b = 1$, $\mathcal{P}$ opens only the commitments that form an Hamiltonian cycle in the permuted matrix committed in step $\mathsf{BL1}$.

$\mathcal{V}$ accepts if and only if all n executions are accepting.

Figure 3.1: Blum's witness indistinguishable proof of knowledge for Hamiltonicity.

Commitment schemes. A commitment scheme is a two-stage (commitment phase, decommitment phase), two-party (sender, receiver) protocol in which the sender binds itself to a value in the commitment phase, keeping secret the value to the receiver (this property is called hiding). Nevertheless in the decommitment phase, the sender reveals the secret value and the receiver is guaranteed that it corresponds to the value committed in the previous stage (this property is called binding). A commitment scheme is statistically binding if the binding property holds even against unbounded malicious sender. Non-interactive statistically binding commitment can be constructed from any one-way permutation [GL89]. In statistically hiding commitment schemes instead the hiding is preserved even against unbounded malicious receiver. Constant-round statistically hiding commitment schemes exist assuming the existence of families of collision-resistant hash functions [HM96a].

Pseudorandom functions. A keyed function f is a two inputs function $f : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^\ell$, where the first input is the *key* and the second is the input, mapping a κ-bit input string to an ℓ-bit string by using a fixed n-bit key. We say that f is efficient if there is a deterministic polynomial time algorithm that computes $f(k,x) \overset{def}{=} f_k(x)$ given k and x in input.

Definition 6 (Pseudorandom Function)**.** *Let $f : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^\ell$ be an efficient keyed function. We say that f is a pseudo-random function (PRF, in short) if for all PPT distinguisher D, there exists a negligible function ϵ such that: $|\mathsf{Pr}(k \overset{\$}{\leftarrow} \{0,1\}^\kappa; D^{f_k(\cdot)}(1^\kappa)) - \mathsf{Pr}(f \overset{\$}{\leftarrow} \{F\}_{\kappa,\ell}; D^{f(\cdot)}(1^\kappa))| \leq \epsilon(\kappa)$ where $F_{\kappa,\ell}$ is the uniform distribution over all functions mapping κ-bit long strings to ℓ-bit long strings.*

ZAPs [DN00, GOS06]. ZAPs are two-round resettably-sound witness indistinguishable proof systems [DN00]. As noted in [DN00] by requiring that the randomness of the prover is generated by applying a pseudo-random function to the first message sent by the verifier, ZAPs are also resettable witness indistinguishable. We refer to such a simultaneously resettable ZAP as $\mathsf{r}ZAP$.

CHAPTER 4

A constant-round resettably sound resettably $\mathcal{WI}$ arguments of knowledge

In order to obtain a constant-round simultaneously resettable $\mathcal{ZK}$ AoK protocol in the BPK model, we first construct a resettably sound resettably witness-indistinguishable Aok protocol ($\mathsf{simres}\mathcal{WI}\mathsf{AoK}$) in the plain model where the protocol enjoys the constant-round complexity. Protocol $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ serves as a main building block of our constant-round simultaneously resettable $\mathcal{ZK}$ AoK protocol in the BPK model. Remark that $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ is the first constant-round protocol in the plain model which is resettably witness-indistinguishable and resettable argument of knowledge.

4.1 The overview of our new techniques

Our goal is to obtain a construction that is resettably-sound resettable $\mathcal{WI}$ *and* argument of knowledge in a constant number of rounds. The only known constant-round simultaneously-resettable $\mathcal{WI}$ protocol is $\mathsf{r}ZAP$ which is not an argument of knowledge and can not be trivially transformed in a argument of knowledge even without considering resettability.

A typical paradigm: determining message and consistency proof. Typically ([CGGM00, BGGL01, DGS09]), protocols dealing with a resetting adversary follow this paradigm: the resetting party is required to provide a special message (called *determining message*) that determines her own action for the rest of the protocol. Namely, for each protocol message the resetting party is required to prove that such message is consistent with the determining message (we call this proof *consistency proof*). Moreover, the actual randomness used by the honest party in the protocol depends on the determining message (typically the honest party applies a pseudorandom function

(PRF) on it). The combination of the randomness depending on the determining message and the consistency proof given by the resetting party suppresses the resetting power of the adversary. Indeed, since the resetting party, upon each protocol message has to prove consistency with the determining message, even after a reset she cannot change a message previously played without first having changed the determining message (unless she is able to fake the consistency proof). However, if she changes the determining message, then the honest party plays the protocol with (computationally) fresh randomness (unless the adversary violates the pseudo-randomness of the PRF). We will follow this paradigm to construct our simultaneously resettable witness indistinguishable argument of knowledge. Recall that as specified above, we can not start from r$ZAP$s that are already simultaneously resettable and try to transform them into an argument of knowledge. Our starting point is Blum's proof of knowledge [Blu86] shown in Fig. 3.1 which is secure only against concurrent adversaries. In the following discussion we show incrementally how to transform such protocol to enjoy resettable witness indistinguishability and resettable soundness (this transformation is already known in literature) to finally present our novel technique to obtain also *resettable* argument of knowledge.

Resettable $\mathcal{WI}$ and stand-alone arguments of knowledge [BGGL01]. Consider the case in which only the verifier can reset the prover. Following the above paradigm, it is easy to construct a resettable $\mathcal{WI}$ system starting from Blum's protocol. In Blum's protocol the only message from $\mathcal{V}$ to $\mathcal{P}$ is the challenge. The modified resettable version requires that $\mathcal{V}$ sends a statistically binding commitment of the challenge as determining message. The only other protocol message of $\mathcal{V}$ is the opening of the commitment which, due to the binding property, is itself a proof that the message is consistent with the determining message. Note that such modified protocol is no longer an argument of knowledge since the extractor has the same power of the malicious verifier. In order to allow only the extractor to cheat, the next step is to avoid the opening as a proof of consistency. Instead of the actual opening of the commitment, $\mathcal{V}$ is required to send the challenge along with a res-sound (non-black-box) $\mathcal{ZK}$ argument ([Bar01]) as a proof of consistency with the commitment. The (non-black box) extractor can send an arbitrary challenge and prove consistency with the determining message by using the (stand-alone) non-black-box simulator (recall that only

$\mathcal{V}$ might reset here).

The resulting protocol is resettable $\mathcal{WI}$ and (stand-alone) argument of knowledge (r$\mathcal{WI}$AoK) and it is known from [BGGL01]. We use a modified version of such protocol. We require that the commitment sent by the verifier is statistically hiding (instead of statistically binding), and we use the statistical zero-knowledge argument of knowledge of [PR05b] instead of the protocol of [Bar01] that is only computational $\mathcal{ZK}$ and is not an argument of knowledge. In the following we refer to such protocol as BGGL.

Achieving Resettable Soundness and *Resettable* Argument of Knowledge: existent solutions do not work. We now deal with the case in which also the prover can reset. By the BGGL compiler [BGGL01], we know that any public-coin $\mathcal{WI}$ argument system can be upgraded to resettable soundness by simply requiring the honest verifier to apply a PRF on the first message received from the prover. However, since our aim is to obtain simultaneous resettability, we need to start from the r$\mathcal{WI}$AoK protocol shown before, which is not public coin. Thus, following the paradigm and the technique of [DGS09], we require that as first message, $\mathcal{P}$ sends the commitment of the randomness that will be used in the protocol: this is the determining message. Then upon each protocol message $\mathcal{P}$ proves that the message is honestly computed using the randomness committed in the determining message: this is the consistency proof. Since we are now in the setting in which both parties can reset each other the consistency proof must be provided with a simultaneous resettable tool. For this purpose we use r$ZAP$s that are constant-round simultaneously resettable $\mathcal{WI}$ proofs. We denote the theorem to be proved with rZAP as "consistency theorem", since $\mathcal{P}$ proves that a message is honestly computed and consistent with the randomness committed in the determining message.

The technical problem using r$ZAP$s is that they are only $\mathcal{WI}$, thus the theorem being proved is required to have more than one witness (note that using the simultaneous $\mathcal{ZK}$ protocol of [DGS09] is not an option for us since we aim to a constant-round construction). Recall that we want to use rZAP to provide the proof of consistency with the determining message. If the determining message is a statistically-binding commitment of the randomness, then there exists a unique opening, which implies the existence of only one witness. On the other hand, if we use a statistically hiding commitment, then any opening is a legitimate witness, and the theorem is always true, which leads to another technical problem when one wants to reduce the resettable soundness of the resulting construction to the soundness of rZAP. The solution to overcome this problem is to change the theorem to be proved with rZAP so that it admits more than one witness.

In [DGS09] the consistency theorem is augmented with the theorem "$x \in L$" that we call "*trapdoor* theorem". We call it trapdoor to stress out that it is an escape for the prover that can avoid the consistency proof essentially having freedom to change messages among resets. Hence in [DGS09, DFG^{+}11], along with each protocol message, $\mathcal{P}$ is required to prove that either the protocol message is computed honestly with the randomness committed in the determining message, the "consistency theorem", or $x \in L$, the "trapdoor theorem".

This solution can be seen as an instance-dependent technique. Indeed, it is easy to see that a malicious prover has the freedom of not proving the consistency of its messages and therefore to exploit the resetting power only when $x \in L$. Instead, when proving soundness, since $x \notin L$, the trapdoor theorem is false, hence due to soundness of r$ZAP$s, the malicious prover is forced to prove the consistency theorem and thus to honestly follow the protocol.

Unfortunately, such an instance-dependent solution serves well to prove resettable soundness but fails completely when one would like to prove witness extraction (i.e., the argument of knowledge property). The reason is that, when proving witness extraction, we have to construct an extractor that works against any malicious prover, even one who uses the witness of the trapdoor theorem instead of proving consistency of the protocol messages.

More specifically, this possible behavior harms the extractor in two ways: 1) upon seeing the challenge of the verifier/extractor, $\mathcal{P}$ resets it and changes the first message of Blum's protocol according to the challenge; 2) it can act as a resetting verifier in the non black-box $\mathcal{ZK}$ protocol, therefore preventing the extractor to use the stand-alone non black-box simulator. Even if this is not harmful for the soundness (a malicious prover can perform this attack only when $x \in L$), this attack kills the existence of the extractor; thus, the final construction is only resettable $\mathcal{WI}$ and resettable sound. Therefore, the instance-dependent technique of [DGS09] inherently prevents the existence of any extractor. New ideas are required to solve the problem.

Achieving *Resettable* Argument of Knowledge: the new technique. We propose a new "trapdoor" theorem that forces the prover to honestly follow the protocol regardless whether $x \in L$ or not.

The idea is the following. We let the protocol consist of the parallel execution of two $\mathsf{B}GGL$ protocols that we call left sub-protocol denoted by π_0 and right sub-protocol execution denoted by π_1. Before starting each sub-protocol $\mathcal{P}$ commits to the randomness that will be used in the sub-protocol. Moreover, $\mathcal{P}$ must commit to a single bit. The determining message therefore consists of the statistically-binding commitment of both randomnesses and of a single bit.

The trapdoor theorem that $\mathcal{P}$ proves in the sub-protocol π_d is the following: "d is the bit committed in the determining message". Clearly, due to statistically-binding property of the commitment, the trapdoor theorem is true only in one sub-protocol. Therefore, in each sub-protocol π_d, along with each message of $\mathsf{B}GGL$ protocol, $\mathcal{P}$ provides as proof of consistency a $\mathsf{r}ZAP$ for the following compound theorem: either the message is honest and consistent with the determining message, or d is the bit committed in the determining message. Finally, the verifier will accept the proof if and only if *both* sub-protocol executions π_0, π_1 are accepting.

It is easy to see that now, regardless of whether $x \in L$ or not, any malicious prover cannot escape from proving the consistency theorem in at least one of the subprotocols. Indeed, let b be the bit committed in the determining message, in sub-protocol π_b, a malicious $\mathcal{P}$ is not forced to be honest and can then use the resetting power to prove any false theorem (indeed among resets $\mathcal{P}$ can change the protocol messages without without changing the determining message). In sub-protocol $\pi_{\bar{b}}$, the trapdoor theorem is false, thus the only way to provide an accepting $\mathsf{r}ZAP$ is to prove the "consistency" theorem (where $\mathcal{P}$ proves that messages are honestly computed using the randomness committed in the determining message). Therefore, in sub-protocol $\pi_{\bar{b}}$, the extractor is guaranteed that 1) for sessions starting with the same determining message, the first round of Blum's protocol does not change, so that playing with two distinct challenges yields the extraction of the witness; 2) it can run the stand-alone non black-box $\mathcal{ZK}$ simulator without being detected. Note that in both sub-protocols, the resettable $\mathcal{WI}$ property is still preserved. Finally we have the following: sub-protocol $\pi_{\bar{b}}$ is resettably-sound and resettable argument of knowledge, while sub-protocol π_b is not sound. Both sub-protocols are r$\mathcal{WI}$.

Summing up, the final protocol consists of a round in which the prover sends its determining message, namely the commitments of the randomnesses used in each sub-protocol, and the commitment of a bit. Then prover and verifier run two parallel executions of BGGL augmented with the additional zaps from $\mathcal{P}$ to $\mathcal{V}$ where $\mathcal{P}$ proves consistency with the determining message. $\mathcal{P}$ runs each sub-protocol with the randomness committed in the first round, $\mathcal{V}$ runs each sub-protocol with the randomness generated as a function of the determining message of the prover.

4.2 The formal Construction of simres$\mathcal{WI}$AoK

In this section, we provide the formal description of our main protocol. We describe how to build a simultaneously resettable $\mathcal{WI}$ AoK (simres$\mathcal{WI}$AoK) starting from Blum's protocol (BL protocol), shown in Fig. 3.1. We denote by SHCom, a two-round statistically hiding commitment scheme. We denote by SBCom the commitment procedure of a non-interactive statistically binding commitment scheme. We denote by $c \leftarrow \mathsf{SBCom}(v, s)$ (resp. SHCom) the output of the commitment of the value v computed with randomness s. We use the resettably-sound statistical (non black-box) $\mathcal{ZK}$ AoK of [PR05b] that we denote by resSZK. In our construction, we require that $\mathcal{P}$, at each round of the protocol (except the last that is the opening of commitments as required by BL protocol), provides a proof that either the messages are honestly computed according to the randomness committed in the first round, or the "trapdoor" condition is satisfied. Formally, $\mathcal{P}$ provides r$ZAP$s for the following **NP** languages during the execution of the main protocol (except the language Λ_{SHCom} that is proved only by $\mathcal{V}$ using resSZK protocol).

Λ_{BL1}**:** correctness and consistency of the first round of Blum's protocol (BL1). A tuple $(x, m, \mathsf{c}_{r_b}, \mathsf{c}_b) \in \Lambda_{\mathsf{BL1}}$ if: there exist (r_b, s_b) such that $\mathsf{c}_{r_b} = \mathsf{SBCom}(r_b, s_b)$ and m is honestly computed according to BL1 for the graph x using randomness $f_{r_b}(\mathsf{c}_b)$.

Λ_{V}**:** correctness and consistency of verifier's messages. A tuple $(m_P, m_V, \mathsf{c}_{r_b}, \mathsf{c}_b) \in \Lambda_{\mathsf{V}}$ if: there exist (r_b, s_b) such that $\mathsf{c}_{r_b} = \mathsf{SBCom}(r_b, s_b)$ and m_V is honestly computed according to the verifier's procedure of the protocol resSZK having in input the prover's message m_P using randomness $f_{r_b}(\mathsf{c}_b)$.

Λ_{trap}: **t**rapdoor theorem (true only for sub-protocol b). The pair $(\mathsf{c}_s, b) \in \Lambda_{\mathsf{trap}}$ if there exists s such that $\mathsf{c}_s = \mathsf{SBCom}(b, s)$.

Λ_{SHCom}: validity of the opening (proved by $\mathcal{V}$). The pair $(\mathsf{c}_s, m) \in \Lambda_{\mathsf{SHCom}}$ if there exists s such that $\mathsf{c}_s = \mathsf{SHCom}(m, s)$. Note that for a statistically-hiding commitment scheme, any pair (c_s, m) is actually in Λ_{SHCom}. Nevertheless, $\mathcal{V}$ proves this theorem using the argument of knowledge resSZK.

The $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ protocol is described in Figure 4.2, while a graphic description is provided in Figure 4.1. It basically consists of two phases. In the first phase, $\mathcal{P}$ and $\mathcal{V}$ generate the random tapes that they will use to run the sub-protocols. $\mathcal{P}$ sends $\mathcal{V}$ the commitments of two random strings $(\mathsf{c}_{r_0}, \mathsf{c}_{r_1})$ and the commitment of a random bit c_s. This message is the *determining message* on which $\mathcal{V}$ applies a PRF to generate a pseudo-random tape (to be used to execute the protocol). The second phase consists of a parallel execution of π_0 and π_1 formally described in Figure 4.3. $\mathcal{P}$ runs each sub-protocol on theorem x, randomness picked in the first stage, and the witnesses for computing the $\mathsf{r}ZAP$s as inputs (i.e., the opening of the commitments of the determining message). $\mathcal{V}$ runs each sub-protocol using the psuedo-randomness computed upon the determining message received from $\mathcal{P}$. Each sub-protocol is resettable $\mathcal{WI}$, while only one of the two sub-protocols is a resettably-sound AoK. Since $\mathcal{V}$ accepts the proof only if *both* executions are accepting, the final protocol is also a resettably-sound argument of knowledge.

The sub-protocol π_d is described in Figure 4.3. We omit the first round of the $\mathsf{r}ZAP$ and the first round of the statistically hiding commitment scheme SHCom. $\mathsf{r}ZAP$s are computed with independent randomness. We stress out that the determining message for $\mathcal{V}$ is the first prover's message: $\mathsf{dm} = (\mathsf{c}_{r_0}, \mathsf{c}_{r_1}, \mathsf{c}_s)$. The determining message for $\mathcal{P}$ is the first verifier's message: $(\mathsf{c}_0, \mathsf{c}_1)$.

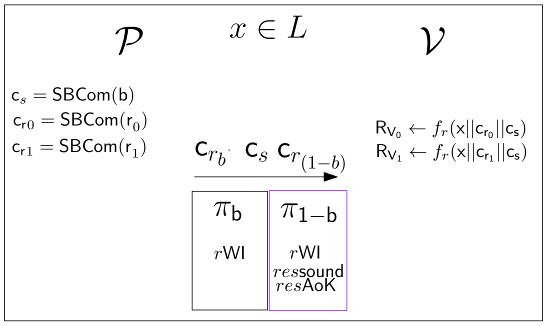

$\mathcal{P}$
$x \in L$
$\mathcal{V}$
$\mathsf{c}_s = \mathsf{SBCom}(\mathsf{b})$
$\mathsf{c}_{r0} = \mathsf{SBCom}(\mathsf{r}_0)$
$\mathsf{c}_{r1} = \mathsf{SBCom}(\mathsf{r}_1)$
$\mathsf{c}_{r_b} \; \mathsf{c}_s \; \mathsf{c}_{r_{(1-b)}}$
$\mathsf{R}_{\mathsf{V}_0} \leftarrow f_r(\mathsf{x}||\mathsf{c}_{\mathsf{r}_0}||\mathsf{c}_\mathsf{s})$
$\mathsf{R}_{\mathsf{V}_1} \leftarrow f_r(\mathsf{x}||\mathsf{c}_{\mathsf{r}_1}||\mathsf{c}_\mathsf{s})$
π_b
$r\mathsf{WI}$
$\pi_{1-\mathsf{b}}$
$r\mathsf{WI}$
$res\mathsf{sound}$
$res\mathsf{AoK}$

Figure 4.1: simres$\mathcal{WI}$AoK.

Protocol simres$\mathcal{WI}$AoK

Inputs: common input $x \in \mathsf{HC}$.
$\mathcal{P}$'s input: witness y, randomness ω. $\mathcal{V}$'s input: randomness r.

1. $\mathcal{P}$: $b \overset{\$}{\leftarrow} \{0,1\}$; $r_0, r_1, s_0, s_1 \overset{\$}{\leftarrow} \{0,1\}^n$.
   Send $\mathsf{c}_{r_0} \leftarrow \mathsf{SBCom}(r_0, s_0)$, $\mathsf{c}_{r_1} \leftarrow \mathsf{SBCom}(r_1, s_1)$, $\mathsf{c}_s \leftarrow \mathsf{SBCom}(b, s)$.
   Run in parallel $\pi_0^{\mathcal{P}}(x, y, r_0, s_0, b, s)$; $\pi_1^{\mathcal{P}}(x, y, r_1, s_1, b, s)$.
2. $\mathcal{V}$: upon receiving $\mathsf{dm} = (\mathsf{c}_{r_0}, \mathsf{c}_{r_1}, \mathsf{c}_s)$ from $\mathcal{P}$.
   $\mathsf{R}_{\mathsf{V}0} \leftarrow f_r(x||\mathsf{c}_{r_0}||\mathsf{c}_s)$; $\mathsf{R}_{\mathsf{V}1} \leftarrow f_r(x||\mathsf{c}_{r_1}||\mathsf{c}_s)$; Run in parallel $\pi_0^{\mathcal{V}}(x, \mathsf{R}_{\mathsf{V}0})$; $\pi_1^{\mathcal{V}}(x, \mathsf{R}_{\mathsf{V}1})$.

Figure 4.2: Simultaneously Resettable $\mathcal{WI}$AoK.

4.3 The proof of security of simres$\mathcal{WI}$AoK

In this section, we provide an high-level proof sketch of the simultaneous resettable witness indistinguishability property and the resettable argument of knowledge property of the protocol depicted in Fig. 4.2.

4.3.1 Resettable Soundness and Argument of Knowledge

In this section we show that protocol shown in Fig. 4.2 is a resettably-sound argument of knowledge.

Resettable-soundness of simres$\mathcal{WI}$AoK**.** We start proving that the scheme is resettable-sound. Recall that the protocol starts with $\mathcal{P}^*$ sending the determining message dm to $\mathcal{V}$. dm consists of three commitments: commitment of two seeds and commitment of a bit (let us call the bit committed b). Consider the following observations.

Sub-protocol: $\pi_d = \langle \pi_d^{\mathcal{P}}(x, y, r_d, s_d, b, s), \pi_d^{\mathcal{V}}(x, \mathsf{R}_{\mathsf{V}d}) \rangle$.
Inputs: common input: x ($\in$ HC). $\mathcal{P}$'s input: witness y for $\mathcal{R}_{\mathsf{HC}}$; (partial) witness (r_d, s_d) to prove rZAP's consistency theorem. $\mathcal{V}$'s input: randomness $\mathsf{R}_{\mathsf{V}d}$. Protocols BL (Fig. 3.1) and resSZK ([PR05b]) are used as sub-protocols.

- $\mathcal{V}$: Pick challenge for BL protocol: $ch_d \overset{\$}{\leftarrow} \{0,1\}^n$. Send $\mathsf{c}_d \leftarrow \mathsf{SHCom}(ch_d)$ to $\mathcal{P}$.
- $\mathcal{P}$: upon receiving c_d (this is the determining message for $\mathcal{P}$):
  1. generates randomness $\mathsf{R}_{\mathsf{P}d} \leftarrow f_{r_d}(x||\mathsf{c}_d)$.
  2. computes the step $\mathsf{BL1}$ for the instance x using randomness $\mathsf{R}_{\mathsf{P}d}$. Let us denote the output as $\mathsf{mBL1}^d$.
  3. Send $\mathsf{mBL1}^d$ to $\mathcal{V}$ along with the rZAP for theorem: $((x, \mathsf{mBL1}^d, \mathsf{c}_{r_d}, \mathsf{c}_d) \in \Lambda_{\mathsf{BL1}} \vee (\mathsf{c}_s, d) \in \Lambda_{\mathsf{trap}})$.
- $\mathcal{V}$: if rZAP is accepting send ch_d to $\mathcal{P}$.
  Prove theorem $(\mathsf{c}_d, ch_d) \in \Lambda_{\mathsf{SHCom}}$ using resSZK protocol. Let $m^d_{\mathsf{P}_{\mathsf{rszk}}}$ be the prover's message of sub-protocol resSZK (sent by $\mathcal{V}$ to $\mathcal{P}$) and $m^d_{\mathsf{V}_{\mathsf{rszk}}}$ be the verifier's messages of resSZK (sent by $\mathcal{P}$ to $\mathcal{V}$):
  1. ($\mathcal{P} \rightarrow \mathcal{V}$) at each round of the protocol resSZK, upon receiving $m^d_{\mathsf{P}_{\mathsf{rszk}}}$ from $\mathcal{V}$, $\mathcal{P}$ computes $m^d_{\mathsf{V}_{\mathsf{rszk}}}$ using randomness $\mathsf{R}_{\mathsf{P}d}$ and sends $m^d_{\mathsf{V}_{\mathsf{rszk}}}$ to $\mathcal{V}$ along with an rZAP for the theorem $((m^d_{\mathsf{P}_{\mathsf{rszk}}}, m^d_{\mathsf{V}_{\mathsf{rszk}}}, \mathsf{c}_{r_d}, \mathsf{c}_d) \in \Lambda_{\mathsf{V}} \vee (\mathsf{c}_s, d) \in \Lambda_{\mathsf{trap}})$.
  2. ($\mathcal{V} \rightarrow \mathcal{P}$) at each round of the protocol resSZK upon receiving $m^d_{\mathsf{V}_{\mathsf{rszk}}}$ from $\mathcal{P}$, if rZAP is accepting $\mathcal{V}$ computes the next resSZK's prover message and sends it to $\mathcal{P}$. Otherwise aborts.
- $\mathcal{P}$: upon successfully completing the resSZK protocol compute step $\mathsf{BL3}$ and send the message $\mathsf{mBL3}^d$ to $\mathcal{V}$.
- If $\mathsf{mBL3}^d$ is the correct third message of BL protocol $\mathcal{V}$ outputs $\mathtt{accept}$, else outputs $\mathtt{abort}$.

Figure 4.3: Sub-protocol $\pi_d = (\pi_d^{\mathcal{P}}(\cdot), \pi_d^{\mathcal{V}}(\cdot))$.

1. The randomness used by $\mathcal{V}$ depends on the determining message. In a resetting attack the malicious prover $\mathcal{P}^*$ activates $\mathcal{V}$ selecting theorem and randomness (x, j), forcing $\mathcal{V}$ to run with the same randomness r_j among several executions. However, the randomness that is actually used by $\mathcal{V}$ at each session is determined by the output of the PRF on seed r_j and input (x, dm) where dm is the determining message. Thus, even if activated with the same random tape r_j, when receiving a new determining message $\mathcal{V}$ plays with fresh random tape. Note that here we are using the assumption that the output of the PRF looks random to the PPT $\mathcal{P}^*$, thus soundness holds against computationally bounded adversary only. For lack of

space we omit the reduction, that however is pretty standard.

2. In sub-protocol π_b the resetting power of $\mathcal{P}^*$ is effective. The reason is that in π_b, $\mathcal{P}^*$ can honestly prove the trapdoor theorem of the $\mathsf{r}ZAP$ therefore she is not forced to use the randomness committed in the determining message among several resetting attacks. More specifically $\mathcal{P}^*$ can mount the following attack. She initiates a session labelled by (x, j, dm), and in the sub-protocol π_b, upon receiving the challenge ch_b from $\mathcal{V}$, $\mathcal{P}^*$ resets $\mathcal{V}$ (keeping the *same* determining message) up to the second round (after $\mathcal{V}$ has sent the commitment of the challenge), and changes the message $\mathsf{mBL1}^b$ according to the challenge ch_b previously seen (indeed, since the determining message is the same $\mathcal{V}$ will use the same challenge in the sub-protocol π_b). Thus, in this sub-protocol $\mathcal{P}^*$ can basically run the algorithm of the simulator of the zero-knowledge version (i.e., when there is a mechanism for the simulator to know in advance the challenge) of BL's protocol and prove any theorem. Therefore in sub-protocol π_b soundness does not hold.

3. In sub-protocol $\pi_{\bar{b}}$ $\mathcal{P}^*$ trapdoor theorem is false thus resetting attacks are ineffective. This means that in order to provide an accepting transcript, $\mathcal{P}^*$ must provide a $\mathsf{r}ZAP$ for the "honest" theorem, i.e., for each protocol message she has to prove that it is honestly computed according to the randomness committed in the determining message. Due to the statistically binding property of SBCom (there exists only one opening for the commitment c_s, $\mathsf{c}_{r_{\bar{b}}}$) and to the soundness of $\mathsf{r}ZAP$ (any unbounded $\mathcal{P}^*$ cannot prove a false theorem) if $\mathcal{V}$ accepts the $\mathsf{r}ZAP$ for messages of protocol $\pi_{\bar{b}}$ it must be the case that $\mathcal{P}^*$ is honest and consistent with the randomness committed in the determining message. Furthermore, if $\mathcal{P}^*$ resets $\mathcal{V}$ forcing to use the same randomness and changes the determining message, due to the fact that $\mathcal{V}$ use of the PRF applied to the determining message, it will use a fresh new randomness therefore making the reset useless. Thus the resetting power of $\mathcal{P}^*$ in the execution of $\pi_{\bar{b}}$ is defeated. Therefore, $\pi_{\bar{b}}$ is resettable sound.

In the light of the observations above consider the follows. Assume that there exists a PPT malicious prover $\mathcal{P}^*$ and a pair (x, j) such that $\mathcal{V}$ accepts x with non-negligible probability, but $x \notin \mathsf{HC}$

. Specifically, this means that $\mathcal{P}^*$ is able to generate an accepting transcript for x when invoking the verifier on randomness r_j. By observation 1, we know that such a transcript is indexed also by a determining message dm. Thus the accepting transcript can be labelled by the triple (x, j, dm). By observation 2, we know that for the same determining message dm there are polynomially many distinct sub-transcripts for sub-protocol π_b ($\mathcal{P}^*$ can reset $\mathcal{V}$ many times and change the protocol messages), and that all these (partial) transcripts of π_b can be accepting for $x \notin \mathsf{HC}$ since for π_b soundness does not hold. However by observation 3 we know that for a fixed triple (x, r_j, dm) there exists only one possible accepting transcript for the sub-protocol $\pi_{\bar{b}}$ since, fixed the determining message, $\mathcal{P}^*$ is forced to honestly follow the (BL) protocol according to the randomness committed in it. Since $\mathcal{V}$ accepts if and only if *both* sub-protocols executions are accepting, since $\pi_{\bar{b}}$ is sound against reset attack, the composition of the two protocols is resettably sound too. This proves the following theorem.

Theorem 1 (Protocol $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ is resettably sound)**.** *If trapdoor permutations and collision resistance hash functions exist then the protocol* $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ *is a resettably sound argument system.*

Resettable argument of knowledge. In this paragraph we show that the protocol depicted in Fig. 4.2 is a *resettable* argument of knowledge. In order to prove this we show an expected polynomial time extractor $\mathbf{E}$ that extracts the witness from any $\mathcal{P}^*$ with probability that is negligibly close to the probability that $\mathcal{P}^*$ convinces an honest verifier.

Let (x, j, dm) be the label of the session in which $\mathcal{P}^*$ provides an accepting proof. The goal of the extractor is to obtain two distinct accepting transcripts for the same label. As observed before one label identifies a unique execution of one sub-protocol in which $\mathcal{P}^*$ is forced to play honestly. Therefore, given a fixed label, the extractor first has to identify the sub-protocol in which $\mathcal{P}^*$ is forced to use the committed randomness. Then it rewinds the prover and in that sub-protocol it tries several times to obtain a new accepting transcript, opening to distinct challenges of the BL's protocol, i.e. it will cheat in the opening of the challenge committed using the non-black-box simulator. Note that since in the execution of this sub-protocol $\mathcal{P}^*$ is bound to the committed randomness, the stand-alone simulator of resSZK is sufficient. The non-black-box simulator takes as input the code of the malicious verifier. In order to use the non-black-box simulator the extractor must prepare an augmented machine $\mathbf{M}$ that simulates the system $\langle \mathcal{P}^*, \mathcal{V} \rangle$ internally and forwards in output only the message of $\mathcal{P}^*$ belonging to the protocol resSZK protocol.

In the following paragraph we describe the extractor **E** and the augmented Machine $\mathbf{M}$.

The Extractor E. Let $s(n)$ be the maximum number of sessions that adversary $\mathcal{P}^*$ opens. This bound is known since we know the size of the circuit of the adversary (we are assuming that the extractor gets the code of $\mathcal{P}^*$). Following the traditional approach, our extractor roughly consists of two phases, in the first phase **E** follows the honest verifier procedure. When $\mathcal{P}^*$ has completed her execution, if there exists an accepting session, labelled by (x, j, dm) (if there are many, the extractor will consider the last accepting session) the extractor goes ahead to the second phase. Let us call the session labelled by (x, j, dm) as target session. In the second phase the extractor tries to obtain another distinct accepting transcript for the target session that allows extraction of the witness. The distinct transcript is obtained by cheating in the opening of the challenge committed in the first phase. The cheating is done by simulating the zero knowledge proof given by the verifier (i.e. running the non black-box simulator Sim). In this phase **E** must detect the sub-protocol in which $\mathcal{P}^*$ is bound to the randomness committed in dm (recall that dm consists of the commitment of two randomnesses and a bit). Indeed it is only in this sub-protocol that **E** can use the stand-alone simulator Sim. The second phase requires polynomially many rewinds for which the extractor needs fresh randomness, i.e. randomness that is distinct from the one used to execute the first phase. The actual extractor is more involved as it requires an intermediate estimation step (as shown in [GK96]) in which the extractor estimates the probability of having another accepting transcript for the label (x, j, dm).

Therefore, the random tape used by **E** can be seen as partitioned in three blocks. The first block is used to activate the malicious prover, the second block is used to executed the first phase as the honest verifier, the size of this block is a fixed polynomial $s(n)$ and depends of the malicious $\mathcal{P}^*$. We denote the first block by R^* and the second block as $\vec{R}$. The last block that we denote as R'

is used to perform the estimation phase and the second phase. The size of the second block is an arbitrary polynomial. The extractor **E** is shown in Fig. 4.4.

Resettable Extractor **E**($desc(\mathcal{P}^*), R^*, \vec{R}, R'$)
Input: Random tape for honest phase $\vec{R} = r_1, \ldots, r_{s(n)}$, R^* for $\mathcal{P}^*$'s activation, for all other computations R'.

- **H**onest Verifier Phase. Run $\langle \mathcal{P}^*(R^*), \mathcal{V}(\vec{R})\rangle$. Upon completion of $\mathcal{P}^*$'s execution: let (x, j, dm) the label of the last accepting session, and let $\tau_{x,j}$ be the accepting transcript (for better readability we omit the subscript dm). If there are no accepting transcripts output $\perp$. From now on, the extractor will focus on obtaining another accepting transcript $\tau'_{x,j}$ for the same session that we call target session. For all the other sessions, the extractor proceeds as the honest verifier. Fixed the label (x, j, dm) the accepting transcript can be seen as the concatenation of the accepting transcript of each sub-protocol $\tau_{x,j} = \tau^0_{x,j}, \tau^1_{x,j}$ and dm.

- **E**stimation Phase. Run $\mathcal{P}^*(R^*)$ and execute the honest verifier procedure using randomness $\vec{R}$ for all sessions except that in the target session execute the protocol resSZK with independent fresh randomness taken from R'. Repeat this step until either n^2 accepting transcripts for the target session have been obtained, or the loop has been repeated 2^n times. Let $q(n)$ be the number of iteration needed to obtain the n^2 accepting transcripts.

