Non-Malleable Multi-Prover Interactive Proofs and Witness Signatures

Vipul Goyal [*] Aayush Jain [†] Dakshita Khurana [‡]

Abstract

We explore a new man-in-the-middle adversarial model for multi-prover interactive proofs (MIPs), and construct round-optimal, unconditionally secure, non-malleable MIPs. We compile from a large sub-class of Σ-protocols to a non-malleable MIP, avoiding the use of expensive NP-reductions to Graph Hamiltonicity or other NP-complete problems. Our compiler makes novel use of non-malleable codes – in particular, we rely on *many-many* non-malleable codes constructed recently by Chattopadhyay, Goyal and Li (STOC 2016).

We introduce another (seemingly unrelated) primitive – witness signatures – which are motivated by the goal of removing central trust assumptions from cryptography. Witness signatures allow any party with a valid witness to an NP statement to sign a message *on behalf* of that statement. These signatures must be unforgeable – that is, signing a new message, even given several signatures, should be as hard as computing a witness to the NP statement itself.

We first observe that most natural notions of witness signatures are impossible to achieve in the plain model. While still wanting to avoid a central trusted setup, we turn to the tamper proof hardware token model of Katz (Eurocrypt 2007). We show that *non-malleable* MIPs yield efficient, unconditional witness signatures in the hardware token model. However, our construction of unconditional witness signatures only supports bounded verification. We also obtain unbounded polynomial verification assuming the existence of one-way functions. Finally, we give a matching lower bound – obtaining unconditional unbounded-verifiable witness signatures with black-box extraction, is impossible even with access to an unbounded number of stateful tamper-proof hardware tokens.

[*]Microsoft Research, India. Email: vipul@microsoft.com.
[†]UCLA, USA. Email: aayushjainiitd@gmail.com. Work done in part while at Microsoft Research, India.
[‡]UCLA, USA. Email: dakshita@cs.ucla.edu. Work done in part during an internship at Microsoft Research, India.

Contents

1 Introduction

Ben-Or, Goldwasser, Kilian and Wigderson [BGKW88] introduced the model of multi-prover interactive proofs, with the goal of obtaining zero-knowledge proofs for NP without any intractability assumptions. They generalized the interactive proof model to allow two isolated provers that share secret randomness and communicate with the same single verifier. This allowed them to obtain perfect zero-knowledge proofs by exhibiting a two-prover protocol that obtained statistical soundness via *sequential* repetition.

Lapidot and Shamir [LS91] proved security under parallel repetition of a two-prover ZK protocol for Graph Hamiltonicity, thereby giving a one-round two-prover ZK proof system. This was later generalized by Raz [Raz95] who showed that parallel repetition of any two-prover one-round proof system decreases the soundness error at an exponential rate. In this work, we focus on Σ-protocols because of their efficiency.

Sigma (Σ) protocols are three-round proof systems that satisfy special-soundness and are honest-verifier zero-knowledge. We demonstrate a compiler from a large class of Σ-protocol to an unconditional one-round two-prover ZK proof. In particular, this gives MIPs for many languages that admit Σ-protocols without going via expensive NP reductions. This result can also be used to obtain MIPs for all of NP via the standard Σ-protocol for Graph Hamiltonicity.

1.1 Non-Malleable MIPs

The concept of non-malleability was introduced in the seminal work of Dolev, Dwork and Naor [DDN91, DDN00]. It has proven fundamental to several areas of cryptography such as CCA2-secure encryption scheme [NY90, DDN91, Sah99], composable multi-party computation [CLOS02, Pas04, LPV09], privacy amplification and non-malleable extractors [DW09, DPW10, ADL14, CGL15], hash functions [BCFW09], etc.

In this paper, we initiate the study of non-malleable MIPs, which are a natural extension of multi-prover interactive proofs. Consider the following scenario: a pair of isolated honest provers (P_1, P_2) execute an MIP to prove some statement x to a pair of isolated verifiers $V_{\mathsf{MIM},1}$ and $V_{\mathsf{MIM},2}$ (instead of to a single verifier). The provers (P_1, P_2) are allowed to share secret randomness, and so are the verifiers $(V_{\mathsf{MIM},1}, V_{\mathsf{MIM},2})$. In turn, these verifiers $(V_{\mathsf{MIM},1}, V_{\mathsf{MIM},2})$ execute an MIP as two isolated cheating provers $(P_{\mathsf{MIM},1}, P_{\mathsf{MIM},2})$ convincing some honest verifier V on a related statement x' (or possibly x itself), for which they lack a witness. Our security definition ensures that two man-in-the-middle verifiers cannot maul an honestly generated proof for any statement $x \in L$ and tag, to a two-prover proof for any other statement $x' \in L'$ on a different tag' unless they know a witness for x'. At the heart of our techniques lies a novel method of using non-malleable codes to achieve non-malleability in protocols.

