University of California

Los Angeles

Studies in Non-Malleable Commitment

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Chen-Kuei Lee

2013

Abstract of the Dissertation

Studies in Non-Malleable Commitment

by

Chen-Kuei Lee

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2013

Professor Rafail Ostrovsky, Chair

Non-malleable commitment is one of the most fundamental cryptographic primitives in that it is often used as a basic building block in modern cryptography. Briefly, a commitment scheme allows a sender to commit to a value while keeping the value secret from the receiver. In addition, when the commitment is opened in a later stage, the scheme guarantees that the sender can reveal only the unique value being committed to in the previous stage. A commitment scheme is said to be non-malleable if no adversary can succeed in committing to a related massage m', after seeing a commitment of a message m. In this dissertation, we study the construction of non-malleable commitments and some of their applications.

A black-box construction of non-malleable commitment is presented in the first part of this dissertation. Our protocol requires only a constant number of rounds and satisfies the standard notion of non-malleability w.r.t. commitment. Our technique is based on ideas from the "zero-knowledge from secure multi-party computation" paradigm by Ishai, Kushilevitz, Ostrovsky, and Sahai (STOC 2007). Furthermore, our construction only uses one-way functions as a black-box and thus does not depend on the actual implementation details of the functions. Prior to our result, under the standard notion of security, no black-box construction of constant-round non-malleable commitment was known. Hence, our main result closes the gap between black-box and non-black-box constructions for the problem of non-malleable commitment.

In the second part of this dissertation, we show that our technique can be applied to multiple applications. Specifically, we extend our work to obtain a constant-round concurrent non-malleable commitment, a constant-round multi-party parallel coin tossing, and a non-malleable statistically hiding commitment (according to the notion of non-malleability w.r.t. opening.) Additionally, all results mentioned above are based only on a black-box use of one-way functions.

The results presented in this dissertation are adapted from joint work with Vipul Goyal, Rafail Ostrovsky and Ivan Visconti.

The dissertation of Chen-Kuei Lee is approved.

Alexander Sherstov

Amit Sahai

Igor Pak

Rafail Ostrovsky, Committee Chair

University of California, Los Angeles

2013

To my Parents.

Table of Contents

List of Figures

Acknowledgments

I would like to thank my advisor, Dr. Rafail Ostrovsky, for introducing me to research in cryptography and for his guidance during my doctoral studies. Working with him has been a remarkable experience and I have learned a lot from the passion he has shown and the inspiration he has provided me. In addition, I was very lucky that Dr. Ivan Visconti was at UCLA during part of my graduate studies. A special thank goes to Ivan for all of his guidance and advice on everything from research to non-malleability. It has been a privilege to have Rafi and Ivan as my advisor and mentor.

I am also grateful to my co-authors and collaborators: Chongwon Cho, Vipul Goyal, Rafail Ostrovsky and Ivan Visconti. It was a great experience working with them. This thesis would not have been possible without their contributions. In addition, I thank Dr. Igor Pak, Dr. Amit Sahai and Dr. Alexander Sherstov for agreeing to serve on my doctoral committee.

I would also like to thank all of my friends and the members of the crypto group, including, but not limited to, Nishanth Chandran, Sanjam Garg, Ran Gelles, Divya Gupta, Bhavana Kanukurthi, Abishek Kumarasubramaniam, Hemanta Maji, Omkant Pandey, Vanishree Rao, Alan Roytman, Alessandra Scafuro and Akshay Wadia, for contributing to my unforgettable journey at UCLA.

I specially thank Dr. Todd Presner and all my colleagues of HyperCities project: Jih-Chung Fan, Yoh Kawano, David Shepard, Lisa Snyder and Lung-Chih Tung. I would particularly like to thank Todd for his continued support.

Finally, I thank my parents for their love and unconditional support over the years. This dissertation is dedicated to them.

Vita

2002	B.S. (Information and Computer Education) National Taiwan Normal University, Taipei, Taiwan.
2004	M.S. (Computer and Information Science) National Chiao Tung University, Hsinchu, Taiwan.
2007–2013	Ph.D. Student, Computer Science Department, UCLA, Los Angeles, California, USA.

Publications

Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, Ivan Visconti. "*Constructing Non-Malleable Commitments: A Black-Box Approach*". In FOCS 2012 - IEEE 53nd Annual Symposium on Foundations of Computer Science, New Brunswick, NJ, USA, Page 51 – 60.

Chongwon Cho, Chen-Kuei Lee, Rafail Ostrovsky. "*Equivalence of Uniform Key Agreement and Composition Insecurity*". In CRYPTO 2010 - Advances in Cryptology, Santa Barbara, CA, USA, Page 447 – 464.

Chongwon Cho, Chen-Kuei Lee, Rafail Ostrovsky. "*Equivalence of Uniform Key Agreement and Composition Insecurity*". Electronic Colloquium on Computational Complexity (ECCC), Report No. 108 (2009)

CHAPTER 1

Introduction

The notion of non-malleable cryptography was introduced in the seminal work of Dolev, Dwork and Naor [DDN91] and has been widely studied since then. These non-malleable primitives, such as non-malleable commitment and non-malleable zero-knowledge, are powerful tools for dealing with man-in-the-middle attacks in cryptographic protocols. Man-in-the-middle attacks could be of concern either if there is a single protocol execution with multiple parties or when there are several executions running at the same time. For instance, non-malleable commitments have been useful in constructing round-efficient multi-party computation protocols [Bar02, KOS03, Pas04, LP09, Wee10, Goy11]. Many protocols also use non-malleable primitives as crucial technical tools to handle the concurrent setting. There has been a large body of literature on getting concurrent security in the plain model [Pas03, PS04, BS05, MPR06, VV08, OPV08, OPV10, CLP10, CVZ11] and on getting universally composable protocols in various settings [CLOS02, BCNP04, Kat07, LPV09].

After the initial feasibility results by Dolev et. al., a fruitful line of research has focused on efficiency. Round complexity, a natural measure of efficiency has been studied in several works. Barak, in a breakthrough work [Bar02] gave the first constant-round construction of non-malleable commitments using the so called non-black-box simulation techniques [Bar01]. Since then, a number of works have investigated the round complexity of non-malleable protocols. There have been super-constant-round protocols based on one-way functions [LP09, Wee10], constant-round protocols using non-standard or sub-exponential hardness assumptions [PPV08, PW10], and constant-round protocols using non-black-box simulation techniques [Bar02, PR05a, PR05b, OPV09, CVZ10]. Very recently, constant-round constructions based only on one-way functions (OWF) (with black-box simulation techniques) were proposed independently by Goyal [Goy11] and Lin and Pass [LP11]. In all of these works, constructions according the the traditional security notion (of non-malleability w.r.t. commitment) make a non-black-box use of underlying cryptographic primitives.

While round complexity is an important measure of efficiency, a fundamental step in obtaining efficient protocols is to obtain a black-box construction (i.e., one where the underlying cryptographic primitives is used only as an oracle). Construction making use of the underlying primitive in a non-black-box way can typically only be seen as a feasibility result (regardless of the round complexity). To see the difference between black-box and non-black-box constructions, consider the following example (due to Ishai et. al. [IKLP06]).

Suppose that due to major advances in cryptanalytic techniques, all basic cryptographic primitives require a full second of computation on a fast CPU. Non-black-box techniques require parties to prove (e.g., in zero-knowledge), statements that involve the computation of the underlying primitives, say a one-way function. These zero-knowledge protocols, in turn, invoke cryptographic primitives for any gate of a circuit computing a one-way function. Since (by our assumption) a one-way function takes one second to compute, its circuit implementation contains trillions of gates, thereby requiring the protocol trillions of second to run. A black-box construction, on the other hand, would make the number of invocations of the primitive independent of the complexity of implementing the primitive.

1.1 Related Work

Obtaining black-box constructions for various cryptographic primitives has been an active line of research in recent years (c.f., [IKLP06, PW09, Wee10]). However the state of art on constructing non-malleable commitments making a black-box use of cryptographic primitives is far from satisfactory. There have only been results according to new and relaxed notions of security [PW09, Wee10, Goy11].

Pass and Wee [PW09] gave a construction of non-malleable commitments in $O(\log(n))$ rounds making a black-box use of one-way functions. In a concurrent setting, they also gave a $O(n)$ rounds construction using one-way functions as a black-box. However, their construction is according to a relaxed security notion called non-malleability w.r.t. extraction (which they introduce). Wee [Wee10] gave a $O(\log^*(n))$ round construction following the same notion of security. A limited black-box constant-round construction was given by Goyal [Goy11] for an even weaker notion called non-malleability w.r.t. replacement. The construction of Goyal was restricted to providing security only against synchronizing adversaries[1] (as opposed to general adversary). This makes it useful in stand-alone settings only. However in settings where there are more than one (uncoordinated) executions, the construction of Goyal does not provide any security. Both these weaker notions of security have been useful in constructing secure protocols for (stand-alone)

[1]Roughly, this means that the man-in-the-middle adversary $\mathcal{M}$ sends the i-th round message on the right interaction immediately after getting the i-th round message on the left interaction.

multi-party computation in a black-box manner.

In both of these notions, the adversary can indeed correlate (in a limited way) the value it commits to in the right execution to the one in the left execution: in particular, if the value on left is 0, adversary may be able to commit to 0, while if the value on left is 1, adversary commits to $\perp$. Such a situation raises the possibility of selective abort attacks. Even in settings where these notions have been useful, the analysis is more complex than if one were using the standard notion of non-malleability w.r.t. commitment. Using the standard security notion allows us to construct commitment scheme which can be useful in a wider range of settings as well as obtain simpler and cleaner proofs of security.

To summarize, there is sharp contrast in what is known using non-black-box construction (constant-round protocols using only one-way functions) and black-box constructions (no construction known as per the traditional definition) for the problem of non-malleable commitments. This raises the following natural question:

Does there exist a black-box construction of non-malleable commitments following the traditional security notion [DDN91, PR05b, PR08a, LPV08] from any cryptographic assumption with any round complexity?

The main difficult in resolving the above question seems to be in developing a "cut-and-choose" technique having the appropriate coding theoretic properties [PW09, Wee10].

1.2 Our Results

We resolve the above question in the affirmative by providing a black-box construction of non-malleable commitments. Our construction follows the traditional notion of non-malleability w.r.t. commitment [DDN91, PR05b, PR08a, LPV08]. Our construction is additionally optimal in terms of round complexity and cryptographic assumptions. That is, our construction uses only a constant number of rounds and is based only on a black-box use of one-way functions. This completely closes the wide gap between the state of knowledge between black-box and non-black-box constructions for non-malleable commitments. Our construction relies on (and can be seen as an instantiation of) the recent non-malleable commitment scheme of Goyal [Goy11]. Our key technical contribution relates to the construction of a commitment scheme which allows one to prove any arbitrary relation over the committed values in zero-knowledge in a black-box manner (which, in turn, we use in the commitment scheme of Goyal [Goy11]).

Once we obtain such a construction, black-box constructions for several other primitives can be obtained in a natural way. We generalize our construction to get concurrent non-malleable commitments. This construction is constant-round as well and is based on one-way functions. In addition, assuming a broadcast channel, we obtain constant-round multi-party parallel coin-tossing based only on a black-box use of one-way functions. This is a direct improvement over the work of Pass and Wee [PW09] which provided such a construction only for the two-party case[2]. Last, we provide a black-box construction of non-malleable statistically hiding commitments which satisfy the notion of non-malleable w.r.t. opening [PR05a, PR08b][3]. Our construction builds on a stand-alone statistically hiding commitment and converts it into a non-malleable statistically hiding commitment.

[2]For the case of two parties, one does not run into issues of man-in-the-middle attacks.

[3]For statistically hiding commitments, the notion of non-malleability w.r.t. commitment is meaningless as the committed value is not well defined. To analyze security in such a setting, the standard notion of non-malleability is w.r.t. opening as studied for instance by Pass and Rosen [PR05a, PR08b].

This allows us to get a non-malleable statistically hiding commitment in constant rounds based on a black-box use of collision resistant hash functions. Furthermore, one can also have a construction based only on one-way functions in $O(n/\log(n))$-rounds. To our knowledge, this is the first black-box construction of non-malleable statistically hiding commitments.

Along the way we also give several corollaries of independent interest most notably a black-box non-malleability amplification preserving security against general non-synchronizing adversaries. This is an improvement over the analogous result of Wee [Wee10] which required non-black-box access to a one-way function.

1.3 Our Techniques

Traditionally, constructions of non-malleable commitment schemes have relied on executing a basic protocol block somehow several times and then proving consistency among all of them. The proof of consistency typically makes use of underlying cryptographic primitives in a non-black-box way. The question of constructing non-malleable commitments in a black-box way has been raised in a number of previous works [LP09, PW09, Wee10, Goy11]. The main difficulty encountered in previous works is in coming up with a "cut-and-choose" technique having the right properties to replace the zero-knowledge proof of consistency.

Our main technical contribution of this work is a novel way of implementing the zero-knowledge proof of consistency that is typically required in non-malleable commitment protocols. Our technique is based on ideas from the "zero-knowledge from secure multi-party computation" paradigm of Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS09]. In this paradigm, we have a prover who runs a multi-party computation protocol "in his head" and proves the correctness of the result to the verifier. This form of computation in the head approach was proposed in [IKOS09] in the context of improving the communication complexity of zero-knowledge protocols.

However, our goal and the way we use these ideas are somewhat different. The key difference from [IKOS09] is that in our setting, the statement we are proving actually inherently involve a non-black-box use of the commitment scheme: "the evaluation of f on the set of committed values results in $f(S)$". Our technique can also be seen as a way of proving a secret but committed statement (in a way that does not involve the circuit of the commitment scheme in a non-black-box way). We believe our technique might allow us to obtain black-box constructions by eliminating zero-knowledge proofs of consistency in other settings as well.

As a side note, since obtaining a constant-round information theoretically secure multi-party computation is still a major open problem connected to the existence of short locally decodable codes [IK04], we will use a non-constant-round multi-party computation protocol such as BGW [BGW88] in our construction. However our final protocol is still constant-round because this computation only needs to be done "in the head" of the committer.

1.4 Outline

We recall some notations, definitions and basic building blocks in Chapter 2. Additional definitions are given in the sections themselves. We present our construction of black-box non-malleable commitments as follows. In Chapter 3, we start by providing a high-level overview of our technique. Then, in Section 3.2, we give a basic version of our protocol for easy understanding. The security properties of the basic protocol are proven in Section 3.3. Finally, we describe how to obtain a full-fledged protocol in Section 3.4. Using similar techniques described in Chapter 3, we show a constant-round multi-party parallel coin-tossing and a non-malleable statistically hiding commitment in Chapter 4.

CHAPTER 2

Preliminaries

We assume familiarity with some standard cryptographic and theoretical definitions, namely, zero knowledge, interactive proofs, hash functions, probabilistic polynomial-time (PPT) machine, computational indistinguishability, etc. For detailed background on these standard notions and primitives, we refer the reader to [Gol01, KL07]. Nevertheless, for the sake of completeness, we shall recall some crucial building blocks whenever necessary. First, we introduce some notions and notations used throughout our constructions.

2.1 Basic Notations and Tools

Throughout our text, we let $\mathbb{N}$ denote the set of all natural numbers and $[m]$ be the set $\{1, 2, \ldots, m\}$ for any $m \in \mathbb{N}$. Unless stated otherwise, we denote by $k \in \mathbb{N}$ the security parameter and all quantities that are polynomial in k will be denoted by $\texttt{poly}(k)$. For any $x \in \{0,1\}^*$, we denote the length of x (in bits) by $|x|$. For two strings x and y, we denote the concatenation of strings by $x \parallel y$, and the bitwise logical exclusive-or (XOR) of strings by $x \oplus y$. We denote by (A, B) a pair of interactive Turing machines A and B, and denote by $\langle A(x), B(y) \rangle$ the random variable that represents the interaction between two interactive Turing machines A and B, where x and y are their inputs. For example, we denote by

$\tau = \langle A(x), B(y) \rangle$ the interactive execution of (A, B) invoked with inputs x for A, y for B, and producing τ as the transcript of the execution. Next, we recall the formal definitions of some tools and facts from information theory.

2.1.1 Indistinguishability

We begin with some definitions of indistinguishability. Recall that an ensemble of probability distribution is a sequence of indexed random variables. A function ε from non-negative integers to reals is called a negligible function if for every $c > 0$ there exists an N_c such that for all $n > N_c$, $\varepsilon(n) \leq 1/n^c$.

Definition 1 (Computational Indistinguishability) *Two probability ensembles $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable if for every* PPT *algorithm $\mathcal{D}$, there exists a negligible function ε such that for all random coin tosses r and r' of $\mathcal{D}$,*

$$|\Pr[\mathcal{D}_r(X_n) = 1] - \Pr[\mathcal{D}_{r'}(Y_n) = 1]| \leq \varepsilon(n).$$

That is, a PPT distinguisher $\mathcal{D}$ cannot tell apart a sample from X and Y. We use $X \approx Y$ to denote X and Y are computationally indistinguishable.

Definition 2 (Statistical Distance) *For random variables X and Y taking values in $\mathcal{U}$, their statistical difference is defined as*

$$\Delta(X,Y) = \mathtt{max}_{T\subseteq\mathcal{U}}|\Pr[X\in T] \; - \Pr[Y\in T]|.$$

We say that X and Y are ε-close if $\Delta(X,Y)\leq\varepsilon$.

Definition 3 (Statistical Indistinguishability) *Two probability ensembles $X=\{X_n\}_{n\in\mathbb{N}}$ and $Y=\{Y_n\}_{n\in\mathbb{N}}$ are called statistically indistinguishable if*

$$\Delta(X_n,Y_n)\leq\varepsilon(n).$$

We use $X\equiv_s Y$ to denote X and Y are statistically indistinguishable. Finally, two probability ensembles $X=\{X_n\}_{n\in\mathbb{N}}$ and $Y=\{Y_n\}_{n\in\mathbb{N}}$ are called perfectly indistinguishable if for all $n\in\mathbb{N}$, X_n and Y_n have the same distribution. We will use $\equiv$ to denote perfect indistinguishability.

2.1.2 Pairwise Independent Hash Functions

Pairwise independent hash functions are useful tools for constructing interactive proofs. We will use a family of pairwise independent hash functions [CW79, WC81] in our construction. The formal definition is the following.

Definition 4 (Pairwise Independent Hash Functions) *A family of hash functions $\mathcal{H}=\{\,h:\{0,1\}^n\rightarrow\{0,1\}^m\,\}$ is said to be pairwise independent iff $\forall x\neq x'\in\{0,1\}^n$ and $\forall y,y'\in\{0,1\}^m$,*

$$\Pr_{h\leftarrow\mathcal{H}}[h(x)=y\wedge h(x')=y'] = 1/(2^m)^2.$$

In other words, if function h is chosen uniformly at random from $\mathcal{H}$, then for all $x \neq x'$, the random variables $h(x)$ and $h(x')$ will be uniformly distributed and pairwise independent. For example, one can construct a pairwise independent hash family from linear maps as follows.