- **E**xtraction Phase. Pick $b \xleftarrow{\$} \{0, 1\}$.
  Repeat $q(n)$ times:
  1. pick $ch' \leftarrow \{0, 1\}^n$ using random tape R'. Run $\tau_{ts} \leftarrow \mathsf{Sim}(\mathbf{M}(\mathcal{P}^*, \mathcal{V}, R^*, \vec{R}, b, (x, j, \mathsf{dm}), \tau^b_{x,j}, ch')$ on the theorem $(\mathsf{c}_b, ch') \in \Lambda_{\mathsf{SHCom}}$; If Sim does not abort, **r**econstruct the new accepting transcript: run $\mathcal{P}^*(R^*)$ and plays as the honest verifier with randomness $\vec{R}$ except that in all sessions labelled with (x, j, dm) run the sub-protocol $\pi^{\mathcal{V}}_b$ opening the commitment c_b as ch' and using the transcript τ_{ts} for the $\mathsf{rs}\mathcal{SZK}$ proof. At the end of the execution obtain the transcript $\tau_{x,j} = \tau'^b_{x,j}, \tau^{(\bar{b})}_{x,j}$ and use $\tau^b_{x,j}$ and $\tau'^b_{x,j}$ to extract the cycle y. Output y and halt.
  2. $b \leftarrow (b + 1) mod\, 2$.

  Output $\perp$.

Figure 4.4: The resettable extractor **E**.

Remark 2. *The above extractor follows the behaviour of any standard extractor, therefore when it halts, it either outputs the witness or the special symbol $\perp$. However, if the* $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ *protocol is used as sub-routine in a larger protocol, when proving the security of the larger protocol, it could be useful to have an extractor that in case of failure provides more information, that can be used by a possible outer simulator/extractor. Therefore it is straightforward to modify the above extractor such that in case of abort it outputs the last message received by $\mathcal{P}^*$.*

Claim 4.3.1. *The extractor **E** runs in expected polynomial time in the security parameter n.*

Proof. The extractor consists of three phases: the honest verifier phase, the estimation phase and the extraction phase. The honest verifier phase consists in executing the (PPT) procedure of the honest verifier, thus this step requires polynomial time $t_{\mathrm{ver}}(n)$. Now, assume that in this phase $\mathcal{P}^*$ has provided an accepting transcript for a session (x, j, dm) (as explained above, a session is determined also by the determining message dm), with probability $\zeta^{(x,j)} = \zeta(R^*, r_j, x)$. Then, with probability $\zeta^{(x,j)}$ **E** initiates to the estimation phase.

The estimation phase, follows the Goldreich Kahan [GK96] technique, and consists of repeating the execution with $\mathcal{P}^*$ until n^2 accepting transcript for the session (x, j, dm) are obtained. Upon each repetition, the view of $\mathcal{P}^*$ is identical for all the other sessions, and for session (x, j, dm) the only change is the randomness used by the verifier in running the protocol resSZK. Therefore, we have that at each repetition, in session (x, j, dm), $\mathcal{P}^*$ produces an accepting transcript with probability $\zeta^{(x,j)}$. Therefore in order to obtain n^2 accepting transcript **E** runs the second step: $q(n) = \frac{n^2}{\zeta^{(x,j)}}$ times.

In the extraction phase **E** runs the simulator Sim on input the augmented machine **M**. The augmented machine takes in input (among the other inputs) the bit of the target sub-protocol in which the extractor wants to cheat and does the following: 1) it executes the honest verifier procedure as long as it does not detect that the prover is successfully resetting the verifier in the target sub-protocol 2) forwards the messages belonging to the resSZK protocol of the target sub-protocol to Sim. Hence, **M** runs in polynomial time. The non-black box simulator of the protocol resSZK runs also in polynomial time t_{sim}. Recall that the extractor has to figure out in which sub-protocol it can safely use the stand-alone simulator. In order to do this, at each attempt **E** invokes Sim first with input **M** that cheats in in sub-protocol b and the then again **M** cheating in sub-protocol $(1 - b)$.

Thus, the extraction phase consists of the repetition of at most $2q(n)$ times of the simulator Sim plus a polynomial time due to the reconstruction of the new transcript that we denote as $t_{\mathtt{rec}}$.

Summing up the total running time of the three phases is the following:

$$t_{\mathrm{ver}} + \zeta^{(x,j)} \cdot \Big[q(n) \cdot \mathsf{poly} + t_{sim} \cdot q(n) + t_{\mathtt{rec}}\Big] = \mathsf{poly}(n)$$

□

Claim 4.3.2. *Let $\mathcal{P}^*$ any PPT prover, let p the probability that $\mathcal{P}^*$ provides an accepting transcript* *for a theorem $x \in \mathsf{HC}$. If trapdoor permutations and collision resistance hash functions exist, then* $\mathbf{E}$ *outputs the witness $y \in \mathcal{R}_{\mathsf{HC}}(x)$ with probability negligibly close to p.*

Proof. Recall that due to the statistically biding of SBCom and to the soundness of the $\mathsf{r}ZAP$s, for each accepting session (x, j, dm) the sub-protocol $\pi_{\bar{b}}$ is resettable sound and that in such sub-protocol $\mathcal{P}^*$ is bound to the randomness committed in the determining message dm. Note that the strategy of the extractor is basically to play honestly in all sessions, while in the session (x, j, dm) tries to make effective rewinds by keeping the same verifier's determining message, i.e. the commitments of the challenges of BL's protocol $\mathsf{c}_0, \mathsf{c}_1$ and cheating in the opening by providing a false proof. Such a cheating is allowed only in the sub-protocol $\pi_{\bar{b}}$ where $\mathcal{P}^*$ is forced to be consistent and in turn for $\mathbf{E}$ is sufficient to cheat invoking the only stand-alone non-black box simulator Sim. Assume that there exists a session (x, j, dm) (that we will denote as target session) in which $\mathcal{P}^*$ generates an accepting transcript with probability p. An accepting transcript consists of a pair of sub-transcripts $\tau_{x,j} = (\tau^0_{x,j}, \tau^1_{x,j})$. We want to argue that $\mathbf{E}$ is able to obtain from $\mathcal{P}^*$ a new accepting sub-transcript $\tau'^{(\bar{b})}_{x,j}$, for the target session with almost the same probability. We show this through hybrids arguments.

$\mathbf{H}_0$: In this hybrid consider a modified version of the extractor $\mathbf{E}$ that runs always as the honest verifier. Namely, in the third phase (the extraction phase), instead of playing with challenge ch', $\mathbf{E}$ activated the augmented machine $\mathbf{M}$ on input the honest openings of the challenge ch (as the honest receiver) and the messages of the augmented machine are held by the real prover instead of the simulator. Obviously in this modified extraction phase, once $\mathbf{E}$ gets another accepting transcript for the target session it does not extract the witness. In this experiment the view of the prover $\mathcal{P}^*$ interacting with an honest $\mathcal{V}$ is indistinguishable from the view of $\mathcal{P}^*$ interacting with the modified $\mathbf{E}$. The only difference is that here the extractor could abort more often then honest $\mathcal{V}$.

Now we want to argue that in this experiment the extractor outputs $\perp$ with negligible probability. Note that the extractor outputs $\perp$ in the third phase if, after repeating the execution with $\mathcal{P}^*$ $2q(n)$ times (where $\frac{1}{q(n)}$ is an estimation of p), $\mathcal{P}^*$ does not provide another accepting transcript for the target session (x, j, dm). Note that by the Goldreich Kahan analysis we have that after $q(n)$ number of repetitions, $\mathcal{P}^*$ provides another accepting transcript with overwhelming probability. Note also that, in the extraction phase, the extractor runs the augmented machine M. More specifically each repetition M is run with a target sub-protocol in input, and it aborts when detects that the prover's messages are not consistent with previous transcript. However, at each repetition M is run twice, once per each sub-protocol. Now, by the soundness of $\mathsf{r}ZAP$ and by the statistically-binding property of the commitment sent by $\mathcal{P}^*$, we have that at least in one sub-protocol $\mathcal{P}^*$ must be consistent with previous transcript. Then, since in the extraction phase, each attempt is repeated once for each sub-protocol, there exists a sub-protocol in which M does not abort. Therefore, at each repetition **E** aborts only if also $\mathcal{P}^*$ aborts. Due to the indistinguishability of the view generated by **E** $\mathcal{P}^*$ aborts with the same probability of the first and second phase. Thus the probability of obtaining a new transcript in the third phase is overwhelming, and in turn the probability that **E** outputs $\perp$ is negligible.

$\mathbf{H}_1$: In this hybrids the extractor works as in experiment $\mathbf{H}_1$ except that in the third phase instead of handling the message of the augmented machine M running the honest prover strategy on the theorem $(\mathsf{c}_b, ch \in \Lambda_{\mathsf{V}})$ it invokes the zero knowledge simulator Sim. By the statistically zero knowledge property of the protocol resSZK $\mathbf{H}_0$ and $\mathbf{H}_1$ are statistically close.

$\mathbf{H}_2$: In this hybrid the extractor works as in the previous hybrid except that in the third phase it invokes the simulator on theorem $(\mathsf{c}_b, ch' \in \Lambda_{\mathsf{V}})$. Due to the statistically hiding property of the commitment SHCom hybrids $\mathbf{H}_2$ and $\mathbf{H}_1$ are statistically close. This is the extractor described in Fig. 4.4.

Remark 3 (Simulation Soundness in not needed.). *Note that in this experiment $\mathcal{P}^*$ is receiving simulated proofs by* Sim *of a false theorem and could maul this proof in other concurrent executions. Since* resSZK *is not simulation sound (very roughly a protocol is simulation sound when even if the adversary receives simulated proof of false theorems is still not able to prove a false theorem to an honest verifier) we cannot rule out this possibility. Note how-*

ever that by the unconditional soundness of $\mathsf{r}ZAP$ $\mathcal{P}^*$ *cannot use the simulated proof within the same execution in which the simulator is used. Still* $\mathcal{P}^*$ *could open new concurrent executions and use the simulated messages to lead* $\mathcal{V}$ *to accept a false theorem. This is not a problem since the extractor, once is in the third phase, has already a target theorem/session on which is trying to extract, therefore, other new (possibly false) theorems proved in other sessions are nor relevant to* ***E***.

□

Augmented Machine In this paragraph we formally define the augmented machine M (depicted in Fig. 4.5). Very roughly, M internally runs the system $\langle \mathcal{P}^*(R^*), \mathcal{V}(\vec{R}) \rangle$ honestly using the same randomness used in the first phase of **E** for all sessions different from the target session. The target session (x, j, dm), and the sub-protocol π_b in which M has to cheat are provided in input. The cheating consists in opening the commitment of the challenge c_b sent in the first phase as a fresh challenge ch' that is also provided as input (note that M is a deterministic machine).

For the target session, M cheats by simulating the ZK protocol proving that ch' is the correct opening of c_b, i.e., the verifier's message of resSZK sent by $\mathcal{P}^*$ are written to the output tape, and M waits the the simulator writes the prover's answer to the input tape. One of the most important task of M is to detect if the sub-protocol π_b in which it is trying to cheat is the wrong one, i.e. is the one in which $\mathcal{P}^*$ is free to reset the verifier without getting caught. In order to do this, M will receive in input the transcript of the target sub-protocol $\tau^b_{x,j}$ generated in the first phase, such that it can check if the messages sent by $\mathcal{P}^*$ are consistent with such transcript and thus detect if $\mathcal{P}^*$ is changing her messages among the resets.

Summing up, M receives the following inputs:

- the code of $\mathcal{P}^*$ and $\mathcal{V}$, the randomness $R^*, \vec{R}$, and the target session (x, j, dm) to reproduce the same execution generated in the first phase by **E** for all sessions except the target session (x, j, dm).
- the bit b indicates the sub-protocol π_b of the target session in which the messages belonging to

the resSZK protocol must be forwarded in output to the simulator.

- the fresh challenge ch' to be open to, that replaces the honest challenge committed in c_d.
- the transcript $\tau^b_{x,j}$ of the sub-protocol π_b obtained in the first step of $\mathbf{E}$ and that $\mathbf{M}$ uses to detect if the sub-protocol π_b is the one in which $\mathcal{P}^*$ is free to cheat.

In the following, we say arrays $(a_1, \dots, a_n) \neq (a'_1, \dots, a'_n)$ if there exists $i \in [n]$ such that $a_i \neq a'_i$. We denote as $\emptyset$ the empty string. When writing the Augmented Machine $\mathbf{M}$ we cannot consider sub-protocols π_0, π_1 as black-boxes, but we have to deal with each sub-protocol round. Following there is some notation for that. We indicate with $\mathsf{Z}AP^0_{\mathsf{BL}}$, $\mathsf{Z}AP^1_{\mathsf{BL}}$ the $\mathsf{r}ZAP$s sent along with messages $\mathsf{mBL1}^0, \mathsf{mBL1}^1$, and with $\mathsf{Z}AP^0_{\mathsf{rszk}}, \mathsf{Z}AP^1_{\mathsf{rszk}}$ the $\mathsf{r}ZAP$s sent along with each message $m^0_{\mathsf{V}_{\mathsf{rszk}}}, m^1_{\mathsf{V}_{\mathsf{rszk}}}$ of the resSZK protocol. We denote as $\tau^b_{x,j}$ the transcript of the sub-protocol b (the sub-session from which we are trying to extract the witness) for the accepting session labelled with (x, j, dm). Recall that $\tau^b_{x,j}$ was generated by the extractor in the first phase (honest verifier phase)(see Fig. 4.4). A sub-protocol π_d consists of three stages, the $\mathsf{BL1}$ step (along with ZAPs), the resSZK protocol and the $\mathsf{BL3}$ phase. In particular $\mathsf{BL1}, \mathsf{BL3}$ consist of a single message from $\mathcal{P}$ to $\mathcal{V}$. The resSZK steps consists of ℓ messages. Thus we denote with $\tau^b_{x,j}[\mathsf{BL1}], \tau^b_{x,j}[\mathsf{BL3}]$ the single messages for $\mathsf{BL1}, \mathsf{BL3}$ steps along with respective $\mathsf{r}ZAP$, and with $\tau^b_{x,j}[\mathsf{resSZK}^i]$ the i-th verifier's message of protocol resSZK. All messages are considered along with the respective $\mathsf{r}ZAP$s. $\mathbf{M}$ stores in the local variable τ_{Sim} the transcript of the messages received by the simulator of resSZK protocol.

4.3.2 Resettable Witness Indistinguishability

In this section we prove that protocol shown in Fig. 4.2 is resettable WI. Recall that the protocol mainly consists of a single message from $\mathcal{P}$ to $\mathcal{V}$ (i.e., the prover's determining message $(\mathsf{c}_{r_0}, \mathsf{c}_{r_1}, \mathsf{c}_s)$) and the parallel execution of π_0, π_1. Therefore the main protocol can be seen as a parallel repetition of two protocols π'_0,π'_1 where $\pi'_0 = (\mathsf{c}_s, \mathsf{c}_{r_0}, \pi_0)$ and $\pi'_1 = (\mathsf{c}_s, \mathsf{c}_{r_1}, \pi_1)$ (the repetition of c_s is only for presentation purpose).

The proof structure is the following. Assume that there exists a WI distinguisher D for the

```
$\mathbf{M}(\mathcal{P}^*, \mathcal{V}, R^*, b, (x, j, \mathsf{dm}, \tau^b_{x,j}), \vec{R}, ch')$ :

$\tau_{\mathsf{Sim}} \leftarrow \emptyset$;
Run $\mathcal{P}^*(R^*)$. When receiving a command $(x', j', \mathsf{dm}', m_{\mathcal{P}^*})$:
  – if $(x', j', \mathsf{dm}') \neq (x, j, \mathsf{dm})$: return $\mathcal{V}(x', r_j, \mathsf{dm}', m_{\mathcal{P}^*})$;
  – else:
      • if $m_{\mathcal{P}^*} = \emptyset$:
          – $(\mathsf{R_{V0}}|\mathsf{R_{V1}}) \leftarrow f_{r_j}(x||\mathsf{dm})$;
          – $\mathsf{c}_b \leftarrow \pi^{\mathcal{V}}_b(x, \mathsf{R_{V}}_b, \mathsf{mBL1}^b, \mathsf{Z}AP^b_{\mathsf{BL}})$; $\mathsf{c}_{\bar{b}} \leftarrow \pi^{\mathcal{V}}_{\bar{b}}(x, \mathsf{R_{V}}_{\bar{b}}, \mathsf{mBL1}^{\bar{b}}, \mathsf{Z}AP^{\bar{b}}_{\mathsf{BL}})$. Return $\mathsf{c}_0, \mathsf{c}_1$ to $\mathcal{P}^*$;
      • if $m_{\mathcal{P}^*} = (\mathsf{mBL1}^0, \mathsf{Z}AP^0_{\mathsf{BL}}, \mathsf{mBL1}^1, \mathsf{Z}AP^1_{\mathsf{BL}})$:
          – if $(\mathsf{mBL1}^b, \mathsf{Z}AP^b_{\mathsf{BL}}) \neq \tau^b_{x,j}[\mathsf{BL}]$ ABORT;
          – else compute $ch_{\bar{b}} \leftarrow \pi^{\mathcal{V}}_{\bar{b}}(x, \mathsf{R_{V}}_{\bar{b}}, \mathsf{mBL1}^{\bar{b}}, \mathsf{Z}AP^{\bar{b}}_{\mathsf{BL}})$. Return $ch', ch_{\bar{b}}$ to $\mathcal{P}^*$.
      • if $m_{\mathcal{P}^*} = (\mathsf{mBL1}^0, \mathsf{Z}AP^0_{\mathsf{BL}}, \mathsf{mBL1}^1, \mathsf{Z}AP^1_{\mathsf{BL}}, m^{0,i}_{\mathsf{V_{rszk}}}, \mathsf{Z}AP^0_{\mathsf{rszk}}, m^{1,i}_{\mathsf{V_{rszk}}}, \mathsf{Z}AP^1_{\mathsf{rszk}})$ for $i \in [\ell]$:
          – if $(\mathsf{mBL1}^b, \mathsf{Z}AP^b_{\mathsf{BL}}) \neq \tau^b_{x,j}[\mathsf{BL}]$ or $(m^{b,i}_{\mathsf{V_{rszk}}}, \mathsf{Z}AP^b_{\mathsf{rszk}}) \neq \tau^b_{x,j}[\mathsf{resSZK}^i]$ ABORT;
          – else $m^{\bar{b}}_{\mathsf{P_{rszk}}} \leftarrow \pi^{\mathcal{V}}_{\bar{b}}(x, \mathsf{R_{V}}_{\bar{b}}, \mathsf{mBL1}^{\bar{b}}, \mathsf{Z}AP^{\bar{b}}_{\mathsf{BL}}, m^{\bar{b},i}_{\mathsf{V_{rszk}}}, \mathsf{Z}AP^{\bar{b}}_{\mathsf{rszk}}$; if there exists a pair
            $(m^{\bar{b},i}_{\mathsf{V_{rszk}}}, m^i_{\mathsf{Sim}}) \in \tau_{\mathsf{Sim}}$, return $m^{\bar{b}}_{\mathsf{P_{rszk}}}, m^i_{\mathsf{Sim}}$ to $\mathcal{P}$; else write on the output tape $m^{\bar{b},i}_{\mathsf{V_{rszk}}}$ and wait
            for the message of $\mathsf{Sim}$.
```

Figure 4.5: Augmented Machine

protocol (π'_0, π'_1), i.e. D distinguishes whether $\mathcal{P}$ runs *both* protocols using witnesses sampled from distribution $\{\bar{y}^0(\bar{x})\}_{\bar{x}}$ or from $\{\bar{y}^1(\bar{x})\}_{\bar{x}}$. This in turn means that there exists d such that D distinguishes in sub-protocol π'_d. The difference between protocol π'_d and $\pi'_{\bar{d}}$ is due to value committed c_s: indeed, fixed sub-protocol π'_d we have that if d is the bit committed in c_s then the trapdoor theorem is true in this protocol while it is false in sub-protocol $\pi_{\bar{d}}$ (more details follow in the formal proof). Then we consider two cases: 1) the case in which D distinguishes in the sub-protocol in which the trapdoor theorem is true, i.e. it distinguishes in the execution of π'_d when c_s is a commitment of d; in claim 4.3.3 we show that if such distinguisher exists then it can be used to break the WI property of the underlying BL protocol used in π'_d. 2) the case in which D distinguishes in the sub-protocol where the trapdoor theorem is false i.e. in sub-protocol $\pi'_{\bar{d}}$ if d is the bit committed in c_s; we show that such a distinguisher can be used to break the hiding of the commitment scheme used for c_s. After analyzing that in both cases D distinguishes with negligible probability, we conclude that D does not distinguish in the main protocol.

Formal Proof. Assume that there exist PPT $\mathcal{V}^*$ and a PPT distinguisher D such that:

$$\mathsf{Pr}[D(\langle \mathcal{P}(\bar{y}^{(0)}), \mathcal{V}^* \rangle(\bar{x})) = 1] = p_0$$

while

$$\mathsf{Pr}[D(\langle \mathcal{P}(\bar{y}^{(1)}), \mathcal{V}^* \rangle(\bar{x})) = 1] = p_1$$

and $p_0 - p_1$ is non negligible, where the probability is taken on the random coins of $\mathcal{P}, \mathcal{V}^*$. For what said above the prover can be seen as it is playing two parallel protocols π'_0, π'_1. Therefore, with a slight abuse of the notation, the above equations can be re-written as follows:

$$\mathsf{Pr}[D(\langle \mathcal{P}(\pi'_0(\bar{y}^{(0)}), \pi'_1(\bar{y}^{(0)})), \mathcal{V}^* \rangle(\bar{x})) = 1] = p_0$$

and

$$\mathsf{Pr}[D(\langle \mathcal{P}(\pi'_0(\bar{y}^{(1)}), \pi'_1(\bar{y}^{(1)})), \mathcal{V}^* \rangle(\bar{x})) = 1] = p_1$$

where by $\mathcal{P}(\pi'_d(\bar{y}^{(d)}), \pi'_{\bar{d}}(\bar{y}^{(d)}))$ we denote the fact that $\mathcal{P}$ is running the protocol π'_d using witness belonging to distribution $\bar{y}^{(d)}$ and is running running protocol $\pi_{\bar{d}}$ using witness belonging to distribution $\bar{y}^{(d)}$.

If such D exists than it must exist a d such that:

$$\mathsf{Pr}[D(\langle \mathcal{P}(\pi'_d(\bar{y}^{(0)}), \pi'_{\bar{d}}(\bar{y}^{(0)})), \mathcal{V}^* \rangle(\bar{x})) = 1] = p_0$$

and

$$\mathsf{Pr}[D(\langle \mathcal{P}(\pi'_d(\bar{y}^{(1)}), \pi'_{\bar{d}}(\bar{y}^{0})), \mathcal{V}^* \rangle(\bar{x})) = 1] = p_1$$

As mentioned before the only difference between protocols π'_d and $\pi'_{\bar{d}}$ is determined by the value of the bit committed in c_s. Recall that each sub-protocol consists of several executions of $\mathsf{r}ZAP$ and the theorem proved in each $\mathsf{r}ZAP$ is the OR of the "honest theorem" and the "trapdoor theorem" (see the par. Overview given in Sec. 4.1 for more details about honest and trapdoor theorem). The trapdoor theorem for protocol π'_d is: $(\mathsf{c}_s, d) \in \Lambda_{\mathsf{trap}}$, and is true only in one sub-protocol. Let b the bit committed in c_s. By the equation above we know that there exists $\mathcal{V}^*$ such that D distinguishes in π'_d. Hence we have two cases:

Case $b = d$**.** In this case then we have that for π'_d the trapdoor theorem (used in the $\mathsf{r}ZAP$ run in the sub-protocol) is true. (We stress that however, regardless from the bit committed in c_s the honest $\mathcal{P}$ always proves the honest theorem, that is it never uses the witness (b, s) where $(\mathsf{c}_s = \mathsf{SBCom}(b, s))$ in proving ZAPs). In Claim 4.3.3 we prove that, if c_s is a commitment of d then sub-protocol π_d is resettable WI. Therefore, in this case D distinguishes with negligible probability.

Case $b \neq d$**.** In this case D distinguishes when the change of the witness is made in the sub-protocol in which the trapdoor theorem is false, i.e. in sub-protocol π'_b. If such $\mathcal{V}^*$ exists then it is possible to construct an adversary $\mathcal{A}$ for the hiding of the commitment scheme SBCom. The reduction works as follows. $\mathcal{A}$ playing in the hiding experiment obtains the challenge commitment C. Then it runs $\mathcal{V}^*$ as sub-routine and simulates the honest prover $\mathcal{P}$ in all the executions $(i', j) \neq (i, j)$. In execution (i, j)[1] it proceeds as follows. It prepares the first message $\mathsf{c}_{r_0}, \mathsf{c}_{r_1}$ as the honest prover, sets $\mathsf{c}_s = C$ and sends the three commitments to $\mathcal{V}^*$. Note that so far the view of $\mathcal{V}^*$ receiving $\mathsf{c}_{r_0}, \mathsf{c}_{r_1}, \mathsf{c}_s$ is identical to the view that she would have obtained playing with $\mathcal{P}$. Now, $\mathcal{A}$ picks $d \in \{0, 1\}$ and proceeds as follows. It runs the protocol π_d using witness belonging to distribution $\bar{y}^1$ and protocol $\pi_{\bar{d}}$ using witness belonging to distribution $\bar{y}^0$. Note that $\mathcal{A}$ can run both sub-protocols without knowing the opening of C since $\mathcal{P}$ never uses such witness in the honest execution. Then when $\mathcal{V}^*$ has completed its executions $\mathcal{A}$ hands the output to D and outputs whatever D outputs. Now if C is a commitment of d, it means that in the sub-protocol π_d where $\mathcal{A}$ changed the witness, the trapdoor theorem is true. By the claim above we know that in this sub-protocol D does not distinguish the change of the witness, therefore by $\mathcal{A}$ says 1 with probability $\approx p_0$.

[1]wlog we can focus only on execution (i, j) since if $\mathcal{V}^*$ is such that D distinguishes $\mathcal{P}^{(i,j)}(\bar{y}^0)$ from $\mathcal{P}^{(i,j)}(\bar{y}^1)$ by hybrid arguments there exists an i in which $\mathcal{V}^*$ distinguishes. Then, picking j from $\{1, \ldots, t\}$ where t is the upper-bound of the random tapes of $\mathcal{P}$, in the reduction we reduce the advantage of the adversary by a factor of $\frac{1}{t}$.

Otherwise if C is a commitment of $\bar{d}$, since the sub-protocol in which $\mathcal{A}$ changed the witness is π_d we have that, by hypothesis assumptions D says 1 with probability p_1. Since $p_0 - p_1$ is non-negligible, we have that $\mathcal{A}$ distinguishes if the bit committed in C with non-negligible probability. By the computational indistinguishability of the commitment scheme we have that D does not distinguish the witness used also in this case.

Claim 4.3.3. *Let d the bit committed in c_s in session (i, j). Then protocol π'_d is Resettable-WI.*

Proof. Intuition. The main idea of the proof is to show that if there exists a PPT $\mathcal{V}^*$ breaking the resettable WI property of the protocol π'_d then it is possible to show a reduction to break the WI of the basic Blum's protocol BL. In order to implement such reduction we have to show how the resetting power of $\mathcal{V}^*$ is disabled by the protocol π'_d. Very informally, the reduction is possible for the following reasons:

1. The randomness of the honest prover executing protocol π'_d depends of the first message sent by $\mathcal{V}^*$ to $\mathcal{P}$, namely $\mathcal{P}$ applies a PRF to the first message received by $\mathcal{V}^*$. This message is the commitment of the challenge for the BL protocol and is the determining message for $\mathcal{P}$. Due to the pseudo-randomness of PRF, whenever $\mathcal{V}^*$ changes the determining message $\mathcal{P}$ plays with fresh randomness.

2. By the resettably-sound argument of knowledge property of the resSZK protocol and by the computationally binding of the commitment scheme SHCom we have that $\mathcal{V}^*$ cannot maintain the same determining message, i.e, the commitment of BL's challenge, and at the same time ask for two distinct challenges (therefore trivially extracting the witness).

3. Finally, in order to be able to show a reduction to the WI of BL's protocol, we have to forward the messages of $\mathcal{V}^*$ to an external prover. However, notice that for each protocol message the honest prover is required to send a ZAP proving that the messages are consistent with the randomness committed in the first round of the protocol. Thus, in order to accomplish the reduction we have to cheat in the proof proving the trapdoor theorem with witness (b, s). Due

to the *resettable* WI property of $\mathsf{r}ZAP$ and the computational hiding of $\mathsf{c}_s, \mathsf{c}_{r_d}$, the cheating is not detected by any PPT $\mathcal{V}^*$.

Formal proof. The proof follows hybrids arguments. In all hybrids we assume that for all sessions opened by $\mathcal{V}^*$, the sub-protocol $\pi'_{\bar{d}}$ is always played honestly by $\mathcal{P}$, therefore for simplicity we omit to write it and we focus only on the sub-protocol π'_d. Recall that we are assuming that in all executions (i,j), in protocol π'_d the theorem $(\mathsf{c}_s, d) \in \Lambda_{\mathsf{trap}}$ is true. Recall that the procedure of π'_d requires that each protocol message sent by $\mathcal{P}$ must be followed by a $\mathsf{r}ZAP$s proving the fact that either the message is consistent with the randomness committed in the first round, or $\mathcal{P}$ knows the opening of the commitment c_s. Consider the following hybrids.

$\mathbf{H}_0$: In this hybrid $\mathcal{P}$ honestly follows the protocol using witness distribution $\bar{y}^{(0)}$.

$\mathbf{H}_1$: This hybrid is the same as before except that here $\mathcal{P}$ computes $\mathsf{r}ZAP$s using the witness (d,s) for the language Λ_{trap} (recall that we are assuming that $(\mathsf{c}_s, d) \in \Lambda_{\mathsf{trap}}$ is true). Note however, that the message are still honestly computed according to the randomness committed in the determining message. Assume that there exists a distinguisher between hybrids $\mathbf{H}_1^{(i,j)}$ and $\mathbf{H}_2$, then it is possible to construct a distinguisher for the resettable WI property of $\mathsf{r}ZAP$.

$\mathbf{H}_3^{(i,j)}$: This hybrids is the same as before except that here $\mathcal{P}$ is session (i,j) in commitment c_{r_d} instead of committing to the seed r_d of the PRF used in the protocol, commits to the string 0^n. However, the string r_d is still used as seed, and the protocol's messages are still computed using the output of the PRF. Therefore, the only difference between the two hybrids concerns the string committed in c_{r_d}. Assume that there exists a distinguisher between $\mathbf{H}_2$ and $\mathbf{H}_3^{(i,j)}$, then it is possible to construct a distinguisher $\mathcal{A}$ for the hiding of the commitment scheme SBCom. $\mathcal{A}$ runs the procedure of $\mathcal{P}$ as in the $\mathbf{H}_2$, except that for session (i,j). For this session $\mathcal{A}$ picks a random r_d and hands to the sender of the commitment scheme SBCom the strings $(r_d, 0^k)$, receiving the challenge commitment C. Then each time $\mathcal{V}^*$ activates session (i,j), $\mathcal{A}$ sends C, c_s as determining message, while it computes all messages of the protocol using the randomness generated by the PRF seeded by r_d. Clearly, if C is the commitment of r_d then the view of $\mathcal{V}^*$ is distributed as $\mathbf{H}_2$, if C is the commitment 0^n then the view of

$\mathcal{V}^*$ is distributed as $\mathbf{H}_3^{(i,j)}$. Thus at the end of $\mathcal{V}^*$'s attack $\mathcal{A}$ hands the output of $\mathcal{V}^*$ to D, then $\mathcal{A}$ outputs whatever D outputs. By the computational hiding of SBCom hybrids $\mathbf{H}_2$ and $\mathbf{H}_3^{(i,j)}$ are computationally indistinguishable. The same argument above can be repeated one by one, for all (i,j), with $j \in \{1,\ldots,t\}$ asked by $\mathcal{V}^*$ during her attack. Therefore at the end of all this sub-hybrids we have that in all session indexed by $(i,\cdot)$ played by the prover, the seed of the PRF is independent of the value committed in the first round in c_{r_d}.

$\mathbf{H}_4^{(i,j)}$: This hybrids is the same as before except that here in session (i,j) $\mathcal{P}$ computes the protocol messages using a truly random function, instead of the PRF. Note that, already in hybrid $\mathbf{H}_3^{(i,j)}$, the seed of PRF did not appear anywhere in the protocol except that in the evaluation of the PRF. Assume that there exists a distinguisher D between hybrids $\mathbf{H}_3^{(i,j)}$ and $\mathbf{H}_4^{(i,j)}$, then it is possible to construct a distinguisher $\mathcal{A}$ for the pseudo-randomness of the PRF. The reduction works as follows. $\mathcal{A}$ has access to a random oracle $\mathcal{O}$ and has to distinguish if it is truly random or is a pseudo-random function. Thus, $\mathcal{A}$ activates $\mathcal{V}^*$ and runs the same procedure of $\mathcal{P}$ in the hybrid $\mathbf{H}_3^{(i,j)}$ expect that, when $\mathcal{V}^*$ sends the determining message for the session (i,j), $\mathcal{A}$ generates the random tape to be used in this session forwarding the determining message received by $\mathcal{V}^*$ to $\mathcal{O}$.

Now, if the oracle $\mathcal{O}$ is a PRF, the view of $\mathcal{V}^*$ is distributed identically to $\mathbf{H}_3^{(i,j)}$, otherwise if $\mathcal{O}$ is a truly random function then the view of $\mathcal{V}^*$ is distributed identically to $\mathbf{H}_4^{(i,j)}$. Thus at the end of $\mathcal{V}^*$'s attack $\mathcal{A}$ hands the output of $\mathcal{V}^*$ to D, then $\mathcal{A}$ outputs whatever D outputs. By the pseudo-randomness of PRF hybrids $\mathbf{H}_3^{(i,j)}$ and $\mathbf{H}_4^{(i,j)}$ are computationally indistinguishable. The same argument above can be repeated one by one, for all (i,j), with $j \in \{1,\ldots,t\}$ asked by $\mathcal{V}^*$ during her attack.

At this point we have that in all the sessions (i,j) $\mathcal{P}$ computes the protocol messages using truly randomness and in all ZAPs $\mathcal{P}$ does not proving theorems related to the randomness used to computed the protocol messages. Thus at this point it seems that we are ready to forward the verifier messages to an external prover of BL protocol thus proving that the protocol is WI.

This is not true, since there is still a possible resetting attack that we did not rule out.

Assume that we proceeds now forwarding the verifier's message to the external prover of BL. Recall that, BL is only concurrent WI, i.e. we cannot reset the external prover. Now consider the following attack of $\mathcal{V}^*$ for a session (i, j). $\mathcal{V}^*$ sends the commitment of the challenge c_d to $\mathcal{P}$. Thus in the reduction we activate the prover of BL and obtain the first message $\mathsf{BL1}$, that we forward to $\mathcal{V}^*$. Then $\mathcal{V}^*$ sends the challenge ch_d and prove using the resSZK protocol that is the opening of the first commitment. If the proof is accepting we forward the challenge to the external prover and finally we forward the message of the external prover to $\mathcal{V}^*$. Then, $\mathcal{V}^*$ reset $\mathcal{P}$ and ask again to play session (i, j) sending the same determining message c_d. Thus, to be consistent we copy the same message sent in the other session. At this point $\mathcal{V}^*$ sends the challenge ch'_d and provides an accepting resSZK. In this case, $\mathcal{V}^*$ would extract the witness, trivially breaking the WI property. Thus, before proceeding to the next hybrid, we have to prove that $\mathcal{V}^*$ is not successful in such attack.