Non-Malleable Codes. Non-malleable codes were introduced by Dziembowski, Pietrzak and Wichs [DPW10]. They allow encoding a message such that an adversary that applies a function f to the codeword, from a limited class of tampering functions, cannot output a codeword that decodes to a related message. Since their introduction, there has been a body of work on obtaining non-malleable codes for various classes of tampering functions [LL12, CCFP11, CKM10, CCP12, FMVW14, CG14, FMNV14, ADL14, CZ14, AGM+15, ADKO15, CGL15].

In this paper, we use non-malleable codes against split-state tampering functions. A split-state non-malleable code encodes some message M into two parts (L, R), such that any adversary tampering independently with L and R, cannot obtain a codeword that decodes to a message that is related to M. The response of the second prover corresponds to a split-state function that a man-in-the-middle must tamper. A split-state non-malleable code is also a non-malleable secret sharing scheme, such that any adversary that tampers shares independently cannot reconstruct to a related message. Our compiler makes use of many-many non-malleable codes in the split-state model, that were recently introduced by Chattopadhyay et al. [CGL15].

Not only are non-malleable MIPs interesting in their own right, we show that they also have strong implications to a special kind of signature scheme, which we call a witness signature. We discuss this primitive next.

1.2 Witness Signatures

Suppose a government promises to send a huge cash prize directly to the address (or bank account) of anybody who can solve a hard puzzle. A scientist solves the puzzle but he does not want to give out the solution to the government unless he gets his prize money.

"How should he convince them that he knows a solution to the puzzle, while ensuring that the money gets delivered to his address?"

In this paper, we study the notion of a *witness signature*, that offers a simple solution to this problem. Roughly, a witness signature allows any party with a witness to some NP statement x to sign a message m, such that anyone can verify that the message was indeed signed by someone with the knowledge of a valid witness for x. These signatures should also be unforgeable: that is, signing a fresh message (even given several other signed messages) should be as hard as computing a witness to the NP statement itself.

Given such a primitive, the scientist can directly sign his address using his solution to the puzzle and send this in clear to the government. The unforgeability of the signature will ensure that nobody without a witness can tamper with the address (or bank account) in the signature; or create a new signature.

Witness-Based Cryptography. Witness signatures can be seen as the signature analogues of witness encryption [GGSW13] and witness PRFs [Zha16]. Witness encryption allows any party to encrypt a message to some NP statement – such that decrypting the message requires access to a witness. A witness PRF is a special kind of pseudo-random function, such that anyone with a valid witness to some NP statement $x \in L$ can evaluate the PRF on x without the secret PRF key, while for $x \notin L$, the PRF evaluation on x is computationally hidden without knowledge of the secret key. The central idea of such witness-based cryptography is to base hardness on NP puzzles, with trapdoors comprised by the solutions to these puzzles. This removes the need for trusted setup or prior communication; such that not even publicly-available keys are required.

However, unlike witness encryption, the notion of witness signatures is not entirely new. Various primitives similar in spirit to witness signatures have been studied in the past decade. Witness signatures appear to be closely related to non-malleable non-interactive zero-knowledge arguments of knowledge (NIZKAoK) [Sah99, SCO+01]. To construct witness signatures from (tag-based) non-malleable NIZKAoKs, the signer can give a non-interactive zero-knowledge argument of knowledge of the witness to the NP statement, non-malleably tagged by the message being signed.

Chase and Lysyanskaya [CL06] studied another very similar primitive called 'signatures of knowledge', that has interesting applications to delegatable anonymous credentials. This allows a signer to sign on behalf of an NP statement, while additionally ensuring that the signature is zero knowledge. However, common to both these primitives is the necessity for a common reference string (CRS) generated by a trusted setup. On the other hand, we regard avoiding any trust in a central setup or key generation/exchange, as one of the primary goals of "witness-based cryptography".

The objective of this work is to explore the problem of constructing witness signatures in alternate models where no central trusted setup or prior communication is required (in keeping with developing cryptography based on hard problems alone and no keys or setups). But we demonstrate that shooting for a construction in the plain model is perhaps too ambitious. In fact, for a very natural definition of witness signatures, obtaining a construction in the plain model with a black-box security reduction is impossible unless $\mathsf{BPP} = \mathsf{NP}$.

Witness Signatures Using Stateful Tamper-Proof Hardware Tokens. To get around the plain model impossibility, we resort to the tamper-proof hardware model of Katz [Kat07] which allows us to bypass any central setup or trust assumptions (and in fact allows us to get an unconditionally secure construction assuming stateful hardware). The key feature of this model is that one does not need to place any trust in the hardware tokens: in particular a dishonest party may construct such tokens maliciously and may query the received hardware tokens in any way it wishes.