Construction 1 (Pairwise Independent Hash Functions) *Let $\mathbb{F}$ be a finite field. Then the family of functions $\mathcal{H} = \{h_{a,b} : \mathbb{F} \to \mathbb{F}\}_{a,b \in \mathbb{F}}$ where $h_{a,b}(x) = ax + b$ is pairwise independent.*

2.1.3 Seeded Randomness Extractors

The security of many cryptographic protocols depends on having access to a source of perfect randomness, but sometimes physical sources of randomness does not guarantee an uniform output. Therefore, it is very useful to use a randomness extractor to remove possible biases and correlations from imperfect random sources. Informally, randomness extractor [NZ96, NT99] is a deterministic polynomial-time function that extracts a sequence of almost uniformly random bits from a weak source of randomness. If an extractor also receives a uniformly distributed seed as a second input, it is called Seeded Randomness Extractor. The most common measure of the "quality of randomness" in cryptography is min-entropy. Intuitively, if a distribution has high min-entropy then every outcome should have small probability. We recall the formal definitions as follow.

Definition 5 (Min-Entropy) *Given a distribution X on $\{0,1\}^n$. Then the* min-entropy *of X is defined as*

$$H_\infty(X) = min_x \Big\{ \log \frac{1}{\Pr[X = x]} \Big\}.$$

Definition 6 (Seeded Randomness Extractor) *We say that a function $\texttt{Ext}(r,s) : \{0,1\}^{|r|} \times \{0,1\}^{|s|} \to \{0,1\}^m$ is a (k,ε)-extractor if for every distribution of r with $H_\infty(r) \geq k$ and any uniformly distributed seed s that is independent of r, the output of $\texttt{Ext}(r,s)$ is ε-close to the uniform distribution over $\{0,1\}^m$.*

Definition 7 (Strong Extractor) *An extractor $\texttt{Ext}(r,s) : \{0,1\}^{|r|} \times \{0,1\}^{|s|} \to \{0,1\}^m$ is a strong (k,ε)-extractor if for every distribution of r with $H_\infty(r) \geq k$ and any uniformly distributed seed s that is independent of r, it holds that $\texttt{Ext}'(r,s) = (s \circ \texttt{Ext}(r,s))$ is a standard (k,ε)-extractor.*

That is, a strong extractor guarantees that even when s is given as part of the output, the output distribution is still ε-close to a uniform distribution. This is especially useful in a setting where in addition to the output of extractor, seed s is available to an adversary. Finally, by Leftover Hash Lemma [HILL99], one can explicit construct a strong extractor from pairwise independent hash functions.

Lemma 1 (Leftover Hash Lemma) *if $\mathcal{H} = \{\, h : \{0,1\}^n \to \{0,1\}^m \,\}$ is a pairwise independent family of hash functions of size 2^d where $m = k - 2\log(1/\varepsilon)$, then $\texttt{Ext}(x,h) = h(x)$ is a strong (k,ε)-extractor with seed length d.*

2.2 Build Blocks

In our constructions, we will make use of Naor's statistically binding commitment scheme [Nao91], statistically binding extractable commitment schemes, secure multi-party computation with statistical security and perfect completeness, and a verifiable secret sharing scheme with a deterministic reconstruction phase. We now briefly introduce these cryptographic primitives.

2.2.1 Commitment Schemes

A (bit) commitment scheme is a two-phase protocol between a sender Com and a receiver Rec. In the former phase, called the *commitment phase*, Com commits to a secret bit b to Rec. Let c be the transcript of the interaction. In the later phase, called the *decommitment phase*, Com reveals a bit b' and proves that it is the same as b that was hidden in the transcript c of the commitment phase. Typically, there are two security properties w.r.t. a commitment scheme. The *binding* property requires that after the commitment phase, a malicious sender cannot decommitment c to two different values in the decommitment phase. The *hiding* property guarantees that a malicious receiver learns nothing about b in the commitment phase. A commitment scheme can be either Statistically Binding (but Computationally Hiding) or Statistically Hiding (but Computationally Binding).

Definition 8 (Commitment Scheme) *A (bit) commitment scheme, denoted by* $\mathtt{CS} = (\mathrm{Com}, \mathrm{Rec})$*, is a two-phase protocol consists of a pair of* PPT *Turing machines* Com *and* Rec*. Unless stated otherwise, both* Com *and* Rec *receive the security parameter* $k \in \mathbb{N}$ *as common input.*

- Commit: Com *runs on a private input* $b \in \{0,1\}$ *and a sequence of coin tosses* ω*. The transcript* $c = \langle \mathrm{Com}(b;\omega), \mathrm{Rec}\rangle$ *is obtained after interacting with* Rec. Com *also get a private output* d *after this phase. Without loss of generality,* d *could be the private coin tosses of* Com.

- Decommitment: Com *reveals a bit* b' *and* d*.* Rec *accepts and outputs the value committed to be* b' *if and only if* Com *can convince* Rec *that* $b' = b$*.*

Definition 9 (Security of Commitment Scheme) *In a commitment scheme, the following security properties hold for any* PPT *adversary* $\mathcal{A}$*.*

- Correctness: *if sender and receiver both follow the protocol, then for all* $b \in \{0,1\}$*, when the sender commits and opens to* b*,* Rec *accepts and outputs* b*.*

- Hiding: *let* $\mathsf{dist}_{\mathtt{CS}}^{\mathcal{A}(k,z)}(b)$ *denote the random variable describing the output of a* PPT *adversary* $\mathcal{A}$ *running on security parameter* k *and auxiliary input* z*, with a honest sender committing to a bit* b *by running* $\mathtt{CS}$*. It holds that for every* PPT *adversary* $\mathcal{A}$ *on security parameter* $k \in \mathbb{N}$ *and auxiliary input* $z \in \{0,1\}^*$*, the probability ensembles* $\{\mathsf{dist}_{\mathtt{CS}}^{\mathcal{A}(k,z)}(0)\} \approx \{\mathsf{dist}_{\mathtt{CS}}^{\mathcal{A}(k,z)}(1)\}$*.*

- Binding: *for every* PPT *adversary* $\mathcal{A}$*, and for all but a negligible probability over the random coin tosses of* Rec*, after the commitment phase, the* *probability that $\mathcal{A}$ can successfully open the commitment both as 0 and 1 is negligible.*

Furthermore, a commitment scheme is statistically biding if its binding property is secure against any unbounded adversary $\mathcal{A}$. We will often use schemes with non-interactive opening. That is, the sender opens a commitment by sending the randomness used in the commit phase. We now recall the existing commitment schemes that are relevant for our constructions.

Statistically binding commitment scheme. We will use Naor's statistically binding commitment scheme [Nao91] in our construction. Naor's scheme only requires a black-box use of a pseudo-random generator (which can be based on the black-box use of any one-way function [HILL99]), and we will use Naor's commitment scheme in a black-box manner. Naor's commitment scheme $\texttt{CS} = (\mathrm{Com}, \mathrm{Rec})$ executed by a sender Com and a receiver Rec with the following notation: $c = \mathrm{Com}_\sigma(b; \omega)$ denotes a commitment to a bit b computed using randomness ω, where σ is the first message generated by Rec to construct the commitment. To decommit and verify the commitment, Com sends (b, ω) and Rec verifies that $c = \mathrm{Com}_\sigma(b; \omega)$. We stress that Naor's commitment scheme can be used to commit to strings by iterating it for each bit of the string. Moreover, we will use it with non-interactive opening.

2.2.2 Extractable Commitment Schemes

Informally, a commitment scheme is said to be extractable if there exists an efficient extractor that having black-box access to any efficient malicious sender ExCom^* that successfully performs the commitment phase, is able to efficiently extract the committed string. The formal definition from [PW09] is as follows.

Definition 10 (Extractable Commitment Scheme) *A commitment scheme* $\mathtt{ExCS} = (\mathrm{ExCom}, \mathrm{ExRec})$ *is an extractable commitment scheme if given an oracle access to any* PPT *malicious sender* ExCom^**, committing to a string, there exists an expected* PPT *extractor* $\mathcal{E}$ *that outputs a pair* (τ, σ^*) *such that the following properties hold:*

- Simulatability: *the simulated view* τ *is identically distributed to the view of* ExCom^* *(when interacting with an honest* ExRec*) in the commitment phase.*
- Extractability: *the probability that* τ *is accepting and* σ^* *correspond to* $\perp$ *is negligible. Moreover, the probability that* ExCom^* *opens* τ *to a value different than* σ^* *is negligible.*

An extractable commitment scheme can be statistically binding (in such case also extractability must hold against an unbounded ExCom^*) or statistically hiding. The construction of an extractable commitment in [PW09] follows the one proposed by Rosen in [Ros04], which is a non-concurrent version of the one originally proposed by Prabhakaran et al. in [PRS02], and formally defined as concurrent extractable commitment by Ong et al. in [MOSV06]. Since we do not require concurrent extractability, we will consider the non-concurrent and round-efficient version only.

We briefly recall the extractable commitment scheme from [PW09]. One can construct an extractable commitment scheme $\mathtt{ExCS} = (\mathrm{ExCom}, \mathrm{ExRec})$ with non-interactive opening from any commitment scheme $\mathtt{CS} = (\mathrm{Com}, \mathrm{Rec})$ with non-interactive opening in a black-box manner as follows. Let ExCom be the sender, ExRec be the receiver, and $\mathrm{Com}(\sigma;\omega)$ denote the commitment to a message σ computed using randomness ω. In Figure 2.1, we show the steps of a statistically binding extractable commitment scheme by assuming that if at any time the received message is inconsistent with the protocol specification then the honest player aborts (e.g., the receiver would output $\perp$).

The proof of hiding and binding can be found in [PW09]. The extractor can simply run as a receiver, and if any of the k commitments is not accepting, it outputs $\sigma^* = \perp$. Otherwise, it rewinds (Step 2) and changes the challenge until another k well formed decommitments are obtained. Then it verifies that for each decommitment, the XOR of all pairs corresponds to the same string. Then the extractor can extract a value from the responses of these two distinct challenges. The extractor by playing random challenges in each execution of Step 2 is perfectly simulating the behavior of the receiver and the analysis in [PW09] shows that its running time is polynomial. However, the extractor described above will produces over extraction, which means that the extractor can output a value different from $\perp$ when the transcript has no valid opening.

In our constructions, we will also need an extractable commit scheme without over extraction, but tolerating extraction failure.

Definition 11 (Weakly Extractable Commitment Scheme)
A weakly extractable commitment scheme `WExCS` $= (\mathrm{WExCom}, \mathrm{WExRec})$ *is a commitment scheme such that given oracle access to any* PPT *malicious sender* WExCom^**, committing to a string, there exists an expected* PPT *extractor* `Ext` *that outputs a pair* (τ, σ^*) *such that the following properties hold:*

- Simulatability: *the simulated view* τ *is identically distributed to the view of* WExCom^* *(when interacting with an honest* WExRec*) in the commitment phase.*

Commitment Phase:

1. ExCom on input a message σ, generates k random strings $\{r_i^0\}_{i\in[k]}$ of the same length as σ, and computes $\{r_i^1 = \sigma \oplus r_i^0\}_{i\in[k]}$, therefore $\{\sigma = r_i^0 \oplus r_i^1\}_{i\in[k]}$. Then ExCom uses `CS` to commit to the k pairs $\{(r_i^0, r_i^1)\}_{i\in[k]}$. That is, ExCom and ExRec produce $\{c_i^0 = \langle \mathrm{Com}(r_i^0, \omega_i^0), \mathrm{Rec}\rangle, c_i^1 = \langle \mathrm{Com}(r_i^1, \omega_i^1), \mathrm{Rec}\rangle\}_{i\in[k]}$.

2. ExRec responses to ExCom by sending a random k-bit challenge string $r' = (r'_1, \dots, r'_k)$.

3. ExCom decommits $\{c_i^{r'_i}\}_{i\in[k]}$ (i.e., non-interactively opens k of previous commitments, one per pair).

4. ExRec verifies that commitments have been opened correctly.

Decommitment Phase:

1. ExCom sends σ and non-interactively decommits the other k commitments $\{c_i^{\bar{r}'_i}\}_{i\in[k]}$, where $\bar{r}'_i = 1 - r'_i$.

2. ExRec checks that all k pairs of random strings $\{r_i^0, r_i^1\}_{i\in[k]}$ satisfy $\{\sigma = r_i^0 \oplus r_i^1\}_{i\in[k]}$. If so, ExRec takes the value committed to be σ and $\perp$ otherwise.

Figure 2.1: Extractable Commitment Scheme `ExCS`

Commitment Phase:

1. WExCom on input a message σ, generates a random strings r^0 of the same length as σ, and computes $r^1 = \sigma \oplus r^0$. That is, WExCom sets $\sigma = r^0 \oplus r^1$. Then WExCom uses `CS` to commit to a pair of values (r^0, r^1). That is, WExCom and WExRec produce $c^0 = \langle \mathrm{Com}(r^0, \omega^0), \mathrm{Rec} \rangle, c^1 = \langle \mathrm{Com}(r^1, \omega^1), \mathrm{Rec} \rangle$.
2. WExRec responses to WExCom by sending a random bit challenge string b.
3. WExCom decommits c^b (i.e., non-interactively opens one of previous commitments).
4. WExRec verifies that c^b has been opened correctly.

Decommitment Phase:

1. WExCom sends σ and non-interactively decommits the other commitment $c^{\bar{b}}$, where $\bar{b} = 1 - b$.
2. WExRec checks that $\sigma = r^0 \oplus r^1$. If so, WExRec takes the value committed to be σ and $\perp$ otherwise.

Figure 2.2: Weakly Extractable Commitment Scheme `WExCS`

– Extractability: *the probability that τ is accepting and σ^* correspond to $\perp$ is at most $1/2$. Moreover if $\sigma^* \neq \perp$ then the probability that* ExCom* *opens τ to a value different than σ^* is negligible.*

In contrast to the previous definition, a weakly extractable commitment scheme can tolerate failures in the extraction procedure. In Figure 2.2, we show a construction that satisfies our definition on top of any commitment scheme $\texttt{CS} = (\mathrm{Com}, \mathrm{Rec})$.

The proof of binding and hiding of `WExCS` are even simpler than the one given in [PW09]. Moreover, there is no issue of over-extraction, rather there is an issue of under-extraction. Since the malicious sender might refuse to open another commitment during rewinds, with probability 1/2, a cheating sender commits successfully but the extractor fails. We will show later that this weak notion of extractability suffices to prove the security of our main theorem.

Notice that in the two above constructions, when `CS` is the parallel version of Naor's commitment scheme (i.e., a statistically binding string commitment scheme where the sender commits to pairs of strings which XOR corresponds to the string to be committed), then we obtain a constant-round extractable (resp. weakly extractable) statistically binding string commitment scheme based on the black-box use of one-way functions. Similarly, a constant-round (weakly) extractable statistically hiding commitment scheme from any family of collision-resistant hash functions can be obtained from the scheme of [HM96], and a $O(n/\log n)$-round (weakly) extractable statistically hiding commitment from any one-way function can be obtained from the scheme of [HR07, HNO$^+$09].

2.2.3 Non-Malleable Commitment Schemes

For the non-malleability of commitments, we follow the definition introduced by Pass and Rosen and by Lin et al. [PR05b, PR08a, LPV08]. Let $\mathcal{M}$ be the man-in-the-middle adversary running on auxiliary input z, and $\mathtt{NMCS} = (\mathcal{C}, \mathcal{R})$ denote a non-malleable commitment scheme executed by a sender $\mathcal{C}$ and a receiver $\mathcal{R}$. In the real experiment, the adversary $\mathcal{M}$ interacts with a committer $\mathcal{C}$ on the left-hand side and a receiver $\mathcal{R}$ on the right-hand side. In the left execution, $\mathcal{M}$ interacts with $\mathcal{C}$ who is committing to a message σ. In the right execution, $\mathcal{M}$

interacts with $\mathcal{R}$ and attempts to commit to a related message $\tilde{\sigma}$. In the simulated execution, a simulator $\mathcal{S}$ incorporates $\mathcal{M}$ and directly interacts with $\mathcal{R}$.

Statistically binding non-malleable commitment schemes. We use the notion of non-malleability w.r.t. commitment from [DDN91] for statistically binding non-malleable commitment schemes. In this setting, the adversary $\mathcal{M}$ is said to succeed in the experiment if $\mathcal{M}$ can commit to a message $\tilde{\sigma}$ that is related to the message σ committed by the honest committer. Furthermore, we will consider tag-based commitments, where an additional identifier string referred to as *tag* is received in input by both sender and receiver. We also define the notion of one-sided non-malleable commitment where we only consider interactions where the players of the left execution use a common value *tag* that is smaller than any value $\tilde{tag}$ used in any right interaction[1].

Formally, let $mim^{\mathcal{M}}_{\mathtt{NMCS}}(\sigma, z, tag)$ denote a random variable that describes the value σ that $\mathcal{M}$ commits to in the right execution and the view of $\mathcal{M}$ in the full experiment. The goal of $\mathcal{M}$ receiving a commitment of σ in an execution with a identifier tag, consists in committing to a related σ' in an execution with a identifier $\tilde{tag}$ such that $tag \neq \tilde{tag}$. Therefore, in $mim^{\mathcal{M}}_{\mathtt{NMCS}}(\sigma, z, tag)$ we will assume that when the identifier used in the right-hand execution is equal to the one used in left-hand execution, the message committed in $mim^{\mathcal{M}}_{\mathtt{NMCS}}(\sigma, z, tag)$ is always defined as $\perp$. In the simulated experiment, a simulator $\mathcal{S}$ directly interacts with $\mathcal{R}$. Let $sim^{\mathcal{S}}_{\mathtt{NMCS}}(z, tag)$ denote the random variable describing the value σ' committed to by $\mathcal{S}$ and the output of $\mathcal{S}$. Notice that if the commitment scheme is statistically binding, in $mim^{\mathcal{M}}_{\mathtt{NMCS}}(\sigma, z, tag)$ and in $sim^{\mathcal{S}}_{\mathtt{NMCS}}(z, tag)$ the values σ and σ' are well

[1]If there exists a right interaction with $\tilde{tag} < tag$, the value b committed to in that right interaction is defined to be $\perp$.

defined .

It is well known that tag-based non-malleable commitments imply plain non-malleable commitments since one can use any signature scheme for this implication. Since it is known how to construct signature schemes by using a one-way function in a black-box manner, we have that the sole notion to care about in this work is that of tag-based non-malleable commitments.