Assume that there exists a pair (i, j),c_d such that $\mathcal{V}^*$ in session (i, j) is able to open the commitment c_d as two distinct strings ch_d, ch'_d, therefore extracting the witness. Observe that since c_d is computed using a statistically-hiding commitment scheme, we have that theorems $(\mathsf{c}_d, ch_d) \in \Lambda_{\mathsf{SHCom}}$ and $(\mathsf{c}_d, ch'_d) \in \Lambda_{\mathsf{SHCom}}$ that are proved using the resSZK protocol are always true, therefore, we cannot conclude that if $\mathcal{V}^*$ exists, we break the soundness of resSZK. Fortunately, the protocol resSZK is also argument of knowledge, therefore if $\mathcal{V}^*$ manages to provides two accepting proof for two distinct opening of the commitment c_d, we can extract the witness in both execution and break the computational binding of the commitment scheme SHCom.

The formal reduction goes as follows: let $\mathcal{A}$ a malicious sender of the statistically hiding commitment scheme and let $\mathcal{V}^*$ as above. Let (i, j) the session in which $\mathcal{V}^*$ is able to open the commitment c_d as two distinct strings ch_d and ch'_d. $\mathcal{A}$ runs $\mathcal{V}^*$ as sub-routine simulating the prover to her as in hybrid $\mathbf{H}_4$. In session (i, j), when $\mathcal{V}^*$ sends the string ch along with an accepting ZK proof of the theorem $(\mathsf{c}_d, ch_d) \in \Lambda_{\mathsf{SHCom}}$, $\mathcal{A}$ runs the code of the extractor of resSZK (this is a black-box extractor). Note that is crucial that in the ZAP $\mathcal{A}$ is proving the trapdoor theorem, therefore, it can reset $\mathcal{V}^*$ and change the protocol messages according to the procedure of the non-black box extractor. Due to the AoK property of resSZK protocol, $\mathcal{A}$ extracts the witness, i.e. the opening of c_d with high probability. Then, when for the same session (i, j), $\mathcal{V}^*$ sends ch'_d along with another accepting proof, $\mathcal{A}$ run the extractor as before to obtain the witness of the theorem $(\mathsf{c}_d, ch'_d) \in \Lambda_{\mathsf{SHCom}}$, and thus a distinct opening. Finally $\mathcal{A}$ forwards to the honest receiver the message c_d along with the opening as ch_d and ch'_d. By the computationally-hiding of the commitment scheme this event happens with negligible probability.

$\mathbf{H}_5$: This hybrid is the same as hybrid $\mathbf{H}_4$ except that here $\mathcal{P}$ runs the protocol using the witness belonging to the distribution $\bar{y}^{(1)}$. Now assume that there exists a distinguisher D that is able to tell apart hybrid $\mathbf{H}_5$ from $\mathbf{H}_4$ then it is possible to construct a distinguisher $\mathcal{A}$ for the witness indistinguishability BL. The distinguisher works as follow. First $\mathcal{A}$ activates the prover of BL with theorems $\bar{x}$ and the two witness distributions $\bar{y}^{(0)}$,$\bar{y}^{(1)}$. When $\mathcal{A}$ activates $\mathcal{P}$ with (i, j), $\mathcal{A}$ follows the prover's procedure as in $\mathbf{H}_4$. Then upon receiving the verifier's determining message c_d it checks if c_d was already asked by $\mathcal{V}^*$. If c_d appears for the first time, $\mathcal{A}$ activates the external prover of BL handing the theorem x_i, and forwards the message $\mathsf{mBL1}^d$ to $\mathcal{V}$, along with the rZAP computed using the trapdoor witness (d, s). If c_d was asked already, then forward the same BL1 previously obtained. Upon receiving the string ch'_d, first run the ZK protocol for the theorem (c_d, ch'_d) with $\mathcal{V}^*$. If the ZK was accepting then, check if there was already an accepting pair (c_d, ch_d). If $ch_d \neq ch'_d$ then aborts. By the soundness of the resSZK this event happens with negligible probability. Otherwise if $ch_d = ch'_d$ then forwards BL3 obtained previously. Finally if there is no such pair, then forwards challenge ch'_d to the external prover. $\mathcal{A}$ follows this procedure for all (i, j) (recall that BL is concurrent WI).

When $\mathcal{V}^*$ terminates its attack, $\mathcal{A}$ hands to D the output of $\mathcal{V}^*$ and outputs whatever D outputs. Now, if the external prover picked distribution $\bar{y}^{(0)}$ then the view of $\mathcal{V}^*$ is distributed identically to $\mathbf{H}_4$, while if the external prover picked distribution $\bar{y}^{(1)}$ then the view of $\mathcal{V}^*$ is distributed identically to $\mathbf{H}_5$. By the concurrent WI property of BL we have that $\mathbf{H}_4$ and $\mathbf{H}_5$

are computationally indistinguishable. This complete the proof.

□

Therefore, the following theorem holds.

Theorem 2. *If trapdoor permutations and collision resistance hash functions exist, then the protocol shown in Fig. 4.2 is simultaneously resettable witness indistinguishable and a resettable argument of knowledge.*

CHAPTER 5

A constant-round Simultaneously resettable $\mathcal{ZK}$ argument of knowledge in the BPK model

In this chapter, we present the first application of our main protocol simres$\mathcal{WI}$AoK. We show how to combine two instances of simres$\mathcal{WI}$AoK to obtain the first construction of constant round Simultaneously Resettable Zero-knowledge Argument of Knowledge (simres$\mathcal{ZK}$AoK) in Bare Public Key (BPK) model.

5.1 Construction of simres$\mathcal{ZK}$AoK

In the BPK model, verifiers first generate public keys and corresponding secret keys. Then they register the public keys in a public file F which is available to all provers. Essentially, malicious PPT prover and verifier employing resetting attacks may run the protocol with polynomially many identities. In the following, let $s(n)$ be a polynomial in security parameter n which upper-bounds the running time of malicious prover and verifier. Hence, the size of public file is also upper-bounded by $s(n)$.

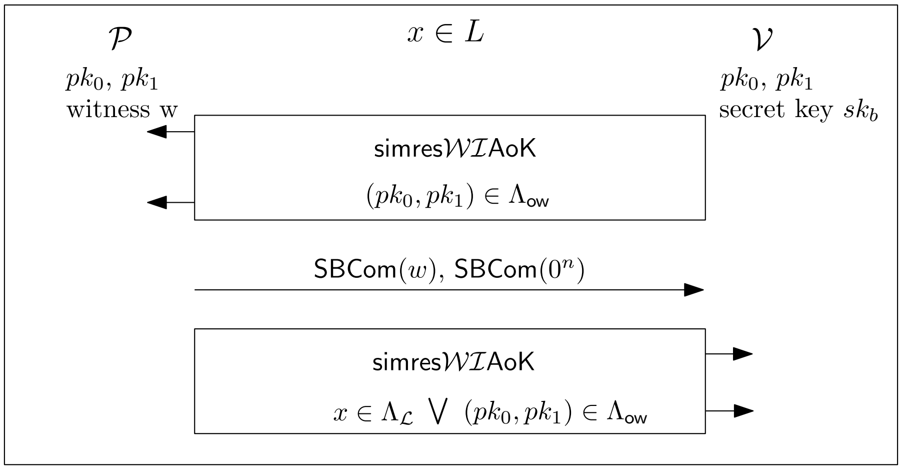
For underlying primitives, we assume to have a non-interactive statistically binding commitment scheme, denoted by SBCom where for easiness $\mathsf{SBCom}(v, s) : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$. Again, $c \leftarrow \mathsf{SBCom}(v, r)$ denotes that c is the commitment to input string v using random tape r. In addition, let $g : \{0,1\}^* \rightarrow \{0,1\}^*$ be a one-way function.

Our protocol consists of two phases, the key registration phase where verifiers create and publish their identities, and the main execution phase where $\mathcal{P}$ proves an **NP**-statement to $\mathcal{V}$ associated to an identity. In particular, the main execution phase consists of three sub-phases which we denote by Π_0, COM and Π_1 for notational convenience in the proof of security later. We now define the **NP**-languages appearing in our construction.

Language Λ_{ow}: A tuple $(y = (y_0, y_1), g)$ is in Λ_{ow} if g is an one-way function and there exists x such that $y_0 = g(x)$ or $y_1 = g(x)$. Hence, x is a witness of the **NP**-statement $(y, g) \in \Lambda_{\mathsf{ow}}$. $(x, y, g) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$ denotes that tuple (x, y, g) satisfies the **NP**-relation of Λ_{ow}.

Language Λ_{SBCom}: A tuple (m, c) is in Λ_{SBCom} if there exists a string r such that $c = \mathsf{SBCom}(m, r)$. Hence, c is the statistically binding commitment to m with respect to randomness r. Hence, r is a witness of **NP**-statement $(m, c) \in \Lambda_{\mathsf{SBCom}}$. $(c, m, r) \in \mathcal{R}_{\mathsf{SBCom}}$ denotes that tuple (c, m, r) satisfies the **NP**-relation of Λ_{SBCom}.

The pictorial description of $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ is provided in Figure 5.1. The formal definition of the protocol is provided in Figure 5.2.



$\mathcal{P}$
$x \in L$
$\mathcal{V}$
pk_0, pk_1
witness w
pk_0, pk_1
secret key sk_b
$\mathsf{simres}\mathcal{WI}\mathsf{AoK}$
$(pk_0, pk_1) \in \Lambda_{\mathsf{ow}}$
$\mathsf{SBCom}(w)$, $\mathsf{SBCom}(0^n)$
$\mathsf{simres}\mathcal{WI}\mathsf{AoK}$
$x \in \Lambda_{\mathcal{L}} \bigvee (pk_0, pk_1) \in \Lambda_{\mathsf{ow}}$

Figure 5.1: Constant-Round Simultaneously Resettable Zero-Knowledge Argument of Knowledge In The BPK Model

5.2 The proof of security of protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$

In this section, we prove that the protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ is resettably zero-knowledge and resettably argument of knowledge. In Section 5.2.1, we first prove that the protocol is resettably zero-

Protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$

Ingredients: One-way function g, statistically binding commitment scheme SBCom, sub-protocol $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$

Key Registration Phase:
$\mathcal{V}$ chooses a pair of secret keys (sk_0, sk_1) where $sk_b \in \{0,1\}^n$ and $b \in \{0,1\}$. Then $\mathcal{V}$ generates the corresponding pair of public keys (pk_0, pk_1) such that $pk_b = g(sk_b)$ for $b \in \{0,1\}$. $\mathcal{V}$ publishes (pk_0, pk_1) in public file F and stores sk_b as its secret trapdoor information with $b \overset{\$}{\leftarrow} \{0,1\}$. We call i an identity of verifier $\mathcal{V}$ as if $\mathcal{V}$'s public keys (pk_0, pk_1) appear in the ith entry of the public file F.

Main Execution Phase:
Common input: **NP**-statement x supposedly in L and the verifier's identity i. Hence, prover $\mathcal{P}$ knows public keys (pk_0^i, pk_1^i) in F, chosen by $\mathcal{V}$.
Input for $\mathcal{P}$: Witness w such that $(x,w) \in \mathcal{R}_L$ and randomness r_P.
Input for $\mathcal{V}$: Randomness r_V, secret key sk_b^i.

- $\mathcal{P}$: Obtain a sufficiently long pseudo-random tape $r_P^{'} \leftarrow f_{r_P}(x||pk_0^i||pk_1^i)$. From now on, $\mathcal{P}$ uses $r_P^{'}$ for the execution in the rest of protocol. For convenience, we assume that $r_P^{'}$ consists of four partitions, $r_P^{'}(1)$, $r_P^{'}(2)$, $r_P^{'}(3)$ and $r_P^{'}(4)$.
- $(\mathcal{V} \to \mathcal{P})$ (Π_0): $\mathcal{V}$ proves, by using $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$, the following statement:
  - There exists sk_b^i such that $(sk_b^i, (pk_0^i, pk_1^i), g) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$.

  For the execution of $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$, $\mathcal{P}$ uses random tape $r_P^{'}(1)$.
- $(\mathcal{P} \to \mathcal{V})$ (COM): If the above proof is rejecting, then $\mathcal{P}$ aborts. Otherwise, $\mathcal{P}$ commits to w and 0^n as $\mathsf{c}_0 \leftarrow \mathsf{SBCom}(w, r_P^{'}(2))$ and $\mathsf{c}_1 \leftarrow \mathsf{SBCom}(0^n, r_P^{'}(3))$. Then, $\mathcal{P}$ sends c_0 and c_1 to $\mathcal{V}$.
- $(\mathcal{P} \to \mathcal{V})$ (Π_1): $\mathcal{P}$ by using $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ and random tape $r_P^{'}(4)$ proves to $\mathcal{V}$ the following statements:
  1. There exists w and r such that $(x,w) \in \mathcal{R}_L$ and $(\mathsf{c}_0, w, r) \in \mathcal{R}_{\mathsf{SBCom}}$ **OR**
  2. There exist sk and r such that $(sk, pk_0^i, g) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$ and $(\mathsf{c}_1, sk, r) \in \mathcal{R}_{\mathsf{SBCom}}$ **OR**
  3. There exist sk and r such that $(sk, pk_1^i, g) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$ and $(\mathsf{c}_1, sk, r) \in \mathcal{R}_{\mathsf{SBCom}}$.
- $\mathcal{V}$: output "accept" if and only if the proof provided by $\mathcal{P}$ is accepting.

Figure 5.2: Constant-Round Simultaneously Resettable Zero-knowledge Argument of Knowledge in the BPK Model

knowledge by providing the resettable zero-knowledge simulator which runs in expected polynomial time. In Section 5.2.2, we prove that protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ is an indeed resettable argument of knowledge by constructing an expected polynomial time knowledge extractor.

5.2.1 Proof of Resettable Zero-Knowledge of $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$

Let $\mathcal{V}^*$ be an $s(n)$-resetting malicious verifier where $s(\cdot)$ is a polynomial, also written as s for short. Given X = $\{x_1, \cdots, x_s\}$ as an input (a set of fixed theorems), $\mathcal{V}^*$ first creates and publishes a public file F = $\{(pk_0^i, pk_1^i)|i \in [s]\}$. Upon its own choice of identity k, $\mathcal{V}^*$ proceeds to interact with at most s^2 incarnations of $\mathcal{P}_{(i,j)}$'s by choosing (i,j) where i is an index of a theorem x_i and j is the jth random tape of $\mathcal{P}$ for $i, j \in [s]$. Note that $\mathcal{V}^*$ possibly invokes each incarnation of the prover with different identities. In the following, we denote each session by an index tuple (i,j,k) where $\mathcal{V}^*$ with the kth identity invokes the incarnation of $\mathcal{P}_{(i,j)}$ in the session. We denote three main sub-phases of session (i,j,k) by Π_0^{ijk}, COM^{ijk}, and Π_1^{ijk}.

The high-level description of our expected probabilistic polynomial time (EPPT) non-black-box simulator Sim is as follows. Sim first honestly plays Π_0 with random tape and statement chosen by $\mathcal{V}^*$. Whenever Sim reaches COM of a session without the corresponding trapdoor information, Sim constructs an augmented machine M (which will be formally defined later in this section) which enables Sim to extract the trapdoor information in Π_0 by applying EPPT knowledge extractor $E_{\mathcal{WI}}$ on M. Note that each session is processed in a sequential way. Once Sim reaches COM of a session for which Sim possesses the corresponding trapdoor information sk, Sim simply commits to 0^n and sk instead of actual witness w and 0^n. At the end of each session, Sim proves to $\mathcal{V}^*$ in Π_1 that sk is the valid pre-image of one of two public keys with respect to $\mathcal{V}^*$'s one-way function. For convenience, we call a vector *empty* if the vector contains only $\perp$'s as its entries. The formal construction of Sim is provided as follows.

Simulator Sim:

Input: The description of $\mathcal{V}^*$ $\mathsf{desc}(\mathcal{V}^*)$, a vector of theorems $\overline{x} = \{x_1, \cdots, x_s\}$.

1. Sim prepares the following entries:

    (a) $\vec{\mathcal{T}}$:= $\langle sk_k \rangle_{k \in [s]}$, a vector of trapdoor secret keys sk_k where k is the index of the identity of $\mathcal{V}^*$. $\vec{\mathcal{T}}$ is initialized to be empty.

    (b) $\vec{\mathcal{R}}$:= $\langle r_{ijk} \rangle_{i,j,k \in [s]}$, a vector of random tapes for Sim and the first phase of extractor $E_{\mathcal{WI}}$ to play as an honest prover except the use of a different witness. Sim samples s^3 random tapes r_{ijk} and initializes $\vec{\mathcal{R}}$ to be $\langle r_{ijk} \rangle_{ijk \in [s]}$. Note that each r_{ijk} consists of four partitions $r_{ijk}(1)$, $r_{ijk}(2)$, $r_{ijk}(3)$, and $r_{ijk}(4)$ where $r_{ijk}(1)$ and $r_{ijk}(4)$ are used in Π_0 and Π_1 respectively while $r_{ijk}(2)$ and $r_{ijk}(3)$ are used in COM.

    (c) $\mathcal{R}^*$:= A random tape which is sufficiently long for extractor $E_{\mathcal{WI}}$ to use in the second (extraction) phase of extraction for all sessions opened by $\mathcal{V}^*$.

    (d) $\mathcal{R}_{\mathcal{M}}$:= $\langle r_{ijk} \rangle_{i,j,k \in [s]}$, a vector of random tapes for an augmented machine M to play the incarnations of honest $\mathcal{P}_{(i,j)}$. $\mathcal{R}_{\mathcal{M}}$ is initialized to be empty.

    (e) $\vec{\mathcal{R}}_{\mathsf{init}}$:= $\langle r_l \rangle_{l \in [s]}$, a vector of $s(n)$ random strings for extractor $E_{\mathcal{WI}}$ of underlying protocol $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ to initiate the malicious prover with. Sim uniformly generates $s(n)$ strings and creates vector $\vec{\mathcal{R}}_{\mathsf{init}}$. A malicious prover might ignore $\vec{\mathcal{R}}_{\mathsf{init}}$ and is allowed to use its own hardwired randomness.

2. **(Simulation of Key Generation Phase)** Sim samples a random tape for $\mathcal{V}^*$ and runs $\mathcal{V}^*$ to obtain file F = $\{(pk_0^i, pk_1^i) | i \in [s]\}$.

3. **(Simulation of Main Execution Phase)** Sim repeats the following steps until $\mathcal{V}^*$ terminates the interaction.

    Step 1: Sim receives a tuple of indexes (i, j, k) as a request from $\mathcal{V}^*$ where $\mathcal{V}^*$ with the kth identity in F and invokes the incarnation $\mathcal{P}_{(i,j)}$. Then, Sim proceeds to Step 2 with random tape $r_{ijk} \in \vec{\mathcal{R}}$ or $\mathcal{R}_{\mathcal{M}}$ and $x_i \in X$.

    Step 2: For the execution of session (i, j, k), Sim follows the instruction described below:

    - If the next message of Sim belongs to Π_0^{ijk}, then Sim honestly generates and sends the next message to $\mathcal{V}^*$ by using random tape $r_{ijk}(1)$.

- If the next message of Sim belongs to COM^{ijk}, then Sim proceeds to Step 3 without sending the commitments.
- If the next message of Sim belongs to Π_1^{ijk}, then Sim generates and sends the next message to $\mathcal{V}^*$ by using $r_{ijk}(4)$, which prove that there exists sk_k such that $(sk_k, pk_b^k, g) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$ for a bit b and that there exists $r_{ijk}(3)$ such that $(0^n, \mathsf{c}_1^{ijk}, r_{ijk}(3)) \in \mathcal{R}_{\mathsf{SBCom}}$. Bit b is easily obtained by executing $g(sk_k)$. Note that if Sim reaches Π_1^{ijk}, then Sim must have succeeded to extract and stores sk_k in $\vec{\mathcal{T}}$ in Step 4.
- Whenever $\mathcal{V}^*$ opens a new session by sending index tuple (i, j, k), Sim proceeds back to Step 1.
- Whenever $\mathcal{V}^*$ sends an empty string $\perp$, Sim terminates the execution and outputs a view including the entire transcripts exchanged between Sim and $\mathcal{V}^*$ and the randomness of $\mathcal{V}^*$.

Step 3: In session (i, j, k), Sim proceeds as follows:

- If $sk_k \in \vec{\mathcal{T}}$, then Sim commits to 0^n and sk_k as $\mathsf{c}_0 \leftarrow \mathsf{SBCom}(0^n, r_{ijk}(2))$ and $\mathsf{c}_1 \leftarrow \mathsf{SBCom}(sk_k, r_{ijk}(3))$ respectively. Sim sends c_0 and c_1 to $\mathcal{V}^*$. Then Sim proceeds to Step 2.
- If $sk_k \notin \vec{\mathcal{T}}$, then Sim proceeds to Step 4.

Step 4: (Extraction thread for extraction) Note that all the transcripts generated during Step 4 will not be included in the output view of Sim.

(a) **(Construction of Augmented machine $M(\vec{\mathcal{T}}, \mathcal{R}_{\mathcal{M}}, \mathcal{P}, \mathsf{desc}(\mathcal{V}^*))$)**

Sim first constructs augmented machine M which takes as inputs $\vec{\mathcal{T}}$, $\mathcal{R}_{\mathcal{M}}$, $\mathcal{P}$ and $\mathsf{desc}(\mathcal{V}^*)$ and follows the instruction described below.

- For any message in Π_0^{ijk} such that $sk_k \notin \vec{\mathcal{T}}$, M *externally* forwards the messages from $\mathcal{V}^*$.
- For any message in Π_0^{ijk} such that $sk_k \in \vec{\mathcal{T}}$, M *internally* emulates the interaction with $\mathcal{V}^*$ as $\mathcal{P}$ by using random tape $r_{ijk}(1) \in \mathcal{R}_{\mathcal{M}}$.
- When M reaches COM^{ijk}, M checks if $sk_k \in \vec{\mathcal{T}}$. If so, M commits to 0^n and

sk_k by using $r_{ijk}(2)$ and $r_{ijk}(3)$. Then, M *internally* sends the commitments to $\mathcal{V}^*$. If $sk_k \notin \vec{\mathcal{T}}$, then M aborts.

- If M reaches Π_1^{ijk}, then it must be the case that $sk_k \in \vec{\mathcal{T}}$. Thus, M as a prover *internally* executes Π_1^{ijk} with $\mathcal{V}^*$ by using the random tape $r_{ijk}(4) \in \mathcal{R}_{\mathcal{M}}$ to prove $(sk_k, pk_b^j, g) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$ for a bit b and that there exists $r_{ijk}(3)$ such that $(0^n, \mathsf{c}_1^{ijk}, r_{ijk}(3)) \in \mathcal{R}_{\mathsf{SBCom}}$.

(b) **(Extraction of Trapdoor Information)**

Sim runs EPPT extractor $E_{\mathcal{WI}}$ for simres$\mathcal{WI}$AoK on input $M(\vec{\mathcal{T}}, \mathcal{R}_{\mathcal{M}}, \mathcal{P}, \mathsf{desc}(\mathcal{V}^*))$, $\vec{\mathcal{R}}_{\mathsf{init}}$, $\vec{\mathcal{R}}$ and $\mathcal{R}^*$. If $E_{\mathcal{WI}}(M, \vec{\mathcal{R}}_{\mathsf{init}}, \vec{\mathcal{R}}, \mathcal{R}^*)$ outputs sk_k for some $k \in [s]$, then Sim proceeds to Step 5. Otherwise, Sim aborts and outputs "Simulation Failure".

Step 5: (Reconfiguration of entities)

Sim adds sk_k to $\vec{\mathcal{T}}$. Also, Sim removes from $\vec{\mathcal{R}}$ all the random tapes $r_{i'j'k'}$ such that $k' = k$ and adds such random tapes $r_{i'j'k'}$ to $\mathcal{R}_{\mathcal{M}}$. Then, Sim proceeds again with Step 2.

Remarks on simulator Sim: Note that the interactions between extractor $E_{\mathcal{WI}}$ and machine M in Step 4 will not be included in the output view of Sim. In addition, note that Sim solves every session in a sequential way (i.e., session by session). That is, whenever there exists a session in which Sim reaches COM without the corresponding trapdoor information, Sim opens an extraction thread and extracts the trapdoor information with overwhelming probability. If the trapdoor information corresponding to an identity has already been extracted in a previously opened session and stored in $\vec{\mathcal{T}}$, then simulator Sim simply uses the extracted trapdoor information to (straight-line) simulate any later opened session of the identical identity of $\mathcal{V}^*$ without opening extraction thread. Since the number of identities is upper-bounded by $s(n)$, the running time of simulator Sim is upper-bounded by the expected probabilistic polynomial time in the security parameter n.

Claim 5.2.1. *Simulator* Sim *runs in expected probabilistic polynomial time in security parameter n.*

Proof. Let the time to prepare entities and key generation phase be $\mathsf{poly}(n)$ for some polynomial

$\mathsf{poly}(n)$. In addition, Let $\mathsf{poly}(n)'$ be the running time of a single completed session of the protocol. Note that $\mathcal{V}^*$ resets at most $s(n)$ times, with possibly $s(n)$ distinct identities in which Sim must open $s(n)$ extraction threads to extract $s(n)$ secret keys. It is easy to see that the running time of executing a extraction thread (in Step 4) is dominated by the running time taken by extractor $E_{\mathcal{WI}}$ on input M, which is an expected probabilistic polynomial time $\mathtt{EPPT}_{E_{\mathcal{WI}}}(n)$ in n. The reconfiguration phase of Step 5 is dominated by finding and removing the used random tapes from $\vec{\mathcal{R}}$, which is at most $s(n)$. Therefore, Sim's running time t_{Sim} is as follows:

$$\begin{aligned}
t_{\mathsf{Sim}} &= \text{Running time for key generation phase} + \text{Running time for main simulation phase}\\
&\approx \mathsf{poly}(n) + s(n)\mathsf{poly}(n)' + s(n)\mathtt{EPPT}_{E_{\mathcal{WI}}}(n) + s(n)\\
&\approx \mathsf{poly}(n) + \mathsf{poly}(n)\mathtt{EPPT}_{E_{\mathcal{WI}}}(n) \text{ by letting all polynomial terms be } \mathsf{poly}(n)\\
&\approx \mathsf{poly}(n)(1 + \mathtt{EPPT}_{E_{\mathcal{WI}}}(n))\\
&= \text{expected polynomial in } n.
\end{aligned}$$

□

Let $\{\langle\mathcal{P}(\overline{w}), \mathcal{V}^*(z)\rangle(\overline{x})\}_{\overline{x},\overline{w}}$ be a random variable denoting the output view of $\mathcal{V}^*$ on input $\overline{x}$ where $\overline{w}$ is the vector of witness for $\overline{x}$. Let $\{\mathsf{Sim}(\overline{x}, \mathsf{desc}(\mathcal{V}^*))\}_{\overline{x}}$ be a random variable denoting the output view of Sim on input $\overline{x}$ and the description of $\mathcal{V}^*$. In the following, we show that the output view of Sim and the view of $\mathcal{V}^*$ in the real execution of the protocol are computationally indistinguishable. Without loss of generality, we assume that $\mathcal{V}^*$ receives its optimal random tape as non-uniform information (i.e., the optimal random tape is hardwired to $\mathcal{V}^*$). Hence, we consider $\mathcal{V}^*$ as an deterministic machine in the analysis. We present a series of hybrids and show that two consecutive hybrids are identical or computationally indistinguishable to each other.

Claim 5.2.2. *For any $s(n)$-resetting malicious verifier $\mathcal{V}^*$ and any PPT distinguisher $\mathcal{D}$, distribution ensemble $\{\mathsf{Sim}(\overline{x}, \mathsf{desc}(\mathcal{V}^*))\}_{\overline{x}}$ is computationally indistinguishable from distribution ensemble $\{\langle\mathcal{P}(\overline{w}), \mathcal{V}^*(z)\rangle(\overline{x})\}_{\overline{x},\overline{w}}$.*

Proof. Consider the following series of hybrids.

$\mathbf{H}_0(\overline{x}, \overline{w})$: is identical to a real interaction between honest prover $\mathcal{P}$ with $\overline{w}$ and $\mathcal{V}^*$ on common input $\overline{x}$. Therefore, it is easy to see that distribution ensemble $\{\mathbf{H}_0(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ is identical to $\{\langle\mathcal{P}(\overline{w}), \mathcal{V}^*(z)\rangle(\overline{x})\}_{\overline{x},\overline{w}}$.

$\mathbf{H}_1(\overline{x}, \overline{w})$: is identical to $\mathbf{H}_1(\overline{x}, \overline{w})$ except that $\mathcal{P}$ does not use pseudo-random function f to obtain random tape. Instead, $\mathcal{P}$ samples $s(n)^3$ random tapes and indexes the tapes as r_{ijk}. Finally, $\mathcal{P}$ interacts with $\mathcal{V}^*$ who invokes $\mathcal{P}(i, j)$ with the kth identity by using the prepared random tapes.

Claim 5.2.3. *Distribution ensemble $\{\boldsymbol{H}_1(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ is computationally indistinguishable from distribution ensemble $\{\boldsymbol{H}_0(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ for any PPT distinguisher $\mathcal{D}$ and all $\overline{x}$ and $\overline{w}$.*

Proof Sketch. Suppose that there exists a PPT distinguisher $\mathcal{D}$, $\overline{x}$, and $\overline{w}$ such that $\mathcal{D}$ distinguishes $\{\mathbf{H}_1(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ from $\{\mathbf{H}_0(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ with non-negligible probability. Then, $\mathcal{D}$ can be used to construct an adversary that distinguishes the ensemble of pseudo-random function from uniform random tapes with the identical advantage as follows. Let $\mathcal{P}_1$ be the prover's strategy described in $\{\mathbf{H}_1(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$. Then, adversary $\mathcal{A}$ for pseudo-random function f executes $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ as $\mathcal{P}$ with using a challenge (i.e., $s(n)^3$ pseudo-random or uniform random strings). Finally, $\mathcal{A}$ outputs whatever the distinguisher $\mathcal{D}$ outputs on the above output view. □

$\mathbf{H}_2(\overline{x}, \overline{w})$: is identical to $\mathbf{H}_1(\overline{x}, \overline{w})$ except that $\mathcal{P}$ runs EPPT extractor $E_{\mathcal{WI}}$ on M as described in Step 4. Hence $\mathcal{P}$ extracts all the secret keys from Π_0. However, $\mathcal{P}$ does not use them in the execution of COM and Π_1.

Claim 5.2.4. *Distribution ensemble $\{\boldsymbol{H}_2(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ is statistically close to $\{\boldsymbol{H}_1(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ for any PPT distinguisher $\mathcal{D}$ and all $\overline{x}$ and $\overline{w}$.*

Proof. The claim directly follows from the fact that the view of extraction thread is not included in the output view and $\mathcal{P}$ does not use any extracted trapdoor information in the protocol. Since the extraction fails only with negligible probability in security parameter n, the above two distribution ensembles are statistically close to each other. □

$\mathbf{H}_3(\overline{x}, \overline{w})$: is identical to $\mathbf{H}_2(\overline{x}, \overline{w})$ except that $\mathcal{P}$ commits to extracted secret key sk_k for commitment c_1 in session (i, j, k). Note that $\mathcal{P}$ still commits to the actual witness w_i for the first commitment c_0.

Claim 5.2.5. *Distribution ensemble $\{\boldsymbol{H}_3(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ is computationally indistinguishable from distribution ensemble $\{\boldsymbol{H}_2(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ for any PPT distinguisher $\mathcal{D}$ and all $\overline{x}$ and $\overline{w}$.*

Proof Sketch. Suppose that there exists a PPT distinguisher $\mathcal{D}$, $\overline{x}$, and $\overline{w}$ such that $\mathcal{D}$ distinguishes $\{\mathbf{H}_3(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ from $\{\mathbf{H}_2(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ with non-negligible probability. Then, we construct adversary $\mathcal{A}$ by using $\mathcal{D}$, which violates the computational hiding property of underlying statistically binding commitment. Essentially, $\mathcal{A}$ can distinguish the commitment to 0^n from the commitment to a non-zero string of the same length with non-negligible probability. $\square$

$\mathbf{H}_4(\overline{x}, \overline{w})$: is identical to $\mathbf{H}_3(\overline{x}, \overline{w})$ except that $\mathcal{P}$ executes Π_1 to prove in session (i, j, k) that $(sk_k, (pk_0^k, pk_1^k), g) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$ by using the extracted secret key sk_k.

Claim 5.2.6. *Distribution ensemble $\{\boldsymbol{H}_4(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ is computationally indistinguishable from distribution ensemble $\{\boldsymbol{H}_3(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ for any PPT distinguisher $\mathcal{D}$ and all $\overline{x}$ and $\overline{w}$.*

Proof Sketch. Suppose that there exists a PPT distinguisher $\mathcal{D}$, $\overline{x}$ and $\overline{w}$ such that $\mathcal{D}$ distinguishes distribution ensemble $\{\mathbf{H}_4(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ from distribution ensemble $\{\mathbf{H}_3(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ with non-negligible probability. We construct PPT adversary $\mathcal{A}$ that violates the witness indistinguishability property of the underlying protocol $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ as follows. The witness indistinguishability challenger receives $\overline{x}$, $\overline{w}$ and uniformly picks random tapes $r_{ijk}(3)$ for all $i, j, k \in [s]$, which is to be used to execute Π_1 as a prover of Π_1. $\mathcal{A}$ interacts with $\mathcal{V}^*$ as a prover by generating all the messages belonging to Π_0 and COM as defined by the $\{\mathbf{H}_4(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ (which is identical up to Π_0 and COM). Whenever $\mathcal{V}^*$ extract a secret key for some session, $\mathcal{V}^*$ sends this extracted secret key to the challenger. For all the messages belonging to Π_1 in every session, $\mathcal{V}^*$ externally forwards the messages between the challenger and $\mathcal{V}^*$ as the challenger is forced to prove either statement 1 using the witnesses in $\overline{w}$ for all the sessions or to prove statement 2 or 3 using the secret keys received from $\mathcal{V}^*$ for all the sessions. Upon the completion of all the sessions, $\mathcal{V}^*$ runs $\mathcal{D}$ on the output transcript and outputs whatever $\mathcal{D}$ outputs. Since the extraction fails only with negligible probability, $\mathcal{V}^*$ violates the witness indistinguishability property of $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ between proofs of actual witnesses and proofs of secret keys with non-negligible probability as does $\mathcal{D}$. □

$\mathbf{H}_5(\overline{x})$: is identical to $\mathbf{H}_4(\overline{x}, \overline{w})$ except that $\mathcal{P}$ commits to 0^n for commitment c_0 in all sessions as prover $\mathcal{P}$ does not receive $\overline{w}$ as an input.