Our goal is to provide unconditional constructions of witness signatures in the hardware token model, while using a minimal number of hardware tokens. A key observation is that witness signatures in the hardware token model are closely related to the notion of *non-malleable* MIPs explained above.

Our construction of witness signatures comprises only two (stateful) tokens generated by the signer, that act as two provers of a non-malleable two-prover zero-knowledge proof. It is straightforward to see that this yields unconditionally secure witness signatures in the stateful hardware token model. However, this construction only allows for a polynomially bounded number of verifications. We also give an alternate construction based on one-way functions, that allows an unbounded number of verifications. Finally, we provide a matching lower bound – it is impossible to obtain unconditionally secure witness signatures that verify an unbounded number of times, using any finite number of stateful hardware tokens.

Other Applications. We believe that similar to witness encryption, witness signatures is a fundamental and theoretically intriguing primitive. Therefore, we believe that a systematic study of this primitive is justified regardless of applications to other cryptographic primitives. However we also remark that several applications of non-malleable NIZKs and signatures of knowledge, in fact, do apply to witness signatures as well (and hence unconditional witness signatures in the hardware token model would lead to those applications in such a model as well). Two such examples are ring signatures and delegatable anonymous credentials, and Chase and Lysyanskaya [CL06] realized these primitives in the CRS model[1]. Our construction of witness signatures would allow us to realize these primitives without a central setup (albeit assuming tamper-proof hardware).

1.3 Our Techniques: Non-Malleable MIPs

As our main contribution, we construct an unconditionally secure non-malleable MIP directly from a subclass of Σ protocols, which we refer to as special Σ-protocols. Informally, special Σ-protocols are Σ protocols with the following additional structural requirement on the third round: Depending on the challenge vector, the third message of the prover must necessarily consist of decommitments to a (proper or improper) subset of the vector committed to in the first message. Our results can be captured by the following informal theorem and its corollary.

Informal Theorem 1. *There exists a round-optimal non-malleable MIP for any language that admits a special Σ-protocol.*

Informal Corollary 1. *There exist round-optimal non-malleable MIPs for all of NP.*

Informal Theorem 1 allows us to avoid expensive NP reductions while obtaining non-malleable MIPs for languages that admit special Σ-protocols. Informal Corollary 1 is obtained by applying Informal Theorem 1 to the Blum protocol for Graph Hamiltonicity, which is also a special Σ-protocol, and using an NP-reduction from Graph Hamiltonicity to any language in NP. Our construction relies heavily on non-malleable codes, and we outline our techniques below.

Two-prover commitments. To provide a good intuition about our ideas, we start by discussing a well-known simple unconditional commitment scheme in the two-prover model. The provers, in order to commit to a value X, secret shares X using a simple 2-out-of-2 secret sharing (XOR encoding) into shares (A, B). After the provers are isolated, the verifier queries the second prover only, and depending upon the challenge string, the prover outputs either A or B.

To decommit, the first (isolated) prover outputs both values (A, B) to the verifier. It is easy to see that this is a perfectly hiding, $1/2$-binding commitment to X. If the first prover tries to decommit, he must change at least one of A or B. Since he is isolated from the second prover, verification fails for such a cheating prover with probability at least $1/2$.

Compiling Σ-Protocols to Two-Prover Proofs. The commitment scheme described above helps directly compile from a large class of Σ-protocols to unconditional single round two-prover proofs. Σ-protocols are proof systems that follow a special commit-challenge-response structure. In particular, we consider the sub-class of Σ-protocols where the prover commits to a vector of values X in the first round. The verifier's

[1] However, the definitions of [CL06] were much stronger and also gave rise to group signatures. Group signatures, by definition, require a trapdoor and therefore cannot exist in our setup-free world.

challenge defines a function f, and in the third message, the prover must decommit to a subset of indices of X defined by $f(X)$. The verifier accepts if the prover's third message passes certain public checks.

In the two-prover setting, the second prover P_2 can be used to commit to a vector X by saving an ordered secret sharing (A, B) of X. The verifier queries P_2 and obtains output either A or B depending upon its challenge. The verifier queries the first prover P_1 directly with the challenge f and P_1 reveals all shares $(f(A), f(B))$ in the subset of indices defined by $f(\cdot)$. The verifier checks these shares against the shares (either A or B) opened by P_2.

At this point, we have a (completely malleable) two-prover ZK proof. In a malleation scenario, however, the two isolated MIM provers could orient themselves arbitrarily in order to interact (as verifiers) with an arbitrary disjoint subset of provers P_1 and P_2. For instance, a setup where $P_{\mathsf{MIM},1}$ obtains the output of P_1 and P_2 while $P_{\mathsf{MIM},2}$ does not interact with any of these provers, is allowed. Our protocol works for arbitrary orientations of the man-in-the-middle, but for this overview, we only focus on the most interesting setting where $P_{\mathsf{MIM},1}$ interacts with P_1 and $P_{\mathsf{MIM},2}$ interacts with P_2.