Definition 12 (Non-Malleable Commitments w.r.t Commitment) *A tag-based commitment scheme* NMCS *is said to be non-malleable if for every* PPT *man-in-the-middle adversary $\mathcal{M}$, there exists a (expected)* PPT *simulator $\mathcal{S}$ such that for all $k \in \mathbb{N}$, $tag \in \{0,1\}^k$, $\sigma \in \{0,1\}^k$, and $z \in \{0,1\}^*$, the following ensembles are computationally indistinguishable:*

$$\{mim^{\mathcal{M}}_{\texttt{NMCS}}(\sigma, z, tag)\} \approx \{sim^{\mathcal{S}}_{\texttt{NMCS}}(z, tag)\}.$$

Similarly, one can define the one-many (resp., many-many) variant of the above definition where the view of $\mathcal{M}$ along with the tuple of values it commits to is required to be indistinguishable regardless of the value (resp., values) committed to in the left interaction (resp., interactions) by the honest sender. We refer the reader to [LPV08] for more details.

2.2.4 Secure Multi-Party Computation (MPC)

A secure multi-party computation (MPC) [BGW88, AL11] scheme allows players to jointly and correctly compute a function based on their private inputs, even in the presence of corrupted players. More precisely, let n be the number of players and t denotes the number of corrupted players. Under the assumption that there exists a synchronous network over secure point-to-point channels, in [BGW88] it is shown that for every n-ary function $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$, there exists a

t-secure MPC protocol Π_f that securely computes f in the semi-honest model for any $t < n/2$, and in the malicious model for any $t < n/3$, with perfect completeness and security. That is, given the private input w_i of player i, after running the protocol Π_f, as long as the adversary corrupts less than t players, each honest player i receives in output the i-th component of the result of the function f applied to the inputs of the players. In addition, nothing is learnt by the adversary from the execution of Π_f other than the output.

Formally, we denote by $\mathcal{A}$ the real-world adversary running on auxiliary input z, and by $\mathcal{S}$ the ideal-world adversary. In a real execution of π where for any $i \in [n]$, party P_i has input x_i. Let set I be the corrupted parties controlled by the adversary and $x = (x_1, \ldots, x_n)$. We denote by $REAL_{\pi,I}^{\mathcal{A}(z)}(x)$ the random variable consisting of the output of $\mathcal{A}$ as well as the outputs of the honest parties. Similarly, we denote by $IDEAL_{f,I}^{\mathcal{S}(z)}(x)$ the analogous output of $\mathcal{S}$ and honest parties after an ideal execution with a trusted party computing f. Our construction will use MPC protocols with perfect completeness and statistical security. We provide the formal definitions in the following.

Definition 13 *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-ary functionality and let π be a protocol. We say that π (n,t)-statistically securely computes f if for every probabilistic adversary $\mathcal{A}$ in the real model, there exists a probabilistic adversary $\mathcal{S}$ of comparable complexity[2] in the ideal model, such that for every $I \subset [n]$ of cardinality at most t, every $x = (x_1, \ldots, x_n) \in (\{0,1\}^*)^n$ where $|x_1| = \ldots = |x_n|$, and every $z \in \{0,1\}^*$, it holds that: $\{IDEAL_{f,I}^{\mathcal{S}(z)}(x)\} \equiv_s \{REAL_{\pi,I}^{\mathcal{A}(z)}(x)\}$.*

Theorem 1 (BGW88) *Consider a synchronous network with pairwise private channels. Then, for every n-ary functionality f, there exists a protocol π_f that (n,t)-perfectly securely computes f in the presence of a static semi-honest adversary*

[2] *It means that $\mathcal{S}$ runs in time that is polynomial in the running time of $\mathcal{A}$.*

for any $t < n/2$, and there exists a protocol that (n, t)-perfectly securely computes f in the presence of a static malicious adversary for any $t < n/3$.

We will refer to such a protocol π_f mentioned in the above theorem as an (n, t)-perfectly secure MPC protocol for functionality f. Also notice that all the above communication requirements to run the MPC protocol will not result in communication requirements for our commitment scheme, since we will use virtual executions of MPC that will be run only locally by players.

Next, we define the *view* of a player in an MPC protocol. The view of a player includes the private inputs given to the player, the randomness used by the player and all messages received by that player during the execution of the protocol. We further denote by $view_i$ the view of player P_i. For a honest player P_i, the final output and all messages sent by that player can be inferred from $view_i$ by running a virtual execution of the protocol. We recall the formal definition of view consistency adapted from [IKOS07].

Definition 14 (View Consistency) *A $view_i$ of an honest player P_i during an MPC computation π contains private inputs and randomness used in the computation, and all messages exchanged on the communication tapes by P_i. We say that a pair of views $(view_i, view_j)$ are consistent with each other if the following hold:*

(a) both the players P_i and P_j individually and honestly computed each outgoing message by using the random tapes, inputs and incoming messages specified in $view_i$ and $view_j$ respectively.

(b) all outgoing messages of P_i to P_j appearing in $view_i$ are consistent with incoming messages of P_j received from P_i appearing in $view_j$, and vice versa.

2.2.5 Verifiable Secret Sharing (VSS)

A verifiable secret sharing (VSS) [CGMA85] scheme is a two-stage secret sharing protocol for implementing the following functionality. In the first stage, a special player, referred to as dealer, shares a secret among the other players, referred to as shareholders, in the presence of at most t corrupted players. In the second stage, players reconstruct the secret shared by the dealer. The functionality ensures that when the dealer is honest, before the second stage begins, all corrupted players have no information about the secret. Moreover, when the dealer is dishonest, at the end of the share phase the honest players would have realized it through an accusation mechanism that disqualifies the dealer.

In contrast to Shamir's Secret Sharing scheme [Sha79], a VSS scheme can tolerate errors on malicious dealer and players on distributing inconsistent or incorrect shares, indeed the critical property is that even in case the dealer is dishonest but has not been disqualified, still the second stage always reconstruct the same bit among the honest players. We will consider a VSS scheme implementing the above VSS functionality, as defined below.

Definition 15 (VSS Scheme) *An $(n+1, t)$-perfectly secure VSS scheme consists of a pair of protocols* $\mathtt{VSS} = (\mathrm{Share}, \mathrm{Recon})$ *that implement respectively the sharing and reconstruction phases as follows.*

– Share*: Player P_{n+1} referred to as dealer runs on input a secret s and randomness r_{n+1}, while any other player P_i, $1 \leq i \leq n$, runs on input a randomness r_i. During this phase players can send (both private and broadcast) messages in multiple rounds.*

– Recon*: Each shareholder sends its view v_i of the sharing phase to each other player, and on input the views of all players (that can include bad or empty views) each player outputs a reconstruction of the secret s.*

All computations performed by honest players are efficient. The computationally unbounded adversary can corrupt up to t players that can deviate from the above procedures. The following security properties hold.

– Commitment: *if the dealer is dishonest then one of the following two cases happen:*

  1) during the sharing phase honest players disqualify the dealer, therefore they output a special value $\perp$ and will refuse to play the reconstruction phase;

  2) during the sharing phase honest players do not disqualify the dealer, therefore such a phase determines a unique value s^* that belongs to the set of possible legal values that does not include $\perp$, which will be reconstructed by the honest players during the reconstruction phase.

– Secrecy: *if the dealer is honest then the adversary obtains no information about the shared secret before running the protocol* Recon*.*

– Correctness: *if the dealer is honest throughout the protocols then each honest player will output the shared secret s at the end of protocol* Recon*.*

Direct implementations of $(n+1, \lfloor n/3 \rfloor)$-perfectly secure VSS schemes can be found in [BGW88, CDD$^+$99]. However since we are interested in a deterministic reconstruction procedure, we will use the scheme of [GIKR01] that implements an $(n+1, \lfloor n/4 \rfloor)$-perfectly secure VSS scheme. We will denote by Π_{Share} the execution of an $(n+1, \lfloor n/4 \rfloor)$-perfectly secure protocol that implements the Share stage of the above VSS functionality. We will denote by Π_{Recon} the corresponding protocol executed by shareholders to implement the deterministic Recon stage.

CHAPTER 3

Construction of Non-Malleable Commitment

To construct non-malleable commitments in a black-box manner, our starting point is the recent constant-round protocol of Goyal [Goy11]. The protocol in [Goy11] makes use of one-way functions in a non-black-box manner. It has a zero-knowledge proof of consistency on committed strings which we will implement using the secure computation in the head approach. However, we note that the protocol of Goyal, very informally, is still too "non-black-box" to admit a simple application of this idea. The protocol of Goyal uses a proof of complex statements involving the randomness with which a commitment is constructed. We present a simplification of the non-malleable commitment scheme of Goyal. Our simplification involves making a part of the protocol *purely information theoretic* using pairwise-independent hash functions and strong randomness extractors. This is done in a way such that the proof of non-malleability w.r.t. commitment still goes through. Finally, we are left with a protocol where the only computational part is an initial commitment to a set of random strings such that the consistency proof only needs to prove a statement above the committed strings, and the MPC in the head technique that we use is powerful enough to handle such a scenario.

3.1 Intuition

We first provide some intuition behind our construction. Let k be a security parameter. Suppose one needs to commit to a set of strings $S = \{s_1, \ldots, s_n\}$ and prove a statement about these strings later on. Our MPC in the head technique works as follows. The committer starts by emulating k virtual players "in his head". Each player is given as input a share of S, where the secret sharing is done using a verifiable secret sharing scheme (e.g., [CGMA85]). Let the view of the players so far be $view_1^0, \ldots, view_k^0$ respectively. The committer commits to these views using a regular computationally secure commitment scheme.

At a later point in the interaction, suppose the committer needs to compute a functionality f on the committed strings, reveal the result $f(S)$ to the verifier, and prove its correctness. The committer can now do as follows:

1) The committer continues to emulate the k virtual player in his head. The players will now compute the following functionality: *take the share of each player, reconstruct the set S and output $f(S)$ to each player.* The players jointly run a secure computation protocol (starting with the views already committed to) to compute this functionality. The secure computation protocol being used is information theoretically secure tolerating up to a constant fraction of corrupted parties such as BGW [BGW88].

2) Let the new views of the players up to this point be $view_1^1, \ldots, view_k^1$. The committer now reveals $f(S)$ and commits to these new views.

3) The receiver chooses a constant fraction of the players at random. The committer decommits to both the views for the selected player P_i. This

includes the initial views $view_i^0$ as well as the new views $view_i^1$.

4) The receiver checks if the players behaved honestly during the entire computation and that their views are "consistent" with each other (see Definition 14 for the precise notion of consistency). If this check is successful, "most" of the virtual players were correctly emulated by the committer. Hence the output of the computation must be correct (since the protocol anyway tolerates a constant fraction of corrupted players).

5) The security of the committer is also preserved since revealing views for a constant fraction of players does not reveal anything about the set of strings S that he started with (other than of course the output $f(S)$).

The key difference from [IKOS09] is that in our setting, the statement we are proving actually inherently involve a non-black-box use of the commitment scheme: "the evaluation of f on the set of committed values results in $f(S)$". The technique of [IKOS09] was later extended (with multiple commitments of views) by Ostrovsky for proving relations over committed values in a black-box fashion and, independently, by Goyal, Ishai and Sahai to get a black-box realization of the commit and prove functionality. However both extensions are insufficient to obtain our results.

Next, we describe a simplified version of our protocol, which only considers "short" tags with one-sided non-malleability. That is, we restrict the tag length to be $\log n$ for an n-round protocol and the players of the left execution use a common value tag that is smaller than any value $\tilde{tag}$ used in any right interaction. In addition, the security of this basic protocol is guaranteed against synchronized adversaries only. A synchronized adversary is a restricted adversary that plays the main-in-the-middle attack by playing exactly one message on the right execution after a message is received from the left execution, and playing exactly one message on the left execution after a message is received from the right execution. In our scheme we will use an extractable commitment scheme `ExCS` as already used in previous work [PW09]. Such a commitment scheme suffers of "over extraction", which means that the extractor can output a value different than $\perp$ even when the committed message is not well formed (therefore the committed message is undefined and can not be opened anymore). In addition, we use the commitment scheme `WExCS` which instead suffers from "under extraction" as described in Definition 11.

3.2 Basic Construction

We assume that each execution has a session identifier $tag \in [2n]$, where n is the length of party identity in bits. Let k be the security parameter and $\ell = \ell(k) = k \cdot tag$. The commitment scheme $\texttt{NMCS} = (\mathcal{C}, \mathcal{R})$ between a committer $\mathcal{C}$ and a receiver $\mathcal{R}$ proceeds as follows to commit to a k-bit string σ. We assume that $\lambda = \lfloor k/4 \rfloor$. In the description below, we have included some intuition in *italics*.

Commitment Phase.

0. **Initial setup.** $\mathcal{R}$ picks λ out of k players (which will be later emulated by the committer) at random. That is, it randomly selects λ distinct indices $\Lambda = \{r_1, \ldots, r_\lambda\}$ where $r_i \in [k]$ for any $i \in [\lambda]$. For each r_i, $\mathcal{R}$ sends an extractable commitment c_i of r_i using `ExCS`.

1. **Primary slot.** Let Π_{Share} be a protocol implementing the Share phase of

a $(k+1, \lambda)$-perfectly secure VSS scheme. We require the VSS protocol to have a deterministic reconstruction phase. The committer $\mathcal{C}$ is given a k-bit string σ to commit.

1.1. **Commit:** $\mathcal{C}$ first generates ℓ pairs of random strings $\{\alpha_i^0, \alpha_i^1\}_{i\in[\ell]}$ of length $4k$ each, and a k-bit random string s. *Here the strings are such that the knowledge of both strings $\{\alpha_i^0, \alpha_i^1\}$ for any pair will allow an extractor to extract the committed value. The string s is meant to serve as a seed of a strong extractor used later on in the protocol. The purpose of the next two stages (1.2 and 1.3) is simply to produce a specialized commitment to the strings $\{\alpha_i^0, \alpha_i^1\}_{i\in[\ell]}, s$ and σ.*

1.2. The committer $\mathcal{C}$ now starts emulating $k+1$ virtual players locally "in his head". $\mathcal{C}$ sets the input of P_{k+1} (i.e., the Dealer) to the string $\sigma \parallel s \parallel \{\alpha_i^0, \alpha_i^1\}_{i\in[\ell]}$, while each other player has no input. Then $\mathcal{C}$ runs Π_{Share} such that for all $i \in [k]$, player P_i obtains share w_i.

1.3. Let $view_1^1, \ldots, view_{k+1}^1$ be the views of the $k+1$ players describing the execution of Π_{Share}. $\mathcal{C}$ uses `WExCS` to send a commitment V_i^1 of $view_i^1$ to $\mathcal{R}$, in parallel for any $i \in [k+1]$. *At this stage, the committer is now committed to σ, s and $\{\alpha_i^0, \alpha_i^1\}_{i\in[\ell]}$.*

1.4. **Challenge:** $\mathcal{R}$ sends a random ℓ-bit challenge string $ch = \{ch_i\}_{i\in[\ell]}$.

1.5. **Response:** $\mathcal{C}$ sends $\{\alpha_i^{ch_i}\}_{i\in[\ell]}$ to $\mathcal{R}$. *The goal of the extractor would be to rewind and learn a pair $\{\alpha_i^0, \alpha_i^1\}$. To ensure non-malleability, this would be done without rewinding the (interleaved) left interaction.*

2. **Verification message.** Let $\mathcal{H}$ be a family of pairwise-independent hash functions with domain $\{0,1\}^{4k}$ and range $\{0,1\}^k$, and `Ext` $: \{0,1\}^{4k} \times \{0,1\}^k \to \{0,1\}^k$ be a strong randomness $(3k, 2^{-k})$-extractor.

2.1. $\mathcal{R}$ picks a function h at random from $\mathcal{H}$ and sends it to $\mathcal{C}$.

2.2. $\mathcal{C}$ sends s, $\{h(\alpha_i^0), h(\alpha_i^1), B_i = \sigma \oplus \mathtt{Ext}(\alpha_i^0, s) \oplus \mathtt{Ext}(\alpha_i^1, s)\}_{i\in[\ell]}$ to $\mathcal{R}$.

Say that in the primary slot phase, the extractor rewinds the adversary and receives a value α_i^j. This phase enables checking such a received value for correctness (and for subsequent recovery of the string σ). This phase is purely information theoretic but still provides for the right binding properties. The corresponding mechanism in the construction of Goyal was implemented using complex computations involving random tapes used to generate various commitments.

3. **Consistency proof.** *Now the sender needs to prove the correctness of the values revealed in stage 1.5 and 2.2.*

3.1. Let Π_{ch} be a (k, λ)-perfectly secure MPC protocol such that given ch as a public input and w_i as the private input of P_i for any $i \in [k]$, at the end of the computation $\{\alpha_i^{ch_i}\}_{i\in[\ell]}$ is received in output by P_i for any $i \in [k]$. $\mathcal{C}$ runs internally Π_{ch} and sends a commitment V_i^2 of the view $view_i^2$ of P_i when executing Π_{ch} using `WExCS` in parallel for any $i \in [k]$ to $\mathcal{R}$.

3.2. Let Π_h be a (k, λ)-perfectly secure MPC protocol such that given a hash function h as a public input and w_i as the private input of P_i for any $i \in [k]$, at the end of the computation $(s, \{h(\alpha_i^0), h(\alpha_i^1)\}_{i\in[\ell]}, \{B_i = \sigma \oplus \mathtt{Ext}(\alpha_i^0, s) \oplus \mathtt{Ext}(\alpha_i^1, s)\}_{i\in[\ell]})$ is received in output by P_i for any $i \in [k]$. $\mathcal{C}$ runs internally Π_h and sends a commitment V_i^3 of the view $view_i^3$ of P_i when executing Π_h using `WExCS` in parallel for any $i \in [k]$.

3.3. $\mathcal{R}$ decommits $\{c_i\}_{i\in[\lambda]}$.

3.4. $\mathcal{C}$ decommits $\{V_{r_i}^1, V_{r_i}^2, V_{r_i}^3\}_{i\in[\lambda]}$ (i.e., it decommits the subset of views $\{view_{r_i}^1, view_{r_i}^2, view_{r_i}^3\}_{i\in[\lambda]}$.)

3.5. For $j = 1, 2, 3$, $\mathcal{R}$ verifies that all pairs of views in $\{view_{r_i}^j\}_{i\in[\lambda]}$ are consistent (according to Definition 14) and that the dealer P_{k+1} has not been disqualified by any player, otherwise $\mathcal{R}$ aborts; moreover for $j = 1, 2$ and $i = 1, \ldots, \lambda$, $\mathcal{R}$ checks that $view_{r_i}^j$ is a prefix of $view_{r_i}^{j+1}$, otherwise $\mathcal{R}$ aborts.

Decommitment Phase.