Claim 5.2.7. *Distribution ensemble $\{\boldsymbol{H}_5(\overline{x})\}_{\overline{x}}$ is computationally indistinguishable from distribution ensemble $\{\boldsymbol{H}_4(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ for any PPT distinguisher $\mathcal{D}$ and all $\overline{x}$ and $\overline{w}$.*

Proof Sketch. Suppose that there exists a PPT distinguisher $\mathcal{D}$, $\overline{x}$, and $\overline{w}$ such that $\mathcal{D}$ distinguishes $\{\mathbf{H}_5(\overline{x})\}_{\overline{x}}$ from $\{\mathbf{H}_4(\overline{x}, \overline{w})\}_{\overline{x},\overline{w}}$ with non-negligible probability. Upon the existence of PPT distinguisher $\mathcal{D}$, we construct adversary $\mathcal{A}$ which violates the computational hiding property of underlying statistically binding commitment. In particular, $\mathcal{A}$ can distinguish the commitment to 0^n from the commitment to a non-zero string of the same length (i.e., commitments to the real witnesses) with non-negligible probability. □

Notice that $\{\mathbf{H}_5(\overline{x})\}_{\overline{x}}$ is identical to $\{\mathsf{Sim}(\overline{x}, \mathsf{desc}(\mathcal{V}^*))\}_{\overline{x}}$ as the description of $\mathcal{P}$ in $\mathbf{H}_5(\overline{x})$ is identical to the description of Sim. Thus, the proof of claim 5.2.2 is completed. □

We showed that Sim is an resettable EPPT zero-knowledge simulator for protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ by claim 5.2.1 and claim 5.2.2. Therefore, we obtain the following theorem.

Theorem 3 (Protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ is resettable Zero-knowledge)**.** *If trapdoor permutations and collision resistant hash functions exist, then protocol* $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ *is resettable Zero-knowledge argument system in bare public key model.*

5.2.2 Proof of Resettable Argument of Knowledge of $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$

In this section, we present EPPT extractor $E_{\mathcal{ZK}}$ to prove the argument of knowledge property of protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$. That is, if any PPT prover $\mathcal{P}^*$ succeeds to convince honest $\mathcal{V}$ of the validity of theorems with probability p, then E can extract the witness of the last completed session with the probability negligibly close to p. Without loss of generality, we assume that malicious prover $\mathcal{P}^*$ is hardwired to its optimal random tape. Hence, we treat $\mathcal{P}^*$ as a deterministic machine in the analysis.

The extraction of $E_{\mathcal{ZK}}$ is directly obtained from the extractor $E_{\mathcal{WI}}$ of underlying protocol $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$. For the high-level description, our non-black-box extractor $E_{\mathcal{ZK}}$ with the description of malicious prover $\mathcal{P}^*$ constructs an augmented machine that as verifier $\mathcal{V}$ internally plays Π_0 and COM with $\mathcal{P}^*$ and externally forwards all messages of Π_1. Then, $E_{\mathcal{ZK}}$ runs $E_{\mathcal{WI}}$ on the prepared augmented machine. The main technical challenge in the construction of $E_{\mathcal{ZK}}$ is the reduction from the case that $\mathcal{P}^*$ succeeds to prove the possession of secret keys in Π_1 of some session to the witness indistinguishability property of the underlying protocol $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ (i.e., the WI property of Π_0). If we simply provide the augmented machine with the secret keys as as input, which are required to play honest prover in Π_0 (denoted by $\mathcal{P}_{\Pi_0}$), then the reduction fails since the machine taken as an witness indistinguishability adversary knows already all the secrets and the relevant random coins. To overcome this difficulty, our extractor $E_{\mathcal{ZK}}$ generates the messages of $\mathcal{P}_{\Pi_0}$ and provides the messages to the augmented machine which will play Π_0 according to the provided messages without knowledge of secret keys.

To prepare $\mathcal{P}_{\Pi_0}$'s messages for the augmented machine, $E_{\mathcal{ZK}}$ exploits the following properties of $E_{\mathcal{WI}}$'s output messages. $E_{\mathcal{WI}}$ takes as inputs the followings: the description $\mathsf{desc}(\mathcal{P}^*)$ of a malicious prover, a vector of random strings $\vec{\mathcal{R}}$ to play a honest verifier, a sufficiently long random string $\mathcal{R}^*$ to rewind and extract the witness, a random string R to activate the prover. Upon the above inputs, $E_{\mathcal{WI}}$ returns a message m as follows:

1. If $m = w_i$ for the last completed session $i \in [s]$, then the extraction is successful (It is still not sufficient to claim the extraction from the larger protocol. See Claim $refzkaokproof$ for the detailed discussion).

2. If $m = \perp$ (denoting "abort") while playing as a honest verifier by using $\vec{\mathcal{R}}$, then the extraction is successful. In particular, since the malicious prover aborts the protocol execution without completing proofs in all opened sessions while $E_{\mathcal{WI}}$ plays an honest verifier, $E_{\mathcal{WI}}$

is not required to extract any witness.

3. If $m = \perp$ while rewinding and extracting a witness from the prover by using $\mathcal{R}^*$, then the extraction fails. By the property of $E_{\mathcal{WI}}$, the extraction failure occurs with only negligible probability in security parameter n.

4. If m is neither of the above cases, then $E_{\mathcal{WI}}$ received message m while playing an honest verifier by using $\vec{\mathcal{R}}$, in which $E_{\mathcal{WI}}$ as a honest verifier did not expect to receive message m.

The formal construction of non-black box extractor $E_{\mathcal{ZK}}$ is provide below.

Extractor $E_{\mathcal{ZK}}$:

Input: The description of $\mathcal{P}^*$ denoted by $\mathsf{desc}(\mathcal{P}^*)$

1. $E_{\mathcal{ZK}}$ prepares the following entries:
   (a) τ := a string of polynomial length, which contains the messages of honest verifier $\mathcal{V}$ as $\mathcal{P}_{\Pi_0}$ to be sent to $\mathcal{P}^*$ as $\mathcal{V}^*_{\Pi_0}$ during the executions of Π_0. $E_{\mathcal{ZK}}$ initializes τ to be empty.
   (b) $\langle \tau_{ijk} \rangle_{i,j,k \in [s]}$:= a vector of $s(n)^3$ strings of polynomial length. Each τ_{ijk} contains messages which are sent from $\mathcal{P}^*$ as $\mathcal{V}^*_{\Pi_0}$ to $\mathcal{V}$ as $\mathcal{P}_{\Pi_0}$ during the executions of Π_0 in session (i, j, k).
   (c) $\vec{\mathcal{R}}$:= $\langle r_j \rangle_{j \in [s]}$, a vector of random strings for $E_{\mathcal{ZK}}$ and extractor $E_{\mathcal{WI}}$ to play as an honest verifier. Sim samples s random strings r_j and initializes $\vec{\mathcal{R}}$ to be $\langle r_j \rangle_{j \in [s]}$.
   (d) $\mathcal{R}^*$:= A random tape which is sufficiently long for extractor $E_{\mathcal{WI}}$ to use in the second (extraction) phase of extraction for all sessions opened by $\mathcal{V}^*$.
   (e) $\vec{\mathcal{R}}_{\mathsf{init}}$:= $\langle r_l \rangle_{l \in [s]}$, a vector of $s(n)$ random strings for extractor $E_{\mathcal{WI}}$ to evoke the malicious prover. $E_{\mathcal{ZK}}$ simply chooses $s(n)$ random strings and creates $\vec{\mathcal{R}}_{\mathsf{init}}$. Note that a malicious prover might ignore $\vec{\mathcal{R}}_{\mathsf{init}}$ and uses its own random tapes.

2. **(Key Generation Phase)**

(a) $E_{\mathcal{ZK}}$ samples $2s$ secret keys sk_0^k and sk_1^k for $k \in [s]$. Then, $E_{\mathcal{ZK}}$ generates and publishes public file F = $\{(pk_0^k, pk_1^k)|k \in [s]\}$ where $pk_0^k = sk_0^k$ and $pk_1^k = sk_1^k$ for some one-way function g.

(b) $E_{\mathcal{ZK}}$ uniformly chooses bit $b_k \in \{0,1\}$ and stores secret key $sk_{b_k}^k$ for all $k \in [s]$.

3. **(Main Extraction Phase)** $E_{\mathcal{ZK}}$ performs the following instructions.

Step 1: (Construction of Augmented machine $M(\mathsf{desc}(\mathcal{P}^*), \tau, F)$) $E_{\mathcal{ZK}}$ constructs an augmented machine M which takes as inputs $\mathsf{desc}(\mathcal{P}^*)$,$\tau$ and F and follows the instructions described below. Then, $E_{\mathcal{ZK}}$ proceeds to Step 2.

- For any message belonging to Π_0^{ijk} or COM^{ijk} sent by $\mathcal{P}^*$, M does not externally forward it.
- For all the messages belonging to Π_1^{ijk}, M externally forwards the messages.
- If the next message that M needs to send to $\mathcal{P}^*$ belongs to the execution of Π_0^{ijk} in session (i, j, k), then M looks up τ and attempts to find the corresponding next message m which is supposed to be the prefix (of a certain size) of τ. If $\tau = \emptyset$, then M externally forwards the previous message from $\mathcal{P}^*$ and the session index (i, j, k). Otherwise, M internally sends m to $\mathcal{P}^*$ and then deletes m from τ.

Step 2: $E_{\mathcal{ZK}}$ runs EPPT extractor $E_{\mathcal{WI}}$ which as inputs takes $M(\mathsf{desc}(\mathcal{P}^*), \tau, F)$, $\vec{\mathcal{R}}_{\mathsf{init}}$, $\vec{\mathcal{R}}$ and $\mathcal{R}^*$. Let m be the message returned by $E_{\mathcal{WI}}(M, \vec{\mathcal{R}}_{\mathsf{init}}, \vec{\mathcal{R}}, \mathcal{R}^*)$. $E_{\mathcal{ZK}}$ proceeds to Step 3.

Step 3: Upon m returned in Step 2 $E_{\mathcal{ZK}}$ follows the instruction described below.

Case 1: If m contains w such that $(x_i, w) \in \mathcal{R}_L$ for some $i \in [s]$, then $E_{\mathcal{ZK}}$ outputs w.

Case 2: If $m = \perp$, then $E_{\mathcal{ZK}}$ outputs $\perp$.

Case 3: If m is neither of the above cases, then $m \in \Pi_0^{ijk}$ for some session (i, j, k) along with the index (i, j, k). Then, $E_{\mathcal{ZK}}$ does the followings:

(a) $E_{\mathcal{ZK}}$ adds m to τ_{ijk} as $\tau_{ijk} = \tau_{ijk}||m$.

(b) $E_{\mathcal{ZK}}$ runs $\mathcal{P}_{\Pi_0}$ on inputs τ_{ijk}, r_j and $sk^k_{b_k}$. Then, $E_{\mathcal{ZK}}$ computes $\mathcal{P}_{\Pi_0}$'s next message $m' = \mathcal{P}_{\Pi_0}(\tau_{ijk}, r_j, sk^k_{b_k})$.

(c) $E_{\mathcal{ZK}}$ adds m' to τ as $\tau = \tau$——m'.

(d) $E_{\mathcal{ZK}}$ proceeds to Step 1.

Remarks on extractor $E_{\mathcal{ZK}}$: Augmented machine M does not receive as inputs secret keys and random tapes of $E_{\mathcal{ZK}}$ as honest verifier $\mathcal{V}$. Instead, M only takes transcript τ which contains the messages of $E_{\mathcal{ZK}}$ as honest verifier $\mathcal{V}$ in the execution of Π_0. Since we treats PPT malicious prover $\mathcal{P}^*$ as a deterministic machine, the messages contained in τ enables M to internally response to the messages of $\mathcal{P}^*$ without knowing the secret keys and random tapes of $E_{\mathcal{ZK}}$ during the internal emulation of Π_0. Whenever M could not find a message in τ to respond to $\mathcal{P}^*$ in an execution of Π_0, $E_{\mathcal{ZK}}$ parses the last message of $\mathcal{P}^*$ to generate the next message and updates τ to construct a new augmented machine M. Once τ contains all the messages to send to $\mathcal{P}^*$ for the execution of Π_0, M can be treated as a malicious prover of Π_1, which enables $E_{\mathcal{ZK}}$ to extract the witness of last completed session by running $E_{\mathcal{WI}}$ on M. We first prove that $E_{\mathcal{ZK}}$ is an EPPT extractor for protocol simres$\mathcal{ZK}$AoK.

Claim 5.2.8. *Extractor $E_{\mathcal{ZK}}$ runs in expected probabilistic polynomial time in security parameter n.*

Proof. The running time of the key registration phase clearly takes a polynomial time in security parameter n. Let c be a constant denoting the round complexity of underlying protocol simres$\mathcal{WI}$AoK. The running time of main extraction phase by $E_{\mathcal{ZK}}$ is dominated by the time of constructing augmented machine M and running time of EPPT extractor $E_{\mathcal{WI}}$ for simres$\mathcal{WI}$AoK on prepared machine M. It is easy to see that $E_{\mathcal{ZK}}$ constructs augmented machine M and runs $E_{\mathcal{WI}}$ on M at most $c \cdot s(n)$. Let $\texttt{EPPT}_{E_{\mathcal{WI}}}(n)$ be the expected probabilistic polynomial running time taken by $E_{\mathcal{WI}}$ on M. Therefore, the running time of $E_{\mathcal{ZK}}$ is also an expected probabilistic polynomial time in security parameter n which is dominated by $c \cdot s(n)(\mathsf{poly}(n) + \texttt{EPPT}_{E_{\mathcal{WI}}}(n))$. □

Claim 5.2.9. *Let $\mathcal{P}^*$ be any $s(n)$-resetting PPT prover for any polynomial $s(n)$ and let p be the* *probability with which $\mathcal{P}^*$ provides an accepting transcript for some theorem $x \in L$. Given the description of $\mathcal{P}^*$, $E_{\mathcal{ZK}}$ outputs witness $w \in \mathcal{R}_L$ with probability negligibly close to p.*

Proof. To prove the claim, we want to show that the following holds: For any PPT prover $\mathcal{P}^*$, and for all $\overline{x}$,

$$|\mathsf{Pr}[\langle \mathcal{P}^*, \mathcal{V} \rangle(\overline{x}) = 1] - \mathsf{Pr}[E_{\mathcal{ZK}}(\mathsf{desc}(\mathcal{P}^*), \overline{x}) \in \mathcal{R}_L]| < \epsilon \tag{5.1}$$

where the probability taken over the random coins of $\mathcal{P}^*$ and ϵ is a negligible function in security parameter n. That is, whenever any PPT prover $\mathcal{P}^*$ outputs an accepting transcript in real interactions with probability p, our extractor, given the description of $\mathcal{P}^*$, outputs the witness with probability negligibly close to p. Essentially, our extractor extracts the witness of last accepting session of the resetting attack without loss of generality. This is due to the fact that the malicious prover can always duplicate any session in the last session [BGGL01].

Upon the completion of protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$, we have three distinct possible events where PPT prover $\mathcal{P}^*$ outputs an accepting transcript by interactions with $\mathcal{V}$ on a vector of theorems $\overline{x}$ as a common input (Hence, $\langle \mathcal{P}^*, \mathcal{V} \rangle(\overline{x}) = 1$) as follows.

$\mathsf{Event1}$**:** $\mathcal{P}^*$ convinces $\mathcal{V}$ that there exists witness w_i such that $(x_i, w_i) \in \mathcal{R}_L$ for some $i \in [s]$.

$\mathsf{Event2}$**:** $\mathcal{P}^*$ convinces $\mathcal{V}$ that there exists $sk^k_{1-b_k}$ such that $(pk^k_{1-b_k}, sk^k_{1-b_k}) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$ while $\mathcal{V}$ holds $sk^k_{b_k}$ for the kth identity.

$\mathsf{Event3}$**:** $\mathcal{P}^*$ convinces $\mathcal{V}$ that there exists $sk^k_{b_k}$ such that $(pk^k_{b_k}, sk^k_{b_k}) \in \mathcal{R}_{\Lambda_{\mathsf{ow}}}$ while $\mathcal{V}$ holds $sk^k_{b_k}$ for the kth identity.

We call an event "Good" when $\mathcal{P}^*$ outputs an accepting transcript in which the last session of the accepting interaction with $\mathcal{V}$ corresponds to the event. Hence, the union bound gives us the following:

$$\mathsf{Pr}[\langle \mathcal{P}^*, \mathcal{V} \rangle(\overline{x}) = 1] \leq \mathsf{Pr}[\mathsf{Event1} = \mathsf{Good}] + \mathsf{Pr}[\mathsf{Event2} = \mathsf{Good}] + \mathsf{Pr}[\mathsf{Event3} = \mathsf{Good}].$$

We first prove that if $\mathsf{Event1}$ is Good with probability p, then $E_{\mathcal{ZK}}$ extracts corresponding witness (in the last accepting session) with probability negligibly close to p. Then, we show that $\mathsf{Event2}$ and $\mathsf{Event3}$ occur with only negligible probability in security parameter n.

Notice that in the core of our non-black-box EPPT extractor $E_{\mathcal{ZK}}$, (non-black-box) EPPT extractor $E_{\mathcal{WI}}$ of underlying $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$ is used as a black-box subroutine to extract the witness of proofs in accepting interactions. One crucial feature of $E_{\mathcal{WI}}$ is that it extracts the witness regardless of the validity of the theorem x. That is, $E_{\mathcal{WI}}$ not only extracts the witness w when the prover proves $x \in L$, but also it extracts (trapdoor) secret keys when the prover proves (in a witness indistinguishable way) its possession of secret keys.

If any PPT prover $\mathcal{P}^*$ generates an accepting transcript with probability p, then $E_{\mathcal{ZK}}$, given the description of $\mathcal{P}^*$ as an input, obtains a new accepting transcript with probability p, where the transcripts belonging to Π_0 and COM are honestly generated by $E_{\mathcal{ZK}}$ and the transcripts belonging to Π_1 is internally generated by $E_{\mathcal{WI}}$. As shown in the proof of Claim $4.3.2$ (the proof of argument of knowledge for our main protocol $\mathsf{simres}\mathcal{WI}\mathsf{AoK}$), $E_{\mathcal{WI}}$ obtains the accepting transcript with the identical probability as $\mathcal{P}^*$ generates the accepting transcript. Once τ contains all the $\mathcal{V}$'s messages for the executions of Π_0 generated by $E_{\mathcal{ZK}}$, $E_{\mathcal{WI}}$ obtains the accepting transcript belonging to the executions of Π_1 by interacting with an augmented machine $M(\mathsf{desc}(\mathcal{P}^*), \tau, F)$ with the same probability and outputs the witness of the last accepting session of the interaction with probability negligibly close to p, which is the output of our extract $E_{\mathcal{ZK}}$. This proves the following claim.

Claim 5.2.10. *For all $\overline{x}$ and for any PPT malicious prover $\mathcal{P}^*$ such that $\langle \mathcal{P}^*, \mathcal{V}\rangle(\overline{x}) = 1$, $E_{\mathcal{ZK}}$ takes as inputs $\mathsf{desc}(\mathcal{P}^*)$ and $\overline{x}$, and outputs the witness in the last accepting session (i, j, k) for some i,j and $k \in [s]$ with probability negligibly close to p.*

Hence, we obtain the following claim immediately.

Claim 5.2.11. *For all $\overline{x}$ and for any PPT malicious prover $\mathcal{P}^*$ such that $\langle \mathcal{P}^*, \mathcal{V}\rangle(\overline{x}) = 1$, $E_{\mathcal{ZK}}$ takes as inputs $\mathsf{desc}(\mathcal{P}^*)$ and $\overline{x}$, and outputs the witness w_i in the last accepting session (i, j, k) for some j and $k \in [s]$ with probability negligibly close to p.*

Now, we show that $\mathsf{Event2}$ and $\mathsf{Event3}$ occur with only negligible probability in security parameter n below.

Claim 5.2.12. *Suppose that there exists PPT malicious prover $\mathcal{P}^*$ and a vector of theorems $\overline{x}$ such that* $\mathsf{Pr}[\mathsf{Event2} = \mathsf{Good}] = p$*. Then, p is negligible in security parameter n.*

Proof. Towards contradiction, suppose that there exists PPT malicious prover $\mathcal{P}^*$ and vector of theorems $\overline{x}$ such that

$$\mathsf{Pr}[\mathsf{Event2} = \mathsf{Good}] = p.$$

where p is a non-negligible function in security parameter n.

Then, we construct an EPPT adversary $\mathcal{A}$ which violates the one-way property of underlying one-way function g. In particular, the adversary follows the extraction strategy of $E_{\mathcal{ZK}}$ above. Adversary $\mathcal{A}$ is a pair of two algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ defined as follows.

A challenger first randomly picks sk and generates a challenge pk such that $pk = g(sk)$ where g is a one-way function to break. The challenger sends pk to adversary $\mathcal{A}$.

Given challenge pk and black-box access to g, $\mathcal{A}_1$ randomly picks an index $k^{'} \in [s]$. For the key generation phase, $\mathcal{A}$ uniformly chooses $2s-1$ secret keys (sk_0^1, sk_1^1), (sk_0^2, sk_1^2), $\cdots$, $(sk_0^{k^{'}})$, $\cdots$, (sk_0^s, sk_1^s) and generates public keys (pk_0^k, pk_1^k) for all $k \in [s]$ by using oracle access to one-way function g. We set $k^{'}$th public key pair to be $(pk_0^{k^{'}}, pk)$ where pk is challenge from the challenger. $\mathcal{A}_1$ sets and publishes public file F which contains all the public key pairs. Finally, $\mathcal{A}_1$ stores s secret keys $sk_{b_1}^1, sk_{b_2}^2, \cdots, sk_{b_s}^s$.

Now, $\mathcal{A}_1$ as a verifier of $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ interacts with $\mathcal{P}^*$ in the following way. For all the messages of Π_0 and COM, $\mathcal{A}_1$ honestly plays $\mathcal{V}$ of $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$. For all the messages of Π_1, $\mathcal{A}_1$ externally forwards the messages. Then, $\mathcal{A}_2$ simply runs $E_{\mathcal{WI}}$ on the messages forwarded by $\mathcal{A}_1$ (Note that $E_{\mathcal{WI}}$ receives the random tape to play honest verifier of Π_1, the random tape to extract the witness, and the random tape to initiate $\mathcal{P}^*$ externally from $\mathcal{A}_2$).

Since $\mathsf{Pr}[\mathsf{Event2} = \mathsf{Good}] = p$ is non-negligible in security parameter n, $\mathcal{A}_2$ outputs sk with probability negligibly to p, which is non-negligible as well. As we randomly chooses index $k^{'}$, the overall advantage where $\mathcal{A}$ inverts challenge pk is negligibly close to p/s which is non-negligible. Therefore, $\mathcal{A}$ violates the one-way property of g with non-negligible probability in security parameter n. $\square$

Claim 5.2.13. *Suppose that there exists PPT malicious prover $\mathcal{P}^*$ and a vector of theorems $\overline{x}$ such that* $\mathsf{Pr}[\mathsf{Event3} = \mathsf{Good}] = p$. *Then, p is negligible in security parameter n.*

Proof. Towards contradiction, suppose that there exist PPT adversary $\mathcal{P}^*$ and vector of theorems $\overline{x}$ such that

$$\mathsf{Pr}[\mathsf{Event3} = \mathsf{Good}] = p.$$

We construct an EPPT adversary $\mathcal{A}$ which upon the description of $\mathcal{P}^*$ as an input violates the witness-indistinguishability property of simres$\mathcal{WI}$AoK on the language Λ_{ow}. Challenger $\mathcal{C}$ receives the following theorems and the corresponding witnesses.

- **Theorems:** A vector of theorems is defined by $\langle (pk_0^k, pk_1^k) \rangle_{k \in [s]}$.
- **Witnesses:** For each theorem, the challenger has two possible witnesses sk_0^k and sk_1^k for $k \in [s]$ such that $pk_b^k = g(sk_b^k)$ where g is a one-way function, all $b \in \{0,1\}$, and all $k \in [s]$.

Our EPPT adversary $\mathcal{A}$ taking a description of $\mathcal{P}^*$ as an input is a pair of algorithms denoted by $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$. As high-level overview, $\mathcal{A}_1$ acts as a man-in-the-middle adversary between $\mathcal{P}^*$ and $\mathcal{C}$. In the left interaction with $\mathcal{P}^*$, $\mathcal{A}_1$ plays a honest verifier of simres$\mathcal{ZK}$AoK. In the right interaction with $\mathcal{C}$, $\mathcal{A}_1$ plays a malicious resetting verifier of simres$\mathcal{WI}$AoK. In particular, $\mathcal{A}_1$ externally forwards all the messages of Π_0 between $\mathcal{C}$ and $\mathcal{P}^*$ (which is a verifier in Π_0). For any message belonging to COM and Π_1, $\mathcal{A}_1$ internally generates and sends the messages to $\mathcal{P}^*$. Finally, $\mathcal{A}_1$ collects the messages from $\mathcal{C}$ to $\mathcal{P}^*$ and outputs an augmented machine M defined as in Main Extraction Phase of $E_{\mathcal{ZK}}$.

Formally, $\mathcal{A}_1$ prepares random tapes to play COM as a receiver and Π_1 as a verifier. Each random tape contains $s(n)^3$ random strings indexed by (i, j, k) for $i, j, k \in [s]$. We denote the random tapes for the execution of COM and Π_1 by $\vec{\mathcal{R}}$. In addition, $\mathcal{A}_1$ prepares random tape $\vec{\mathcal{R}}_{\mathsf{init}}$ for initiating $\mathcal{P}^*$. Also, $\mathcal{A}_1$ publishes $\langle (pk_0^k, pk_1^k) \rangle_{k \in [s]}$ as a public file. For all the messages belonging to the execution of Π_0, $\mathcal{A}_1$ externally forwards the messages between $\mathcal{P}^*$ and $\mathcal{C}$ including the resetting request from $\mathcal{P}^*$. For all the messages belonging to the executions of COM and Π_1, $\mathcal{A}_1$ internally plays honest receiver and verifier of COM and Π_1 by using the prepared random tapes. Upon the completion of interaction with $\mathcal{P}^*$, $\mathcal{A}_1$ collects the transcript from $\mathcal{C}$ to $\mathcal{P}^*$, denoted by τ. Then, $\mathcal{A}_1$ outputs an augmented machine $M(\mathsf{desc}(\mathcal{P}^*), \tau)$ defined as in Main Extraction Phase of $E_{\mathcal{ZK}}$.

On the augmented machine output by $\mathcal{A}_1$, $\mathcal{A}_2$ runs extractor $E_{\mathcal{WI}}$ on inputs M, $\vec{\mathcal{R}}_{\mathsf{init}}$, $\vec{\mathcal{R}}$ and $\mathcal{R}^*$ as $E_{\mathcal{WI}}(M(\mathsf{desc}(\mathcal{P}^*), \tau)$, $\vec{\mathcal{R}}_{\mathsf{init}}$, $\vec{\mathcal{R}}$, $\mathcal{R}^*)$. Finally, $\mathcal{A}_2$ outputs b if secret key sk_b^k is returned by $E_{\mathcal{WI}}$ for some $k \in [s]$. Otherwise, $\mathcal{A}_2$ outputs 0. $\mathcal{A}$ outputs whatever $\mathcal{A}_2$ outputs.

Note that $\mathcal{A}_1$ construct transcript τ from $\mathcal{C}$ to $\mathcal{P}^*$ in the execution of Π_0 by externally forwarding the messages between $\mathcal{C}$ and $\mathcal{P}^*$. Since $E_{\mathcal{WI}}$ run by $\mathcal{A}_2$ takes $\vec{\mathcal{R}}$ to play honest verifier of Π_1 and augmented machine M responses all the messages from $\mathcal{P}^*$ by referring to τ, $E_{\mathcal{WI}}$ outputs sk_b^k for some $k \in [s]$ with probability negligibly close to p as $\mathsf{Pr}[\mathsf{Event3} = \mathsf{Good}] = p$. Therefore, $\mathcal{A}$ distinguishes the proof of sk_0^k and sk_1^k for some $k \in [s]$ with non-negligible probability in security parameter n. □

By Claim 5.2.12 and Claim 5.2.13, we let $\epsilon_{\mathsf{Event2}}$ and $\epsilon_{\mathsf{Event3}}$ be the negligible probabilities where $\mathsf{Event2}$ is Good and $\mathsf{Event3}$ is Good respectively. Hence, we have the following:

$$\begin{aligned}\mathsf{Pr}[\langle \mathcal{P}^*, \mathcal{V}\rangle(\overline{x}) = 1] &\leq \mathsf{Pr}[\mathsf{Event1} = \mathsf{Good}] + \mathsf{Pr}[\mathsf{Event2} = \mathsf{Good}] + \mathsf{Pr}[\mathsf{Event3} = \mathsf{Good}] \\ &= p + \epsilon_{\mathsf{Event2}} + \epsilon_{\mathsf{Event3}}\end{aligned}$$

Hence, from (5.1), we complete the proof of Claim 5.2.9 by the following:

$$\begin{aligned}&|\mathsf{Pr}[\langle \mathcal{P}^*, \mathcal{V}\rangle(\overline{x}) = 1] - \mathsf{Pr}[E_{\mathcal{ZK}}(\mathsf{desc}(\mathcal{P}^*), \overline{x}) \in \mathcal{R}_L]| \\ &\qquad \leq |p + \epsilon_{\mathsf{Event2}} + \epsilon_{\mathsf{Event3}} - \mathsf{Pr}[E_{\mathcal{ZK}}(\mathsf{desc}(\mathcal{P}^*), \overline{x}) \in \mathcal{R}_L]| = \epsilon(n) \quad \text{by Claim 5.2.11}\end{aligned}$$

□

Resettable Soundness of $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$**:** The resettable soundness of protocol $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$

follows from its property of argument of knowledge. This is because the above extractor $E_{\mathcal{ZK}}$ extracts the witness regardless of the validity of the theorems to be prove as so does the underlying extractor $E_{\mathcal{WI}}$. For any PPT $\mathcal{P}^*$ and all $\overline{x}$, Event2 and Event3 occur with only negligible probabilities (denoted by $\epsilon_{\mathsf{Event2}}$ and $\epsilon_{\mathsf{Event3}}$ respectively) by Claim 5.2.12 and Claim 5.2.13. If $x \notin L$ for all $x \in \overline{x}$, then Event1 occurs with probability 0. Therefore, for all $\overline{x}$ such that $x \notin L$ for all $x \in \overline{x}$ and for any PPT prover $\mathcal{P}^*$,

$$\mathsf{Pr}[\langle \mathcal{P}^*, \mathcal{V} \rangle(\overline{x}) = 1] \leq \epsilon_{\mathsf{Event2}} + \epsilon_{\mathsf{Event3}} = \epsilon(n).$$

Therefore, we immediately obtain the following claim on the resettable soundness of $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$.

Claim 5.2.14. *If trapdoor permutations and collision resistance hash functions exist, then protocol* $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ *is resettably-sound argument system.*

Notice that the above argument on the soundness works because because the extraction is not based on instance-dependent primitives. Consequently, by Claim 5.2.8, Claim 5.2.9, and Claim 5.2.14, we derive the following theorem.

Theorem 4. *If trapdoor permutations and collision resistance hash functions exist, then protocol* $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ *is resettably-sound argument of knowledge system.*

Combining Theorem 3 and Theorem 4, we obtain the following conclusion.

Theorem 5. *If trapdoor permutations and collision resistance hash functions exist, then protocol* $\mathsf{simres}\mathcal{ZK}\mathsf{AoK}$ *is resettably-sound resettable Zero-Knowledge argument of knowledge system.*

Part II

Constant-round concurrently secure multi-party computation protocols in the cross-domain model

CHAPTER 6

Introduction

Consider the following toy scenario: British intelligence and CIA want to enable their employees to be able to independently and freely collaborate with each other over the internet. In particular for example, Alice an agent of the British intelligence and Bob a member of the staff at CIA should be able to evaluate joint functions on the confidential databases that these countries own. Unfortunately, Alice and Bob (also, British intelligence and CIA) do not trust each other and want to protect their own confidential information from one another. Even worse, neither are they willing to trust a third-party setup. The well studied notion of *secure computation* [Yao86, GMW87] allows them to do so, however only in the *stand-alone* setting where security holds only if a single protocol session is executed in isolation. However the requirement of free collaborations between employees of British intelligence and CIA requires security to hold even when multiple sessions are executed concurrently as in the Internet. Can this be enabled? What if in parallel Bob also wants to collaborate with a secret service agent Charlie in Germany while protecting itself even in a setting where Alice and Charlie collude?

Background: Concurrent Security. The last decade has seen a push towards obtaining secure computation protocols in the demanding network setting where there might be multiple concurrent protocol executions. A large number of secure protocols (in fact under an even stronger notion of security called Universal Composability (UC)) based on various trusted third-party setup assumptions [CF01, CLOS02, BCNP04b, CPS07, Kat07, CGS08, LPV09, GO07, GK08, GGJS11] have been proposed. One of main aims to this line of work has been to reduce the level of trust that honest parties need to place in the trusted third-party setup. For example, Katz [Kat07] considered the hardware token model. In his model, honest parties program tokens and send them to other parties. Since honest parties can program their own tokens, they only need to trust their hardware token manufacturer. Groth and Ostrovsky [GO07] initiated the study of constructing UC secure protocols without relying on a single trusted external entity. In other words, one of the main goals in this line of works is to achieve those notions of security in a setting which is as close to the "plain model" as possible (also see [GK08, GGJS11] for subsequent works). In this work, we are interested in the standard notion of secure computation.

The Dark Side of Concurrency. Unfortunately, very strong impossibility results ruled out the existence of secure protocols in the concurrent setting. UC secure protocols for most functionalities of interest have been ruled out by [CF01, CKL06]. Concurrent self-composability[1] for a large class of interesting functionalities (i.e., bit transmitting functionalities) was first ruled out [Lin04] only in a setting in which the honest parties choose their inputs *adaptively* (i.e., "on the fly"). Subsequently, a series of works [BPS06, Goy11a, AGJ$^+$12, GKOV12] show that it is impossible to achieve concurrent self-composition even in the very natural setting of *static* (pre-specified) inputs. In summary, these results have firmly established that for obtaining the most general result some setup is needed unless we are willing to consider more constrained models. Finally even in a setting with bounded number of players [JORV12], an impossibility result has been established. However, this is for the more demanding setting in which honest parties choose their inputs adaptively.

6.1 Our setting and our results

We introduce a new set-up model, called the Cross-Domain(CD) model. In this model, there exist multiple domains and each of them comes with a key certification authority (KCA). Furthermore each party in a domain trusts the KCA of its domain only. In this setting, each party in a domain can gets its own public key certified by KCA it trusts. We assume that the public-key of each of the KCA is shared with every other KCA. We prove the following for the setting of n-domain multi-party protocols:

[1]Concurrent self-composition requires that a protocol remain secure even when multiple copies of the same protocol are executed concurrently.

Positive result if $n = 2$. We give a multi-party protocol that concurrently and securely evaluates any function in the CD model of two fixed domains where each domain may contain arbitrarily many parties. Additionally, our protocol has a *constant round* complexity, a *black-box* proof of security and relies only on *standard assumptions*.

Impossibility results when $n \geq 3$. We show that there does not exist a two-party protocol such that parties from three distinct domain can concurrently and securely realizes any *complete* asymmetric (only one party gets the output) deterministic functionality[2] in the stand-alone setting [Kil88, KMO94, BMM99, Kil00, KKMO00]. Our impossibility results hold even in the *very restricted* setting of *static* inputs (inputs of honest parties are pre-specified) and *fixed roles* (i.e, the adversary can corrupt only two parties who plays the same role across all executions).

This answers the motivating question we started with. We can equip the employees of British intelligence and CIA to collaborate freely but not if collaboration with Germany are also desired.

Our results directly extend to the setting of n-domain protocols. In particular an n-party protocol for concurrently and securely computing any function on the joint inputs of n parties form distinct domains exists, if and only if there are exactly n domains in the system.

6.2 Previous results with weaker notions of security

To address the problem of concurrent security for general secure computation in the *plain model*, a few candidate definitions have been proposed, including input-indistinguishable security [MPR06, GGJS12] and super-polynomial simulation [Pas03, PS04, BS05, LPV09, CLP10]. Both of these notions, although very useful in specialized settings, *do not* suffice in general. Additionally, other models that limit the level of concurrency have also been considered [Pas04, Goy11a] or allow simulation using additional outputs from the ideal functionality [GJO10]. Among these models the model of m-bounded concurrency [PR03, Pas04] which allows m different protocol executions

[2] A functionality is *complete* if it can be used to securely realize any other functionality.

to be interleaved has received a lot of attention in the literature [PR03, Pas04, Lin04, Lin08]. Unbounded concurrent oblivious transfer in the restricted model where all the inputs in all the executions are assumed to be independent has been constructed in [GM00]. Finally the only known positive results for concurrently secure composition in the plain model are for the zero-knowledge functionality [DNS98, RK99, KP01, PRS02, BPS06].

6.3 Technical Overview

In this section, we give the intuition behind our results. We assume some basic familiarity with the notion of concurrent secure computation.

Impossibility. We start by giving the intuition behind the impossibility result for constructing a protocol that concurrently securely realizes the Oblivious Transfer(OT) functionality in the setting of three parties. The extension to general asymmetric two party functionalities follows using a Theorem from [AGJ$^+$12]. In the following, we consider the simplest setting where three domains exist where each domain contains a single party.

Our impossibility result builds on top of ideas developed by [AGJ$^+$12, GKOV12] for the setting of plain model. Even though their result hold for the two party setting, we recall their technique for the setting of three parties. Consider a scenario with three parties Alice, Bob and Charlie. Now, consider an adversary that corrupts Bob and Charlie who (as receivers of the OT protocol) are allowed to participate in an arbitrary polynomial number of executions of the protocol with honest Alice (who plays as the sender). In this setting, we can construct a real-world adversary that interacts with Alice in an execution of the protocol, referred to as the *main* execution, that cannot be simulated in the ideal world. The key idea is that the adversary has secure computation at its disposal and it can use it to its advantage. More specifically, the adversary on behalf of Charlie may interact with the Alice in multiple *additional* executions of the secure computation protocol and use these executions to generates messages that it needs to send in the main execution on behalf of Bob. More specifically, the adversary securely realizes Bob by using garbled circuits such that the adversary needs to evaluate the garble circuit in order to generate the messages it sends on behalf of Bob. However, the adversary does not have the OT keys necessary for evaluation of the garbled circuit. Instead, the OT keys are given to the honest Alice from which the adversary obtains the desired OT keys by the (additional) concurrent executions of the OT protocol as Charlie. Finally, the existence of a simulator simulating such an adversary that is securely implementing Bob contradicts the stand-alone security of the OT protocol.

In the CD model, each domain containing Alice, Bob and Charlie generates a certificate associated with their public-keys. The key insight in our impossibility result is to use the setting described above and to enable the garbled circuit securely evaluating Bob to generate Bob's public key as well. The adversary however will generate Charlie's public key and secret key by himself, which enables the adversary to interact freely on Charlie's behalf. In particular, this allows the adversary to still obtain the OT keys for the garbled circuit from Alice as in the plain model. Finally, the existence of a simulator simulating such an adversary that is securely implementing Bob (along with its key registration) contradict the stand-alone security of the OT protocol in the CD model.

Positive result for two parties. The intuition behind the impossibility result above makes it abundantly clear that the adversary must be able to do secure computation with honest Alice if it wants to securely simulate Bob. However, if we restrict ourselves to the setting of two parties then the adversary essentially looses this ability. In fact, it is exactly this loss in its ability that allows us to give a positive result.

Our protocol can roughly be partitioned into two phases– the preamble phase and the post-preamble phase. In the preamble phase, a party needs to demonstrate the knowledge of the secret key corresponding to its public and the certificate issued by its KCA. Subsequently in the post-preamble phase the actual secure computation happens. In the simulation for the proof of security, obtaining the knowledge of the adversary's secret key suffices for straight-line simulation.

Our protocol proceeds to the post-preamble phase only after the adversary has demonstrated knowledge of its secret key in the preamble phase. The adversary can interleave sessions arbitrarily and among these interleaved sessions consider the first session in which the protocol reaches the post-preamble phase. Let's call this session as the target session. Now note that since the target session was the first session in which the the post-preamble phase was reached, we can expect the same thing to happen with some probability on appropriate re-windings as well. We formalize this appropriately using *swapping* argument introduced in [PRS02]. Now note that throughout this process of re-windings we never execute the post-preamble phase for any session. This allows us to avoid the technical difficulties that generally arise when construction concurrently secure two-party computation protocols. Our protocol with this limited re-windings is able to extract the secret key of the adversary and this allows our simulator to subsequently simulate all the sessions in straight-line. For our construction and the proof we build on the techniques developed in [BPS06, GJO10, GGJS12].

CHAPTER 7

Preliminaries

7.1 Concurrently secure multi-party computation

In this section, we recall the notion of concurrently secure computation. Our definitions of concurrent security build upon the the definition of Lindell [Lin08]. Some of the text has been taken verbatim from [Lin08]. A protocol is modeled by Interactive Turing Machines (ITM) as in [GMR89], which essentially represent participants of the protocol. Intuitively, in the concurrent computation, an adversary, controlling a subset of protocol participants, may interact with the honest parties in polynomially many sessions while the schedule of the protocol messages is interleaved in any way desired by the adversary. In this work, we consider the case of concurrently secure MPC with a static adversary where a static adversary corrupts before beginning the execution of the protocol and then fully controls the corrupted parties during the execution of the protocol.

Ideal model of concurrent MPC Let $f : (\{0,1\}*)^N \rightarrow (\{0,1\}*)^N$ be an (desired) N-party functionality where $N = N(n)$ is a polynomial in security parameter n. In the ideal model, we assume that a trusted party denoted by $\mathcal{F}$ exists where the trusted party computes f on the inputs received from the participants and then returns the outputs to the corresponding parties. Let P_1, P_2, $\cdots$, P_N be N parties which wish to participate in the execution of the ideal model. The adversary $\mathcal{A}$ corrupts a subset of N parties denoted by $\mathcal{C}$ and fully controls the parties in $\mathcal{C}$. Let t be the number of sessions in which the adversary desires to interact with the honest party. Note that $t = t(n)$ is an arbitrary polynomial in security parameter n. The ideal model proceeds as follows:

Ideal model of concurrent MPC Let $f : (\{0,1\}*)^N \rightarrow (\{0,1\}*)^N$ be an (desired) N-party functionality where $N = N(n)$ is a polynomial in security parameter n. In the ideal model, we assume that a trusted party denoted by $\mathcal{F}$ exists where the trusted party computes f on the inputs received from the participants and then returns the outputs to the corresponding parties. Let P_1, P_2, $\cdots$, P_N be N parties which wish to participate in the execution of the ideal model. The adversary $\mathcal{A}$ corrupts a subset of N parties denoted by $\mathcal{C}$ and fully controls the parties in $\mathcal{C}$. Let t be the number of sessions in which the adversary desires to interact with the honest party. Note that $t = t(n)$ is an arbitrary polynomial in security parameter n. The ideal model proceeds as follows:

Inputs: Each honest party $P_i \in \mathcal{H}$ holds an input vector $\vec{x}_i = \{x_i^{\mathsf{sid}}\}_{\mathsf{sid} \in [t]}$ where the honest party always uses input x_i^{sid} for the session sid. The adversary sim holds distinct input vectors $\vec{x}_i$ for each party $P_i \in \mathcal{C}$.

Session Initiation: Whenever the adversary wishes to initiate a new session, it sends a special message $\mathsf{new\text{-}session}$ to the trusted party. Upon the special message $\mathsf{new\text{-}session}$, the trusted party sends $\mathsf{sid} \in [t]$ to all N parties where sid is the index of the new session.

Input phase of the honest party: Whenever the honest party P_i receives sid from the trusted party $\mathcal{F}$, the honest party sends $(x_i^{\mathsf{sid}}, \mathsf{sid})$ to the trusted party.

Input phase of the corrupted party: The adversary by instructing the corrupted parties P_j in $\mathcal{C}$ sends $(x_j^{\mathsf{sid}}, \mathsf{sid})$ for every $j \in \mathcal{C}$ where x_j^{sid} is an arbitrarily (possibly maliciously) chosen input string of appropriate length.

The adversary receives the result: If there exists a session sid such that the trusted party has recived inputs (x_i, sid) from all parties for every $i \in [N]$, then the trusted party $\mathcal{F}$ computes $(y_1^{\mathsf{sid}}, y_2^{\mathsf{sid}}, \cdots, y_N^{\mathsf{sid}}) = f(x_1^{\mathsf{sid}}, x_2^{\mathsf{sid}}, \cdots, x_N^{\mathsf{sid}})$. The trusted party first sends to the adversary $(y_j^{\mathsf{sid}}, \mathsf{sid})$ for every j such that $P_j \in \mathcal{C}$ and waits for the adversary's response.

The adversary responds to the trusted party: Upon the reception of $(y_j^{\mathsf{sid}}, \mathsf{sid})$ for every j such that $P_j \in \mathcal{C}$, the adversary sends a messages of the following form: (R, sid) where R is a subset of honest parties to which the trusted party sends the corresponding outputs in session

sid. Then, upon receiving the message (R, sid) from the adversary, the trusted party sends y_i^{sid} to P_i in session sid if $P_i \in R$. Otherwise, it sends a special message $(\perp, \mathsf{sid})$.

Output of Ideal Model: The honest parties output the values received from the trusted party. The adversary outputs the values received from the trusted party along with its entire view of the above procedure. We let $\mathrm{IDEAL}_{\mathsf{sim}}^{\mathcal{F}}(n, z, \{\vec{x}_i\}_{i \in N})$ be a random variable denoting the outputs of the honest parties and the adversray in the Ideal model.

Real model of concurrent MPC We next consider the real model where N real parties jointly compute the desired N-ary functionality $f : (\{0,1\}*)^N \to (\{0,1\}*)^N$ on a network. Let $t = t(n)$ be a polynomial in security parameter n. Let Π be a N-party protocol for computing the desired functionality f. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial time (PPT) machine which corrupts a subset of N parties. In the real concurrent execution of protocol Π, $\mathcal{A}$ concurrently interacts with the honest parties at most t times with its own (maliciously chosen) inputs and schedule of messages. The output of real concurrent execution is denoted by $\mathrm{REAL}_{\mathcal{A}}^{\Pi}(n, z, \{\vec{x}_i\}_{i \in N})$ which includes the outputs of the honest parties, and the outputs and the view of the adversary. Note that the schedule of messages of real concurrent execution is fully controlled by the adversary. That is, the adversary sends a message of the form (α, sid) to the honest parties. Then, the honest parties add α to the view of Π of session sid, and follows the instruction of Π on the view to responds to the adversary. This process repeats in the way that the adversary desires.

Security of concurrent MPC protocols Intuitively, a concurrent multi-party protocol is said to securely carry out a given task (a desired functionality f) if the execution of the protocol in real model emulates the ideal model. More specifically, for any real world concurrent adversary, we argue that there exists an ideal world adversary (so-called *simulator*) such the the distribution ensembles of outputs from ideal and real models are computationally indistinguishable. The formal definition follows below.

Definition 7. *Let $\mathcal{F}$ be an ideal functionality and let Π be a N-party computation protocol where both compute a desired N-ary functionality f. Then, protocol Π is said to be a concurrently secure* *computation protocol for computing f if for every probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a probabilistic polynomial-time simulator* sim *in the ideal model such that for all inputs $\{\vec{x}_i\}_{i\in N}$,*

$$\left\{\mathrm{IDEAL}^{\mathcal{F}}_{\mathsf{sim}}(n, z, \{\vec{x}_i\}_{i\in N})\right\}_{n\in\mathbb{N}; z\in\{0,1\}*} \overset{\mathsf{c}}{\equiv} \left\{\mathrm{REAL}^{\Pi}_{\mathcal{A}}(n, z, \{\vec{x}_i\}_{i\in N})\right\}_{n\in\mathbb{N}; z\in\{0,1\}*}.$$

For simplicity of exposition, we consider a network which supports authenticated communication in this work. That is, the adversary can deliver only the messages which were actually sent by a party.[1]

7.2 The Cross-Domain (CD) model

We introduce a model of multiple key certification authorities (KCA) where multiple parties exist under each of the KCAs. In this model, a party might not trust only all other KCAs but the KCA to which the party belongs. Therefore, each KCA defines a domain which consists of multiple parties. Intuitively, whenever a party in one domain wants to jointly compute something with a party in another domain, each party registers its public key to its own KCA (trusted by the registering party) and obtains a certification on the public key. No party communicates with the KCA of other domains as it does not trust the other untrusted KCAs. Instead, only KCAs communicate with each other to obtain the certification information of other parties within other KCAs and then distribute the certification information to the parties under its own domain before the interaction between parties in distinct domains. Once the interaction between parties across the domains, parties uses the certification information received from the trusted KCA throughout the subsequent interaction. We formalize this as an interaction between parties and multiple KCA functionalities denoted by a set of functionalities denoted by $\mathbf{F}_{\mathsf{KCA}} = \{\mathbf{F}^1_{\mathsf{KCA}}, \mathbf{F}^2_{\mathsf{KCA}}, \ldots, \mathbf{F}^N_{\mathsf{KCA}}\}$ where N is the number of domains.

More specifically, $\mathbf{F}^i_{\mathsf{KCA}}$ for some $i \in [N]$, upon a registration request from a party within

[1]Note that, as discussed in [BCL+05], this extra treatment does not undermine our result.

the domain computes a signing key and a verification key, then it signs a part of the registration request with the signing key. Finally, it returns the signature and the verification key to the party and records the signing key (certification creation key) and the verification key (certification key) associated with the player in its storage. Note that each party may register polynomially many public keys to $\mathbf{F}^{i}_{\mathsf{KCA}}$ but all under the identical signing key. That is, we associate the signing and verification keys *with the domain* rather than separately with each player. The honest parties do not obtain the signing key associated with their registered public keys while corrupted parties have freedom to choose whatever they want to create signing and verification keys. The crucial restriction is that the number of domains is exactly N. Moreover, in this cross-domain model, we consider an adversary such that if an adversary corrupts a party in a domain, then it consequently assumes that all parties in the domain is corrupted.[2] Below, we provide the formal definition of our KCA functionality $\mathbf{F}_{\mathsf{KCA}}$.

Definition 8. ***[Functionality*** $\boldsymbol{F}_{\mathsf{KCA}}$***]:*** *Let n be the security parameter. Let* $\mathsf{KeyGen}_{\mathsf{sig}} : \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ *be a deterministic key generation algorithm of a digital signature scheme where* $\mathsf{KeyGen}_{\mathsf{sig}}$ *takes a random string $r \in \{0,1\}^n$ and outputs (master) signing key msk and (master) verification key mvk. We also denote the signing algorithm and the verification algorithm by* $\mathsf{Sign}(\cdot,\cdot)$ *and* $\mathsf{Ver}(\cdot,\cdot)$ *respectively. Each party is assigned to a unique identity* id. *Also, for each $j \in [N]$,* $\boldsymbol{F}^{j}_{\mathsf{KCA}}$ *internally maintains a set of records including good signing keys and corresponding verification keys, denoted by R. That is, R is a set consisting of at most one element of the form* (id, msk, mvk)*. Initially, R is set to be empty. Let P^j_i be the i-th party under domain* $\boldsymbol{F}^{j}_{\mathsf{KCA}}$.

1. ***Public-Key Registration by a honest party:*** *When* $\boldsymbol{F}^{j}_{\mathsf{KCA}}$ *receives a message of the form* $(\mathsf{register}, \mathsf{id}, \mathit{sid}, pk)$ *from a party with* id *within the domain. If there is a tuple* $(\mathsf{id}', msk, mvk) \in R$ *for some* id'*, then* $\boldsymbol{F}^{j}_{\mathsf{KCA}}$ *computes the signature* $\sigma \leftarrow \mathsf{Sign}(pk, msk)$*, sends* (id, σ) *to the party of* id *and to $\mathcal{A}$. If R is empty, then* $\boldsymbol{F}^{j}_{\mathsf{KCA}}$ *obtains a fresh* $(msk, mvk) \leftarrow \mathsf{KeyGen}_{\mathsf{sig}}(r)$ *where r is a uniform string and adds (msk, mvk) to R. Then,* $\boldsymbol{F}^{j}_{\mathsf{KCA}}$ *signs*

[2]For better understanding, imagine an intranet. Once an adversary succeeds to deploy a malware in a computing system within an intranet, the malware is likely to spread over other systems within the intranet, resulting in the compromise of security within the entire domain.

pk as $\sigma \leftarrow \mathsf{Sign}(m, sk)$ and sends (id, σ) to the party of id *and to $\mathcal{A}$. Finally, it adds (id, vk) to R.*

2. ***Public-Key Registration by a corrupted party:*** *When $\boldsymbol{F}^j_{\mathsf{KCA}}$ receives a message of the form $(\mathsf{register}, \mathsf{id}, \mathit{sid}, r)$ from a corrupted party of* id*, $\boldsymbol{F}^j_{\mathsf{KCA}}$ checks if the $\boldsymbol{F}^j_{\mathsf{KCA}}$ computes $(msk, mvk) \leftarrow \mathsf{KeyGen}_{\mathsf{sig}}(r)$. Then, $\boldsymbol{F}_{\mathsf{KCA}}$ records (id, mvk). Note that the adversary can choose one of the existing (msk, mvk) recorded previously by simply choosing the same r.*

3. ***Certificate Retrieval between KCAs:*** *A domain $\boldsymbol{F}^j_{\mathsf{KCA}}$ may request a verification key of any any domain $\boldsymbol{F}^k_{\mathsf{KCA}}$ by sending a message of the form $(\mathsf{retrive}, j)$ to $\boldsymbol{F}^k_{\mathsf{KCA}}$. Then, $\boldsymbol{F}^k_{\mathsf{KCA}}$ sends a recorded mvk in R to $\boldsymbol{F}^j_{\mathsf{KCA}}$. If the adversary corrupts a party in $\boldsymbol{F}^k_{\mathsf{KCA}}$, $\boldsymbol{F}^k_{\mathsf{KCA}}$ forwards the message $(\mathsf{retrive}, j)$ to $\mathcal{A}$. Then, $\mathcal{A}$ sends mvk to $\boldsymbol{F}^k_{\mathsf{KCA}}$. Subsequently, $\boldsymbol{F}^k_{\mathsf{KCA}}$ sends mvk to $\boldsymbol{F}^j_{\mathsf{KCA}}$.*

Definition 9. ***[Concurrently Security of MPC protocols in the Cross-Domain model]:*** *Let f be a desired functionality to be computed where $N(n)$ is a polynomial in security parameter n. Assume that all parties which desire to participate in the main protocol execution belongs to one of exactly N domains. The concurrently secure MPC in the CD model is defined identically to the concurrently secure MPC as in Definition 7, with the following conditions;*

1. *Before any interaction between parties starts, the parties registers their public keys by interacting with their own trusted domain as defined in Definition 8. Also, each party obtains a certificate (verification key) of parties in other domains from the own KCA who obtains the certificate by interacting with other KCAs.*

2. *If an adversary corrupts a party in a domain, then all parties in the domain are considered to be corrupted.*

Remarks: Our main aim of the above definition is to protect the privacy of inputs of parties in domains in which no corrupted party exists from interacting with corrupted parties in the other domains. Note that we can extend the model and security definition by removing the second condition. That is, a adversary is allowed to corrupt a subset of parties in one domain. We emphasize that the the security of honest parties (co-existing with corrupted parties) within a single domain can be achieved by previously known constructions based on various set-up assumptions and simulation techniques [BCNP04a, LPV09].

7.3 Garbled Circuits

In this section we briefly recall the notion and constructions for garbled circuits that we need in this paper. Garbled circuits were first constructed by Yao [Yao86] and have since then found numerous uses in secure multiparty computation and elsewhere. A vast body of literature explaining this notion exists. However since our construction relies heavily on garbled circuits in order to clarify the notation that we use in this paper we present an informal overview. We assume that the reader is familiar with the notion and constructions. We first start by considering the semi-honest (or, honest-but-curious) case and then extend the construction to the malicious case.

7.3.1 Garbled circuits for semi-honest adversaries

A formal simulation based security definition for garbled circuit construction in the case of an honest but curious adversary is presented in [HK07, KO04]. Some parts of the following text have been taken verbatim from [HK07, KO04].

Let $F_k : \{0,1\}^k \rightarrow \{0,1\}^k$ denote any polynomial time computable function[3]. Note that a garbled circuit only hide the nature of a gate used in a circuit and does not hide the number of gates used etc. However this can be achieved by using some form of canonicalization. This is fairly standard when using garbled circuits and we refer the reader to Section 4 of [GHV10] for more discussion. Formally,

Definition 10. *(garbled circuits) Let $F_k : \{0,1\}^k \rightarrow \{0,1\}^k$ be the description of any function as above. Yao [Yao86]'s* garbled circuit *technique is a pair of algorithms* $(\mathtt{Yao}_1, \mathtt{Yao}_2)$ *such that,*

- $\mathtt{Yao}_1$ *is a randomized algorithm which takes as input F_k and outputs a tuple $(GC, \mathbf{Z})$ where*

[3]The garbled circuit technique also extends to functions whose input and output are polynomial in k. However for simplicity of exposition we limit ourselves to functions with input and output lengths exactly k.

GC is a garbled circuit *and $\mathbf{Z} = \{Z_{i,\sigma}\}_{i \overset{\$}{\in} 1,\ldots,k,\sigma\in\{0,1\}}$ are* keys *corresponding to input wires.*

- $\texttt{Yao}_2$ *is a deterministic algorithm which takes as input a garbled circuit, GC and keys corresponding to an input x. More specifically it takes as input a set $\{Z_{i,x_i}\}_{i \overset{\$}{\leftarrow} 1,\ldots,k}$ (denoted by $\mathbf{Z}_x$) of keys where x_i is the i^{th} bit of x. It outputs an invalid symbol $\perp$ or a value $v \in \{0,1\}^k$.*
- ***C**orrectness: For any function F_k, let $(GC, \mathbf{Z}) \leftarrow \texttt{Yao}_1(F_k)$, and for any input x, let $v = \texttt{Yao}_2(GC, \mathbf{Z}_x)$. Then we require that $v = F_k(x)$.*
- ***P**rivacy (honest-but-curious case): $\exists$ a PPT simulator* $\texttt{YaoSim}$ *such that for all PPT adversaries $\mathcal{A}$ with auxiliary input z,*

$$\Big\{(GC, \mathbf{Z}) \leftarrow \texttt{Yao}_1(F_k) : A(k, z, x, (GC, \mathbf{Z}))\Big\}_{k\in\mathbb{N}, z\in\{0,1\}^*, x\in\{0,1\}^k}$$
$$\overset{c}{\equiv} \Big\{v = F_k(x) : A(k, z, x, \texttt{YaoSim}(k, x, v))\Big\}_{k\in\mathbb{N}, z\in\{0,1\}^*, x\in\{0,1\}^k}$$

7.3.2 Garbled circuits for malicious adversaries

Although garbled circuits were designed for the honest-but-curious case, since [GMR89], they have been used in the presence of malicious adversaries in the interactive setting via compilations with zero-knowledge proofs. [GKR08, BPS06] consider garbled circuit constructions secure against a malicious adversary. In the following for the sake of completeness we provide a formal definition of security of garbled circuits in the malicious, adaptive setting and provide a construction for the same. The construction follows from the work of [GKR08]. In our construction we sill assume a garbled circuit constructions secure in the semi-honest setting (according to Definition 10). Some of the following text has been taken verbatim from [GKR08].

Before we get into details we highlight the main concern. A malicious adversary evaluating the garbled circuits can be *adaptive* in it choice of the input x. In particular, a malicious adversary may choose the input (and thus the keys it obtains) adaptively based on the actual circuit (and also the keys so far). Therefore our simulator needs to "hold off" on embedding the output at the time of the generation of the garbled circuit. Goldwaseer et. al. [GKR08] consider such a setting. They deal with this problem by providing the adversary with a garbled circuit that computes masked functions. The unmasking values are provided along with the keys. More specifically, a simulator can be designed to *hold off* until all the input keys have been queried (so fully specified). Finally when all the input bits have been specified the simulator can manipulate the output masks based on the output that it needs to world.

Next we formalize the notion of garbled circuits in the setting of malicious adversaries below by using the simulation based definition for the functionality $\mathcal{F}$ (that can be invoked n times) to evaluate functions $F_1, F_2 \ldots F_n$.

Reactive functionality $\mathcal{F}$. A reactive functionality $\mathcal{F}$ consists of a sequence of functions F_1, F_2, ..., $F_n : \{0,1\}^k \to \{0,1\}^k$.[4] On the i^{th} invocation of the functionality $\mathcal{F}$ function F_i is evaluated and the output of F_i can potentially depend on the inputs provided to functions F_j's for all $1 \leq j < i$. We remark that in our setting we will use the next message function of a party (in an execution of an n round protocol) and model it as a functionality $\mathcal{F} = (F_1, F_2, \ldots, F_n)$.

Towards the goal of defining security, we first describe the ideal world process; next we describe the real process; and finally present the definition.

Ideal execution of a reactive functionality $\mathcal{F}$. Consider an ideal execution of an adversary sim on input $\bar{x}$ and auxiliary input z interacting with the ideal functionality $\mathcal{F}$. Let $\mathcal{F} = (F_1, F_2, \ldots, F_n)$. Execution proceeds as follows,

- **I**nput: sim receives an input $\bar{x} = (x_1, \ldots, x_n)$. It is also provided with an auxiliary input z.
- **E**valuation: For each $i \in [n]$, sim sends as input x'_i and obtains a response $F_j(x'_i)$, where as mentioned before, F_i can additionally depend on all of the inputs $x'_j, \forall 1 \leq j < i$.

[4]As in the case of semi-honest setting technique extends to functions whose input and output are polynomial in k.

- **O**utput: sim outputs any arbitrary probabilistic polynomial time function of its view. Let $\text{IDEAL}_{\mathcal{F},\mathsf{sim}}(k, \bar{x}, z)$ denote the output distribution of the ideal world adversary sim in interaction with the ideal functionality $\mathcal{F}$.

Real execution of garbled circuits by an adversary. Let $\mathcal{F} = (F_1, \ldots, F_n)$ be a reactive functionality as above and let $(\texttt{Yao}_1, \texttt{Yao}_2)$ be as above. A real model evaluation of $\mathcal{F}$ using garbled circuits uses a pair algorithms $(\texttt{Yao}_1, \texttt{Yao}_2)$ such that,

- $\texttt{Yao}_1$: $\texttt{Yao}_1$ on input a description of a reactive functionality $\mathcal{F} = (F_1, \ldots, F_n)$ outputs a tuple $(GC, \mathbf{Z})$ denoting the garbled circuit and the keys for input wires.

- $\texttt{Yao}_2$: $\texttt{Yao}_2$ is a deterministic algorithm which on input a garbled circuit GC and keys $(Z_{1,b_1}, Z_{2,b_2}, \ldots, Z_{t,b_t})$ where $t = i \cdot k$ (where b (of length t) is the concatenation of the inputs $x_1, \ldots, x_i$) outputs an evaluation of F_i.

Let $\mathcal{A}$ be any polynomial time algorithm. Then the real world execution proceeds as follows:

- **I**nput: Let $\bar{x} = (x_1, \ldots, x_n)$ be the input of the adversary $\mathcal{A}$. It is also provided with an auxiliary input z.

- **S**etup: Let $(GC, \mathbf{Z}) \leftarrow \texttt{Yao}_1(\mathcal{F})$. Let g be the number of keys where $g = n \cdot k$.

- **K**ey Queries: Let $\mathcal{K}$ be a key oracle. At any point $\mathcal{A}$ sends a message of the form $(i, b), i \in [g], b \in \{0, 1\}$ to $\mathcal{K}$ to which $\mathcal{K}$ responds by $Z_{i,b}$.

- **O**utput: $\mathcal{A}$ outputs any arbitrary probabilistic polynomial time function of its view. Let $\text{REAL}_{\mathcal{K},\mathcal{A}}(k, \bar{x}, z)$ denote the output distribution of $\mathcal{A}$.

Security: The definition of security, following the ideal-real model is as follows. [5]

[5] We provide a stronger black-box definition of security. Known constructions satisfy this definition of security.

Definition 11. *(garbled ciruits - malicious case) Let $\mathcal{F}$ be any reactive functionality as above. Then $\exists$ a PPT ideal-world adversary (i.e a black box simulator)* `YaoSim` *such that for every PPT real-world adversary $\mathcal{A}$,*

$$\left\{\text{IDEAL}_{\mathcal{F},\texttt{YaoSim}^{\mathcal{A}}}(k,\bar{x},z)\right\}_{k\in\mathbb{N},z\in\{0,1\}^*,\bar{x}\in(\{0,1\}^k)^n} \overset{c}{\equiv} \left\{\text{REAL}_{\mathcal{K},\mathcal{A}}(k,\bar{x},z)\right\}_{k\in\mathbb{N},z\in\{0,1\}^*,\bar{x}\in(\{0,1\}^k)^n}$$

Next using a construction for garbled circuits $(\widehat{\texttt{Yao}_1},\widehat{\texttt{Yao}_2})$ secure in the semi-honest setting (Definition 10) we give a construction $(\texttt{Yao}_1,\texttt{Yao}_2)$ that offers security against malicious adversaries (Definition 11). A formal construction is provided in Construction 7.1

Let $\mathcal{F} = (F_1,\ldots F_n)$ where each F_i is such that $F_i : \{0,1\}^k \to \{0,1\}^k$. Let $(\widehat{\texttt{Yao}_1},\widehat{\texttt{Yao}_2})$ be a pair of algorithms that satisfy Definition 10. Then let algorithms $(\texttt{Yao}_1,\texttt{Yao}_2)$ be as follows.

$\texttt{Yao}_1$:

1. For each $i \in \overset{\$}{\leftarrow} 1,\ldots n$, choose masks $m_i \leftarrow \{0,1\}^k$.
2. Let $\mathcal{F}' = (F_1(x_1)\oplus m_1, F_2(x_2)\oplus m_2,\ldots F_n(x_n)\oplus m_n)$. Let $(\widehat{GC},\widehat{\mathbf{Z}}) = \widehat{\texttt{Yao}_1}(\mathcal{F}')$. Note that $\widehat{\mathbf{Z}}$ consists of key pairs for kn input wires and one key from the first ki pairs of keys is needed to evaluate F_i. The specific keys needed depend on the inputs $x_1, x_2\ldots x_i$.
3. For each $i \in [n]$, for each $j \in [ik]$, choose a sub-mask $s_{i,j} \leftarrow \{0,1\}^k$ subject to $\oplus_{j=1}^{ik} s_{i,j} = m_i$. Append sub-mask $s_{i,j}$ to the keys $\widehat{Z}_{j,0}$ and $\widehat{Z}_{j,1}$
4. **O**utput: $(GC,\mathbf{Z}) = (\widehat{GC},\widehat{\mathbf{Z}})$ after the transformations above.

$\texttt{Yao}_2$: On input GC, and input x_i (where b is the concatenation of x_i with the previous inputs $x_1, x_2\ldots x_i$) proceed as follows:

1. Observe that the keys Z_{j,b_j} for every $j \in [(i-1)k]$ have already been obtained for evaluations $F_1,\ldots F_{i-1}$. Obtain the keys Z_{j,b_j} for every $j \in (i-1)k,\ldots ik$.
2. For each $j \in [ik]$, extract the values $\widehat{Z}_{j,b_j}$ and the sub-mask $s_{i,j}$ from Z_{j,b_j}.
3. Compute the mask $m_i = \oplus_{j=1}^{ik} s_{i,j}$ and let $O_i = \widehat{\texttt{Yao}_2}(GC, \{\widehat{\mathbf{Z}}_{j,b_j}\}_{j\in\{1\ldots ik\}})$.
4. **O**utput: Output $m_i \oplus O_i$.

Figure 7.1: Garbled Circuits: malicious case

Lemma 1. ***[Garbled circuits secure against malicious adversaries]*** *Assuming garbled circuit secure against honest-but-curious adversaries exist, there exists a garbled circuit construction that is secure against malicious adversaries (Definition 11).*

Proof. In order to argue the above lemma we need to argue that for every adversary $\mathcal{A}$ (that obtains its keys *adaptively* from the oracle $\mathcal{K}$) we construct a simulator YaoSim that can simulate its view interacting directly with the functionality $\mathcal{F}$. Note that the adversary $\mathcal{A}$ expects to receive as input a garbled circuit. Our simulator YaoSim starts by generating a garbled circuit that just outputs random values $m_1, m_2 \ldots m_n$. Now our simulator YaoSim needs to simulate the key oracle $\mathcal{K}$ for the adversary $\mathcal{A}$ in a way such that $\mathcal{A}$ obtains the correct output values from the evaluation when unmasked with the sub-masks. We will describe our strategy in ensuring that the correct output F_i is obtained. Every $i \in [n]$ can be handled in an analogous manner. For the adversary $\mathcal{A}$ to be able to evaluate F_i, the adversary $\mathcal{A}$ needs to obtain all the keys $\{Z_{j,b_j}\}_{j\in[ik]}$ where b is the concatenation of $x_1, x_2 \ldots x_i$. Furthermore each key Z_{j,b_j} contains a sub-mask $s_{i,j}$ that is needed to evaluate F_i. All these masks are set randomly until the last key is queried. Let Z_{t,b_t} be the last key among $\{Z_{j,b_j}\}_{j\in[ik]}$ that the adversary $\mathcal{A}$ queries to $\mathcal{K}$. Note that the entire input (i.e., $x_1, \ldots x_i$) of the adversary gets specified once it has made all these queries. Our simulator YaoSim, when queried with the last key Z_{t,b_t} obtains the appropriate output for F_i from the ideal functionality $\mathcal{F}$. It sets the sub-mask $s_{i,t} = \oplus_{j=1,j\neq t}^{ik} s_{i,j} \oplus m_i$. This allows our simulator to force the output obtained from the ideal functionality $\mathcal{F}$. □

7.4 Building Blocks for our positive results

We review the main cryptographic primitives which are used for our construction of black-box constant-round concurrently secure MPC in the CD model. The notations and discussions on the following primitives are directly adopted or slightly modified from the recent work of Garg et al. [GGJS12].

7.4.1 Statistically Binding String Commitments

Such a protocol can be obtained by executing, in parallel, Naor's statistically binding bit commitment [Nao91] based on an one-way function. In fact, we use a non-interactive perfectly binding string commitment for the simpler exposition.[6] Such a non-interactive bit commitment scheme can be constructed from any 1-to-1 one-way function. We denote a non-interactive perfectly binding string commitment to string x using random coin r by $\text{PB}\textsc{com}(x;r)$. In the following, we slightly abuse the notation and let $\text{PB}\textsc{com}(x)$ denote the commitment to string x by taking random coin r to be implicit.