1. $\mathcal{C}$ decommits $\{V_i^1\}_{i\in[k]}$ as $\{view_i^1\}_{i\in[k]}$.
2. $\mathcal{R}$ checks that all commitments to the views are opened correctly in the previous step. If a commitment is opened incorrectly, $\mathcal{R}$ sets the corresponding revealed view to 0^k (instead of just aborting).
3. Let Π_{Recon} be a protocol implementing the Recon phase corresponding to the $(k+1, \lambda)$-perfectly secure VSS Share phase (which includes the string σ) used in the commitment phase. $\mathcal{R}$ runs Π_{Recon} using $view_1^1, \ldots, view_k^1$ as input to reconstruct and output the first substring of the value that the majority of the players would output in the reconstruction. If there is no majority, then consider the committer to have aborted during the decommitment phase (and output $\perp$).

   We stress that the receiver does not perform any additional checks. In particular, even if it detects that some of the views are not correctly constructed (and hence the committer behaved in a dishonest way), it still accepts the decommitment phase as long as a majority of the players agree on a value during reconstruction. This is crucial for getting security as per the non-malleability *w.r.t. commitment notion.*

3.3 Proof of Security

Theorem 2 *The commitment scheme* NMCS *is a one-sided non-malleable commitment scheme with short tags secure against synchronized adversaries.*

Proof. To prove the non-malleability of NMCS, we show that there exists a black-box simulator $\mathcal{S}$ such that for all $k \in \mathbb{N}$, $n = \texttt{poly}(k)$, $tag \in [2n]$, $\sigma \in \{0,1\}^k$, $z \in \{0,1\}^*$, and for any man-in-the-middle adversary $\mathcal{M}$, $\mathsf{dist}_1 = \{mim^{\mathcal{M}}_{\texttt{NMCS}}(\sigma, z, tag)\}$ and $\mathsf{dist}_2 = \{sim^{\mathcal{S}}_{\texttt{NMCS}}(z, tag)\}$ are computationally indistinguishable.

3.3.1 Construction of the Simulator $\mathcal{S}$

The simulator $\mathcal{S}$ is constructed as follows. $\mathcal{S}$ uses the adversary $\mathcal{M}$ as a subroutine and interacts with an external receiver $\mathcal{R}$; in the left interaction, $\mathcal{S}$ honestly commits to the string 0^k to $\mathcal{M}$, while in the right interaction it simply forwards the messages being sent out by $\mathcal{M}$ to $\mathcal{R}$ and vice versa. We claim that given the above $\mathcal{S}$, the ensembles dist_1 and dist_2 are computationally indistinguishable.

Suppose, towards contradiction, that there exists a distinguisher $\mathcal{D}$ which can distinguish dist_1 and dist_2 with an advantage $r(k) \geq \varepsilon(k)$ for some negligible function ε over infinitely many values of k. That is,

$$\big|\Pr[\mathcal{D}(\mathsf{dist}_1) = 1] - \Pr[\mathcal{D}(\mathsf{dist}_2) = 1]\big| \geq 2r(k).$$

Now, fix any such generic k and consider the full experiment where the adversary $\mathcal{M}$ interacts with a committer $\mathcal{C}$ in the left interaction and with a receiver $\mathcal{R}$ in the right one. Given the view of $\mathcal{M}$ in such real experiment, we then show how to construct an extractor $\mathcal{E}$ which outputs the value $\tilde{\sigma}$ committed by $\mathcal{M}$ in the right interaction with probability at least $1 - r(k)$ without rewinding $\mathcal{C}$ (i.e., without having access to the value and the random coins used by $\mathcal{C}$ in the left interaction, similarly to [DDN91, LPV08]). However, the existence of such an extractor $\mathcal{E}$ along with the successful malleability attack, contradicts the (stand-alone) computational hiding property of the commitment scheme `NMCS`. Therefore, to show that there exists no such a distinguisher $\mathcal{D}$, the only thing that remains to show is how to construct an extractor that succeeds with probability at least $1 - r(k)$. For simplicity, we analyze the failure probability of our extractor conditioned on the event that given the completed (i.e., all messages have been played in both sessions and no party aborted) main thread, there is exactly one value $\sigma \neq \bot$ consistent with the transcript of the right interaction. We prove it in the following lemma. Note that the corresponding lemma in [Goy11] was immediate because of the usage of zero-knowledge proofs.

Lemma 2 *Let $\mathcal{R}$ be an honest receiver that completes without aborting the commitment phase with a PPT sender $\mathcal{M}$. Then with all but negligible probability, the commitments $\tilde{V}_1^1, \ldots, \tilde{V}_k^1$ sent by $\mathcal{M}$ uniquely define the string corresponding to the concatenation of σ, s, and $\{\alpha_i^0, \alpha_i^1\}_{i \in [\ell]}$ (i.e., the string that the reconstruction phase of the VSS scheme would reconstruct), and, this string is not equal to $\bot$. In fact, at least 99 percent of the commitments are commitments to valid views such that these views are also all consistent with each other.*

Proof. For the receiver to not abort, it means that in step 3.5, the set of views selected by Λ does not contain any pair of inconsistent views, and no view in the set disqualifies the dealer. We then consider the following two cases for any arbitrary constant c.

Case 1: Large enough consistent set of views. Assume that there is a set of views of size $t > k - \lambda/c$ such that all pairs of views in the set are consistent. By the definition of consistency, it holds that all such views consist of honestly performed local computations. Moreover, the fact that all pairs of views in the set are consistent means that outgoing messages in a view of each player in the set, correspond to incoming messages of another player in the set (of course this is true only for messages sent to players in the set). Therefore, we have that the above t views correspond to an execution of VSS where these t players are honest and the remaining less than λ/c players can instead be corrupted. Since the number of corrupted players is below the security threshold of the implemented VSS, we can now rely on the commitment property of VSS. Indeed, it guarantees that there exists one fixed string that Π_{Recon} would reconstruct and give in output to those honest players, unless all those players accused the dealer in the sharing phase. However this last case can not hold because (by pigeon hole principle) at least one index of those t views is in Λ and therefore the commitment phase would have showed the disqualification of the dealer, and this has been already ruled out at the beginning of the proof.

Case 2: Too many inconsistent pairs of views. In contrast to Case 1, assume that there is no set of views of size $t > k - \lambda/c$ such that all pairs of views in the set are consistent. Similar to [IKOS07], consider the following inconsistency graph G, defined based on the k committed views. The graph G has k vertices (corresponding to the k views) and there is an edge between two vertices in G if the corresponding pair of views is *inconsistent* (see Definition 14).

Observe that the size of the minimum vertex cover of G is at least λ/c. This is because otherwise, we get a fully consistent set of size greater than $k - \lambda/c$ (which includes views corresponding to all vertices except those in the vertex cover). Now, we observe that G must have a matching of size at least $\lambda/2c$. This follows from the well known connection between the size of minimum vertex cover and the size of a maximal matching. The following combinatorial claim is implicit in [IKOS07].

Claim 1 *Consider a graph G having k vertices and a matching of size at least k/C in G (where C is a constant). Select a constant fraction of vertices at random. Then the probability that the resulting subgraph does not cover any edge of the matching is negligible in k. [IKOS07]*

This means that if we were to choose the set Λ at random *after* the graph is defined, Case 2 will happen only with negligible probability. This is because the challenge Λ will select at least one inconsistent pair of views (making the receiver abort). Instead in our protocol, Λ is committed by the receiver before the graph G is fixed by the committer. Hence, we will now rely on the hiding of the commitment to Λ and (weak) extractability of the commitments to views. Consider the following experiment.

An external challenger gives a commitment to Λ consisting of λ random indices in $[k]$. We will construct an adversary $\mathcal{A}$ which guesses "non-trivial and hard to guess" information about Λ with non-negligible probability thus contradicting the hiding property of the commitment to Λ.

The adversary $\mathcal{A}$ works as follows. It receives the commitment to Λ and simply forwards it to $\mathcal{M}$. Now $\mathcal{A}$ receives from $\mathcal{M}$ the commitments to the k views using `WExCS`. Next, $\mathcal{A}$ will rewind (multiple times) $\mathcal{M}$ and extract at least $k - \log^2 k$ views (by simply choosing a random k-bit challenge string each time $\mathcal{M}$ is rewound). We now prove that this can be done in polynomial time. The proof relies on the specific construction of weakly extractable commitment scheme shown in Definition 11.

Claim 2 *If $\mathcal{M}$ completes successfully the commitment phase with non-negligible probability, then the extraction of the views fails for at most $\log^2 k$ commitments.*

Proof. The fact that $\mathcal{M}$ completes successfully the commitment phase implies that, for some polynomial p, $\mathcal{M}$ answers correctly (i.e., without aborting) to a $1/p(k)$ fraction of the k-bit challenge used in the second round of the commitments of the k views through `WExCS`. Assume there are $\log^2 k$ commitments that are completed with probability at most $1/(p(k)k)$ (taken over the entire challenge space). Note that any random challenge selects at least one of them except with negligible probability. Thus, with probability $1/p(k) - \epsilon(k)$, for a negligible function ϵ, $\mathcal{M}$ opens one of these shares. However by union bound, this probability can be at most $(\log^2 k)/(p(k)k)$. This is a contradiction. Therefore there exists at least $k - \log^2 k$ commitments that are opened with probability at least $1/p(k)k$, and thus they can all be extracted in polynomial time. □

Thus, observe that $\mathcal{A}$ has now extracted a subgraph of G consisting of at least $k - \log^2 k$ vertices (corresponding to the views it could extract). Since we are in Case 2, even this subgraph must have a matching of size at least $\lambda/2c - \log^2 k$ (this is because G is guaranteed to have a matching of size at least $\lambda/2c$).

The adversary $\mathcal{A}$ simply outputs the indices of the vertices of each edge from this matching and halts. If Case 2 happens with non-negligible probability (i.e., there is no fully consistent set of size greater than $k - \lambda/c$ and still $\mathcal{M}$ manages to successfully finish the commitment stage), we have that with non-negligible probability, $\mathcal{A}$ outputs a matching of size at least $\lambda/2c - \log^2 k$ such that the challenge Λ does not cover any edge in this matching (this follows from the fact when $\mathcal{M}$ is successful, all pairs of views with indexes in Λ are consistent). If this happens, we say that $\mathcal{A}$ is successful.

By Claim 1, this should happen only with negligible probability if Λ was chosen at random after the graph G was defined. However now since we are taking the commitment to Λ from an external challenger, we can simply rely on the semantic security of the commitment scheme. Even if the external challenger gives a commitment to an all zero string (instead of Λ), $\mathcal{A}$ will still be successful with non-negligible probability. However this is a contradiction. $\square$

3.3.2 Construction of the Extractor $\mathcal{E}$

Let $\ell = \ell(k) = k \cdot tag$ and $\tilde{\ell} = \tilde{\ell}(k) = k \cdot \tilde{tag}$. Given the view of $\mathcal{M}$ in the real experiment as input, $\mathcal{E}$ first honestly simulates the view, by replaying the same messages; we refer to this part of the execution as the "main thread" and denote it by MT. If the adversary $\mathcal{M}$ aborts before the main thread is completed, $\mathcal{E}$ simply outputs $\perp$ and halts (similar to [Goy11], if the parties $\mathcal{C}$ or $\mathcal{R}$ terminate the protocol before it was finished due to an obvious cheating by $\mathcal{M}$, we consider the behavior of $\mathcal{M}$ as an abort). Otherwise, $\mathcal{E}$ rewinds $\mathcal{M}$ up to δ times. For $j \in [\delta]$, the extractor $\mathcal{E}$ do as follows.

- $\mathcal{E}$ rewinds the right interaction to the beginning of the Step 1.4 of the protocol. It sends to $\mathcal{M}$ a new random challenge $\tilde{ch}[j] \in \{0,1\}^{\tilde{\ell}}$ in the right interaction and receives from $\mathcal{M}$ a challenge $ch[j] \in \{0,1\}^{\ell}$ in the left interaction.
- Since $\mathcal{E}$ is not allowed to rewind the committer $\mathcal{C}$ to make additional queries, it has to prepare the *simulated response* to the challenge $ch[j]$ on its own. Notice that the challenge $ch[j]$ induces a selection of ℓ secrets on the left, and there are ℓ secret values $\{\alpha_i^{ch_i}\}_{i\in[\ell]}$ on the left already "recovered" in the

main thread (i.e., they were asked by $\mathcal{M}$ and given by $\mathcal{C}$ in the main thread). Therefore, $\mathcal{E}$ should respond to $\mathcal{M}$ the same value if the same $\alpha_i^{ch_i}$ appears in this recovered set. Otherwise, $\mathcal{E}$ simply chooses a random string and uses that string as the response to $\mathcal{M}$.

- $\mathcal{E}$ receives the response corresponding to $\tilde{ch}[j]$ from $\mathcal{M}$ in the right interaction. $\mathcal{E}$ checks if there exists an index i such that one of $(\tilde{\alpha}_i^0, \tilde{\alpha}_i^1)$ was received during the main thread while the other was received as part of the current response in rewinding j. If so, $\mathcal{E}$ further computes the hash values of $(\tilde{\alpha}_i^0, \tilde{\alpha}_i^1)$ using the same function $\tilde{h}$ sent in the main thread and compares these new hash values with all the value it got in the main thread. If they match, $\mathcal{E}$ computes $\texttt{Ext}(\tilde{\alpha}_i^0; s)$ and $\texttt{Ext}(\tilde{\alpha}_i^1; s)$, and then recovers and outputs the committed value $\tilde{\sigma}$ from $\tilde{B}_i$. Otherwise, $\mathcal{E}$ goes to the beginning of this loop.

If $\mathcal{E}$ did not success in outputting the value $\tilde{\sigma}$ after δ rewindings (e.g., due to $\mathcal{M}$ aborting or not revealing the correct values associated with the commitments), it aborts and outputs $\mathsf{Ext\_Fail}$. Rest of the proof will have two part: (a) we will analyze the probability with which the string $\tilde{\sigma}$ output by the extractor does not correspond to the committed string (conditioned on the event $\mathcal{E}$ does not abort), and, (b) we will analyze the probability with which $\mathcal{E}$ aborts and outputs $\mathsf{Ext\_Fail}$ in the rest of the proof.

Lemma 3 *The probability with which the string $\tilde{\sigma}$ output by the extractor does not correspond to the committed string (conditioned on the event $\mathcal{E}$ does not abort) is negligible.*

Proof. In this proof, we will make use of the pairwise independence property of the hash function h. In particular, we will rely on the fact that if during rewinding, the extractors extracts a value $\tilde{\alpha}_i^j$ which is different from what was committed to, w.h.p., it will hash to a different value than the one appearing in the main thread.

Recall that the extractor runs as honest receiver (in the main thread), gets $\tilde{\ell}$ secret values $\{\tilde{\alpha}_i^{ch_i}\}_{i\in[\tilde{\ell}]}$ first, and then performs the extraction procedure to get other secret values $\{\tilde{\alpha}_i^{\tilde{ch_i}}\}_{i\in[\tilde{\ell}]}$ such that at least one of the hash values of these secrets (i.e., $\{h(\tilde{\alpha}_i^{\tilde{ch_i}})\}_{i\in[\tilde{\ell}]}$) was already received when playing as honest receiver. Now, suppose that the honest receiver gets some α_i^0 first (in the main thread). Also, it received $u = h(\alpha_i^1)$ in the main thread. Now say during rewinding, the extractor receives another value $\alpha_i'^1 \neq \alpha_i^1$. Observe that at the point the extractor receives this value from the adversary, the adversary still has not received the function h from $\mathcal{E}$. This implies that by the pairwise independence of h, probability that $h(\alpha_i'^1) = h(\alpha_i^1)$ is negligible. Hence, it follows that except with negligible probability, $\alpha_i'^1 = \alpha_i^1$. However this means that the extractor has extracted the correct value $\tilde{\sigma}$. □

Lemma 4 *The probability that the extractor $\mathcal{E}$ aborts is bounded by $r(k)$ for large enough k.*

Proof. We will closely follow the definition and proof technique used in [Goy11]. First, we call a main thread "bad" if the probability (over the randomness used in the rewinds) of $\mathcal{E}$ outputting $\mathsf{Ext\_Fail}$ is noticeable. Then we divide these "bad" main threads into three different categories, where each of them satisfy a different property. We also define the prefix of a given main thread as the transcript of the left and the right interaction up to Step 1.3. For a fixed prefix, we denote by p the probability (over the randomness used by $\mathcal{C}$ and $\mathcal{R}$ after Step 1.3) that $\mathcal{M}$ completes the main thread without aborting, and let q be the probability that $\mathcal{E}$ succeeds in extracting in a single rewinding using a simulated response.

Next, we recall the notion of a *fraction* of main threads. If the fraction of main threads with a particular property is f, it means that the probability (over the randomness of the entire experiment) that $\mathcal{E}$ receives a *completed* main thread with that property is f. We choose three arbitrary constants C_1, C_2, C_3 such that $\frac{1}{C_1} + \frac{1}{C_2} + \frac{1}{C_3} \leq \frac{3}{4}$. Note that though these constants could in fact be the same and arbitrarily big, we will use three distinct constants to make the connections between the different parts of proof more clear.

Lemma 5 *The fraction of main threads for which $p < \frac{r(k)}{C_1}$ is bounded by $\frac{r(k)}{C_1}$. We call these threads as MT of type $\mathsf{bad1}$.*

PROOF. Note that $\mathcal{E}$ never aborts and outputs $\mathsf{Ext\_Fail}$ if a main thread was not completed. Therefore,

$$\begin{aligned}
& \Pr\left[\mathsf{MT} \text{ is of type } \mathsf{bad1}\right] & (3.1)\\
\leq\ & \Pr\left[\mathsf{MT} \text{ has a prefix with } p < \frac{r(k)}{C_1}\right] \cdot \Pr\left[\mathsf{MT} \text{ is completed} \mid p < \frac{r(k)}{C_1}\right] \\
\leq\ & 1 \cdot \frac{r(k)}{C_1}.
\end{aligned}$$

□

Next, given a main thread, we define the *dependent set of secrets* S in the right interaction as follows. Recall that the committer $\mathcal{C}$ will choose ℓ pairs of secrets $\{\alpha_i^0, \alpha_i^1\}_{i\in[\ell]}$ which will later be used with `Ext` to "encrypt" the value σ. Notice that these secret values were shared among k players implementing the Share functionality of a VSS scheme (and hence, being committed in views $\{V_i^1\}_{i\in[k]}$), and the challenge $ch[j]$ in fact induces a selection to recover ℓ of such secret values on the left. These ℓ values are in the "recovered" set of secrets, since they were revealed by the committer in the main thread and hence known by the extractor. Intuitively, the secrets in the set S are the secrets that were added by mauling one (or more) of the "unrecovered" secrets that are hidden in committed views in the left interaction. Hence, to correctly reveal the views which are consistent with the secrets in S in the right interaction, $\mathcal{M}$ has to get the correct value from the committed views $\{V_i^1\}_{i\in[k]}$ in the left interaction and use the underlying secrets to compute $\{\tilde{V}_i^1\}_{i\in[k]}$. Throughout the paper, we say the secret is "revealed correctly" by $\mathcal{M}$, if there exist openings of the committed views which are consistent to this value.

Definition 16 (Dependent Set of Secrets) *Let ch be the challenge from $\mathcal{M}$ in the left interaction in the main thread. The probabilities below are taken over* *the randomness of the experiment* after *the prefix completion. We say S is a* dependent set of secrets *of a main thread iff the following two conditions hold: for every secret $\tilde{\alpha}_i^b$ in S,*

1. *The probability that the secret $\tilde{\alpha}_i^b$ is selected by $\mathcal{R}$ AND its value is revealed correctly by $\mathcal{M}$ on the right is at least $\frac{r(k)}{3C_1}$ (for this prefix).*
2. *The probability that the secret $\tilde{\alpha}_i^b$ is selected by $\mathcal{R}$ AND its value is revealed correctly by $\mathcal{M}$ on the right is less than $\frac{r(k)}{2C_2\tilde{\ell}(k)}$ conditioned on the event that the challenge by $\mathcal{M}$ in the left interaction is ch.*

Observe that the first probability in the above definition is dependent only on what the prefix in the main thread is, while, the second one depends on the prefix as well as what the left challenge ch appearing in the main thread is. Both these probabilities values are well defined for a given main thread.

Lemma 6 *Let S be the dependent set of secrets of a main thread and $|S|$ denote the number of secrets in the set S. $|S| > \ell + \log^2 k$ for at most a $\frac{r(k)}{C_2} + negl(k)$ fraction of the main threads. Call these threads as* MT *of type* $\mathsf{bad2}$.

Intuition. Consider the following. Assume that the extractor had the power of sampling multiple transcripts with the same prefix and having the same left challenge. Then all except for at most $\ell + \log^2 k$ secrets on the right have a good probability of being revealed correct by $\mathcal{M}$ (i.e., except in $\frac{r(k)}{C_2} + \text{negl}(k)$ fraction of the main threads). Hence such an extractor will be successful except for $\frac{r(k)}{C_2} + \text{negl}(k)$ fraction of the main threads. At a high level, this is because there is an exponential number of right challenges for each left challenge (on an average) and obtaining a correct response for any two such right challenges enables extraction.

Proof. For a given prefix, consider a set S for a challenge ch such that

$|S| > \ell + \log^2 k$ and a random challenge $\tilde{ch}$ given by $\mathcal{R}$ in the right interaction. We observe the following:

- The probability that the set of secrets selected by $\tilde{ch}$ and the set S are disjoint is at most $\frac{1}{2^{\ell+\log^2 k}}$. More precisely, this probability is either 0 or $\frac{1}{2^{|S|}}$; it is 0 when a pair of secrets $(\tilde{\alpha}_i^0, \tilde{\alpha}_i^1)$ appear in S and $\frac{1}{2^{|S|}}$ otherwise. This is because each secret in S is selected independently with probability one half.

- Depending on the choice of the challenge $ch \in \{0,1\}^\ell$, there are at most 2^ℓ possibilities for such a set S. Taking the union bound over all such sets, we get that the probability that the set of secrets selected by $\tilde{ch}$ is disjoint with *any* such set S (with $|S| > \ell + \log^2 k$) is at most $\frac{2^\ell}{2^{\ell+\log^2 k}} = \text{negl}(k)$.

- By the second condition of Definition 16, the probability that for *some* $\tilde{\alpha}_i^b \in S$, $\mathcal{M}$ revealed the correct value in the right interaction in the main thread is bounded by $\frac{r(k)}{C_2}$. This is by taking the union bound over all $\tilde{\alpha}_i^b \in S$, given that $|S|$ cannot exceed $2\tilde{\ell}(k)$.

Therefore, we have the following:

$$\begin{aligned}
& \Pr[\mathsf{MT} \text{ is of type } \mathsf{bad2}] \\
\leq\ & \Pr\left[\tilde{ch} \text{ does not select any secret in } S\right] + \\
& \Pr\left[\mathsf{MT} \text{ is completed} \mid \tilde{ch} \text{ selects a secret in } S\right] \\
\leq\ & \Pr\left[\exists\ \tilde{\alpha}_i^b \in S \text{ s.t. } \mathcal{M} \text{ revealed the correct value of } \tilde{\alpha}_i^b \text{ in } \mathsf{MT}\right] + \text{negl}(k)
\end{aligned}$$

Hence, the fraction of main threads of type $\mathsf{bad2}$ is bounded by $\frac{r(k)}{C_2} + \text{negl}(k)$. □

This lemma shows that there are at most $\ell + \log^2 k$ secret values on the right which are "dependent" on the left secrets whose value $\mathcal{E}$ did not recover in the main thread. However the total number of secrets on the right is $2 \cdot \tilde{\ell} > 2(\ell + \log^2 k)$ (since $\tilde{tag} > tag$). Hence, there should exists at least one pair of secrets on the right such that $\mathcal{M}$ can correctly reveal both the values (without asking for values unrecovered in the main thread). If that is the case, $\mathcal{E}$ is successful in extracting the value σ committed on the right without any "additional queries" on the left.

Continuing the intuition from Lemma 6. The primary hurdle in completing the proof is the following. Our PPT extractor will not have the power of sampling transcripts with "collision" (i.e., with the same left challenge). The extractor gets a different challenge from $\mathcal{M}$ (compared to the main thread) while rewinding $\mathcal{M}$ and provides a "simulated" response. We now need to analyze such an experiment. Intuitively, suppose there is a secret on the right which is revealed correctly with good probability in the "absence" of values from the unrecovered set of secrets (i.e., conditioned on the event when the left challenge is the same as the main thread). Then this means that the right secrets were not formed by "mauling" one of secrets in the unrecovered set. Hence even if a secret in the unrecovered set was given incorrectly, $\mathcal{M}$ hopefully should still reveal that secret value correctly on the right. We first introduce the following definition and formally analyze this case.

Definition 17 (Strictly Dependent Set of Secrets) *G is the* strictly dependent set of secrets *for a main thread iff the following two conditions are satisfied. For every secret value $\tilde{\alpha}_i^b$ in G,*

1. *The probability that the secret $\tilde{\alpha}_i^b$ is selected by $\mathcal{R}$ AND its value is revealed correctly by $\mathcal{M}$ is at least $\frac{r(k)}{3C_1}$ (for this prefix). Notice that this condition is the same as in the definition of dependent set of commitments and refers to the real honest experiment with the given prefix.*
2. *The probability that the secret $\tilde{\alpha}_i^b$ is selected by $\mathcal{E}$ in a rewinding AND its value is revealed correctly by $\mathcal{M}$ on the right is less than $\frac{r(k)^3}{50\tilde{\ell}(k)^2 C_1 C_2 C_3}$ (i.e., the probability in the experiment where $\mathcal{M}$ gets random strings in places of*

left secret values unrecovered in the main thread).

Again, the first probability in the above definition is dependent only on what the prefix in the main thread is, while, the second one depends on the prefix as well as the left challenge ch appearing in the main thread. We now prove the following lemma.

Lemma 7 *Let S and G respectively be the dependent set and strictly dependent set of secrets of a main thread. Then $G \not\subseteq S$ for at most $\frac{r(k)}{C_3}$ fraction of the main threads. Call these threads as* MT *of type* $\mathsf{bad3}$.

PROOF. Assume, towards contradiction, that for at least a fraction $\frac{r(k)}{C_3}$ of the main threads, there exists a secret value $\tilde{\alpha}_i^b$ in G but not in S. This means the following three conditions hold for these main threads (where the probabilities are taken over the randomness of the experiment after the prefix completion).

1. Consider the condition of a secret not being in S. This means that the second condition of being in S is not satisfied (since the first condition is the same as that in G). Conditioned on the event that $\mathcal{M}$ does not ask any of the values from the unrecovered set of secrets (i.e., its challenge on the left is ch w.r.t. which S and G are defined), $\mathcal{M}$ reveals the correct value of $\tilde{\alpha}_i^b$ on the right with "large" probability (i.e., at least $\frac{r(k)}{2C_2\tilde{\ell}(k)}$).
2. Consider the first condition of being in G. If the secret values in the unrecovered set are given correctly on the left, $\mathcal{M}$ reveals the correct value of $\tilde{\alpha}_i^b$ on the right with "large" probability (i.e., at least $\frac{r(k)}{3C_1}$).
3. Consider the second condition of being in G. That is, if the secret values in the unrecovered set are given randomly on the left (i.e., the response is simulated), $\mathcal{M}$ reveals the correct value of $\tilde{\alpha}_i^b$ on the right with "small"

probability (i.e., less than $\frac{r(k)^3}{50\tilde{\ell}(k)^2 C_1 C_2 C_3}$).

We now construct an adversary $\mathcal{A}$ to show that the above conditions violate the (computational) hiding property of the commitment scheme `WExCS`. Consider the following experiment between the adversary $\mathcal{A}$ and an external challenger *Chal*.

1. $\mathcal{A}$ starts the execution of $\mathcal{M}$ and gives it honestly the messages in the right session. The messages received from $\mathcal{M}$ in the left session are forwarded to *Chal* and its reply is forwarded to $\mathcal{A}$ until the protocol is completed till Step 1.5 (on both left and right interactions).
2. Now *Chal* provides to $\mathcal{A}$ a total of $M = \frac{25\tilde{\ell}(k)^2 C_1 C_2 C_3}{r(k)^3}$ candidate tuples for the values in the unrecovered set of secrets on the left. Exactly one of the candidate tuples has correct values for all the secrets in the unrecovered set. All the values in the rest of the candidate tuples are generated by *Chal* randomly. The goal of $\mathcal{A}$ would be to guess which of the M tuples is the correct one. $\mathcal{A}$ is not allowed any further interaction with *Chal* (including running the protocol beyond step 1).
3. $\mathcal{A}$ now rewinds $\mathcal{M}$ exactly M times. In the i-th rewind, $\mathcal{M}$ gives a challenge $ch[i]$ on the left (if it aborts at any point, we move on the next rewinding). To construct the response, for the secrets in the unrecovered set picked by $ch[i]$, $\mathcal{A}$ uses the values in the i-th candidate tuple. Observe that for exactly one rewind, the response given by $\mathcal{A}$ would be correct and in all other cases, it would be the *simulated* response as given by the extractor $\mathcal{E}$ when it rewinds.
4. $\mathcal{A}$ proceeds as follows. It selects a secret $\tilde{\alpha}_i^b$ from the right interaction as a guess for a secret in $G - S$ (if one exists).
5. Now we consider the case where the following happens. In the main thread,

the secret $\tilde{\alpha}_i^b$ was selected and received by $\mathcal{A}$. There is exactly one rewind (say index *ind*), such that the secret $\tilde{\alpha}_i^b$ was selected by $\mathcal{A}$ AND a value $\tilde{\alpha}_i^b[ind] = \tilde{\alpha}_i^b$ was received (i.e., the values seen in the main thread and this rewind match). If that is the case, $\mathcal{A}$ outputs the index *ind* to *Chal* as its guess for the correct value tuple. In all other cases, $\mathcal{A}$ aborts and outputs $\perp$.

We now analyze the success probability of $\mathcal{A}$. Let E denote the event that main thread is of type $\mathsf{bad3}$, we define the following events:

- $E1$: $(E \wedge \tilde{\alpha}_i^b \in (G - S))$.
- $E2$: ($E1 \wedge$ correct value $\tilde{\alpha}_i^b$ appears in the main thread).
- $E3$: ($E1 \wedge$ correct value $\tilde{\alpha}_i^b$ appears in the rewind with correct response).
- $E4$: ($E1 \wedge$ correct value $\tilde{\alpha}_i^b$ does not appear in any rewind with simulated response).

Then we have the following:

$$\begin{aligned}
&\Pr\big[\mathcal{A} \text{ outputs the correct guess}\big] \\
\geq\ &\Pr\big[E\big] \cdot \Pr\big[E1|E\big] \cdot \Pr\big[E2|E1\big] \cdot \Pr\big[E3|E1\big] \cdot \Pr\big[E4|E1\big].
\end{aligned}$$

Note that the last three probability terms are results of experiments run with independent random coins and hence are independent.

$$\Pr\big[\mathcal{A} \text{ outputs the correct guess}\big] \geq \frac{r(k)}{C_3} \cdot \frac{1}{2\tilde{\ell}(k)} \cdot \frac{r(k)}{2C_2\tilde{\ell}(k)} \cdot \frac{r(k)}{3C_1} \cdot \frac{1}{2}$$

Also note that the expected number of times correct value $\tilde{\alpha}_i^b$ appears in simulated responses is $\frac{r(k)^3}{50\tilde{\ell}(k)^2 C_1 C_2 C_3} \cdot (\frac{25\tilde{\ell}(k)^2 C_1 C_2 C_3}{r(k)^3} - 1) < \frac{1}{2}$, hence at least with probability $\frac{1}{2}$, there are 0 such appearances.)

$$\Pr\left[\mathcal{A} \text{ outputs the correct guess}\right] \geq \frac{r(k)^3}{24\tilde{\ell}(k)^2 C_1 C_2 C_3} \tag{3.2}$$

Claim 3 *In the above experiment, assuming the commitment scheme com is computationally hiding, the probability of any PPT $\mathcal{A}$ outputting the correct guess is bounded by $\frac{r(k)^3}{25\tilde{\ell}(k)^2 C_1 C_2 C_3} + negl(k)$.*

Here we provide a sketch of the proof. This claim relies on a hybrid argument[1]. In the i-th hybrid experiment, in the chosen tuple (out of M tuples) *Chal* keeps the values for the first i unrecovered secrets to be random and the rest correct. In the $\ell(k)$-th hybrid, clearly the probability of $\mathcal{A}$ winning is exactly $\frac{1}{M}$ since the chosen tuple distribution is identical to the rest. Hence, there should exists a hybrid i in which the probability of $\mathcal{A}$ winning changes by a noticeable amount from the last hybrid. Then it can be shown that the hiding property of the commitment scheme WExCom can be broken with a noticeable advantage.

The above claim is in contradiction to the equation (3.2). Thus concludes the proof of Lemma 7. □

Concluding the Analysis. We now conclude the proof of Lemma 4. Very roughly, we have already established that there are only a "small" number of secrets on the right (i.e., secrets in set G) which go from being correct with "large" probability (given a correct response on the left) to being correct only with "small" probability (given a simulated response on the left). Thus, there are sufficiently large number of secrets on the right such that given a simulated response, they are revealed correctly by $\mathcal{M}$ (thus implying success for the extractor $\mathcal{E}$). In more

[1]Since *Chal* provides just the committed values and not any opening to the commitments, there are no issues related to "selective opening attacks" (see [BHY09] and the reference therein).

detail, for the prefix of the given main thread, let p denote the probability that $\mathcal{M}$ completes the main thread (i.e., the real experiment) without aborting (i.e., the probability is taken over the random coins after step 1.(c)). For the given main thread, let q denote the probability of $\mathcal{E}$ succeeding in extracting in a rewinding using a simulated response. Since $\mathcal{E}$ rewinds $\mathcal{M}$ at most δ times,

$$\Pr\left[\mathcal{E} \text{ aborts}\right] \leq p \cdot (1-q)^{\delta}$$

We note that the exact equality may not be satisfied because $\mathcal{M}$ may abort even before prefix completion. Now let $\delta = \frac{k\tilde{\ell}(k)}{r(k)^3}$. Then since $r(k) \geq \frac{1}{\text{poly}(k)}$, it is clear that $\mathcal{E}$ runs in probabilistic polynomial time. Now this value is noticeable only if $q = o(\frac{r(k)^3}{\tilde{\ell}(k)})$, or, in other words, $q < \frac{r(k)^3}{50\tilde{\ell}(k)}C_1C_2C_3$. On the other hand,

$$\begin{aligned}\Pr\left[\mathcal{E} \text{ aborts}\right] \quad &\leq \quad \Pr\left[\mathsf{MT} \text{ is of type } \mathsf{bad1} \text{ or } \mathsf{bad2} \text{ or } \mathsf{bad3}\right] + \\ &\quad\quad \Pr\left[\mathcal{E} \text{ aborts} \mid \mathsf{MT} \text{ is neither of these 3 types}\right]\end{aligned}$$

To compute the second term, we first compute q for the main thread. Note that the main thread being not of type $\mathsf{bad2}$ or $\mathsf{bad3}$ implies that $|G| \leq \ell + \log^2 k$ (since $|S| \leq \ell + \log^2 k$ and $G \subseteq S$). Also, since the main thread is not of type $\mathsf{bad1}$, there are at most $O(\log k)$ secrets in the right interaction for which the probability of getting asked on the right (which happens with probability $\frac{1}{2}$) AND revealed correctly by $\mathcal{M}$ is less than $\frac{r(k)}{3C_1}$ (otherwise, it is easy to show that $p < \frac{r(k)}{C_1}$). Or in other words, there are at least $2\tilde{\ell} - \log^2 k$ secret values on the right with probability of getting asked and revealed correctly is at least $\frac{r(k)}{3C_1}$. Out of these, at most $\ell + \log^2 k$ are in G. Hence, (for large enough k) there are at least $\tilde{\ell} + 1$ secret values (i.e., in other words at least one pair of secrets on the right) such that the probability that such a secret is selected by $\mathcal{E}$ in a rewinding and $\mathcal{M}$ reveals the correct value is at least $\frac{r(k)^3}{50\tilde{\ell}(k)}C_1C_2C_3$. This means for such a main thread, $q \geq \frac{r(k)^3}{50\tilde{\ell}(k)}C_1C_2C_3$. Thus, we complete the proof by having

$$\begin{aligned}\Pr[\mathcal{E} \text{ aborts}] &\leq \frac{r(k)}{C_1} + \frac{r(k)}{C_2} + \frac{r(k)}{C_3} + \text{negl}(k) \\ &\leq \frac{3}{4}\, r(k) + \text{negl}(k).\end{aligned}$$

This concludes the proof of Theorem 2. □

3.4 Getting Full-Fledged Non-Malleable Commitment

We now extend the above construction to get constant round many-many non-malleable commitments (for small tags). The basic construction in Section 3.2 can be extended with only an additive constant increase in the round complexity. Furthermore, the extended scheme is still based only on a black-box use of one-way functions.