7.4.2 Statistically Hiding Commitments

Such a commitment scheme is implied by the existence of one-way functions due to the recent work by Haitner and Reingold [HR07]. In this work, we use the two round statistically hiding commitments [HM96b] and denote it by $\mathsf{SHCom}(x;r)$ a (statistically hiding) commitment to a string x using random coins r. We abuse notation and sometimes skip the random coins for notational convenience.

7.4.3 Constant Round Public-Coin Non-Malleable Commitments

For the construction of our constant-round concurrently secure MPC in the CD model, we use a constant-round public-coin non-malleable commitment, denoted by NMCom. Such a protocol exists due to Pass and Rosen [PR05a]. In addition, Garg at al. [GGJS12] showed the transformation of Goyal's constant-round non-malleable commitment [Goy11b] into a public-coin constant-round non-malleable commitment.

7.4.4 Extractable Commitment Scheme

We use a challenge-response based extractable statistically-binding string commitment scheme $\langle C, R\rangle$ which has been used in previous works [PRS02, Ros04]. In contrast to [PRS02] where multiple slots of challenge and response were used resulting in the poly-logarithmic round complexity of the protocol, we only require a one-slot protocol, similarly to [Ros04], of constant round complexity.

[6]We emphasize that our construction does not require the underlying commitment scheme to be non-interactive.

Protocol $\langle C, R\rangle$. Let $\text{PB}\textsc{com}(\cdot)$ be a non-interactive perfectly binding string commitment scheme described in Appendix 7.4.1. Let n be the security parameter. The extractable statistically-binding string commitment scheme $\langle C, R\rangle$ is defined as follows.

- COMMIT PHASE:

1. To commit to a string $\texttt{str}$, committer C chooses $k = \omega(\log(n))$ pairs of independent random strings $\{\alpha_i^0, \alpha_i^1\}_{i=1}^k$ such that $\texttt{str} = \alpha_i^0 \oplus \alpha_i^1$ for $\forall i \in [k]$. Then, it commits to $\texttt{str}$, α_i^0 and α_i^1 for every $i \in [k]$ by using the non-interactive perfectly binding commitment $\text{PB}\textsc{com}$ and sends those $2k + 1$ commitments to the receiver R. Let B, A_i^0, and A_i^1 respectively denote $\text{PB}\textsc{com}(\texttt{str})$, $\text{PB}\textsc{com}(\alpha_i^0)$, and $\text{PB}\textsc{com}(\alpha_i^1)$ for every $i \in [k]$.

2. Receiver R chooses and sends k independent random bits $b_1, \ldots, b_n$ to committer C.

3. For every $i \in [k]$, if $b_i = 0$, committer C opens A_i^0, otherwise it opens A_i^1 to R by sending the corresponding decommitment.

- OPEN PHASE: Committer C opens all the commitments by sending the decommitment to the receiver R; C sends R all the committed strings and the corresponding random coins used to commit for every commitment.

7.4.5 Constant-Round Strong Witness-Indistinguishable Proof

We use a constant-round strong witness-indistinguishable (strong WI) proof system for every language in **NP**, denoted by $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}}\rangle$. Intuitively, a strong WI proof system guarantees that the distribution of two proofs (each with a distinct witness) are computationally indistinguishable if the theorem statements are computationally indistinguishable. For more details, we refer the readers to [Gol01, Gol04].

7.4.6 Special Non-Malleable Witness-Indistinguishable Argument of Knowledge

In the construction of a constant-round concurrently secure MPC in the CD model, we use a special (constant-round) non-malleable zero-knowledge argument of knowledge (NMWIAok) denoted by $\langle P', V' \rangle$ for every language in **NP**.

In the following, let P' and V' be the prover and the verifier respectively. Let L be any NP-language and let R_L be the NP-relation of L. Let $g : \{0,1\}^n \to \{0,1\}^{3n}$ be a length-tripling pseudo-random generator. Then, we define $\mathcal{R}_2$ to be a NP-relation where the NP theorem is a string mvk and the witness is a tuple $(\tilde{pk}, \tilde{sk}, \tilde{\sigma})$ such that $(mvk, \tilde{pk}, \tilde{sk}, \tilde{\sigma}) \in \mathcal{R}_2$ if and only if $\tilde{pk} = g(sk)$ and $\mathsf{Ver}(\tilde{pk}, \tilde{\sigma}, mvk) = 1$.

PROTOCOL $\langle P', V' \rangle$. Both parties P' and V' get x, mvk as common inputs. Additionally the prover P' gets a witness w such that $(x, w) \in R_L$ or $(mvk, w) \in \mathcal{R}_2$.

2. P' commits to 0 by using a statistically-hiding commitment as $c = \mathsf{SHCom}(0)$. In addition, P' proves the knowledge of a valid decommitment to c by using a statistical zero-knowledge argument of knowledge SZKAOK.

3. P' commits to witness w and sends the commitment to V' by using a constant-round public-coin non-malleable commitment NMCom.

4. P' proves the following statement to V' by using SZKAOK:

   (a) *either* the string w committed to in Step 3 is a valid witness to x such that $(x, w) \in R_L$, *or*

   (b) the value w committed to in Step 2 is such that $(mvk, w) \in \mathcal{R}_2$.

The indistinguishability between transcripts with distinct witnesses. Here, we do not provide the formal proof on the computational indistinguishability between the views using distinct witnesses. Instead, the proof is formally provided as a part of computational indistinguishability between hybrids experiments in Section 8.4.

7.4.7 Constant-Round Statistically Witness Indistinguishable Arguments

Additionally, we use a constant-round statistically witness-indistinguishable (SWI) argument system, denoted by $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle$. Such a protocol can be obtained by executing Blum's Hamiltonicity protocol [Blu87] in parallel $\omega(\log n)$ times, where we replace the prover's commitments in the Blum's Hamiltonicity protocol with a constant-round statistically hiding commitment scheme [NY89, HM96b, DPP97].

7.4.8 Semi-Honest Two Party Computation

We also use a constant-round semi-honest two party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ for any functionality $\mathcal{F}$ in the stand-alone setting. The existence of such a protocol follows from the existence of constant-round semi-honest 1-out-of-2 oblivious transfer [Yao86, GMW87, Kil88].

CHAPTER 8

A constant-round concurrently secure MPC protocol in the Cross-Domain model

In this section, we present the positive side of our result by constructing a constant-round concurrently secure MPC protocol in the CD model. Let $\mathcal{F}$ be a well-formed functionality where such a functionality admits a constant-round two-party computation protocol in the semi-honest setting.[1] In fact, for simplicity, we present a constant-round concurrently secure two-party computation protocol in the CD model, denoted by Π, where a party belongs to either of the two domains.

We emphasize that this two-party protocol easily extends to a concurrently secure protocol for any polynomially many parties in CD model of two fixed domains where a party is under either of two domains. Subsequently, our protocol extends to a concurrently secure protocol for any polynomially many parties in CD model of N fixed domains where each party belongs to one of the N domains. We postpone the explanation on these extensions.

8.1 Building Blocks and Notations

Due to the space restrictions, we provide the details of the building blocks for our construction in Section 7.4. Let $g : \{0,1\}^n \to \{0,1\}^{3n}$ be a length tripling pseudo-random generator. Let $\text{PB}\textsc{com}(\cdot)$ denote a non-interactive perfectly binding commitment scheme, and let $\langle C, R\rangle$ denote an one-slot extractable commitment scheme. Furthermore, we will denote a constant round strong WI proof system by $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}}\rangle$ and a special constant-round NMWI argument of knowledge protocol by $\langle P', V'\rangle$. Finally we denote a constant-round SWI argument by $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle$, and a

[1]See [CLOS02] for the notion of well-formed functionality.

constant-round *semi-honest* two-party computation protocol by $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ which securely computes $\mathcal{F}$ as per the standard simulation-based definition of secure computation.

8.2 Construction of the protocol

We now provide the formal construction of concurrently secure two-party computation protocol in the CD model. Some notations and the protocol description closely resemble those of [GGJS12]. Let $\mathbf{F}_{\mathsf{KCA}} = \{\mathbf{F}^1_{\mathsf{KCA}}, \mathbf{F}^2_{\mathsf{KCA}}\}$ be the key certification authority(KCA) functionality with two domains in the CD model, which is a special case of $\mathbf{F}_{\mathsf{KCA}}$ described in Section 8 where $N = 2$. See the formal description in Section 7.2.

Let n be the security parameter. Let P_1 and P_2 be two parties with private inputs x_1 and x_2 respectively. Without loss of generality, let P_1 and P_2 be in the domains $\mathbf{F}^1_{\mathsf{KCA}}$ and $\mathbf{F}^2_{\mathsf{KCA}}$ respectively. Also, P_1 and P_2 have unique identifiers id_1 and id_2 respectively. Protocol $\Pi = \langle P_1, P_2 \rangle$ proceeds as follows. We omit session identifiers for the succinct specification.

I. Key Registration Phase.

1. P_1 samples random strings sk_1^0 and sk_1^1 and sets $\mathsf{pk}_1^0 := g(\mathsf{sk}_1^0)$ and $\mathsf{pk}_1^1 := g(\mathsf{sk}_1^1)$.

2. P_1 registers both public keys pk_1^0 and pk_1^1 by sending $(register, \mathsf{id}_1, \mathsf{pk}_1^0)$ and $(register, \mathsf{id}_1, \mathsf{pk}_1^1)$ to functionality $\mathbf{F}^1_{\mathsf{KCA}}$.[2]

3. P_1 obtains $(\sigma_{\mathsf{pk}_1^0}, mvk_1)$ and $(\sigma_{\mathsf{pk}_1^1}, mvk_1)$ from $\mathbf{F}^1_{\mathsf{KCA}}$ where $\sigma_{\mathsf{pk}_1^0}$ and $\sigma_{\mathsf{pk}_1^1}$ are signatures on pk_1^0 and pk_1^1 respectively where mvk_1 is the respective verification key.

4. P_1 chooses a random bit $b_1 \in \{0, 1\}$ and sets $\mathsf{pk}_1 = \mathsf{pk}_1^{b_1}$ and $\mathsf{sk}_1 = \mathsf{sk}_1^{b_1}$. We now denote the corresponding signature by σ_{pkpo}.

[2]The registration request is not required to be two distinct requests to the functionality. Registering pk_1^0 and pk_1^1 can be viewed as a registering one public key which is a concatenation of two public keys and the functionality simply decomposes it into two strings, signs both and returns them to the party.

5. P_2 acts analogously, registers pk_2^0 and pk_2^1 with $\mathbf{F}_{\mathsf{KCA}}^2$, and obtains $(\sigma_{\mathsf{pk}_2^0}, mvk_2)$ and $(\sigma_{\mathsf{pk}_1}, mvk_2)$. It sets $\mathsf{pk}_2 = \mathsf{pk}_2^{b_2}$ and $\mathsf{sk}_2 = \mathsf{sk}_2^{b_2}$ where b_2 is a random bit. Finally, σ_{pkpt} is the corresponding signature.

II. Trapdoor Creation Phase. Let $\mathcal{R}_1$ be a NP-relation where the NP theorem is a string mvk and the witness is a tuple $(\tilde{pk}, \tilde{sk}, \tilde{\sigma}, \tilde{c})$ such that $(mvk, \tilde{pk}, \tilde{sk}, \tilde{\sigma}, \tilde{c}) \in \mathcal{R}_1$ if and only if $\tilde{c}$ is the commitment to $\tilde{pk}||\tilde{sk}||\tilde{\sigma}$ with respect to protocol $\langle C, R\rangle$, $\tilde{pk} = g(\tilde{sk})$, and $\mathsf{Ver}(\tilde{pk}, \tilde{\sigma}, mvk) = 1$. For convenience, we let $(mvk, t, \tilde{c}) \in \mathcal{R}_1$ if $t = \tilde{pk}||\tilde{sk}||\tilde{\sigma}$ and $(mvk, \tilde{pk}, \tilde{sk}, \tilde{\sigma}, \tilde{c}) \in \mathcal{R}_1$. In addition, let $\mathcal{R}_2$ be a NP-relation where the NP theorem is a string mvk and the witness is a tuple $(\tilde{pk}, \tilde{sk}, \tilde{\sigma})$ such that $(mvk, \tilde{pk}, \tilde{sk}, \tilde{\sigma}) \in \mathcal{R}_2$ if and only if $\tilde{pk} = g(\tilde{sk})$ and $\mathsf{Ver}(\tilde{pk}, \tilde{\sigma}, mvk) = 1$. Similarly, we denote we let $(mvk, t) \in \mathcal{R}_2$ if $t = \tilde{pk}||\tilde{sk}||\tilde{\sigma}$ and $(mvk, \tilde{pk}, \tilde{sk}, \tilde{\sigma}) \in \mathcal{R}_2$. The trapdoor creation phase proceeds as follows.

1. $P_1 \Rightarrow P_2$: P_1 sends a request $(retrieval, \mathsf{id}_2)$ to $\mathbf{F}_{\mathsf{KCA}}^1$ and obtains mvk_2, a verification key from $\mathbf{F}_{\mathsf{KCA}}^1$. Recall that $\mathbf{F}_{\mathsf{KCA}}^1$ obtains mvk_2 by interacting with $\mathbf{F}_{\mathsf{KCA}}^2$. P_2 analogously obtains mvk_1 from $\mathbf{F}_{\mathsf{KCA}}^2$.

2. $P_1 \Rightarrow P_2$: P_1 executes $\langle C, R\rangle$ with P_2, where P_1 commits to $\mathsf{trap}_1 = \mathsf{pk}_1||\mathsf{sk}_1||\sigma_{\mathsf{pk}_1}$. We denote this execution by $\langle C, R\rangle_{1\to 2}^{\mathsf{trap}_1}$ and the commitment by $\tilde{c_1}$. Next P_1 proves to P_2 by using strong WI proof system $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}}\rangle$ with common input mvk_1, the following NP-statement: there exists $(\mathsf{pk}_1, \mathsf{sk}_1, \sigma_{\mathsf{pk}_1})$ where $(mvk_1, \mathsf{pk}_1, \mathsf{sk}_1, \sigma_{\mathsf{pk}_1}) \in \mathcal{R}_1$. If the verifier V in $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}}\rangle_{1\to 2}^{\mathsf{sk}_1}$ aborts, then P_2 aborts. We denote this execution by $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}}\rangle_{1\to 2}^{\mathsf{trap}_1}$.

3. $P_2 \Rightarrow P_1$: P_2 acts analogously in Step 2 and 2 by first committing to $\mathsf{trap}_2 = \mathsf{pk}_2||\mathsf{sk}_2||\sigma_{\mathsf{pk}_2}$ using $\langle C, R\rangle$ and then giving a proof using $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}}\rangle$. We denote this execution by $\langle C, R\rangle_{2\to 1}^{\mathsf{trap}_2}$ and $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}}\rangle_{2\to 1}^{\mathsf{trap}_2}$.

4. $P_1 \Rightarrow P_2$: P_1 commits to bit 0 as $com_1 = \text{PB}\textsc{com}(0)$ and sends com_1 to P_2. Next P_1 and P_2 executes constant-round NMWI argument of knowledge $\langle P', V'\rangle$ in which P_1 and P_2 respectively play as P' and V'. The common inputs for this execution of $\langle P', V'\rangle$ are com_1 and mvk_2. In this execution, P_1 proves to P_2 that com_1 is a commitment to 0 or there exists

a string t such that $(mvk_2, t) \in \mathcal{R}_2$. Honest party P_1's private input is the de-commitment information of com_1.[3] That is, by the execution of $\langle P', V' \rangle$, P_1 proves to P_2 that com_1 is a commitment to bit 0.

5. $P_2 \Rightarrow P_1$: P_2 proceeds symmetrically as does P_1 above. In summary, it generates a commitment com_2 to bit 0 and then proves the same using $\langle P', V' \rangle$.

III. Input Commitment Phase. Let $\mathsf{Enc}_{\mathsf{pk}}(\cdot)$ denote the encryption algorithm of an dense encryption scheme with pseudo-random public keys with public-keys of length ℓ.

1. $P_1 \Leftrightarrow P_2$: P_1 samples a random string $\alpha_1 \in \{0,1\}^\ell$ and sends $c_1' = \text{PB}\textsc{com}(\alpha_1)$ to P_2. Upon receiving c_1', P_2 responds with a random string $\beta_1 \in \{0,1\}^\ell$. At this point, P_1 reveals the value α_1 to P_2 and proves the following NP-statement to P_2 by executing $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}} \rangle$:

   (a) *either* there exists randomness such that c_1' is a commitment to the string α_1,

   (b) *or* com_1 is a commitment to 1.

   Both parties set $\mathsf{pk}_\mathsf{c}^1 = \alpha_1 \oplus \beta_1$ (public key generated using the coin flipping).

2. $P_2 \Leftrightarrow P_1$: P_2 and P_1 proceed symmetrically as above to generate the public key $\mathsf{pk}_\mathsf{c}^2 = \alpha_2 \oplus \beta_2$.

3. $P_1 \Rightarrow P_2$: P_1 samples a random string r_1 of appropriate length which is to be used as randomness to execute semi-honest two-party computation $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$. P_1 computes $y_1 = \mathsf{Enc}_{\mathsf{pk}_\mathsf{c}^2}(x_1 || r_1)$. Then, it sends y_1 to P_2.

4. $P_2 \Rightarrow P_1$: P_2 proceeds symmetrically as does P_1 above. Let x_2 and r_2 be the input and the random string chosen by P_2 to be used in the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$. Let $y_2 = \mathsf{Enc}_{\mathsf{pk}_\mathsf{c}^1}(x_2 || r_2)$ be the cipher-text generated.

[3]Looking ahead the secret key corresponding to the public key pk_2 will allow the simulator to cheat in the simulation.

IV. Secure Computation Phase. In the secure computation phase, parties P_1 and P_2 jointly evaluate the desired functionality $\mathcal{F}$ based on a constant-round semi-honest two-party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$. Party P_1 plays P_1^{sh} while party P_2 plays P_2^{sh}. Note that $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ is secure against semi-honest adversaries. Thus, we require that the coins of participating parties are indeed uniform. Moreover, we require each party to prove the validity of every message it sends to the other party. That is, whenever a party generates and sends a message to the other party, it is required to prove by using $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}} \rangle$ that the message is honestly generated with respect to its input, random coins and the instructions of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$. In the following, let t be the round complexity of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ where each round consists of two messages: w.l.o.g. a message from P_1 followed by a message from P_2. We denote the next message generators of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ simply by P_1^{sh} and P_1^{sh}. We define transcript $T_{1,i}$ (resp., $T_{2,i}$) by the set (or vector) of all the messages (belonging to $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$) which are exchanged between P_1 and P_2 before P_1 (resp., P_2) needs to send the i-th round message of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ for $i \in [t]$. In particular, P_1 obtains the i-th round message, denoted by $\beta_{1,i}$, of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ as it computes $\beta_{1,i} = P_1^{\mathsf{sh}}(T_{1,i}, x_1, r_1'')$. The P_2's i-th message is symmetrically defined as $\beta_{2,i} = P_1^{\mathsf{sh}}(T_{2,i}, x_2, r_2'')$. The formal definition of the secure computation phase is provided as follows.

1. $P_1 \Rightarrow P_2$: P_1 samples a random string r_2' of appropriate length and sends it to P_2.
2. $P_1 \Leftarrow P_2$: P_2 similarly samples a random string r_1' of appropriate length and sends it to P_1.
3. P_1 computes $r_1'' = r_1 \oplus r_1'$ and P_2 computes $r_2'' = r_2 \oplus r_2'$. Then, r_1'' and r_2'' are the random coins to be used respectively by P_1 and P_2 in the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$.
4. For $i \in [t]$, parties P_1 and P_2 repeats the following procedure.
   (a) $P_1 \Rightarrow P_2$: P_1 computes $\beta_{1,i} = P_1^{\mathsf{sh}}(T_{1,i}, x_1, r_1'')$ and send it to P_2.
   (b) $P_1 \Rightarrow P_2$: P_1 proves to P_2 by using $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}} \rangle$, the NP-statement which is a disjunction of the following NP-statements:
      i. There exist values $\hat{x}_1$, $\hat{r}_1$ such that
         A. there exists randomness such that $y_1 = \mathsf{Enc}_{\mathsf{pk}_c^2}(\hat{x}_1 || \hat{r}_1)$

B. and $\beta_{1,i} = P_1^{\mathsf{sh}}(T_{1,i}, \hat{x}_1, \hat{r}_1 \oplus r'_1)$

ii. com_1 is a commitment to bit 1.

(c) $P_2 \Rightarrow P_1 : P_2$ acts symmetrically.

This completes the formal definition of protocol Π. We claim the following.

Theorem 6. *If there exist a constant-round semi-honest OT, an encryption system with dense(pseudo-random) public keys, and a family of collision-resistant hash functions, then there exists a constant-round concurrently secure two-party computation protocol for every well-formed functionality $\mathcal{F}$ in the CD model.*

8.3 Proof of Theorem 6

In this section, we prove Theorem 6 by constructing an Expected Probabilistic Polynomial-Time (EPPT) simulator sim for protocol Π, which satisfies Definition 9. That is, the EPPT simulator sim with a black-box access to the adversary $\mathcal{A}$ simulates the view of adversary which is computationally indistinguishable from the view of adversary interacting with a honest party in the real world execution of Π. Notice that the NP-statement for an instance of SWI (in Step 4b of Secure Computation Phase) is a disjunction of two NP-statements (Statement 4(b)i and Statement 4(b)ii). In the rest of the work, we refer to Statement 4(b)i as *real* theorem while we refer to Statement 4(b)ii as the *trapdoor* theorem. We call the witness corresponding to statement 4(b)i (resp. statement 4(b)ii) as *real* (resp. *trapdoor*) witness. In the following, we first describe the construction of sim in Section 8.3.1. The formal proof on the indistinguishability between views of real execution and simulated execution is provided in Section 8.4.

8.3.1 Construction of simulator sim

Notations. In the following, we denote the honest party and the adversary by H and $\mathcal{A}$ respectively. Also, let $\mathbf{F}^{H}_{\mathsf{KCA}}$ be the domain to which the honest party belong. Similarly, we use $\mathbf{F}^{\mathcal{A}}_{\mathsf{KCA}}$ to denote the domain where the adversary corrupts a party. Without loss of generality, we define our simulator in the case where the honest party (thus, the simulator in the following) sends the first message in the protocol. We omit the other case where the corrupted party sends the first message. Let $m = \mathsf{poly}$ be the running time of the PPT adversary $\mathcal{A}$. And let l be the number of public keys registered by the corrupted party. The running time of $\mathcal{A}$ serves as an upper-bounds on the number of concurrent sessions and also on the number of registered public keys. In the course of simulation, simulator sim maintains two sets denoted by `Database1` and `Database2`. `Database1` contains an element of the form $(\mathsf{pk}, \mathsf{sk}, \sigma_{pk})$ for $i \in [l]$. `Database2` contains elements of the form $(\mathsf{sid}, x_i^{\mathsf{sid}}, r_i^{\mathsf{sid}})$ where $\mathsf{sid} \in [m]$ and $i \in [l]$. Initially, `Database1` and `Database2` are set to be empty. We sometimes omit the session identifier sid in order to simplify notations.

We preserve the notations for the execution of building blocks as in Section 8.2. For example, we denote by $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle_{\mathsf{sim} \to \mathcal{A}}$, an instance of $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle$ where simulator sim and corrupted party $\mathcal{A}$ play as the prover P and the verifier V respectively in the execution of the protocol $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle$. We demarcate the following two *special* messages in the protocol Π:

- **Message Σ_1^{sid}**: Σ_1^{sid} denotes the second message of $\langle C, R \rangle_{\mathcal{A} \to \mathsf{sim}}^{\mathsf{trap}_{\mathcal{A}}}$ in session sid. Recall that the second message of the protocol $\langle C, R \rangle$ is a random string (challenge) from the receiver to the committer. In the execution of $\langle C, R \rangle_{\mathcal{A} \to \mathsf{sim}}^{\mathsf{trap}_{\mathcal{A}}}$, this message is sent by the simulator (on behalf of H) to the adversary $\mathcal{A}$.

- **Message Σ_2^{sid}**: Σ_2^{sid} denotes the message of session sid when the simulator (on behalf of the honest party H) sends the commitment to 0 using the commitment scheme PBCOM. The simulator will behave honestly until this point and will cheat only after this point is reached.

Description of sim. We provide the simulation strategy of sim in each phase of Π as follows.

I. SIMULATION OF KEY REGISTRATION PHASE: In the key registration phase, simulator sim follows an honest party's strategy. That is, sim interacting with $\mathbf{F}_{\mathsf{KCA}}^{H}$ registers public keys $\mathsf{pk}_{\mathsf{sim}}^{0}$ and $\mathsf{pk}_{\mathsf{sim}}^{1}$ (on behalf of the honest party H) where $(\mathsf{pk}_{\mathsf{sim}}^{0}, \mathsf{sk}_{\mathsf{sim}}^{0})$ and $(\mathsf{pk}_{\mathsf{sim}}^{1}, \mathsf{sk}_{\mathsf{sim}}^{1})$ are obtained as in the honest setting. Finally, sim completes the simulation of key registration phase by setting $\mathsf{pk}_{\mathsf{sim}}$, $\mathsf{sk}_{\mathsf{sim}}$, and $\sigma_{\mathsf{pk}_{\mathsf{sim}}}$ following the honest strategy.

II. SIMULATION OF TRAPDOOR CREATION PHASE: Simulator sim behaves according to the honest party strategy until it needs to send the Σ_2^{sid} for some session session $\mathsf{sid} \in [m]$. At this point, sim by interacting with $\mathbf{F}_{\mathsf{KCA}}^{H}$ obtained a verification key $mvk_{\mathcal{A}}$ of $\mathbf{F}_{\mathsf{KCA}}^{\mathcal{A}}$. To successfully simulate trapdoor creation phase, sim wants to do the following:

1. For all sessions, sim commits to 1 (recall that this differs from the real execution in the fact that honest party commit to 0) by executing $com_{\mathsf{sim}} = \text{PBCOM}(1)$ and then sends com_{sim} in to the adversary.

2. For all sessions, sim proves to $\mathcal{A}$ by executing $\langle P', V' \rangle_{\mathsf{sim} \to \mathcal{A}}$ using a trapdoor information $\mathsf{trap}_{\mathcal{A}}$ (stored in the database `Database1`) as its witness that $(mvk_{\mathcal{A}}, \mathsf{trap}_{\mathcal{A}}) \in \mathcal{R}_2$.

Thus, before sending the commitment to 1, sim checks if `Database1` contains $\mathsf{trap}_{\mathcal{A}}$ such that $(mvk_{\mathcal{A}}, \mathsf{trap}_{\mathcal{A}}) \in \mathcal{R}_2$. If so, then sim proceeds as above. Otherwise, sim employs a rewinding strategy to extract the trapdoor information. Note that session sid (called *target session*) is the session in which the simulator needs to send the commitment to bit 1 using the commitment scheme PBCOM without the corresponding trapdoor information in `Database1`. We will denote this session by sid^{target}. When this point is reached, our simulator sim executes the following look-ahead thread strategy.[4]

1. sim rewinds adversary $\mathcal{A}$ back to the point before sim had sent $\Sigma_1^{\mathsf{sid}^{target}}$ to $\mathcal{A}$.

2. In the look-ahead thread, the simulator sim sends to $\mathcal{A}$ a fresh random challenge for the message $\Sigma_1^{\mathsf{sid}^{target}}$ and behaves honestly subsequently. If in this look-ahead thread, the first session in which the simulator needs to send Σ_2^{sid} is not the target session (in other words $\mathsf{sid} \neq \mathsf{sid}^{target}$), then sim rewinds again and repeats this step. If the number of rewindings reaches 2^n, then sim aborts completely and outputs `Rewind Abort`.

3. Since sim need to send $\Sigma_2^{\mathsf{sid}^{target}}$ in both the main thread and the rewound thread, it must have obtained two distinct *valid* de-commitments[5] of $\langle C, R \rangle_{\mathcal{A} \to \mathsf{sim}}^{\mathsf{trap}_{\mathcal{A}}}$ in the target session sid^{target}

[4]Note that the transcript generated by the execution of look-ahead threads will not be included in the view of the main thread simulation.

[5]We prove this as Lemma 8 in Section 8.4.2.

in both the main thread and the look-ahead threads. At this point, using two distinct valid de-commitments, sim obtains $\mathsf{trap}_{\mathcal{A}}$ (see Section 7.4.4). sim executes the rest of the main concurrent execution with the updated `Database1`. Notice that a single successful extraction of $\mathsf{trap}_{\mathcal{A}}$ in one session suffices to simulate all other sessions.

III. SIMULATION OF INPUT COMMITMENT PHASE.

1. The simulator behaves honestly in the generation of the public key $\mathsf{pk}_c^{\mathcal{A}}$.

2. Now, we describe simulation strategy in generation of the public key $\mathsf{pk}_c^{\mathsf{sim}}$. sim starts by generating a a fresh public key $\mathsf{pk}_c^{\mathsf{sim}}$ along with the secret key $\mathsf{sk}_c^{\mathsf{sim}}$. It generates the commitment c'_{sim} as the commitment to the zero string. Then, sim receives β_{sim} from $\mathcal{A}$. Finally sim opens α_{sim} as $\mathsf{pk}_c^{\mathsf{sim}} \oplus \beta_{\mathsf{sim}}$. sim executes $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}} \rangle_{\mathsf{sim} \to \mathcal{A}}$ where sim uses the trapdoor witness. sim possesses the trapdoor witness since it committed to bit 1 instead of 0 during the simulation of the trapdoor creation phase.

3. sim generates y_{sim} as encryption of the zero string using the public key $\mathsf{pk}_{\mathsf{c}}^{\mathcal{A}}$ and sends it to the adversary. (instead of using its actual input and random coins needed for the semi-honest two-party computation)

4. Upon the receiving $y_{\mathcal{A}}$, the simulator sim extracts the input and randomness $x_{\mathcal{A}}^{\mathsf{sid}}$ and $r_{\mathcal{A}}^{\mathsf{sid}}$ of $\mathcal{A}$ using the secret key $\mathsf{sk}_c^{\mathsf{sim}}$. Now, sim adds $(\mathsf{sid}, x_{\mathcal{A}}^{\mathsf{sid}}, r_{\mathcal{A}}^{\mathsf{sid}})$ to `Database2`.

IV. SIMULATION OF SECURE COMPUTATION PHASE. Let S_{sh} denote the simulator for the semi-honest two-party protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ used in our construction. sim internally runs simulator S_{sh} on adversary $\mathcal{A}_{\mathsf{sh}}$'s input $x_{\mathcal{A}} \in$ `Database2`. S_{sh} at some point makes a call to ideal functionality $\mathcal{F}$ in the ideal world with an input string $x_{\mathcal{A}}$. Then, sim makes a query $(\mathsf{sid}, x_{\mathcal{A}})$ to $\mathcal{F}$. Then, sim forwards the output returned by $\mathcal{F}$ to S_{sh}. At some point of internal simulation of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$, S_{sh} finally halts and outputs a transcript $\beta_{S_{\mathsf{sh}},1}, \beta_{\mathcal{A}_{\mathsf{sh}},1}, \ldots, \beta_{S_{\mathsf{sh}},t}, \beta_{\mathcal{A}_{\mathsf{sh}},t}$ and associated random coin $\hat{r}_{\mathcal{A}}$. sim proceeds with the following instructions.

1. sim computes a random string $\tilde{r}_{\mathcal{A}}$ such that $\tilde{r}_{\mathcal{A}} = r_{\mathcal{A}} \oplus \hat{r}_{\mathcal{A}}$. Then, sim sends $\tilde{r}_{\mathcal{A}}$ to $\mathcal{A}$.

2. For each round $j \in [t]$, sim sends $\beta_{S_{\mathsf{sh}},j}$ to $\mathcal{A}$. Then, sim executes $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle_{\mathsf{sim}\to\mathcal{A}}$ with $\mathcal{A}$ where sim uses the trapdoor witness, decommitment information of com_{sim} (commitment to 1 instead of 0). If $\mathcal{A}$ aborts upon $\beta_{S_{\mathsf{sh}},j}$ for some $j \in [t]$, sim outputs a special abort message $\mathtt{ABORT}_1$.

3. Upon receiving $\mathcal{A}$'s next message $\beta_{\mathcal{A},j}$ in the protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$, sim plays the honest verifier in an execution of $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle_{\mathcal{A}\to\mathsf{sim}}$. For any $j \in [t]$, if the j^{th} message $\beta_{\mathcal{A},j}$ sent by adversary $\mathcal{A}$ is not identical to $\beta_{\mathcal{A}_{\mathsf{sh}},j}$ (obtained from the internal execution of S_{sh}) and if $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle_{\mathcal{A}\to\mathsf{sim}}$ on $\beta_{\mathcal{A},j}$ is accepting, then sim aborts and outputs a special abort message $\mathtt{ABORT}_2$.

Finally, the output of simulator sim contains all messages exchanged between the simulator and the adversary including the output of the adversary in the communication of all sessions.

Lemma 2. *Let $\mathcal{F}$ be a well-formed functionality. Let Π be a protocol defined as in Section 8.2 and $\mathcal{A}$ be any PPT concurrent adversary in the CD model. Given a black-box access to $\mathcal{A}$, simulator* sim *described in Section 8.3.1 runs in expected probabilistic polynomial time and satisfies the following: for any inputs $\{\vec{x}_i\}_{i\in[2]}$,*

$$\left\{\mathrm{IDEAL}^{\mathcal{F}}_{\mathsf{sim}}(n, z, \{\vec{x}_i\}_{i\in[2]})\right\}_{n\in\mathbb{N};z\in\{0,1\}^*} \overset{c}{\equiv} \left\{\mathrm{REAL}^{\Pi}_{\mathcal{A}}(n, z, \{\vec{x}_i\}_{i\in[2]})\right\}_{n\in\mathbb{N};z\in\{0,1\}^*}.$$

In Section 8.4, we provide the formal proof of Lemma 2. In particular we prove that sim runs in EPPT time and that the distribution of its output is computationally indistinguishable from the distribution of the adversary's output.

8.4 Computational Indistinguishability Between The Views

In this section, we formally prove Lemma 2. We first define two experiments denoted by H_0 and H_1 as follows.

Experiment $\mathcal{H}_0$: Simulator sim follows the honest party's algorithm as specified in the definition of Π interacting with adversary $\mathcal{A}$. sim obtains honest party's inputs. At the end of the experiment, sim generates the output of the honest party along with the view (including the output) of $\mathcal{A}$. The output of sim in experiment $\mathcal{H}_0$ is identical to the output from the real execution of protocol Π between an honest party and $\mathcal{A}$.

Experiment $\mathcal{H}_7$: Simulator sim simulates the honest parties by employing the simulation strategy explained in Section 8.3.1. For every session controlled by the adversary, the honest party queries the ideal functionality on its input and outputs the response returned by the ideal functionality. The output of this hybrid corresponds to the the output of the honest party and the view of the adversary $\mathcal{A}$.

We additionally consider a series of six intermediate hybrid experiments denoted by $\mathcal{H}_1$, $\mathcal{H}_2$, $\cdots$, $\mathcal{H}_6$. We formally defined these hybrid experiments later in Section 8.4.2. We denote by ν_i the distribution ensemble of experiment $\mathcal{H}_i$ for every i. Thus, Lemma 2 may be re-written in a simpler notation as follows:

Lemma 3. $\nu_0 \overset{c}{\equiv} \nu_7$.