3.4.1 Non-Malleable Commitments for Small Tags

We first present a non-malleable commitment scheme for small tags against synchronizing adversaries, therefore removing the "one-sided" limitation from the construction in Section 3.2. This can be achieved in a way very similar to the construction by Goyal [Goy11] (which is based on ideas from Pass and Rosen [PR05a, PR08b]). Let $tag \in [2n]$, $\ell = k \cdot tag$, $\ell' = k \cdot (2n - tag)$ and $\lambda = \lfloor k/4 \rfloor$. Intuitively, the new protocol will sequentially execute two slots such that for exactly one of them, the "tag being used on the right" is larger than the one being used on the left. The idea is that each slot will represent a rewinding opportunity, and the extractor will now rewind the slot that has a larger tag on the right and extract the committed string σ. The extended scheme $\mathtt{NMCS_S}$ for the committer $\mathcal{C}$ and the receiver $\mathcal{R}$ is as follows.

Commitment Phase.

0. **Initial setup.** $\mathcal{R}$ picks λ distinct players at random (i.e., randomly selects λ distinct indices $\Lambda = \{r_1, \ldots, r_\lambda\}$ where $r_i \in [k]$ for any $i \in [\lambda]$). For each r_i, $\mathcal{R}$ sends an extractable commitment c_i of r_i using $\mathtt{ExCS}$.

1. **Primary slots.** Let Π_{Share} be a protocol implementing the Share phase of a $(k+1, \lambda)$-perfectly secure VSS scheme. We require the VSS protocol to have a deterministic reconstruction phase .

   1.1. Given the string σ to commit, $\mathcal{C}$ generates a k-bit random string s, ℓ pairs of random strings $\{\alpha_i^0, \alpha_i^1\}_{i\in[\ell}$ and ℓ' pairs of random strings $\{\alpha'^0_i, \alpha'^1_i\}_{i\in[\ell']}$ of length $4k$ each.

   1.2. $\mathcal{C}$ sets the input of P_{k+1} (i.e., the Dealer) to the concatenation of $\sigma, s, \{\alpha_i^0, \alpha_i^1\}_{i\in[\ell]}$, and $\{\alpha'^0_i, \alpha'^1_i\}_{i\in[\ell']}$ while each other player has no input. Then $\mathcal{C}$ runs Π_{Share} and each player P_i obtains shares w_i, for any $i \in [k]$.

   1.3. Let $view_1^1, \ldots, view_{k+1}^1$ be the views of the $k+1$ players describing the execution of Π_{Share}. $\mathcal{C}$ uses $\mathtt{WExCS}$ to send a commitment V_i^1 of $view_i^1$ to $\mathcal{R}$, in parallel for any $i \in [k]$.

   1.4. $\mathcal{R}$ sends a random ℓ-bit challenge string $ch = (ch_1, \ldots, ch_\ell)$.

   1.5. $\mathcal{C}$ sends $\{\alpha_i^{ch_i}\}_{i\in[\ell]}$ to $\mathcal{R}$.

   1.6. $\mathcal{R}$ sends a random ℓ'-bit challenge string $ch' = (ch'_1, \ldots, ch'_{\ell'})$.

   1.7. $\mathcal{C}$ sends $\{\alpha'^{ch'_i}_i\}_{i\in[\ell']}$ to $\mathcal{R}$.

2. **Verification message.** Let $\mathcal{H}$ be a family of pairwise-independent hash

functions with domain $\{0,1\}^{4k}$ and range $\{0,1\}^k$, and $\texttt{Ext} : \{0,1\}^{4k} \times \{0,1\}^k \rightarrow \{0,1\}^k$ be a strong randomness $(3k, 2^{-k})$-extractor.

2.1. $\mathcal{R}$ picks a function h at random from $\mathcal{H}$ and sends it to $\mathcal{C}$.

2.2. $\mathcal{C}$ sends to $\mathcal{R}$ $s, \{h(\alpha_i^0), h(\alpha_i^1), B_i = \sigma \oplus \texttt{Ext}(\alpha_i^0, s) \oplus \texttt{Ext}(\alpha_i^1, s)\}_{i\in[\ell]}$, and $\{h({\alpha'}_i^0), h({\alpha'}_i^1), B'_i = \sigma \oplus \texttt{Ext}({\alpha'}_i^0, s') \oplus \texttt{Ext}({\alpha'}_i^1, s')\}_{i\in[\ell']}$.

3. **Consistency proof.**

3.1. Let $\Pi_{chch'}$ be a (k, λ)-statistically secure MPC protocol such that given ch, ch' as public input and w_i as private input of P_i for any $i \in [k]$, at the end of the computation $\{\alpha_i^{ch_i}\}_{i\in[\ell]}, \{{\alpha'}_i^{ch'_i}\}_{i\in[\ell']}$ are received in output by P_i for any $i \in [k]$. $\mathcal{C}$ runs internally $\Pi_{chch'}$ and sends a commitment V_i^2 of the view $view_i^2$ of P_i when executing $\Pi_{chch'}$ using $\texttt{WExCS}$ in parallel for any $i \in [k]$ to $\mathcal{R}$.

3.2. Let Π_h be a (k, λ)-statistically secure MPC protocol such that given h as public input and w_i as private input of P_i for any $i \in [k]$, at the end of the computation, the following messages are received in output by P_i for any $i \in [k]$:

– s

– $\{h(\alpha_i^0), h(\alpha_i^1)\}_{i\in[\ell]}, \{B_i = \sigma \oplus \texttt{Ext}(\alpha_i^0, s) \oplus \texttt{Ext}(\alpha_i^1, s)\}_{i\in[\ell]})$

– $\{h({\alpha'}_i^0), h({\alpha'}_i^1)\}_{i\in[\ell']}, \{B'_i = \sigma \oplus \texttt{Ext}({\alpha'}_i^0, s) \oplus \texttt{Ext}({\alpha'}_i^1, s)\}_{i\in[\ell']})$

$\mathcal{C}$ runs internally Π_h and sends to $\mathcal{R}$ a commitment V_i^3 of the view $view_i^3$ of P_i when executing Π_h using $\texttt{WExCS}$ in parallel for any $i \in [k]$.

3.3. $\mathcal{R}$ decommits $\{c_i\}_{i\in[\lambda]}$.

3.4. $\mathcal{C}$ decommits $\{V_{r_i}^1, V_{r_i}^2, V_{r_i}^3\}_{i\in[\lambda]}$.

(i.e., the subset of views $\{view_{r_i}^1, view_{r_i}^2, view_{r_i}^3\}_{i\in[\lambda]}$.)

3.5. For $j = 1, 2, 3$, $\mathcal{R}$ verifies that all pairs of views in $\{view^j_{r_i}\}_{i \in [\lambda]}$ are consistent (according to Definition 14) and that the dealer P_{k+1} has not been disqualified by any player, otherwise $\mathcal{R}$ aborts; moreover for $j = 1, 2$ and $i = 1, \ldots, \lambda$, $\mathcal{R}$ checks that $view^j_{r_i}$ is a prefix of $view^{j+1}_{r_i}$, otherwise $\mathcal{R}$ aborts.

Decommitment Phase.

1. $\mathcal{C}$ decommits $\{V^1_i\}_{i \in [k]}$ as $\{view^1_i\}_{i \in [k]}$.
2. $\mathcal{R}$ checks that all commitments are opened correctly in the previous step and sets a revealed valued to 0^k otherwise.
3. Let Π_{Recon} be a protocol implementing the Recon phase corresponding to the $(k + 1, \lambda)$-perfectly secure VSS Share phase (which includes the string σ) used in the commitment phase. $\mathcal{R}$ runs Π_{Recon} using $view^1_1, \ldots, view^1_{k+1}$ as inputs to reconstruct and output the first substring of the value that the majority of the players would output in the reconstruction. If there is no majority, then output $\perp$.

Theorem 3 *The commitment scheme* $\mathtt{NMCS_S}$ *is a constant-round non-malleable commitment scheme with short tags secure against synchronized adversaries and with black-box use only of one-way functions.*

Proof. The security proof of the protocol $\mathtt{NMCS_S}$ is essentially identical to that of our basic protocol $\mathtt{NMCS}$.

Indeed Lemma 2 will still hold and thus the very same k-bit string is used twice by any PPT sender, including $\mathcal{M}$. Therefore during the proof of non-malleability it will be sufficient to extract that k-bit string from any of the two slots (i.e., Steps 1.4-1.5 and Steps 1.6-1.7). Notice that $\mathcal{M}$ is a synchronizing adversary, $\ell = k \cdot tag$

and $\ell' = k \cdot (2n - tag)$. Assuming $\tilde{tag} \neq tag$, we only need to consider exactly one of the following two cases:

- $\ell < \tilde{\ell}$. The proof of this case is the same as the "one-side" protocol. The extractor $\mathcal{E}$ simply performs its rewindings on Step 1.4 by giving simulated responses for the challenges of $\mathcal{M}$ on the left in Step 1.5.

- $\ell' < \tilde{\ell}'$. In this case $\mathcal{E}$ will rewind to Step 1.6 by giving simulated responses for the challenges of $\mathcal{M}$ on the left in Step 1.7.

In both cases, the proof of security (and in particular the proof of all of our 3 key lemmas bounding the fraction of bad main threads) remains essentially identical (similar to Goyal [Goy11]). $\square$

Concurrent Non-Malleable Commitments for Small Tags. To prove that the above construction is also a many-many (or concurrent) non-malleable commitment scheme for small tags, we first focus on showing one-many security. That is, we consider only a left execution with tag tag and several right executions with tags $\tilde{tag}_1, \ldots, \tilde{tag}_m$. The interesting case is when $\tilde{tag}_i \neq tag$ for all $i \in [m]$. The idea is to simply apply the extractor $\mathcal{E}$ one by one for all m sessions (as in the construction of Goyal [Goy11]). In more detail, for each session $i \in [m]$, do the following.

- Let $\mathcal{M}_i$ be a machine that "emulates" all the right sessions on its own except session i.

- $\mathcal{M}_i$ exposes the i-th session to an outside receiver $\mathcal{R}_i$.

- Given the left view and the right view of the i-th session in the main thread as inputs, run the extractor $\mathcal{E}$ on the machine $\mathcal{M}_i$.

The probability that the extractor fails can be computed by a union bound over the m right sessions (and can be made smaller than $\frac{1}{\text{poly}(k)}$ for any polynomial function poly(k) as in the previous section). By using a result proved in [LPV08], that shows that any one-left many-right non-malleable commitment is also a many-left many-right non-malleable commitment, we obtain the following lemma.

Lemma 8 *There exists a many-many non-malleable commitment scheme for tags of length* $\log(n) + 1$ *that is secure against synchronizing adversaries and only makes a black-box use of a one-way function.*

3.4.2 Security Against Non-Synchronizing Adversaries

Based on the commitment scheme that is secure against synchronizing adversaries, some well known techniques can be applied to extend the basic scheme to obtain one that is even secure against a non-synchronizing adversary. For example, the constructions in [Wee10] and [Goy11] can be used to transform any non-malleable commitment scheme that is secure against synchronizing adversaries into one that is secure against arbitrary scheduling strategies. Similar to Goyal's construction, briefly, the basic idea to get security against non-synchronizing adversaries is to rely on the techniques of robust non-malleability due to Lin and Pass [LP09]. We increase the number of rewinding opportunities such that the primary and the secondary slot become robust w.r.t. the proof of consistency. We are able to achieve this by only relying on one-way functions in a black-box way. We now show how to apply a similar transformation to the commitment scheme $\texttt{NMCS}_\texttt{S}$ to obtain a constant-round non-malleable commitment scheme with black-box use of any one-way function and with security against non-synchronizing adversaries.

The intuition behind the transformation is similar to the one in Goyal's construction [Goy11]. Consider a non-synchronizing adversary $\mathcal{M}$. In this case, the proof already given for synchronizing adversaries does not go through when $\mathcal{M}$ asks for the execution of Step 2 in the left interaction *before* finishing the primary slot in the right interaction. This is because if $\mathcal{M}$ asks for Step 2 during rewinding, our extractor will not be able to answer queries consistently with the messages played in the primary slots. Therefore, by applying the similar ideas from [LP09], we will add additional "secondary slots" to make our proof of security go through. More specifically, each of these additional slots will provide an extra rewinding opportunity. If $\mathcal{M}$ asks for Step 2 on the left *before* finishing Step 1 on the right (in the main thread), it will be possible to exploit these additional rewinding opportunities on the right (such that $\mathcal{M}$ does not ask for messages in the left interaction while $\mathcal{E}$ is rewinding such slots).

Assuming that Step 4 (i.e., consistency proof) in the protocol below requires c_v messages sent out by the committer, the modified protocol $\texttt{NS-NMCS}_\texttt{S}$ proceeds as follows.

Commitment Phase.

0. **Initial setup.** $\mathcal{R}$ picks λ distinct players at random (i.e., randomly selects λ distinct indices $\Lambda = \{r_1, \dots, r_\lambda\}$ where $r_i \in [k]$ for any $i \in [\lambda]$). For each r_i, $\mathcal{R}$ sends an extractable commitment c_i of r_i using $\texttt{ExCS}$.

1. **Primary slots.** Let Π_{Share} be a protocol implementing the Share phase of a $(k+1, \lambda)$-perfectly secure VSS scheme. We require the VSS protocol to

have a deterministic reconstruction phase .

1.1. Given the string σ to commit, $\mathcal{C}$ generates a k-bit random string s, ℓ pairs of random strings $\{\alpha_i^0, \alpha_i^1\}_{i\in[\ell}$ and ℓ' pairs of random strings $\{\alpha'^0_i, \alpha'^1_i\}_{i\in[\ell']}$ of length $4k$ each.

1.2. $\mathcal{C}$ sets the input of P_{k+1} (i.e., the Dealer) to the concatenation of $\sigma, s, \{\alpha_i^0, \alpha_i^1\}_{i\in[\ell]}$, and $\{\alpha'^0_i, \alpha'^1_i\}_{i\in[\ell']}$ while each other player has no input. Then $\mathcal{C}$ runs Π_{Share} and each player P_i obtains shares w_i, for any $i \in [k]$.

1.3. Let $view_1^1, \ldots, view_{k+1}^1$ be the views of the $k+1$ players describing the execution of Π_{Share}. $\mathcal{C}$ uses `WExCS` to send a commitment V_i^1 of $view_i^1$ to $\mathcal{R}$, in parallel for any $i \in [k]$.

1.4. $\mathcal{R}$ sends a random ℓ-bit challenge string $ch = (ch_1, \ldots, ch_\ell)$.

1.5. $\mathcal{C}$ sends $\{\alpha_i^{ch_i}\}_{i\in[\ell]}$ to $\mathcal{R}$.

1.6. $\mathcal{R}$ sends a random ℓ'-bit challenge string $ch' = (ch'_1, \ldots, ch'_{\ell'})$.

1.7. $\mathcal{C}$ sends $\{\alpha'^{ch'_i}_i\}_{i\in[\ell']}$ to $\mathcal{R}$.

2. **Secondary slots.**

For all $j \in [c_v + 1]$, $\mathcal{C}$ sequentially do as follows.

2.j. $\mathcal{C}$ uses `WExCS` to send a commitment W_i^j of $view_i^1$ to $\mathcal{R}$, in parallel for any $i \in [k]$.

3. **Verification message.** Let $\mathcal{H}$ be a family of pairwise-independent hash functions with domain $\{0,1\}^{4k}$ and range $\{0,1\}^k$, and $\texttt{Ext} : \{0,1\}^{4k} \times \{0,1\}^k \to \{0,1\}^k$ be a strong randomness $(3k, 2^{-k})$-extractor.

2.1. $\mathcal{R}$ picks a function h at random from $\mathcal{H}$ and sends it to $\mathcal{C}$.

2.2. $\mathcal{C}$ sends to $\mathcal{R}$ $s, \{h(\alpha_i^0), h(\alpha_i^1), B_i = \sigma \oplus \texttt{Ext}(\alpha_i^0, s) \oplus \texttt{Ext}(\alpha_i^1, s)\}_{i\in[\ell]}$, and $\{h(\alpha'^0_i), h(\alpha'^1_i), B'_i = \sigma \oplus \texttt{Ext}(\alpha'^0_i, s') \oplus \texttt{Ext}(\alpha'^1_i, s')\}_{i\in[\ell']}$.

4. **Consistency proof.**

   4.1. Let $\Pi_{chch'}$ be a (k, λ)-statistically secure MPC protocol such that given ch, ch' as public input and w_i as private input of P_i for any $i \in [k]$, at the end of the computation $\{\alpha_i^{ch_i}\}_{i\in[\ell]}, \{{\alpha'}_i^{ch'_i}\}_{i\in[\ell']}$ are received in output by P_i for any $i \in [k]$. $\mathcal{C}$ runs internally $\Pi_{chch'}$ and sends a commitment V_i^2 of the view $view_i^2$ of P_i when executing $\Pi_{chch'}$ using `WExCS` in parallel for any $i \in [k]$ to $\mathcal{R}$.

   4.2. Let Π_h be a (k, λ)-statistically secure MPC protocol such that given h as public input and w_i as private input of P_i for any $i \in [k]$, at the end of the computation, the following messages are received in output by P_i for any $i \in [k]$:

   – s

   – $\{h(\alpha_i^0), h(\alpha_i^1)\}_{i\in[\ell]}, \{B_i = \sigma \oplus \texttt{Ext}(\alpha_i^0, s) \oplus \texttt{Ext}(\alpha_i^1, s)\}_{i\in[\ell]})$

   – $\{h({\alpha'}_i^0), h({\alpha'}_i^1)\}_{i\in[\ell']}, \{B'_i = \sigma \oplus \texttt{Ext}({\alpha'}_i^0, s) \oplus \texttt{Ext}({\alpha'}_i^1, s)\}_{i\in[\ell']})$

   $\mathcal{C}$ runs internally Π_h and sends to $\mathcal{R}$ a commitment V_i^3 of the view $view_i^3$ of P_i when executing Π_h using `WExCS` in parallel for any $i \in [k]$.

   4.3. $\mathcal{R}$ decommits $\{c_i\}_{i\in[\lambda]}$.

   4.4. $\mathcal{C}$ decommits $\{V_{r_i}^1, V_{r_i}^2, V_{r_i}^3\}_{i\in[\lambda]}$.

   (i.e., the subset of views $\{view_{r_i}^1, view_{r_i}^2, view_{r_i}^3\}_{i\in[\lambda]}$.)

   4.5. $\mathcal{C}$ decommits $\{W_{r_i}^j\}_{i\in[\lambda], j\in[c_v+1]}$.

   4.5. For $j = 1, 2, 3$, $\mathcal{R}$ verifies that all pairs of views in $\{view_{r_i}^j\}_{i\in[\lambda]}$ are consistent (according to Definition 14) and that the dealer P_{k+1} has not been disqualified by any player, otherwise $\mathcal{R}$ aborts; moreover for $j = 1, 2$ and $i = 1, \ldots, \lambda$, $\mathcal{R}$ checks that $view_{r_i}^j$ is a prefix of $view_{r_i}^{j+1}$, otherwise $\mathcal{R}$ aborts. Finally $\mathcal{R}$ verifies that decommitment of $V_{r_i}^1$ is equal to the decommitment of $W_{r_i}^j$ for $i \in [\lambda]$ and $j \in [c_v + 1]$.