The proof of this lemma requires a careful analysis on the distribution ensembles of the above hybrid experiments, where the proof structure follows those in the previous works such as [BPS06, GJO10, GGJS12].

8.4.1 The Proof of Lemma 3

In this section, we formally prove that ν_0 is computationally indistinguishable from ν_7 by a hybrid argument. We first start with proving general lemmas that we will use throughout.

Soundness Condition. Looking ahead, while proving the indistinguishability of the outputs of our hybrid experiments, we will need to argue that in each session $\mathsf{sid} \in [m]$, the soundness property holds for $\langle P', V' \rangle_{\mathcal{A} \to \mathsf{sim}}$ and that the trapdoor condition is false for each instance of

$\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle_{\mathcal{A}\to\mathsf{sim}}$. In the sequel, we will refer to this as the *soundness condition*. Now consider the instance $\langle P', V'\rangle^{\mathsf{sid}}_{\mathcal{A}\to\mathsf{sim}}$ in session sid. Let $\pi^{\mathsf{sid}}_{\mathcal{A}}$ denote the proof statement[6] for $\langle P', V'\rangle^{\mathsf{sid}}_{\mathcal{A}\to\mathsf{sim}}$, where, informally speaking, $\pi^{\mathsf{sid}}_{\mathcal{A}}$ states that $\mathcal{A}$ committed to bit 0 (earlier in the trapdoor creation phase). Note that the soundness condition "holds" if in each session $\mathsf{sid} \in [m]$, $\mathcal{A}$ commits to a valid witness to the $\pi^{\mathsf{sid}}_{\mathcal{A}}$ in NMCOM inside $\langle P', V'\rangle^{\mathsf{sid}}_{\mathcal{A}\to\mathsf{sim}}$.

To show this, we define m random variables, denoted by $\{\rho^{\mathsf{sid}}_i\}^m_{\mathsf{sid}=1}$ for $i \in [0, 7]$ such that ρ^{sid}_i is the value committed to in the NMCOM inside $\langle P', V'\rangle^{\mathsf{sid}}_{\mathcal{A}\to\mathsf{sim}}$ in hybrid experiment $\mathcal{H}_i$. In the following, we first show that the soundness condition holds in hybrid experiment $\mathcal{H}_0$ (i.e., the real execution). In order to argue this we first prove the following useful fact.

Lemma 4. *In experiment $\mathcal{H}_0$, no PPT adversary $\mathcal{A}$ corrupting the party $P_{\mathcal{A}}$ can output* trap_H *(the trapdoor information of the honest party) with non-negligible probability.*

Notice that trap_H is of the form of $(\mathsf{pk}, \mathsf{sk}, \sigma_{\mathsf{pk}})$ such that $\mathsf{pk} = g(\mathsf{sk})$ where g is a pseudo-random function. Therefore, in the following proof, we focus on an adversary outputting $(\mathsf{pk}, \mathsf{sk})$.

Proof. Towards the contradiction, assume that such a PPT adversary $\mathcal{A}$ exits. We start by arguing for the case when the adversary $\mathcal{A}$ corrupts P_1. In this case, the adversary $\mathcal{A}$ is assumed to output the trapdoor information $\mathsf{trap}^0_2 = (\mathsf{pk}^0_2, \mathsf{sk}^0_2, \sigma_{\mathsf{pk}^0_2})$ or $\mathsf{trap}^1_2 = (\mathsf{pk}^1_2, \mathsf{sk}^1_2, \sigma_{\mathsf{pk}^1_2})$. The other case when the adversary corrupts P_2 follows in an analogous manner. We first observe that the simulator on behalf of the honest party P_2 uses the trapdoor information $\mathsf{trap}^{b_2}_2$ where b_2 is a uniformly chosen bit.

Two cases arise in this setting. Firstly it is possible that the adversary $\mathcal{A}$ always (expect with negligible probability) outputs the same secret key $\mathsf{sk}^{b_2}_2$ as the one used by the simulator on behalf of honest P_2 or the simulator outputs $\mathsf{sk}^{1-b_2}_2$ with a non-negligible probability. If the first case happens with non-negligible probability then we can contradict the strong WI property of the $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}}\rangle$ protocol. On the other hand, if the secret key that the adversary outputs is different from the one

[6]Recall that the argument system $\langle P', V'\rangle$ has a trapdoor condition of the knowledge of the trapdoor information of the other party. We do not consider it as a part of the theorem statement being proven.

used by the simulator with non-negligible probability, then we can use the adversary to break the security of the pseudo-random generator g. The detailed proofs are provided below.

Towards a contradiction, suppose that there exists a PPT adversary $\mathcal{A}$ corrupting P_1 which outputs $\mathsf{sk}_2^{b_2}$ almost always (except with non-negligible probability), where $\mathsf{trap}_2^{b_2}$ is the trapdoor randomly used by the honest party P_2. Then, we construct a PPT witness-distinguishing adversary $\mathcal{A}_{wi}$ which breaks the strong witness-indistinguishability of $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle$ as follows. We will consider a sequence of m hybrids. In the i^{th} hybrid we require that the adversary uses the trapdoor $\mathsf{trap}_2^{1-b_2}$ in the first i sessions while it uses $\mathsf{trap}_2^{b_2}$ in the rest of the sessions. When we change the trapdoor used to $\mathsf{trap}_2^{1-b_2}$ in all the session, relying on the strong WI property, we can claim that the adversary still outputs the secret key $\mathsf{trap}_2^{b_2}$. Subsequently, we can use the argument presented for case two to reach a contradiction.

Consider a challenger C defined as follows: C is provided with the trapdoor trap_2^0 and trap_2^0 and it commits to one of them in $\langle C, R \rangle$ as a committer. It subsequently gives a proof using the $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle$ protocol. Observe that the value committed to in $\langle C, R \rangle$ computationally hiding. In other words the theorem statements for the strong WI are chosen from computationally indistinguishable distributions.

Now we give proof of indistinguishability among hybrids $i-1$ and i. $\mathcal{A}_{wi}$ proceeds as follows. $\mathcal{A}_{wi}$ first samples two trapdoors. It samples a random bit b_2 and executes all sessions using the trapdoor $\mathsf{trap}_2^{b_2}$ except the i^{th} session for which it uses the challenger. Note that if C outputs $\mathsf{trap}_2^{b_2}$, then we are in the first hybrid $i-1$ and otherwise we are in hybrid i.

Towards a contradiction of the second case, suppose that there exists a PPT adversary $\mathcal{A}$ corrupting P_1 which outputs $\mathsf{trap}_2^{1-b_2}$ with non-negligible probability, where $\mathsf{trap}_2^{b_2}$ is the trapdoor picked by the honest party P_2. Then, we construct a PPT adversary $\mathcal{A}_{ow}$ which breaks the security of PRG g as follows. Consider a challenger that provides pk_{ext} such that $\mathsf{pk}_{ext} = g(\mathsf{sk}_{ext})$. Our adversary $\mathcal{A}_{ow}$ internally generates $\mathsf{pk}_2' = g(\mathsf{sk}_2')$, registers the public key to obtain the corresponding signature, and form $\mathsf{trap}_2^{b_2}$. Then, in the concurrent executions, $\mathcal{A}_{ow}$ as honest party P_2 uses trapdoor $\mathsf{trap}_2^{b_2}$ in $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle_{P_2 \to \mathcal{A}}^{\mathsf{sid}}$ for every $\mathsf{sid} \in [m]$ (Note that this is identical to the honest player's strategy) and plays honestly in the rest of the executions. Since $\mathcal{A}$ outputs the trapdoor of public key which is not used in $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle$ with non-negligible probability, $\mathcal{A}_{ow}$ breaks the security of PRG g, which leads to a contradiction. $\square$

Next we claim that the soundness condition holds in hybrid experiment $\mathcal{H}_0$.

Lemma 5. *Let $\langle P', V' \rangle^{\mathsf{sid}}_{\mathcal{A}\to\mathsf{sim}}$ and $\pi^{\mathsf{sid}}_{\mathcal{A}}$ be as described as above. For every session $\mathsf{sid} \in [m]$, if the honest party does not abort session sid before the beginning of the Input Commitment Phase in $\mathcal{H}_0$, then ρ^{sid}_0 is a valid witness to the statement $\pi^{\mathsf{sid}}_{\mathcal{A}}$ except with negligible probability.*

Roughly speaking, Lemma 4 claimed that no adversary can output the secret key of the honest party. This coupled with the knowledge soundness of the statistical zero knowledge argument of knowledge used in $\langle P', V' \rangle$ implies that the theorem statement $\pi^{\mathsf{sid}}_{\mathcal{A}}$ must be true. Again based on the soundness property of the argument of knowledge we can conclude that ρ^{sid}_0 is a valid witness to the statement $\pi^{\mathsf{sid}}_{\mathcal{A}}$ except with negligible probability. For the detailed proof, see [Claim 2.5, [BPS06]].

The following Lemma generalizes the above lemma over the series of hybrids experiments. We provide the proof of the following lemma later in Section 8.4.3 as its proof requires the use of other claims on our hybrids.

Lemma 6. *Let $\langle P', V' \rangle^{\mathsf{sid}}_{\mathcal{A}\to\mathsf{sim}}$ and $\pi^{\mathsf{sid}}_{\mathcal{A}}$ be as described as above. For every $i \in [7]$ and each session $\mathsf{sid} \in [m]$, if the honest party does not abort session sid before the beginning of the Input Commitment Phase in $\mathcal{H}_i$, then ρ^{sid}_i is a valid witness to the statement $\pi^{\mathsf{sid}}_{\mathcal{A}}$, except with negligible probability.*

Public-coin property of NMCOM. We now describe a strategy that we will repeatedly use in our proofs in order to argue that for every session $\ell \in [m]$, the random variable ρ^{ℓ} (i.e., the value committed by $\mathcal{A}$ in the NMCOM inside $\langle P', V' \rangle^{\ell}_{\mathsf{sim}\to\mathcal{A}}$ remains indistinguishable as we change our simulation strategy from one hybrid experiment to another. Intuitively, we will reduce our indistinguishability argument to a specific cryptographic property (that will be clear from context) that holds in a stand-alone setting. Specifically, we will consider a stand-alone machine M^* that runs sim and $\mathcal{A}$ internally. Here we explain how for any session $\ell \in [m]$, M^* can "expose" the NMCOM inside $\langle P', V' \rangle^{\ell}_{\mathsf{sim} \to \mathcal{A}}$ to an external party R (i.e., M^* will send the commitment messages from $\mathcal{A}$ to R and vice-versa, instead of handling them internally). Note that sim will be rewinding $\mathcal{A}$ during the simulation. However, since R is a stand-alone receiver; M^* can use its responses only on a single thread of execution.

In order to deal with this problem, we will use the following strategy. When $\mathcal{A}$ creates the NMCOM inside $\langle P', V' \rangle^{\ell}_{\mathsf{sim} \to \mathcal{A}}$, any message in this NMCOM from $\mathcal{A}$ on the main-thread is forwarded externally to R; the responses from R are forwarded internally to $\mathcal{A}$ on the main-thread. On the other hand, any message in this NMCOM from $\mathcal{A}$ on a look-ahead thread is handled internally; M^* creates a response on its own and sends it internally to $\mathcal{A}$ on that look-ahead thread. We stress that this is possible because NMCOM is a public-coin protocol.

In the sequel, whenever we use the above strategy, we will omit the details of the interaction between M^* and R.

8.4.2 Definition and Indistinguishability of Intermediate Hybrid Experiments

In this section, we provide the formal description of intermediate hybrid experiments and analyze their distribution ensembles. First, we define the hybrid experiment $\mathcal{H}_1$ as follows.

Experiment $\mathcal{H}_1$: In this hybrid, sim behaves identically as in $\mathcal{H}_0$ except using the rewinding strategy described in the description of simulation. More specifically, simulator sim behaves according to the honest party strategy until it needs to send the Σ_2^{sid} for some session session $\mathsf{sid} \in [m]$. In other words, session sid (called *target session*) is the *first* session in which the simulator needs to send the commitment to bit 0 using the commitment scheme PBcom. We will denote this session by sid^{target}. When this point is reached then our simulator sim executes the following rewinding strategy (i.e., the execution of look-ahead threads).

1. sim rewinds adversary $\mathcal{A}$ back to the point before sim had sent $\Sigma_1^{\mathsf{sid}^{target}}$ to $\mathcal{A}$. Recall that the message Σ_1^{sid} (or the second message of $\langle C, R \rangle^{\mathsf{trap}_{\mathcal{A}}}_{\mathcal{A} \to \mathsf{sim}}$ for session sid) sent by the simulator

sim corresponds to a random challenge string.

2. In the rewound execution the simulator sim sends to $\mathcal{A}$ a fresh random challenge for the message $\Sigma_1^{\mathsf{sid}^{target}}$ and behaves honestly subsequently. If in this rewound execution the first session in which the simulator needs to send Σ_2^{sid} is not the target session (in other words $\mathsf{sid} \neq \mathsf{sid}^{target}$) then simulator rewinds again and repeats[7] this step. Otherwise:

3. Since sim needs to send $\Sigma_2^{\mathsf{sid}^{target}}$ in both the main thread and the rewound thread it must have obtained valid de-commitments[8] in the execution of $\langle C, R\rangle^{\mathsf{trap}_\mathcal{A}}_{\mathcal{A}\to\mathsf{sim}}$ in the target session sid^{target} in both the main thread and the look-ahead thread. At this point using the decommitments generated in the main-thread and the look-ahead thread the simulator obtains the secret key $\mathsf{sk}_\mathcal{A}$ corresponding to the public key $\mathsf{pk}_\mathcal{A}$ of the adversary and makes an entry for the same in `Database1`. sim executes the rest of the concurrent execution in the main thread with the updated database `Database1`. (i.e., the continuation of main thread simulation with the updated `Database1`)

Subsequently the simulator follows the honest party strategy. Note that sim does not use the extracted information anywhere in $\mathcal{H}_1$. First in Lemma 7, we prove that the rewinding strategy halts in the expected probabilistic polynomial time. Secondly in Lemma 8, we will prove that following this strategy the database `Database1` is populated with the secret key of the adversary before Σ_2^{sid} is sent for any session $\mathsf{sid} \in [m]$. Here we will also prove that these keys are indeed valid keys.

Lemma 7. *For any PPT adversary $\mathcal{A}$, simulator* sim *of experiment $\mathcal{H}_1$ halts in expected probabilistic polynomial time of security parameter n.*

Proof. To prove the lemma, we give an analysis similar to the one in Rosen [Ros04]. Let $\mathcal{A}$ be any PPT adversary (given as a black-box) of running time m. Then, sim starts the simulation interacting with $\mathcal{A}$ by fixing some random coins r for $\mathcal{A}$. At some point of the simulation, sim reaches a point where it needs to to send Σ_2^{sid} for some session $\mathsf{sid} \in [m]$ in the main thread. We

[7]If the number of rewindings reaches 2^n, then sim aborts completely and outputs `Rewind Abort`.
[8]We will prove this fact in Lemma 8.

refer to this session as the target session (denoted by sid^{target}). In other words sid^{target} is the first session in which sim needs to send the commitment to 0 using PBCOM in the trapdoor creation phase. We define the event $\mathsf{Reach}(\mathsf{sid}^{\mathsf{target}})$ to happen when sid^{target} is the first session in which sim needs to send the commitment to 0 using PBCOM in the trapdoor creation phase.

We denote the sequence of messages exchanged between sim and $\mathcal{A}$ *before* $\Sigma_1^{\mathsf{sid}^{target}}$ by prefix. Then, let $\delta = \delta(r, \mathsf{prefix})$ be the probability that $\mathsf{Reach}(\mathsf{sid}^{\mathsf{target}})$ occurs over sim's random coins (the random coins used after prefix is completed). Note that these random coins include the challenge string used in $\langle C, R\rangle^{\mathsf{sid}^{target}}_{\mathcal{A}\to\mathsf{sim}}$. Further note that sim behaves in an identical manner in both the main thread and every look-ahead thread. Therefore, we have[9] $\Pr[\mathsf{Reach}(\mathsf{sid}^{\mathsf{target}}) = 1] = \delta$ in both the main thread and a look-ahead thread.

Consider one simulator verifier execution. Let $p_1(n), p_2(n)$ and $p_3(n)$ denote the time taken by such the execution from start to the point when Σ_1^{sid} is sent, then to the point when Σ_2^{sid} needs to be sent and and then finally to the completion point. Note that all these are polynomials.

Observe that sim is expected to obtain at least one look-ahead thread such that $\mathsf{Reach}(\mathsf{sid}^{\mathsf{target}}) = 1$ after $\frac{1}{\delta}$ executions of the look-ahead thread. Therefore, the running time of sim in $\mathcal{H}_1$ is:

$$p_1(n)+(1-\delta)\cdot p_2(n)+\delta\cdot\left(p_2(n)+\frac{1}{\delta}\cdot p_2(n)+p_3(n)\right) \le p_1(n)+p_2(n)+p_2(n)+p_3(n) \le \mathsf{poly}.$$

□

Lemma 8. *Let E be the event that* sim *sends Σ_2^{sid} for some session $\mathsf{sid} \in [m]$ however the database* `Database1` *does not contain a valid secret key for the public key of the adversary. We claim that E happens with negligible probability.*

Proof. Observe that the simulator sends Σ_2^{sid} for some session $\mathsf{sid} \in [m]$ only if it has successfully obtained two threads of execution with the adversary such that in each of the two threads it needs to send $\Sigma_2^{\mathsf{sid}^{target}}$ such that both of them correspond to the same session id sid^{target}, which is also

[9]For simplicity of analysis we ignore the negligible probability cases in which the main thread and the look-ahead thread have the same challenge messages for the target session and the case when `Rewind Abort` happens.

the first session in which Σ_2^{sid}. Finally note that these two threads differ in $\Sigma_1^{\mathsf{sid}^{target}}$ for the session sid^{target}. This allows the simulator to evaluate the extract the committed value. Furthermore we claim that the extracted value will indeed correspond to a valid secret key. This follows from the soundness of the $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle$ protocol in the main thread and the look ahead thread. Observe that the commitment $\langle C, R \rangle_{\mathcal{A} \to \mathsf{sim}}^{\mathsf{trap}_{\mathcal{A}}}$ is statistically binding and therefore if the extracted key is not correct with non-negligible probability then we can conclude that the adversary is proving a false theorem and use this adversary to contradict the soundness of the $\langle \mathsf{P}_{\mathsf{rszk}}, V_{\mathsf{rszk}} \rangle$ protocol. □

Finally we claim indistinguishability of hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$.

Lemma 9.

$$\nu_0 \overset{s}{\equiv} \nu_1, \tag{8.1}$$

$$\forall \textsf{\textit{sid}} \in [m] \quad \rho_0^{\textsf{\textit{sid}}} \overset{s}{\equiv} \rho_1^{\textsf{\textit{sid}}}. \tag{8.2}$$

Proof. The only difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is that sim outputs a `Rewind Abort` since the main-thread and the look-ahead threads are identically distributed. By Lemma 7, the probability that the simulator outputs a `Rewind Abort` is negligible in security parameter n. Hence, the lemma follows. □

Experiment $\mathcal{H}_2$: Identical to $\mathcal{H}_1$ except that sim uses the extracted secret key $\mathsf{sk}_{\mathcal{A}} \in$ `Database1` to execute $\langle P', V' \rangle_{\mathsf{sim} \to \mathcal{A}}^{\mathsf{sid}}$ for all sessions $\mathsf{sid} \in [m]$. We now claim that,

Lemma 10.

$$\nu_1 \overset{c}{\equiv} \nu_2, \tag{8.3}$$

$$\forall \textsf{\textit{sid}} \in [m] \quad \rho_1^{\textsf{\textit{sid}}} \overset{c}{\equiv} \rho_2^{\textsf{\textit{sid}}}. \tag{8.4}$$

The proof of the lemma is similar to [BPS06] and is provided in Appendix 8.5.

Experiment $\mathcal{H}_3$: Identical to $\mathcal{H}_2$ except that sim commits to bit 1 instead of 0 in the trapdoor creation phase in all sessions. That is, in Step 4 of trapdoor creation phase in Section 8.2, sim computes $com_{\mathsf{sim}} = \mathrm{PB}\textsc{com}(1)$ instead of $com_{\mathsf{sim}} = \mathrm{PB}\textsc{com}(0)$ in all sessions. We denote by $com^{\mathsf{sid}}_{\mathsf{sim}\to\mathcal{A}}$ the commitment in session sid.

Lemma 11.

$$\begin{aligned} \nu_2 &\overset{c}{\equiv} \nu_3, && (8.5)\\ \forall \mathsf{sid} \in [m] \quad \rho_2^{\mathsf{sid}} &\overset{c}{\equiv} \rho_3^{\mathsf{sid}}. && (8.6) \end{aligned}$$

The proof of the lemma is provided in Appendix 8.5.

Experiment $\mathcal{H}_4$: Identical to $\mathcal{H}_3$ except that sim uses the trapdoor witness (instead of the real witness) in each instance of $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle$ in session sid, denoted by $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}}\rangle^{\mathsf{sid}}_{\mathsf{sim}\to\mathcal{A}}$. The trapdoor witness (the extracted secret key of the adversary $\mathcal{A}$) for the executions of SWI must be stored in `Database1` since it committed to bit 1 instead of bit 0 in session sid.

Lemma 12.

$$\begin{aligned} \nu_3 &\overset{s}{\equiv} \nu_4, && (8.7)\\ \forall \mathsf{sid} \in [m] \quad \rho_3^{\mathsf{sid}} &\overset{c}{\equiv} \rho_4^{\mathsf{sid}}. && (8.8) \end{aligned}$$

The proof of the lemma is provided in Appendix 8.5.

Experiment $\mathcal{H}_5$: Identical to $\mathcal{H}_4$ except that sim for the input commitment phase of all sessions employs the simulation strategy of the Input Commitment Phase described in Section 8.3.1. In particulary:

1. sim generates the commitment c'_{sim} as the commitment to the zero string.

2. Simulator changes its strategy in generation of the public key $\mathsf{pk}_c^{\mathsf{sim}}$. sim starts by generating a a fresh public key $\mathsf{pk}_c^{\mathsf{sim}}$ along with the secret key $\mathsf{sk}_c^{\mathsf{sim}}$. Then, sim receives β_{sim} from $\mathcal{A}$. Finally sim opens α_{sim} as $\mathsf{pk}_c^{\mathsf{sim}} \oplus \beta_{\mathsf{sim}}$.

   Note that sim is hybrid $\mathcal{H}_4$ was already executing $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}} \rangle_{\mathsf{sim} \to \mathcal{A}}$ with the trapdoor witness.

3. sim generates y_{sim} as encryption of the zero string using the public key $\mathsf{pk}_{\mathsf{c}}^{\mathcal{A}}$ and sends it to the adversary. (instead of using its actual input and random coins needed for the semi-honest two-party computation)

4. Upon the receiving $y_{\mathcal{A}}$, the simulator sim extracts the input and randomness $x_{\mathcal{A}}^{\mathsf{sid}}$ and $r_{\mathcal{A}}^{\mathsf{sid}}$ of $\mathcal{A}$ using the secret key $\mathsf{sk}_c^{\mathsf{sim}}$. Now, sim adds $(\mathsf{sid}, x_{\mathcal{A}}^{\mathsf{sid}}, r_{\mathcal{A}}^{\mathsf{sid}})$ to `Database2`.

Lemma 13.

$$\nu_4 \overset{c}{\equiv} \nu_5, \tag{8.9}$$

$$\forall \mathit{sid} \in [m] \quad \rho_4^{\mathit{sid}} \overset{c}{\equiv} \rho_5^{\mathit{sid}}. \tag{8.10}$$

The proof of the lemma is provided in Appendix 8.5.

Experiment $\mathcal{H}_6$: Identical to $\mathcal{H}_5$ except that sim simulates the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ in each session sid as follows. That is, sim behaves identically as in $\mathcal{H}_5$ and additionally executes the simulation strategy for the Secure Computation Phase IV in Section 8.3.1. Let S_{sh} denote the PPT simulator for the underlying semi-honest two-party secure computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$. sim internally executes S_{sh} on $\mathcal{A}$'s input in `Database2` in session sid. When S_{sh} makes a query to the trusted party with some inputs, sim queries the ideal functionality with this input for session sid. sim forwards the response from the trusted party to S_{sh}. At some point, S_{sh} halts and outputs a transcript of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ and an associated random string for the adversary.

Then, sim forces $\mathcal{A}$ to use this transcript and random coin in the subsequent execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$. If $\mathcal{A}$ aborts upon any of the messages sent by sim, sim aborts all communication and outputs a special message $\texttt{ABORT}_1$. If $\mathcal{A}$ responds differently than the simulated response while it succeeds to generate the accepting SWI proof corresponding to the message, then sim aborts all communication and outputs $\texttt{ABORT}_2$.

We claim that during the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$, the messages of $\mathcal{A}$ must be consistent with the transcript generated by S_{sh} via the internal simulation of sim except with negligible probability. Notice that *soundness condition* holds except with negligible probability at this point. This means that the trapdoor condition is false for every instance of $\langle P_{\mathsf{swi}}, V_{\mathsf{swi}} \rangle_{\mathcal{A} \to H}$ in any session sid.

Lemma 14.

$$\nu_5 \overset{c}{\equiv} \nu_6, \tag{8.11}$$

$$\forall \mathsf{sid} \in [m] \quad \rho_5^{\mathsf{sid}} \overset{c}{\equiv} \rho_6^{\mathsf{sid}}. \tag{8.12}$$

The proof of the lemma is provided in Appendix 8.5.

8.4.3 Proof of Lemma 6

We consider the amount of increment in probability where $\mathcal{A}$ violates the soundness condition from one hybrid experiment to its successive one. Lemma 5 tells us that the probability is negligible in the first hybrid (the real experiment). We will now argue that the increment is bounded a negligible value for each pair of consecutive hybrids we consider. The argument follows based on the indistinguishability of the random variables ρ_i^{sid} and $\rho_{i+1}^{\mathsf{sid}}$ for every $i \in [0, 6]$ and every $\mathsf{sid} \in [m]$ by Equations 8.2, 8.4, 8.6, 8.8, 8.10 and 8.12.

8.5 Hybrid Indistinguishability Details

The following text is adapted from the proofs in [GGJS12], which in turn was based in part on [GJO10, BPS06].

8.5.1 Proof of Equation 8.3 and 8.4

The proof of this claim follows directly from [BPS06]. However since we do not use any of their formal claims directly for completeness, we provide a sketch of the argument here.

Recall that in $\langle P', V' \rangle_{\mathsf{sim} \to \mathcal{A}}$ both parties P' and V' get x, y as common inputs and the prover P' gets a witness w such that $(x, w) \in R_L$ or $(y, w) \in \mathcal{R}_2$. Our simulator sim performs the following steps in $\mathcal{H}_2$ to simulate $\langle P', V' \rangle_{\mathsf{sim} \to \mathcal{A}}$:

1. sim first commits to $\mathsf{trap}_{\mathcal{A}}$ (instead of a string of all zeros) using the statistically hiding commitment scheme SCOM and follows it up with an honest execution of SZKAOK to prove knowledge of the decommitment.

2. sim commits to an all zeros string (instead of a valid witness to w such that $(x, w) \in R_L$) using the non-malleable commitment scheme NMCOM.

3. Finally, sim proves the following statement using SZKAOK: (a) either the value w committed to in NMCOM earlier is such that $(x, w) \in R_L$, or (b) the value w committed to in SCOM is such that $(y, w) \in \mathcal{R}_2$. Here it uses the witness corresponding to the *second* part of the statement. Below, we will refer to this witness as the *trapdoor* witness, while the witness corresponding to the first part of the statement will be referred to as the *real* witness.

Consider the sequence of sub-hybrids $\mathcal{H}_{1:1}, \mathcal{H}_{1:2}, \dots \mathcal{H}_{1:m}$ where hybrid $\mathcal{H}_{1:i}$ represents the case in which the simulator sim uses the extracted trapdoor $\mathsf{trap}_{\mathcal{A}} \in \texttt{Database1}$ to execute $\langle P', V' \rangle^{\mathsf{sid}}_{\mathsf{sim} \to \mathcal{A}}$ for the first i sessions and follows the honest party strategy for the rest of the session. In particular, in hybrid $\mathcal{H}_{1:i}$, P' proves to V' that $(x, w) \in R_L$ for the last $m-i$ sessions while it uses the trapdoor $\mathsf{trap}_{\mathcal{A}} \in \texttt{Database1}$ to prove that $(y, w) \in \mathcal{R}_2$ in the first i sessions. Additionally note that the hybrid $\mathcal{H}_{1:m}$ is same as hybrid $\mathcal{H}_2$. Now, to prove equations 8.3 and 8.4, we will create three intermediate hybrids $\mathcal{H}_{1:i:1}$, $\mathcal{H}_{1:i:2}$, and $\mathcal{H}_{1:i:3}$. Hybrid $\mathcal{H}_{1:i:1}$ is identical to $\mathcal{H}_{1:i}$, except that it changes its strategy to perform step 1 (as described above). Hybrid $\mathcal{H}_{1:i:2}$ is identical to $\mathcal{H}_{1:i:1}$, except that it changes its strategy to perform step 3. Finally, hybrid $\mathcal{H}_{1:i:3}$ is identical to $\mathcal{H}_{1:i:2}$, except that it changes its strategy to perform step 2. Note that $\mathcal{H}_{1:i:3}$ is identical to $\mathcal{H}_{i:2}$.

We now for all values of $i \in \{1, \ldots, m\}$ claim the following:

$$\nu_{i:1} \overset{s}{\equiv} \nu_{1:i:1} \tag{8.13}$$

$$\forall \ell \in [m] \quad \rho^{\ell}_{i:1} \overset{c}{\equiv} \rho^{\ell}_{1:i:1} \tag{8.14}$$

$$\nu_{1:i:1} \overset{s}{\equiv} \nu_{1:i:2} \tag{8.15}$$

$$\forall \ell \in [m] \quad \rho^{\ell}_{1:i:1} \overset{c}{\equiv} \rho^{\ell}_{1:i:2} \tag{8.16}$$

$$\nu_{1:i:2} \overset{c}{\equiv} \nu_{1:i:3} \tag{8.17}$$

$$\forall \ell \in [m] \quad \rho^{\ell}_{1:i:2} \overset{c}{\equiv} \rho^{\ell}_{1:i:3} \tag{8.18}$$

Note that equation 8.3 follows by combining the results of equations 8.13, 8.15, and 8.17. Similarly, equation 8.4 follows by combining the results of equations 8.14, 8.16, and 8.18. We now prove the above set of equations.

Proving Equations 8.13 and 8.14. We first note that SCOM and SZKAOK can together be viewed as a statistically hiding commitment scheme. Let $\overline{\text{SCOM}}$ denote this new commitment scheme. Then, equation 8.13 simply follows from the (statistical) hiding property of $\overline{\text{SCOM}}$.

In order to prove equation 8.14, let us first assume that the claim is false, i.e., $\exists \ell \in [m]$ such that $\rho^{\ell}_{i:1}$ and $\rho^{\ell}_{1:i:1}$ are distinguishable by a PPT distinguisher D. We will create a standalone machine M^* that is identical to $\mathcal{H}_{i:1}$, except that instead of simply committing to a string of all zeros using $\overline{\text{SCOM}}$, M^* takes this commitment from an external sender C and "forwards" it internally to $\mathcal{A}$. Additionally, M^* "exposes" the NMCOM in $\langle P', V' \rangle^{\ell}_{\mathcal{A} \to \mathsf{sim}}$ to an external receiver R by relying on the public-coin property of NMCOM, as described earlier. Let us describe the interaction between M^* and C in more detail. M^* first sends the trapdoor $\mathsf{trap}_{\mathcal{A}}$ to C. Now, when C starts the execution of $\overline{\text{SCOM}}$ in $\langle P', V' \rangle^{\mathsf{sid}}_{\mathsf{sim} \to \mathcal{A}}$, M^* forwards the messages from C to $\mathcal{A}$; the responses from $\mathcal{A}$ are forwarded externally to C. Note that if C commits to a string of all zeros in the $\overline{\text{SCOM}}$ execution, then the (C, M^*, R) system is identical to $\mathcal{H}_{i:1}$. On the other hand, if C commits to $\mathsf{trap}_{\mathcal{A}}$, then the (C, M^*, R) system is equivalent to $\mathcal{H}_{1:i:1}$. We will now construct a computationally unbounded distinguisher D' that distinguishes between these two executions, thus contradicting the statistically hiding property of $\overline{\text{SCOM}}$. D' simply extracts the value inside the NMCOM received by R and runs D on this input. D' outputs whatever D outputs. By our assumption, D's output must be different in these two experiments; this implies that D' output is different as well, which is a contradiction.

Proving Equations 8.15 and 8.16. Equation 8.15 simply follows due to the statistical witness indistinguishability property of SZKAOK. Equation 8.16 also follows from the same fact; the proof details are almost identical to the proof of equation 8.14 and therefore omitted.

Proving Equations 8.17 and 8.18. Equation 8.17 simply follows from the hiding property of NMCOM. To see this, we can construct a standalone machine M that internally runs sim and $\mathcal{A}$ and outputs the view generated by sim. M is identical to $\mathcal{H}_{1:i:2}$ except that inside $\langle P', V' \rangle^{\mathsf{sid}}_{\mathsf{sim} \to \mathcal{A}}$, instead of simply committing (using NMCOM) to a valid witness, it takes this commitment from an external sender C (who is given the valid witness) and "forwards" it internally to $\mathcal{A}$. If the external sender C honestly commits to the witness, then the (C, M) system is identical to $\mathcal{H}_{1:i:2}$; otherwise if C commits to an all zeros string, then the above system is identical to $\mathcal{H}_{1:i:3}$. Equation 8.17 therefore follows from the hiding property of NMCOM.

In order to prove equation 8.18, we will use the non-malleability property of NMCOM. Let us assume that equation 8.18 is false, i.e., $\exists \ell \in [m]$ such that $\rho^{\ell}_{1:i:2}$ and $\rho^{\ell}_{1:i:3}$ are distinguishable by a PPT machine. We will construct a standalone machine M^* that is identical to the machine M described above, except that it will "expose" the non-malleable commitment inside $\langle P', V' \rangle^{\ell}_{\mathcal{A} \to \mathsf{sim}}$ to an external receiver R by relying on the public-coin property of NMCOM, as described earlier. Now, if C commits to the witness to π_H, then the (C, M^*, R) system is identical to $\mathcal{H}_{1:i:2}$, whereas if C commits to a random string, then the (C, M^*, R) system is identical to $\mathcal{H}_{1:i:3}$. From the non-malleability property of NMCOM, we establish that the value committed by M^* to R must be computationally indistinguishable in both cases.

8.5.2 Proof of Equation 8.5 and 8.6

Equation 8.5 simply follows from the (computationally) hiding property of the commitment scheme c. In order to prove equation 8.5, let us first consider the simpler case where sim changes the committed value only in the *first* instance (in the order of execution) of c in the session with session identifier sid where the honest party plays the role of the prover. Then, by a standard hybrid argument, we can extend this proof for multiple commitments.