Decommitment Phase.

1. $\mathcal{C}$ decommits $\{V_i^1\}_{i\in[k]}$ as $\{view_i^1\}_{i\in[k]}$.
2. $\mathcal{R}$ checks that all commitments are opened correctly in the previous step and sets a revealed valued to 0^k otherwise. Also notice that $\mathcal{R}$ does not need to check decommitment of views committed in Step 2.
3. Let Π_{Recon} be a protocol implementing the Recon phase corresponding to the $(k+1, \lambda)$-perfectly secure VSS Share phase (which includes the string σ) used in the commitment phase. $\mathcal{R}$ runs Π_{Recon} using $view_1^1, \ldots, view_{k+1}^1$ as inputs to reconstruct and output the first substring of the value that the majority of the players would output in the reconstruction. If there is no majority, then output $\perp$.

We first prove the following lemma.

Lemma 9 *Except with negligible probability, $\mathcal{M}$ is successful in the commitment phase only if in all (i.e., both primary and secondary) slots the committed string (i.e., the one that would be reconstructed from the committed views) is the same.*

Proof. If different strings are committed in different slots, we have that at least 1/4 of the views committed in one slot must be different with respect to the views committed in the other slot, otherwise the reconstructions of the strings from the views would generate the same output. The fact that $\mathcal{M}$ is successful, implies that all the indexes of those different views do not belong to Λ. The probability that this happens is negligible if Λ is just a random challenge generated after the commitment of the views. However, since Λ was previously committed, similarly to Lemma 2, here one can break the hiding property of the commitment of Λ by relying on the weak extractability of the commitments of the views. $\square$

Now we claim the security of the new scheme.

Theorem 4 *The above scheme* $\texttt{NS-NMCS}_\texttt{S}$ *is a constant-round non-malleable commitment scheme with short tags secure against non-synchronized adversaries and with black-box use only of a one-way function.*

Proof. We consider the following two different interleavings in the main thread.

- **Case 1: The verification message (i.e., Step 3) in the left interaction is completed *before* the end of the primary slots (i.e., Step 1) in the right interaction.** This is the case where the original proof fails, and secondary slots will be useful. Observe that when this case happens:
  - Since in each session, the verification message is played in the protocol after the secondary slots, *all* the secondary slots in the left interaction are executed (along with the verification message) before the primary slot finishes on the right.
  - Now consider the point where the primary slot in the right interaction finishes. There are at most c_v messages remaining in the left interaction (i.e., message in the consistency proof step) and $c_v + 1$ secondary slots remaining in the right interaction.
  - Hence, by pigeon-hole principle, there exists at least one secondary slot in the right interaction such that during its execution, there are no message in the left interaction. Call this the secondary slot j.

Now the extractor $\mathcal{E}$ can rewind the secondary slot j in the right interaction and extract the value σ in the following way. First, it runs the extractor of $\texttt{WExCS}$. We have that $\mathcal{E}$ obtains a large portion of the views[2] that allows to run the reconstruction of the VSS scheme to output a value in $\{0,1\}^n$ or $\perp$. By Lemma 9, we have that the reconstructed value from views committed in secondary slot j corresponds to the value committed in the primary slot, i.e., the committed value that will be considered in the opening.

If during rewinding, $\mathcal{M}$ changes the scheduling to ask for a message in the consistency proof step (as opposed to its strategy in the main thread), $\mathcal{E}$ simply rewinds and runs again the extractor of $\texttt{ExCS}$ with a different randomness. Since $c_v + 1$ is a constant, these additional rewindings do not harm the polynomial time of $\mathcal{E}$, so that given any $r(k) = \frac{1}{\text{poly}(k)}$, one can construct an extractor which performs a strict polynomial number of rewinds and succeeds with probability at least $(1 - r(k))$.

[2]Because of weak extractability there will be at most $\log^2 k$ non-extracted views.

- **Case 2: The verification message in the left interaction appears *after* the end of primary slots in the right interaction.** The proof for this case is similar to the one for synchronizing adversaries. The only difference is that the secondary slots in the left interaction might now appear before the primary slots in the right interaction finish. However during the rewinds, $\mathcal{E}$ does not have to provide the verification message or the consistency proof (if upon rewinding, $\mathcal{M}$ changes its scheduling to ask for such messages, $\mathcal{E}$ simply rewinds again). Hence during the rewinds, $\mathcal{E}$ can run the required secondary slots without any problem by simply committing to random views. Summing up, the hiding property of `WExCS` guarantees that the extraction goes through.

$\square$

Similarly to before, to prove many-many security of the above scheme, we first prove one-many security by simply applying the extractor $\mathcal{E}$ one by one on all sessions on the right and then resort to a general result of Lin et al. [LPV08]. Therefore we obtain the following theorem.

Theorem 5 *There exists a constant-round many-many non-malleable commitment scheme for tags of length* $\log(n) + 1$ *which is secure against non-synchronizing adversaries and uses a one-way function in a black-box manner.*

CHAPTER 4

Applications

In this chapter, we show how to use our concurrent non-malleable commitment scheme `NMCS` combined with our new use of the *computation in the head* paradigm in order to achieve two additional new results. Assuming a broadcast channel, our first result is a multi-party constant-round parallel coin-tossing protocol with the sole use of one-way functions in a black-box fashion. Our second result is the first non-malleable (with respect to opening) statistically hiding commitment scheme with the sole use of any statistically hiding commitment scheme in a black-box fashion.

4.1 Constant-Round Multi-Party Parallel Coin-Tossing

One of the natural and basic applications of the secure multi-party computation is coin-tossing, which allows parties to jointly generate a common unbiased random output. In this section, we show a constant-round protocol based on the black-box use of any one-way function. In our protocol the adversary can control up to $n-1$ players and computations are performed through a broadcast channel. We will show a simulator that is able to bias the outcome of the joint computation to a specific string that it receives as the input.

4.1.1 Preliminaries

We assume that there exists a broadcast channel and the communication is synchronized but allowing a rushing adversary. That is, our protocol will proceed in rounds, and in each round, all messages exchanged by the players must be delivered before the next round begins. However, in each round, the adversary is allowed to see all the messages sent by honest players before deciding how the corrupted players should behave in current round. Also, the broadcast channel assures that all players heard the same message and that the message cannot be disavowed. Note that in this notion, one string is tossed in a multi-party protocol, which is different to the notions in [Lin01, PW09], where many (single bit) coins are tossed in parallel (i.e, multiple protocol pairs are executed simultaneous by two parties). In the two-party case, a constant-round parallel coin-tossing protocol was proposed by Lindell [Lin01]. In addition, the first constant-round non-malleable string-tossing protocol in the plain model is achieved by Barak [Bar02].

Let Π_C be a synchronized multi-party coin-tossing protocol, let $\mathcal{A}$ denotes the real-world adversary running on auxiliary inputs z, and let S be the ideal-world adversary. We then denote by $REAL_{\Pi_C,\mathcal{A}(z)}(1^n)$ the random variable consisting of the output of $\mathcal{A}$ and the outputs of all parties. We denote by $IDEAL_{f,S(z)}(1^n)$ the analogous output of S and honest parties after an ideal execution with a trusted party computing the n-ary function $f : (1^k)^n \rightarrow \{0,1\}^k$, where each party has only the security parameter k as its input and the output of f is uniformly and independently chosen from $\{0,1\}^k$.

Definition 18 (Synchronized Multi-Party Coin-Tossing)
A synchronized multi-party protocol Π_C implements an n-party coin-tossing functionality f if for every PPT *adversary $\mathcal{A}$ in the real model, there exists a* PPT

adversary S of comparable complexity in the ideal model, such that

$$REAL_{\Pi_C,\mathcal{A}(z)}(1^k) \approx IDEAL_{f,S(z)}(1^k).$$

Notice that the many-many non-malleable commitment scheme based on the black-box use of any one-way function that we have presented in previous chapter also enjoys a (stand-alone) extractability property. The extractor works by running the extractor associated to `ExCS`, therefore obtaining a large number of views from which the reconstruction of the committed value can be easily computed. Given a constant-round many-many non-malleable extractable commitment scheme `NMExCS`, we now present a fully black-box constant-round multi-party parallel coin-tossing protocol.

4.1.2 Our Construction

Let k be the security parameter (i.e, the number of coins being tossed in parallel). Each party P_i acts as sender to send its contribution to the coin-tossing protocol, by running a $(k+1,\lambda)$-perfectly secure VSS scheme with deterministic reconstruction where $\lambda = \lfloor k/4 \rfloor$. To share a random k-bit string σ_i, P_i will use `NMExCS` to commit to the views of the above k VSS players, moreover the above process is repeated twice. Then, a (k,λ)-perfectly secure MPC protocol is invoked to ensure that the same string σ_i has been shared in the two above VSS executions.

Formally, P_i behaves as sender S to commit to his contribution σ to every other party and as receiver R to interact with each of all other parties. An execution of S and R goes as follows.

The $\langle S(\sigma), R\rangle$ sub-protocol.

0. **Initial setup.**

   1. R picks at random λ distinct indices $\Lambda = \{r_1, \ldots, r_\lambda\}$ where $r_i \in [k]$ for all $i \in [\lambda]$. For each r_i, R sends extractable commitments c_i (of r_i) using `ExCS`.
   2. Let $\Pi_{VSSshare}$ be a protocol implementing the Share phase of a $(k+1, \lambda)$-perfectly secure VSS scheme. We require the VSS protocol to have a deterministic reconstruction phase. S picks a random string σ and sets the input of P_{k+1} (i.e., the Dealer) to σ, while each other player has no input. Then S runs $\Pi_{VSSshare}$. Let $view_i^0$ be the view of player P_i for all $i \in [k]$. The same VSS computation is repeated to share again σ but players are allowed to use fresh randomness. Let $view_i^1$ be the view of player P_i for all $i \in [k]$ in this second execution.

1. **1st Slot.** S commits to the views of $P_1, \ldots, P_k$ during the first VSS execution using `NMExCS`. That is, S sends non-malleable extractable commitments $\alpha_{0,i}$ (of $view_i^0$) for all $i \in [k]$ in parallel.
2. **2nd Slot.** S commits to the views of $P_1, \ldots, P_k$ during the second VSS execution using `NMExCS`. That is, S sends non-malleable extractable commitments $\alpha_{1,i}$ (of $view_i^1$) for all $i \in [k]$ in parallel.
3. **Consistency proof.**
   1. Let α_0 be the value reconstructed from all views $view_i^0$, and α_1 be the value reconstructed from all views $view_i^1$ for all $i \in [k]$.
   2. Let Π_{eq} be a (k, λ)-statistically secure MPC protocol such that party P_i is given $view_i^0$ and $view_i^1$ as input for all $i \in [k]$, at the end of the computation, if $\alpha_0 = \alpha_1$, 1 is given in output by all honest parties, otherwise, all honest parties output 0.
   3. S runs internally Π_{eq} and let $view_i^2$ be the view of player P_i during the

execution of Π_{eq} for all $i \in [k]$.

4. S sends non-malleable extractable commitments β_i (of $view_i^2$) using `NMExCS` for all $i \in [k]$ in parallel.

4. **Output.**

   1. R decommits $\{c_i\}_{i\in[\lambda]}$.
   2. S sends σ and decommits $\{\alpha_{0,i}, \alpha_{1,i}, \beta_i\}_{i\in[\lambda]}$ (i.e., the subset of views $\{view_{r_i}^0, view_{r_i}^1, view_{r_i}^2\}_{i\in[\lambda]}$).

      For $j = 0, 1, 2$, R verifies that all pairs of views in $\{view_{r_i}^j\}_{i\in[\lambda]}$ are consistent (according to Definition 14) and that the dealer P_{k+1} has not been disqualified by any player, otherwise R aborts; moreover for $j = 0, 1$ and $i = 1, \ldots, \lambda$, R checks that $view_{r_i}^j$ is a prefix of $view_{r_i}^{j+1}$, otherwise R aborts. If the output in $\{view_{r_i}^2\}_{i\in[\lambda]}$ is not 1, R aborts.
   3. S decommits $\alpha_{1,i}$ for all $i \in [k]$, R verifies that each decommitment is correct otherwise it sets to 0^k a revealed view.
   4. Let $\Pi_{VSSrecon}$ be a protocol implementing the Recon phase corresponding to the $(k+1, \lambda)$-perfectly secure VSS Share phase used above. R runs $\Pi_{VSSrecon}$ using $view_1^1, \ldots, view_k^1$, and computes and outputs the value σ' that the majority of the players obtains in output during the computation. If $\sigma' \neq \sigma$ or there is no majority, then output $\perp$, otherwise output σ.

The fully black-box multi-party constant-round coin-tossing protocol. Each party P_i selects a random k-bit string σ_i and runs in parallel the above sub-protocol $\langle S(\sigma_i), R\rangle$ with every other party. Next, P_i checks that each party played the same string with all other parties, otherwise it aborts. If there is no abort, the final outcome of the protocol is $\sigma = \sigma_1 \oplus \cdots \oplus \sigma_n$.

The use of the broadcast channel guarantees that an adversary is forced in using the same string in all parallel executions, since otherwise if two different strings are noticed in the broadcast channel, honest players would abort.

To show the above protocol is a secure coin-tossing protocol. We shall use the ideal-world v.s. real-world paradigm. In this paradigm, the above protocol is executed in the real-world while the ideal-world has a trusted third-party that implements the functionality and security requirements of the protocol. Then, we show that for any real-world adversary $\mathcal{A}$ there exists an ideal-world adversary $\mathcal{S}$ (i.e., a simulator) such that the following lemma holds.

Lemma 10 *The output views of an execution of the coin-tossing protocol with $\mathcal{A}$ in the real-world is computationally indistinguishable from the output views of running $\mathcal{S}$ in the ideal-world.*

Proof. We first provide the high level idea on how to construct the simulator. Assuming w.l.o.g. that the adversary $\mathcal{A}$ controls $n-1$ players, to simulate the adversary that runs the protocol honestly is trivial. Therefore, in the rest of the proof we show that the simulator $\mathcal{S}$, on input a target random string σ, is able to bias the output of the protocol to σ. First, the simulator $\mathcal{S}$ runs Step 0 and then extracts the set of challenge indices (i.e., Λ) when playing as receiver against each party controlled by the adversary. Then it runs honestly during the 1st slot (Step 1), by using a random string σ^*. Next, the simulator $\mathcal{S}$ extracts the shares committed by parties controlled by the adversary (using the extractor of `NMExCS`), and can therefore use in the 2nd slot (Step 2) an adjusted string σ' so that the XOR of the extracted strings and σ' produces the target string σ. Finally, the simulator can exploit knowledge of Λ to cheat in Step 3 on the views of the player P_{r_i} for all $i \in [\lambda]$, so that the combination of both non-opened views and opened views will be indistinguishable from a honest player execution, even though in this case σ' is different from σ^*.

The security proof of the protocol relies on the extractability and the non-malleability of the underlying commitment scheme. The following sequence of hybrid experiments show the correctness of the above simulator.

Experiment $\mathcal{H}_1$. In this experiment, the simulator $\mathcal{S}$ plays honestly but extracts Λ when playing as sender, and extracts $\sigma_{0,i}$ when playing as receiver with player P_i. More precisely, it honestly runs Step 0 and then extracts the set of challenge indices Λ using the extractor of `ExCS`. Then it runs honestly during the Step 1 (1st slot) by using a random string σ^*. Finally, the simulator $\mathcal{S}$ extracts the shares $\sigma_{0,i}$ committed by parties controlled by the adversary using the extractor of `NMExCS`. By the extractability of `ExCS` and `NMExCS`, the shares revealed by P_i correspond to σ_i and are distributed identically to the shares revealed in the real game.

Experiment $\mathcal{H}_2$. Now consider the experiment where in contrast to $\mathcal{H}_1$, the simulator will use knowledge of Λ when running Step 3. It commits to random strings in the positions out of Λ and commits to honestly simulated views in the positions in Λ. Here the perfect security of the VSS scheme, the fact that only a small portion of the views are opened, and the fact that the employed commitment scheme is non-malleable, guarantee that the views revealed in $\mathcal{H}_2$ by the adversary are computationally indistinguishable from the ones revealed in $\mathcal{H}_1$.

Experiment $\mathcal{H}_3$. Next consider the experiment where in Step 2 the simulator actually uses the adjusted string σ' to generate the views so that the XOR of the extracted σ_i and σ' produces the target string σ. Again, the non-malleability of `NMExCS` guarantees that the distribution of views committed by the adversary in Step 2 does not change, and thus the distribution of the shares opened by the adversary does not change too. This final experiment corresponds to the actual simulation that is therefore successful. $\square$

4.2 Non-Malleable Statistically Hiding Commitment

In contrary to our main construction, another complementary notion of commitment schemes is statistically hiding but computationally binding. We now show that our techniques and our non-malleable commitment scheme can also be used to construct the first non-malleable statistically hiding commitment scheme `NM-SHCS` with black-box use of any statistically hiding commitment scheme.

4.2.1 Preliminaries

Statistically hiding commitment schemes. A commitment scheme is statistically hiding if its hiding property is secure against any unbounded adversary $\mathcal{A}$. That is, in this setting, the hiding property holds even against unbounded adversarial receivers for all but a negligible probability (i.e., statistical hiding), while the binding property is required to hold only for polynomially-bounded senders (i.e., computational binding). It is known how to construct a two-round statistically hiding commitment scheme from any family of collision-resistant hash functions [HM96] or an $O(n/\log n)$-round statistically hiding commitment from any one-way function [HR07, HNO+09]. We will use $\mathtt{SHCS} = (\mathrm{SHCom}, \mathrm{SHRec})$ to denote a two-round statistically hiding commitment scheme.

Statistically hiding non-malleable commitment schemes. The notion of non-malleability that we use for the statistically hiding commitment is that of non-malleability with respect to opening. In the statistically hiding case, the previous definition of non-malleability w.r.t. commitment does not make sense, because the committed value is not necessary well defined. To analyze the non-malleability in such a setting, the standard notion of non-malleability is w.r.t. opening and was studied by Di Crescenzo et al. [DIO98] and by Pass and Rosen [PR05a, PR08b]. Briefly, in the notion of non-malleability w.r.t. opening, the adversary is considered successful if after the commitment phase (where $\mathcal{M}$ commits to a message σ), and after observing the decommitment to σ from a honest committer, $\mathcal{M}$ can decommit a message $\tilde{\sigma}$ that is related to σ.

Let $mim^{\mathcal{M}}_{\mathtt{NMCS}}(\sigma, z, tag)$ denote a random variable that describes the view of $\mathcal{M}$ in the full experiment and the value that $\mathcal{M}$ decommits to in the right execution when the sender commits and decommits to σ. In the simulated experiment, a simulator $\mathcal{S}$ directly interacts with $\mathcal{R}$, and will receive the value σ only after the commitment phase has been completed. Let $sim^{\mathcal{S}}_{\mathtt{NMCS}}(\sigma, z, tag)$ denote the random variable describing the output of $\mathcal{S}$.

Definition 19 (Non-Malleable Commitments w.r.t. Opening)
We say that a tag-based commitment scheme NMCS *is non-malleable w.r.t. opening if for every* PPT *man-in-the-middle adversary $\mathcal{M}$, there exists a (expected)* PPT *simulator $\mathcal{S}$ such that for all $k \in \mathbb{N}$, $tag \in \{0,1\}^k$, $\sigma \in \{0,1\}^k$, and $z \in \{0,1\}^*$, the following ensembles are computationally indistinguishable:*

$$\{mim^{\mathcal{M}}_{\mathtt{NMCS}}(\sigma, z, tag)\} \approx \{sim^{\mathcal{S}}_{\mathtt{NMCS}}(\sigma, z, tag)\}.$$

In our construction, we will need an extractable statistically hiding commitment scheme `SHExCS` $=$ (SHExCom, SHExRec), which however can be easily constructed, similarly to the statistically binding case, by using any statistically hiding commitment scheme as a black-box.

4.2.2 Our Construction

Let k be the security parameter and $\lambda = \lfloor k/4 \rfloor$. Similarly to the construction in Chapter 3, to commit a string σ to a receiver $\mathcal{R}$, the committer $\mathcal{C}$ will run a $(k+1, \lambda)$-perfectly secure VSS protocol in his head with deterministic reconstruction.

Commitment Phase.

0. **Initial setup.** $\mathcal{R}$ picks λ distinct players at random (i.e., randomly selects λ distinct indices $\Lambda = \{r_1, \ldots, r_\lambda\}$ where $r_i \in [k]$ for all $i \in [\lambda]$). For each r_i, $\mathcal{R}$ sends an extractable commitment c_i of r_i using `ExCS`.

1. **Commitment.** Let Π_{Share} be a protocol implementing the Share phase of a $(k+1, \lambda)$-perfectly secure VSS scheme. We require the VSS protocol to have a deterministic reconstruction phase. Given the string σ to commit, $\mathcal{C}$ first sets the input of P_{k+1} (i.e., the Dealer) to σ, while each other player has no input. Then $\mathcal{C}$ runs Π_{Share}. Let $view_i^1$ be the view of player P_i for all $i \in [k]$.

   1.1. $\mathcal{C}$ commits to $view_i^1$ using an extractable statistically hiding commitment scheme `SHExCS`. That is, $\mathcal{C}$ runs `SHExCS` k times in parallel, and let V_i^1 be the transcript of the commitment of $view_i^1$ computed through `SHExCS` for all $i \in [k]$.

Decommitment Phase.

1. Let Π_{Recon} be a protocol implementing the Recon phase corresponding to the $(k+1, \lambda)$-perfectly secure VSS Share phase used in the commitment phase. $\mathcal{C}$ runs Π_{Recon} using $view_1^1, \ldots, view_k^1$ so that the k players reconstruct the shared secret. Let $view_i^2$ be the resulting view of player P_i for all $i \in [k]$.

   1.1. $\mathcal{C}$ sends the original string σ to $\mathcal{R}$.

   1.2. $\mathcal{C}$ commits to $view_i^2$ using a many-many non-malleable commitment scheme NS-NMCS. That is, $\mathcal{C}$ invokes NS-NMCS k times in parallel. Let V_i^2 be the transcript of the commitment of $view_i^2$ computed through NS-NMCS for all $i \in [k]$.

2. $\mathcal{R}$ decommits $\{c_i\}_{i \in [\lambda]}$.

3. $\mathcal{C}$ decommits $\{V_{r_i}^1\}_{i \in [\lambda]}$ (i.e., the subset of views $\{view_{r_i}^1\}_{i \in [\lambda]}$) and views $\{V_{r_i}^2\}_{i \in [\lambda]}$ (i.e., the subset of views $\{view_{r_i}^2\}_{i \in [\lambda]}$.)

4. $\mathcal{R}$ verifies that views $\{view_{r_i}^1\}_{i \in [\lambda]}$ are consistent according to Definition 14, $\{view_{r_i}^2\}_{i \in [\lambda]}$ are consistent according to Definition 14, that $view_{r_i}^1$ corresponds to the first part of $view_{r_i}^2$ for all $i \in [\lambda]$, and that the output in all views $view_i^2$ for all $i \in [\lambda]$ is σ, and outputs σ, otherwise it outputs $\perp$.

Theorem 6 *The commitment scheme* NM-SHCS *is a non-malleable (with respect to opening) statistically hiding commitment scheme with black-box use only of a statistically hiding commitment scheme.*

We now prove that NM-SHCS satisfies the following three properties: statistically hiding, computationally binding and non-malleability with respect to opening by the following lemmas.

Lemma 11 *The commitment scheme* NM-SHCS *is statistically hiding.*

Proof. Since the sender in the commitment phase only commits using statistically hiding commitments (and sends no other information), we have that the claim holds. □

Lemma 12 *The commitment scheme* `NM-SHCS` *is computationally binding.*

Proof. Assume by contradiction that the binding of `NM-SHCS` does not hold. Therefore there is an efficient adversary that provides two accepting openings of the same commitment. Since the output σ of the receiver is the value in views $\{view^2_{r_i}\}_{i\in[\lambda]}$, we have that the adversary succeeds in committing (and decommitting) during the decommitment phase to different views $\{view^2_{r_i}\}_{i\in[\lambda]}$ and $\{view'^2_{r_i}\}_{i\in[\lambda]}$ depending on the message to be opened.

Notice that if different strings are committed in $\{view^1_{r_i}\}_{i\in[\lambda]}$ and $\{view^2_{r_i}\}_{i\in[\lambda]}$ (resp., $\{view'^2_{r_i}\}_{i\in[\lambda]}$), we have that at least 1/4 of the views committed in one slot must be different with respect to the views committed in the other slot, otherwise the reconstructions of the strings from the views would generate the same output. The fact that $\mathcal{M}$ is successful, implies that all the indexes of those different views do not belong to Λ. The probability that this happens is negligible if Λ is just a random challenge generated after the commitment of the views. However, since Λ was previously committed, similarly to Lemma 2, here one can break the hiding property of the commitment of Λ by relying on the weak extractability of the commitments of the views.

□

Lemma 13 *The commitment scheme* `NM-SHCS` *is non-malleable with respect to opening.*

Proof. The security proof of the non-malleability relies on hybrid arguments such that when changing the opened value on the left execution, the value opened in the right execution remains the same. In the following, let $\mathsf{dist}_{\mathcal{M}}(m)$ denote the random variable describing the opening of the adversary in `NM-SHCS` when $\mathcal{C}$ opens to m, and let $\mathsf{dist}_i(m)$ (resp. $\mathsf{dist}_i^{(j)}(m)$) denote the random variable describing the opening of the adversary of `NM-SHCS` when $\mathcal{S}$ opens to m in experiment $\mathcal{H}_i$ (resp. $\mathcal{H}_i^{(j)}$). Then, we consider the following sequence of hybrid experiments.

Experiment $\mathcal{H}_0$. In this experiment, the simulator $\mathcal{S}$ honestly runs $\mathcal{C}$ and interacts with $\mathcal{M}$. That is, in the left interaction, $\mathcal{S}$ honestly commits to the string m_0 to $\mathcal{M}$, while in the right interactions it simply forwards the messages being sent out by $\mathcal{M}$ to $\mathcal{R}$ and vice versa. Clearly, $\mathsf{dist}_{\mathcal{M}}(m_0) \equiv \mathsf{dist}_0(m_0)$.

Experiment $\mathcal{H}_1$. In this experiment, $\mathcal{S}$ executes the protocol identically to $\mathcal{H}_0$ except that it also runs the extractor of `ExCS` to retrieve all the indices r_i selected by $\mathcal{M}$, and it aborts if the extraction fails. After getting r_i for all $i \in [\lambda]$, $\mathcal{S}$ executes the rest of the protocol as in $\mathcal{H}_0$. Since the only difference in the view of $\mathcal{H}_0$ and $\mathcal{H}_1$ consists in the aborts performed when the extraction fails, by the extractability of `ExCS` we have that $\mathsf{dist}_0(m_0) \approx \mathsf{dist}_1(m_0)$. (Notice that if $\mathcal{M}$ completes the commitment phase in the left interaction with non-negligible probability, then the extractor of `ExCS` fails with negligible probability only.)

Experiment $\mathcal{H}_2^{(1)}$ to $\mathcal{H}_2^{(n(k-\lambda)+1)}$. These experiments deviate from $\mathcal{H}_1$ as follows. For all commitments of the views that will not be opened to $\mathcal{M}$ the simulator $\mathcal{S}$ will gradually change the committed values to commitments of random strings. Notice these changed commitments are never opened in the decommitment phase. The statistical hiding of `SHExCS` guarantees that the distribution of the opened message by $\mathcal{M}$ does not change in this considered sequence of experiments. Therefore we have that $\mathsf{dist}_1(m_0) \approx \mathsf{dist}_2^{(1)}(m_0) \approx \cdots \approx \mathsf{dist}_2^{(n(k-\lambda)+1)}(m_0)$.

Experiment $\mathcal{H}_3$. In this experiment, the simulator $\mathcal{S}$ proceeds identically to $\mathcal{H}_2^{(n(k-\lambda)+1)}$ with the following exception. Let $\mathcal{S}_{VSS}$ be the simulator of the Share phase of the underlying $(k+1, \lambda)$-perfectly secure VSS scheme, where $\mathcal{S}_{VSS}$ will simulate the VSS computations with malicious players in the positions of Λ and forcing the output to be m. Then for every player $i \in \Lambda$, $\mathcal{S}$ will commit the shares being generated by $\mathcal{S}_{VSS}$. The main idea is that since λ is below the threshold for the perfect security VSS protocol, the outputs of $\mathcal{S}_{VSS}$ generate these shares so that they are perfectly indistinguishable from the shares of $\mathsf{dist}_2^{(n(k-\lambda)+1)}(m_0)$.

Experiment $\mathcal{H}_4^{(1)}$ to $\mathcal{H}_4^{(n(k-\lambda)+1)}$. These steps are analogous to experiment $\mathcal{H}_2^{(1)}$ to $\mathcal{H}_2^{(n(k-\lambda)+1)}$. In these experiments we proceed identically to $\mathcal{H}_3$, except that for all commitments that will not be opened to $\mathcal{M}$, $\mathcal{S}$ will gradually changes back the committed values from the random strings to shares that are consistent with committed value m_1. By the same arguments in experiments $\mathcal{H}_2^{(m)}$, relying on the binding of the statistically hiding commitment scheme and on the non-malleability of `NS-NMCS`, we have that $\mathsf{dist}_3(m_0) \approx \mathsf{dist}_4^{(1)}(m_1) \approx \cdots \approx \mathsf{dist}_4^{(n(k-\lambda)+1)}(m_1)$.

Experiment $\mathcal{H}_5$. In this experiment, the simulator honestly executes the protocol by committing to m_1 and giving in output the corresponding views. The only difference between $\mathcal{H}_5$ and $\mathcal{H}_4^{(n(k-\lambda)+1)}$ is that now the opened shares are not simulated anymore. However, notice that these opened shares were previously perfectly indistinguishable with respect to simulated shares. Thus, $\mathcal{H}_4^{(n(k-\lambda)+1)} \equiv$

$\mathsf{dist}_5(m_1) \equiv \mathsf{dist}_{\mathcal{M}}(m_1)$.

Therefore, by applying the sequences of hybrid experiments shown above, we have $\mathsf{dist}_{\mathcal{M}}(m_0) \approx \mathsf{dist}_{\mathcal{M}}(m_1)$.

$\square$

References

[AL11] Gilad Asharov and Yehuda Lindell. A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:36, 2011.

[Bar01] Boaz Barak. How to Go Beyond the Black-Box Simulation Barrier. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 106–115, 2001.

[Bar02] Boaz Barak. Constant-Round Coin-Tossing with a Man in the Middle or Realizing the Shared Random String Model. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '02, pages 345–355, 2002.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally Composable Protocols with Relaxed Set-Up Assumptions. In *Proceedings of the 45rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 186–195, 2004.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC '88, pages 1–10, 1988.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and Impossibility Results for Encryption and Commitment Secure under Selective Opening. In *Advances in Cryptology — EUROCRYPT '09*, volume 5479 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2009.

[BS05] Boaz Barak and Amit Sahai. How To Play Almost Any Mental Game Over The Net - Concurrent Composition via Super-Polynomial Simulation. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 543–552, 2005.

[CDD$^+$99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient Multiparty Computations Secure Against an Adaptive Adversary. In *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 1999.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '85, pages 383–395, 1985.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, STOC '02, 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 *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '10, pages 541–550, 2010.

[CVZ10] Zhenfu Cao, Ivan Visconti, and Zongyang Zhang. Constant-round concurrent non-malleable statistically binding commitments and decommitments. In *Public Key Cryptography - PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 193–208. Springer, 2010.

[CVZ11] Zhenfu Cao, Ivan Visconti, and Zongyang Zhang. On constant-round concurrent non-malleable proof systems. *Inf. Process. Lett.*, 111(18):883–890, 2011.

[CW79] J. Lawrence Carter and Mark N. Wegman. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.*, 18(2):143 – 154, 1979.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography (Extended Abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 542–552, 1991.

[DIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 141–150. ACM, 1998.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, STOC '01, pages 580–589. ACM, 2001.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2001.

[Goy11] Vipul Goyal. Constant Round Non-malleable Protocols Using One-way Functions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 695–704. ACM, 2011.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HM96] Shai Halevi and Silvio Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 201–215. Springer-Verlag, 1996.

[HNO$^+$09] Iftach Haitner, Minh-Huyen Nguyen, Shien Jin Ong, Omer Reingold, and Salil P. Vadhan. Statistically Hiding Commitments and Statistical Zero-Knowledge Arguments from Any One-Way Function. *SIAM J. Comput.*, 39(3):1153–1218, 2009.

[HR07] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 1–10. ACM, 2007.

[IK04] Yuval Ishai and Eyal Kushilevitz. On the Hardness of Information-Theoretic Multiparty Computation. In *Advances in Cryptology — EUROCRYPT '04*, pages 439–455, 2004.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, STOC '06, pages 99–108, 2006.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from Secure Multiparty Computation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, STOC '07, pages 21–30, 2007.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-Knowledge Proofs from Secure Multiparty Computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[Kat07] Jonathan Katz. Universally Composable Multi-party Computation Using Tamper-Proof Hardware. In *Advances in Cryptology — EUROCRYPT '07*, pages 115–128, 2007.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round Efficiency of Multi-party Computation with a Dishonest Majority. In *Advances in Cryptology — EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 578–595. Springer, 2003.

[Lin01] Yehuda Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 171–189. Springer-Verlag, 2001.

[LP09] Huijia Lin and Rafael Pass. Non-malleability Amplification. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 189–198, 2009.

[LP11] Huijia Lin and Rafael Pass. Constant-round Non-malleable Commitments from Any One-way Function. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 705–714, 2011.

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent Non-malleable Commitments from Any One-Way Function. In *Theory of Cryptography, 5th Theory of Cryptography Conference, TCC 2008*, pages 571–588, 2008.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A Unified Framework for Concurrent Security: Universal Composability from Stand-alone Non-malleability. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 179–188, 2009.

[MOSV06] Daniele Micciancio, Shien Jin Ong, Amit Sahai, and Salil P. Vadhan. Concurrent Zero Knowledge Without Complexity Assumptions. In *Theory of Cryptography, Third Theory of Cryptography Conference,TCC 2006*, pages 1–20, 2006.

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. Input-Indistinguishable Computation. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 367–378, 2006.

[Nao91] Moni Naor. Bit Commitment Using Pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NT99] Noam Nisan and Amnon Ta-Shma. Extracting Randomness: A Survey and New Constructions. *J. Comput. Syst. Sci.*, 58(1):148 – 173, 1999.

[NZ96] Noam Nisan and David Zuckerman. Randomness is Linear in Space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[OPV08] Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. Constant-round concurrent non-malleable zero knowledge in the bare public-key model. In *Automata, Languages and Programming - ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 548–559. Springer, 2008.

[OPV09] Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. Simulation-based concurrent non-malleable commitments and decommitments. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 91–108. Springer, 2009.

[OPV10] Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. Efficiency Preserving Transformations for Concurrent Non-malleable Zero Knowledge. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010*, pages 535–552, 2010.

[Pas03] Rafael Pass. Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition. In *Advances in Cryptology — EUROCRYPT '03*, pages 160–176, 2003.

[Pas04] Rafael Pass. Bounded-Concurrent Secure Multi-Party Computation with a Dishonest Majority. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, STOC '04, pages 232–241, 2004.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive One-Way Functions and Applications. In *Advances in Cryptology — CRYPTO '08*, pages 57–74, 2008.

[PR05a] 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, 2005.

[PR05b] Rafael Pass and Alon Rosen. Concurrent Non-Malleable Commitments. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of ComputerScience*, FOCS '05, pages 563–572, 2005.

[PR08a] Rafael Pass and Alon Rosen. Concurrent Nonmalleable Commitments. *SIAM J. Comput.*, 37(6):1891–1925, 2008.

[PR08b] Rafael Pass and Alon Rosen. New and Improved Constructions of Nonmalleable Cryptographic Protocols. *SIAM J. Comput.*, 38(2):702–752, 2008.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent Zero Knowledge with Logarithmic Round-Complexity. In *Proceedings of the 43th Annual IEEE Symposium on Foundations of ComputerScience*, FOCS '02, pages 366–375, 2002.

[PS04] Manoj Prabhakaran and Amit Sahai. New Notions of Security: Achieving Universal Composability without Trusted Setup. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 242–251, 2004.

[PW09] Rafael Pass and Hoeteck Wee. Black-Box Constructions of Two-Party Protocols from One-Way Functions. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009*, pages 403–418, 2009.

[PW10] Rafael Pass and Hoeteck Wee. Constant-Round Non-malleable Commitments from Sub-exponential One-Way Functions. In *Advances in Cryptology — EUROCRYPT '10*, pages 638–655, 2010.

[Ros04] Alon Rosen. A Note on Constant-Round Zero-Knowledge Proofs for NP. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, pages 191–202, 2004.

[Sha79] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[VV08] Carmine Ventre and Ivan Visconti. Completely non-malleable encryption revisited. In *Public Key Cryptography - PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2008.

[WC81] Mark N. Wegman and J. Lawrence Carter. New Hash Functions and Their Use in Authentication and Set Equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

[Wee10] Hoeteck Wee. Black-Box, Round-Efficient Secure Computation via Non-malleability Amplification. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '10, pages 531–540, 2010.