In order to prove equation 8.6, we will leverage the hiding property of PBCOM and the extractability property of the non-malleable commitment scheme in NMZK. Let us first assume that equation 8.6 is false, i.e., $\exists \ell \in [m]$ such that ρ_2^ℓ and ρ_3^ℓ are distinguishable by a PPT distinguisher. Note that it cannot be the case that the NMCOM inside $\langle P', V' \rangle^\ell_{\mathcal{A} \to \mathsf{sim}}$ concludes *before* sim sends the non-interactive commitment $com^{\mathsf{sid}}_{\mathsf{sim} \to \mathcal{A}}$ in session sid, since in this case, the execution of NMCOM is independent of $com^{\mathsf{sid}}_{\mathsf{sim} \to \mathcal{A}}$. Now consider the case when the NMCOM inside $\langle P', V' \rangle^\ell_{\mathcal{A} \to \mathsf{sim}}$ concludes *after* sim sends $com^{\mathsf{sid}}_{\mathsf{sim} \to \mathcal{A}}$.

We will create a standalone machine M^* that is identical to $\mathcal{H}_2$, except that instead of committing to bit 0 in $com^{\mathsf{sid}}_{\mathsf{sim} \to \mathcal{A}}$, it takes this commitment from an external sender C and forwards it internally to $\mathcal{A}$. Additionally, it "exposes" the NMCOM inside $\langle P', V' \rangle^\ell_{\mathcal{A} \to \mathsf{sim}}$ to an external receiver R by relying on the public-coin property of NMCOM, as described earlier. Note that if C commits to bit 0 then the (C, M^*, R) system is identical to $\mathcal{H}_2$, otherwise it is identical to $\mathcal{H}_3$. Now, recall that NMCOM is an extractable commitment scheme. Therefore, we now run the extractor (say) E of NMCOM on (C, M_ℓ) system. Note that E will rewind M_ℓ, which in turn may rewind the interaction between C and M_ℓ. However, since PBCOM is a non-interactive commitment scheme, M_ℓ simply re-sends the commitment string received from C to $\mathcal{A}$ internally. Now, if the extracted values are different when C commits to bit 0 as compared to when it commits to bit 1, then we can break the (computationally) hiding property of PBCOM, which is a contradiction.

8.5.3 Proof of Equation 8.7 and 8.8

Equation 8.7 simply follows from the statistical witness indistinguishability of SWI by a standard hybrid argument.

In order to prove equation 8.8, let us first consider the simpler case where sim uses the trapdoor witness only in the *first* instance (in the order of execution) of SWI in the session with session identifier sid where the honest party plays the role of the prover. In this case, we can leverage the "statistical" nature of the witness indistinguishability property of SWI in a similar manner as in the proof of equation 8.14. Then, by a standard hybrid argument, we can extend this proof for multiple SWI.

8.5.4 Proof of Equation 8.9 and 8.10

In order to prove these equations, we will define three intermediate hybrids $\mathcal{H}_{4:1}$, $\mathcal{H}_{4:2}$ and $\mathcal{H}_{4:3}$. Experiment $\mathcal{H}_{4:1}$ is the same as $\mathcal{H}_4$, except that sim commits to the zero string as in Step 1 above. Experiment $\mathcal{H}_{4:2}$ is the same as $\mathcal{H}_{4:1}$, except that sim also generates α_{sim} as described in Step 2 above. Finally Experiment $\mathcal{H}_{4:3}$ is the same as $\mathcal{H}_{4:2}$, except that sim also follows the Step 3 as described above. Therefore, by definition, $\mathcal{H}_{4:3}$ is identical to $\mathcal{H}_5$.

We now claim the following:

$$\nu_4 \overset{c}{\equiv} \nu_{4:1} \tag{8.19}$$

$$\forall \ell \in [m] \quad \rho_4^\ell \overset{c}{\equiv} \rho_{4:1}^\ell \tag{8.20}$$

$$\nu_{4:1} \overset{c}{\equiv} \nu_{4:2} \tag{8.21}$$

$$\forall \ell \in [m] \quad \rho_{4:1}^\ell \overset{c}{\equiv} \rho_{4:2}^\ell \tag{8.22}$$

$$\nu_{4:2} \overset{c}{\equiv} \nu_{4:3} \tag{8.23}$$

$$\forall \ell \in [m] \quad \rho_{4:2}^\ell \overset{c}{\equiv} \rho_{4:3}^\ell \tag{8.24}$$

Note that equation 8.9 follows by combining the results of equations 8.19, 8.21 and 8.23. Similarly, equation 8.10 follows by combining the results of equations 8.20, 8.22 and 8.24. We now prove the above set of equations.

Proving Equations 8.19 and 8.20. Proof of equations 8.19 and 8.20 follows is a way identical to the proof of equations 8.5 and 8.6.

Proving Equations 8.21 and 8.22. Proof of equations 8.21 and 8.22 follows in a way identical to the proof of equations 8.5 and 8.6 except that we use the pseudo-randomness property of public keys of the encryption scheme instead of the (computational) hiding property of the commitment scheme PBCOM.

Proving Equations 8.23 and 8.24. Let us first consider the simpler case where sim makes the changes only in the *first* instance (in the order of execution) in the session with session identifier sid. Then, by a standard hybrid argument, we can extend this proof for multiple encryptions.

We start by giving the proof for Equation 8.23. Consider a distinguisher D such that it can distinguish between $\mathcal{H}_{4:2}$ and $\mathcal{H}_{4:3}$ with a non-negligible probability. We will use this distinguisher to contradict the semantic security of the encryption scheme or pseudo-randomness property of the public keys. Given D we can construct a distinguisher D' that before sending $\beta_{\mathcal{A}}$ has already extracted $\alpha_{\mathcal{A}}$ by rewinding and still succeeds in distinguishing $\mathcal{H}_{4:2}$ and $\mathcal{H}_{4:3}$ with a non-negligible probability. We will now describe D'. At the point when the adversary expects to receive $\beta_{\mathcal{A}}$ D' starts a look-ahead thread and sends a random string $\beta'_{\mathcal{A}}$.[10] The adversary will respond by opening $\alpha_{\mathcal{A}}$ with a non-negligible probability. In case the adversary does not open our distinguisher D' just generates a random guess. On the other hand if the adversary opens $\alpha_{\mathcal{A}}$ then D' uses D to generate its guess. It is easy to see that our distinguisher D' still distinguishes with a non-negligible probability. Now the indistinguishability follows just like proof of Equation 8.21.

The proof of equation 8.24 additionally relies on the non-malleability property of the NMCOM. First assume that equation 8.24 is false, i.e., $\exists \ell \in [m]$ such that $\rho^{\ell}_{4:2}$ and $\rho^{\ell}_{4:3}$ are distinguishable by a PPT distinguisher D. Then just like in the proof of Equation 8.23 we will use the adversary $\mathcal{A}$ and

[10]We assume that the adversary does open the committed value $\alpha_{\mathcal{A}}$ with a non-negligible probability. In case the probability of opening is negligible then the argument used to prove Equations 8.21 and 8.22 directly extends to our setting.

construct another adversary that extracts the value $\beta_{\mathcal{A}}$ and "exposes" the NMCOM in $\langle P', V' \rangle^{\ell}_{\mathcal{A}\to\mathsf{sim}}$ with a non-negligible probability. Now, we can use an argument similar to the proof of Equation 8.23 along with the extractor E of NMCOM to argue that equation 8.24. We omit the details.

8.5.5 Proof of Equation 8.11 and 8.12

Informally speaking, equation 8.11 follows from the semi-honest security of the two-party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ used in our construction. We now give more details. In order to prove equation 8.11, let us first consider the simpler case where sim uses simulator for the two-party computation protocol only in the *first* instance (in the order of execution) of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ in the session with session identifier sid. Then, by a standard hybrid argument, we can extend this proof for multiple executions of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$.

We will construct a standalone machine M that is identical to $\mathcal{H}_5$, except that instead of engaging in an honest execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ with $\mathcal{A}$ in session sid, it obtains a protocol transcript from an external sender C and forces it on $\mathcal{A}$ in the following manner. M first queries the ideal world trusted party on the extracted input of $\mathcal{A}$ for session sid in the same manner as explained above for sim. Let $x_{\mathcal{A}}^{\mathsf{sid}}$ denote the extracted input of $\mathcal{A}$. Let x_H^{sid} denote the input of the honest party in session sid. Let O be the output that M receives from the trusted party. Now M sends x_H^{sid} along with $x_{\mathcal{A}}^{\mathsf{sid}}$ and O to C and receives from C a transcript for $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ and an associated random string. M forces this transcript and randomness on $\mathcal{A}$ in the same manner as sim does. Now, the following two cases are possible:

1. C computed the transcript and randomness by using *both* the inputs - x_H^{sid} and $x_{\mathcal{A}}^{\mathsf{sid}}$ - along with the output O. In this case, the transcript output by C is a real transcript of an honest execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$.

2. C computed the transcript and randomness by using only adversary's input $x_{\mathcal{A}}^{\mathsf{sid}}$, and the output O. In this case C simply ran the simulator S_{sh} on input $x_{\mathcal{A}}^{\mathsf{sid}}$ and answered its query with O. The transcript output by C in this case is a simulated transcript for $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$.

In the first case, the (C, M) system is identical to $\mathcal{H}_5$, while in the second case, the (C, M) system is identical to $\mathcal{H}_6$. By the (semi-honest) security of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$, we establish that the output of M must be indistinguishable in both the cases, except with negligible probability. This proves equation 8.11.

Proving Equation 8.12. We will leverage the semi-honest security of the two-party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ and the extractability property of the non-malleable commitment scheme in NMZK to prove equation 8.12.

Specifically, we will construct a standalone machine M^* that is identical to M as described above, except that it "exposes" the NMCOM in $\langle P', V' \rangle^{\ell}_{\mathcal{A} \to \mathsf{sim}}$ to an external receiver R by relying on the public-coin property of NMCOM, as described earlier. Note that if C produces a transcript $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ according to case 1 (as described above), then the (C, M^*, R) system is identical to $\mathcal{H}_5$. On the other hand, if C produces a transcript for $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ according to case 2, then the (C, M^*, R) system is identical to $\mathcal{H}_6$. We can now run the extractor E of NMCOM on (C, M^*) system. Note that E will rewind M^*, which in turn may rewind the interaction between C and M^*. However, since this interaction consists of a single message from C, M^* simply re-uses (if necessary) the transcript received from C in order to interact with $\mathcal{A}$ internally. Now, if the extracted values are different in case 1 and case 2, then we can break the semi-honest security of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$, which is a contradiction.

CHAPTER 9

An impossibility result in the Cross-Domain model

9.1 Introduction

In this section we provide strong impossibility results ruling out constructions for secure MPC protocols in the CD model. We heavily rely on the recent works of [AGJ$^+$12, GKOV12] in proving these results. In fact, we show the impossibility result in the simplest case of the CD model: We show that there does not exist a concurrently secure protocol in the CD model of two domains when three domains and three parties exist in the system. Since each party belongs to the distinct domains in the following discussion, we discuss the impossibility result simply focusing on the parties without considering the KCA functionalities.

We start by showing that string Oblivious Transfer (OT) functionality cannot be concurrently and securely realized even in the setting of *static* inputs in the CD model in the setting of *three* parties even against adversaries that corrupt two parties playing the *same role*, i.e. of the sender or the receiver. Next we generalize this impossibility to essentially all functionalities of interest. Finally we extend our impossibility result to the setting of larger number of parties. In particular we show that no n-party protocol in the CD model (for a large class of functionalities, discussed later) can be concurrently secure in the setting of $n + 1$ parties. We use the notation used by [GKOV12] and some of the text has been taken verbatim from there.

9.2 The String OT functionality and our impossibility result

String OT is a two-party functionality between a sender S, with input (m_0, m_1) and a receiver R with input b which allows R to learn m_b without learning anything about m_{1-b}. At the same time the sender S learns nothing about b. More formally string OT functionality $\mathcal{F}_{OT} : (\{0,1\}^{p(k)} \times \{0,1\}^{p(k)}) \times \{0,1\} \rightarrow \{0,1\}^{p(k)}$ is defined as, $\mathcal{F}_{OT}((m_0, m_1), b) = m_b$, where $p(\cdot)$ is any polynomial and only R gets the output.

Note that string OT is a two-party functionality, however, the protocol realizing the string OT functionality can be executed among multiple parties. We consider the setting of three parties and each of the parties belongs to distinct domains. We show that for some polynomial $p(\cdot)$ (to be fixed later), there does not exist a protocol π that concurrently securely realizes the $\mathcal{F}_{OT}$ functionality among these three parties. More specifically we show that there exists an adversary $\mathcal{A}$ who corrupts 2 parties, registers keys on their behalf, starts a polynomial number of sessions (say $\ell(k)$) of the protocol π with the honest (with pre-specified inputs drawn from a particular distribution $\mathcal{D}$) such that no ideal-world adversary whose output is computationally indistinguishable from the output of real-world adversary $\mathcal{A}$ exists. We stress that the parties corrupted by the adversary (we construct) corrupts two parties playing the same role – either the sender S or the receiver R in all the $\ell(k)$ sessions.

Theorem 7. *(impossibility of static input concurrent-secure string OT in CD model) Let π be any protocol which implements*[1] *the $\mathcal{F}_{OT}$ functionality for a particular (to be determined later) polynomial p in the CD model. Then, in the setting of 3 parties (assuming one-way functions exist) there exists a polynomial ℓ and a distribution $\mathcal{D}$ over ℓ-tuple vectors of inputs and an adversarial strategy $\mathcal{A}$, that corrupts 2 parties, such that for every probabilistic polynomial-time simulation strategy* sim, *Definition 9 of concurrent security,* cannot *be satisfied when the inputs of the parties are drawn from $\mathcal{D}$.*

[1]We say that a protocol implements a functionality if the protocol allows two parties to evaluate the desired function. This protocol however may not be secure.

Implications for bounded concurrency. Observe that the attack described in the above proof (in the unbounded concurrent setting) has natural implications in the bounded setting as well. In particular, the number of sessions that our adversary executes, or the "extent" of concurrency used by the adversary in the proof above in order to arrive at a contradiction is bounded by the communication complexity of the protocol. More specifically the adversary needs to make one additional OT call for every bit that the Sender sends in the protocol.

9.3 The Proof of Theorem 7

We start by recalling and building some notation that we will use in our proof. Next we will consider specifics of our setting. This in particular will include the details on the specifications of inputs of all parties. Then we will define formally the strategy of the real-world adversary. Finally, we will argue that the output of this real-world adversary can not be computationally indistinguishable from the output of any ideal-world adversary.

Notation. Let π be a two-party protocol in the CD model between a Sender S with inputs (m_0, m_1) and a Receiver R with a choice bit b. Without loss of generality we assume that S sends both the first and the last message in the protocol. Further assume that S sends exactly n messages in the execution of this protocol π. Therefore R sends $n-1$ messages. For the sake of contradiction lets assume that π concurrently securely realizes the $\mathcal{F}_{OT}$ functionality when executed among three parties. These parties register their public keys before any execution of the protocol π.

Let π' be a protocol between a sender S with inputs (m_0, m_1) and a receiver R with a choice bit b and additional inputs $\tilde{m}$ and w. In the protocol π', S and R first proceed by executing π with inputs (m_0, m_1) and b respectively. At the end of the execution of π, R obtains m_b. R checks to see if $\tilde{m} = m_b$ and sends w to S if this is indeed the case. Otherwise, it just sends $\perp$ to S. Note that R sends n messages in the execution of this protocol π'.

Now, consider the next message function $F(pk, sk, b, m, w, r, M_1, M_2 \ldots M_i)$ where $i \in [n]$ of

R for the protocol π' with public key pk and secret key sk. More specifically, $F_i(pk, sk, b, m, w, r, M_1, M_2 \ldots M_i)$ generates the ith message that an honest R on input (b, m, w) and random coins r would generate corresponding to the execution in which $M_1, M_2 \ldots M_i$ are sent to R for the protocol π'. Let $\mathcal{F}[pk, sk, b, m, w, r]$ be a reactive functionality parameterized by pk, sk, b, m, w, r that can be invoked n times. The ith invocation of $\mathcal{F}[pk, sk, b, m, w, r]$ expects an input M_i and outputs the ith message that R would have sent in π'. The functionality has the values pk, sk, b, m, w, r which are built into the functionality itself. Let $(\mathtt{Yao}_1, \mathtt{Yao}_2)$ be an implementation of the garbled circuit technique as defined in Definition 11. Roughly speaking $\mathtt{Yao}_1$ is an algorithm that takes the description of a reactive functionality as input and outputs a garbled circuit and the associated keys. Note that there are two keys for every input wire. $\mathtt{Yao}_2$ on the other hand takes as input the garbled circuit and a key corresponding to each wire and outputs an evaluation of the garbled circuit. Since, we are in the malicious case the adversary can obtain the keys for input wire adaptively. We deal with this issues using a technique from [GKR08]. We refer the reader to Section 7.3.2 for details.

Our setting. Here, we describe our setting in which we intend to draw our impossibility. Let S, R_1 and R_2 be three parties executing some polynomial invocations of π. Since we are the in the CD model, the three parties are required to register their public keys before any execution of the protocol π.[2] S plays the role of the sender in all these executions and on the other hand R_1 and R_2 act as receivers. Since we are in the *static* input setting we need to specify the inputs of all parties. More specifically we specify the distribution $\mathcal{D}$ according to which the inputs of the parties S, R_1 and R_2 are sampled.

- Sample (pk, sk) as the public-secret key pair corresponding to a receiver of the protocol π.
- Let $m_0, m_1 \leftarrow \{0,1\}^{p(k)}, b \leftarrow \{0,1\}, r \leftarrow \{0,1\}^*, w \leftarrow \{0,1\}^k, (GC, \mathbf{Z}) = \mathtt{Yao}_1(\mathcal{F}[pk, sk, b, m_b, w, r])$. Let $\mu(k)$ be the number of input wires in the garbled circuit GC. Then $\ell(k) = \mu(k) + 1$. Also $\mathbf{Z}$ contains two garbled circuit keys corresponding to each wire. More

[2]For our impossibility result we consider the setting in which there is a synchronization barrier and the adversary is required to register its public key before any execution of the protocol. Our impossibility naturally extends to the setting when there is no synchronization barrier.

specifically it consists of values $\binom{Z_{1,0},Z_{2,0}\ldots Z_{\mu(k),0}}{Z_{1,1},Z_{2,1}\ldots Z_{\mu(k),1}}$ where $Z_{i,0}, Z_{i,1}$ correspond to the keys for the ith input wire. We will distinguish the sessions into two categories. We will refer to one of the sessions as the *main session*. Rest of the sessions are referred to as *additional sessions*.

- Input to Honest Sender: Let (m_0, m_1) be the input of S in the main session. For each $i \in [\mu(k)]$, let $(Z_{i,0}, Z_{i,1})$ be the input of the honest sender in the ith additional session.

- Input to Receivers: Let $\bar{y}$ be a vector of $\ell(k)$ bits all chosen randomly. Set $\bar{y}$ as the input of the both the receivers.

Description of our real-world adversary $\mathcal{A}$. The real-world adversary corrupts the receivers R_1 and R_2 and receives the garbled circuit GC and the public key pk generated in the above described sampling procedure (distribution $\mathcal{D}$) as auxiliary input. Our adversary registers pk as the public key for R_2 and proceeds as follows. It ignores the inputs $\mathcal{D}$ generates for the honest receivers. Let the messages that S sends to R in the main session be $M_1, M_2, \ldots, M_n$ (recall that n is the number of messages S sends to R in the protocol π). Upon receiving M_i where $i \in [n]$ from S, the adversary obtains its response to be sent in the main session by evaluating the garbled circuit GC on input $M_1, M_2, \ldots M_i$. Let B denote the concatenation of $M_1, M_2, \ldots M_i$. In order to achieve this, the adversary needs the keys Z_{j,B_j} where $j \in |B|$. Note that among these some of the keys have previously already been received. $\mathcal{A}$ obtains the ones that have not been obtained previously by initiating multiple, concurrent, OT protocols (on behalf of the receiver R_1) to which, by construction, the sender provides $(Z_{i,0}, Z_{i,1}), 1 \leq i \leq \mu(k)$ as inputs. On obtaining these keys, $\mathcal{A}$ invokes $\texttt{Yao}_2$, and computes the output. For ever $i \in [n-1]$ it responds to S in the main session using the obtained output. Finally for $i = n$ it outputs the obtained value as its output.

Finally, we remark that the real-world adversary described above always (except with negligible probability) succeeds in outputting the value w generated by the sampling procedure $\mathcal{D}$. The pictorial description of our real-world adversary is provided in Figure 9.1.

Adversarial behavior in the ideal world. Recall that we started by assuming that the protocol π is concurrently secure. Therefore there exists an ideal-world adversary, sim that outputs a distribu-

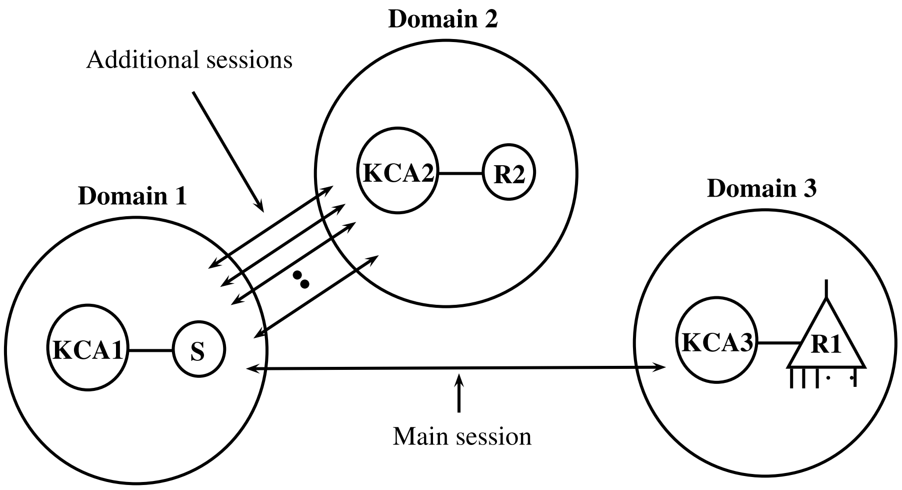


Figure 9.1: The description of real-world adversary $\mathcal{A}$

tion that is computationally indistinguishable from the one that our real world adversary described above generates. Let us recall that sim interacts with the ideal functionality $\mathcal{F}_{OT}$ in a main session and a sequence of additional sessions. The additional sessions however are used just to obtain garbled circuit keys. For notational convenience let $\mathcal{K}$ denote the oracle that is used to obtain garbled circuit keys. More specifically, sim obtains the garbled circuit keys from the key oracle $\mathcal{K}$ rather than the ideal functionality $\mathcal{F}_{OT}$. Next we will convert the simulator sim into an algorithm M that interacts with the $\mathcal{F}_{OT}$ ideal functionality corresponding to the main session only. However, unlike sim, M does not obtain a garbled circuit and it does not query the key oracle $\mathcal{K}$. Instead it interacts with an ideal functionality $\mathcal{F}[pk, sk, b, m_b, w, r]$.

Removing the garbled circuit. Consider an algorithm M that internally executes the simulator sim. According to the security definition of garbled circuits (Definition 11), there exists a simulator $\texttt{YaoSim}_{\mathsf{sim}}$ which interacts with sim in a black-box manner and the ideal function $\mathcal{F}[pk, sk, b, m_b, w, r]$ and outputs a computationally indistinguishable distribution.

Note that the machine M just constructed interacts with $\mathcal{F}[pk, sk, b, m_b, w, r]$ and with $\mathcal{F}_{OT}$ for the main session (in which the honest sender talking to $\mathcal{F}_{OT}$ has input (m_0, m_1)) and is such that its output and the output or the real-world adversary are computationally indistinguishable.

This in particular means that the output of M always (except with negligible probability) includes the value w. However, in order to be able to obtain w from $\mathcal{F}[pk, sk, b, m_b, w, r]$, M must query $\mathcal{F}_{OT}$ (for the main session) with the bit b. Otherwise, M will fail to output w with probability at least $\frac{1}{2}$. In particular this mean that M can be used to extract the choice bit b of an external honest receiver R. This is a contradiction to the stand-alone security of the receiver.

9.4 Extending to all asymmetric functionalities

The goal of this section is to generalize the impossibility result for string OT provided in the previous section to all finite deterministic "non-trivial" asymmetric functionalities $\mathcal{F}$. Consider a two-party functionality $\mathcal{F}_{asym}$ between a sender S, with input x and a receiver R with input y which allows R to learn $f(x, y)$ and at the same time S should not learn anything. More formally, let $f : X \times Y \rightarrow Z$ be any finite function[3] then an asymmetric functionality $\mathcal{F}_{asym}$ is defined as, $\mathcal{F}_{asym}(x, y) = (\perp, f(x, y))$ where S gets no output and R gets $f(x, y)$. We show that there does not exist a protocol π that concurrently securely realizes any *complete* $\mathcal{F}_{asym}$ functionality as defined below.

$\mathcal{F}_{asym}$ is said to be *complete* [Kil00][4] in the setting of stand-alone two-party computation in the presence of *malicious* adversaries iff $\forall b_0, \exists b_1, a_0, a_1$ such that

$$f(a_0, b_0) = f(a_1, b_0) \wedge f(a_0, b_1) \neq f(a_1, b_1).$$

Lemma 15 (Theorems 1 and 3, [AGJ$^+$12]). *Given any protocol ρ that concurrently securely realizes a non-trivial asymmetric functionality $\mathcal{F}$ secure under concurrent self-composition in the static-input, fixed-role setting we have that there exists a protocol Π that securely realizes the $\mathcal{F}_{OT}$ functionality secure under concurrent self-composition in the static-input, fixed-role setting.*

The proof of [AGJ$^+$12] is for the setting of plain model but extends to the setting of the CD model in a direct manner.

[3]Recall that a function is said to be finite if both the domain and the range are of finite size.
[4]Recall that a functionality is said to be complete if it can be used to securely realize any other functionality.

Now any hypothetical protocol for any non-trivial asymmetric functionality $\mathcal{F}$ (using Lemma 15), we will obtain a protocol for $\mathcal{F}_{OT}$, contradicting Theorem 7. This gives our impossibility result for the setting of two parties:

Theorem 8. *(impossibility of static input concurrent security for asymmetric complete functionalities) Let π be any protocol which implements any $\mathcal{F}_{asym}$ functionality that is complete in the stand-alone setting in the CD model. Then, in the setting of 3 parties (assuming one-way functions exist) there exists a polynomial ℓ and a distribution $\mathcal{D}$ over ℓ-tuple vectors of inputs and an adversarial strategy $\mathcal{A}$, that corrupts two parties, such that for every probabilistic polynomial-time simulation strategy* sim, *Definition 9 of concurrent security,* cannot *be satisfied when the inputs of the parties are drawn from $\mathcal{D}$.*

Extending to n-party protocols So far we have only considered the setting of 3-parties only. We now explain how these results can be extended to the setting of $n + 1$ parties executing an n party protocol. Consider an n-party functionality $f(x_1, x_2 \ldots x_n)$ with $x_1, x_2 \ldots x_n$ as input. Let S and $\overline{S}$ be disjoint partitions of the n parties such that only a subset of the parties in $\overline{S}$ get the outputs. Let g be a two-argument function obtained by viewing f as a function of $\{x_i\}_{i\in S}$ and $\{x_i\}_{i\in\overline{S}}$. For any f, if there exist such partitions S and $\overline{S}$ such that g is a complete two-party asymmetric functionality,[5] then we can use our impossibility result for concurrently securely realizing g in the CD model in the setting of 3 parties to argue that f can not be concurrently securely realized in CD model in the setting of $n + 1$ parties. The proof follows in a very similar manner and we omit the details.

[5]Note here this implies that at least one party in S and at least one party in $\overline{S}$ has an input.

References

[AGJ$^+$12] Shweta Agrawal, Vipul Goyal, Abhishek Jain, Manoj Prabhakaran, and Amit Sahai. New impossibility results for concurrent composition and a non-interactive completeness theorem for secure computation. In *CRYPTO*, pages 443–460, 2012.

[APV05] Joël Alwen, Giuseppe Persiano, and Ivan Visconti. Impossibility and feasibility results for zero knowledge with public keys. In *Advances in Cryptology – Crypto '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 135–151. Springer Verlag, 2005.

[Ari11] Seiko Arita. A constant-round resettably-sound resettable zero-knowledge argument in the bpk model. Cryptology ePrint Archive, Report 2011/404, 2011. `http://eprint.iacr.org/`.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[BCL$^+$05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In *CRYPTO*, pages 361–377, 2005.

[BCNP04a] B. Barak, R. Canetti, J.B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, pages 186–195, 2004.

[BCNP04b] Boaz Barak, Ron Canetti, Jesper B. Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *Foundations of Computer Science (FOCS'04)*, pages 394–403, 2004.

[BFGM01] Mihir Bellare, Marc Fischlin, Shafi Goldwasser, and Silvio Micali. Identification protocols secure against reset attacks. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 495–511. Springer, 2001.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001.

[Blu86] Manuel Blum. How to Prove a Theorem So No One Else Can Claim I t. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1986.

[Blu87] Manual Blum. How to prove a theorem so no one else can claim it. In *International Congress of Mathematicians*, pages 1444–1451, 1987.

[BMM99] Amos Beimel, Tal Malkin, and Silvio Micali. The all-or-nothing nature of two-party secure computation. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 1999.

[BPS06] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *FOCS*, pages 345–354, 2006.

[BS05] 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. IEEE Computer Society, 2005.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO*, Lecture Notes in Computer Science, pages 19–40. Springer, 2001.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[CGS08] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. pages 545–562, 2008.

[CKL06] R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[CLP10] 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.

[CPS07] Ran Canetti, Rafael Pass, and Abhi Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *FOCS*, pages 249–259, 2007.

[DFG$^+$11] Yi Deng, Dengguo Feng, Vipul Goyal, Dongdai Lin, Amit Sahai, and Moti Yung. Resettable cryptography in constant rounds - the case of zero knowledge. In *ASIACRYPT*, 2011.

[DGS09] 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. IEEE Computer Society, 2009.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *In 41st FOCS*, pages 283–293. IEEE, 2000.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *Proceedings of the 20th annual ACM symposium on Theory of computing*, STOC '98, pages 409–418. ACM, 1998.

[DNW08] Ivan Damgård, Jesper Buus Nielsen, and Daniel Wichs. Isolated proofs of knowledge and isolated zero knowledge. In *EUROCRYPT*, pages 509–526, 2008.

[DPP97] Ivan Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *J. Cryptology*, 10(3):163–194, 1997.

[DPV04] Giovanni Di Crescenzo, Giuseppe Persiano, and Ivan Visconti. Constant-round resettable zero knowledge with concurrent soundness in the bare public-key model. In *Advances in Cryptology – Crypto '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 237–253. Springer-Verlag, 2004.

[GGJS11] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Bringing people of different beliefs together to do uc. In *TCC*, pages 311–328, 2011.

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In *EUROCRYPT*, pages 99–116, 2012. Full version available at `http://www.cs.ucla.edu/~sanjamg/Research.html/`.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable yao circuits. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2010.

[GJO10] Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Password-authenticated session-key generation on the internet in the plain model. In *CRYPTO*, pages 277–294, 2010.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996.

[GK08] Vipul Goyal and Jonathan Katz. Universally composable multi-party computation with an unreliable common reference string. In *TCC*, pages 142–154, 2008.

[GKOV12] Sanjam Garg, Abishek Kumarasubramanian, Rafail Ostrovsky, and Ivan Visconti. Impossibility results for static input secure computation. *IACR Cryptology ePrint Archive*, 2012:433, 2012.

[GKR08] Shafi Goldwasser, Yael T. Kalai, and Guy. N. Rothblum. One-time programs. In *Advances in Cryptology – CRYPTO'08*, volume 5157, pages 39–56, 2008.

[GL89] Oded Goldreich and Leonid A. Levin. A hard predicate for all one-way functions. pages 20–31, 1989.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GM00] Juan A. Garay and Philip D. MacKenzie. Concurrent oblivious transfer. In *FOCS*, pages 314–324, 2000.

[GM11] Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In *In 52nd Annual IEEE Symposium on Foundations of Computer Science*, FOCS, 2011.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the 17th annual ACM symposium on Theory of computing*, STOC '85, pages 291–304. ACM, 1985.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the 19th annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *CRYPTO*, pages 323–341, 2007.

[Gol01] Oded Goldreich. *Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *Advances in Cryptology – CRYPTO 06*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2006.

[Goy11a] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, pages 695–704, 2011.

[Goy11b] Vipul Goyal. Personal communication., 2011.

[GS09] Vipul Goyal and Amit Sahai. Resettably secure computation. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '09, pages 54–71. Springer, 2009.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 111–129. Springer, 2007.

[HM96a] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *CRYPTO*, pages 201–215, 1996.

[HM96b] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *CRYPTO*, pages 201–215, 1996.

[HR07] Iftach Haitner and Omer Reingold. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC 07, pages 1–10, 2007.

[JORV12] Abhishek Jain, Rafail Ostrovsky, Silas Richelson, and Ivan Visconti. Concurrent zero knowledge in the bounded player model. Cryptology ePrint Archive, Report 2012/279, 2012. `http://eprint.iacr.org/`.

[Kat07] J. Katz. Universally composable multi-party computation using tamper-proof hardware. In *Eurocrypt*, 2007.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. pages 20–31, 1988.

[Kil00] Joe Kilian. More general completeness theorems for secure two-party computation. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 316–324, New York, NY, USA, 2000. ACM.

[KKMO00] Joe Kilian, Eyal Kushilevitz, Silvio Micali, and Rafail Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput.*, 29(4):1189–1208, 2000.

[KMO94] Eyal Kushilevitz, Silvio Micali, and Rafail Ostrovsky. Reducibility and completeness in multi-party private computations. In *FOCS*, pages 478–489, 1994.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 335–354. Springer, 2004.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in polyloalgorithm rounds. In *STOC*, pages 560–569, 2001.

[KPR98] Joe Kilian, Erez Petrank, and Charles Rackoff. Lower bounds for zero knowledge on the internet. In *Proceedings of 39th IEEE Conference on the Foundations of Computer Science*, FOCS '98, pages 484–492, 1998.

[Lin04] Yehuda Lindell. Lower bounds for concurrent self composition. pages 203–222, 2004.

[Lin08] Yehuda Lindell. Lower bounds and impossibility results for concurrent self composition. *J. Cryptology*, 21(2):200–249, 2008.

[LPV09] 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.

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *FOCS*, pages 367–378. IEEE Computer Society, 2006.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In *Advances in Cryptology – Crypto '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 542–565. Springer-Verlag, 2001.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th STOC*, pages 232–241. ACM, 2004.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–413, 2003.

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *In 46th Annual Symposium on Foundations of Computer Science*, FOCS '05, pages 563–572. IEEE Computer Society Press, 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *Proceedings of the 37th annual ACM symposium on Theory of computing*, STOC '05, pages 533–542. ACM, 2005.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *In 43rd FOCS*, pages 366–375, 2002.

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251, 2004.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999.

[Ros04] Alon Rosen. A note on constant-round zero-knowledge proofs for np. In *TCC*, pages 191–202, 2004.

[SV12] Alessandra Scafuro and Ivan Visconti. On round-optimal zero knowledge in the bare public-key model. In *EUROCRYPT*, Lecture Notes in Computer Science. Springer-Verlag, 2012.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167. IEEE, 1986.

[Yil10] Scott Yilek. Resettable public-key encryption: How to encrypt on a virtual machine. In *Topics in Cryptology, CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 41–56. Springer, 2010.

[YZ07] Moti Yung and Yunlei Zhao. Generic and practical resettable zero-knowledge in the bare public-key model. In *EUROCRYPT*, pages 129–147, 2007.