Non-Malleable Secret Sharing. Let us explore what happens if we use a split-state non-malleable secret sharing scheme to obtain the shares (A, B). Now, a man-in-the-middle verifier ($P_{\mathsf{MIM},2}$) that obtains only one share (either A or B) from the P_2, can only tamper one of the shares at a time, while the other one is perfectly hidden. Thus, the commitment to X becomes non-malleable. It appears that porting Σ-protocols to a two-prover proof while using this new non-malleable commitment scheme should directly yield a non-malleable two-prover ZK proof.

Unfortunately, this idea does not directly work. This is because Σ-protocols (including the Graph Hamiltonicity protocol of Lapidot-Shamir [LS91]), in order to have the honest-verifier zero knowledge property, require the prover to open only a subset of his commitments from the first message (i.e., a subset of components of X), depending upon the challenge query f. Opening a subset of components of X is easily possible if a XOR encoding (ordinary secret sharing) is used. But if a non-malleable encoding was used to commit to the entire X vector at once, it is no longer clear how to reveal only a subset of components of X.

Therefore, we we separately encode each component of X using a non-malleable encoding. However, proving security of this construction requires a reduction to one-many split-state non-malleable codes (that are a generalization of the standard split-state non-malleable codes). Such codes were constructed recently by Chattopadhyay, Goyal and Li [CGL15]. For technical reasons, we make a minor modification their construction to obtain an additional property from one-many non-malleable codes (we call this the symmetric decryption property). The resulting codes suffice to complete the argument outlined above.

We prove security of our construction by demonstrating the existence of a simulator-extractor which builds on the simulator for standalone 2-prover ZK. Note that the man-in-the-middle (MIM) cheating provers could orient themselves as verifiers of any *disjoint*[2] subset of honest provers, as long as they are isolated. For example, a $P_{\mathsf{MIM},1}$ could be acting as a verifier to both left provers P_1 and P_2, while the second MIM prover interacts with none of them.

This requires us to perform a meticulous cases analysis over various possible orientations. However, we show that in the most technically challenging case, the second MIM prover behaves like a split-state tampering function over the shares output by the second honest prover, in the real and rewinding executions. Then, we are able to argue non-malleability via a careful reduction to the security of the underlying many-many non-malleable codes. Proving security across all possible cases requires some additional ideas. Please refer to Section 3 for a detailed analysis.

1.4 Our Techniques: Witness Signatures

Formalizing the Notion. We begin by formalizing the most natural definition of witness signatures, without setup, in the standard model. To capture the notion of unforgeability, we require that for any forger that creates a forgery on some NP-hard instance x, which successfully verifies with probability p, there exists a reduction which interacts with such a forger in a fully black-box manner, and outputs a witness for x with probability at least $\mathsf{poly}(p)$, for some fixed polynomial $\mathsf{poly}(\cdot)$. An additional natural property we also require is that the signatures should be witness indistinguishable. This property gives anonymity to the signers, and is crucial in some applications.

[2]Since $P_{\mathsf{MIM},1}$ and $P_{\mathsf{MIM},2}$ are isolated, they cannot both be verifiers to the same left prover (P_1 or P_2) in the same session.

However, we observe that (unlike their encryption/PRF counterparts) witness signatures for this most natural definition are impossible to securely realize without setup in the standard model, unless $\mathsf{BPP} = \mathsf{NP}$. Intuitively, this is because the black-box reduction – which neither has a witness, nor any trapdoor in the standard model without setup – must issue valid signatures to obtain any output from a forger. However, since the signatures are non-interactive, the reduction cannot even rewind the forger in order to simulate 'fake' signatures. This means that the black-box reduction must necessarily have some extra power over a real signer; rendering most natural definitions impossible unless $\mathsf{BPP} = \mathsf{NP}$.

Witness Signatures in the Hardware Token Model. As our main contribution, we construct efficient witness signatures in the hardware token model. This model has been used previously in the literature, in the context of UC-secure computation, to remove setup or trust assumptions. In the *stateful* hardware token model, we use K-OTM tokens, which are simple tokens that implement K parallel executions of the string OT functionality. We also make the following natural assumption – two (possibly malicious) tokens issued by a signer, when queried in isolation, cannot communicate with each other. Then, we obtain the following main results.

Informal Theorem 2. *There exists an unconditionally secure protocol for witness signatures in the stateful hardware token model with malicious encapsulation, for any language that admits a special Σ-protocol, using only two tokens, each of which implements K parallel oblivious transfers. This protocol realizes a-priori bounded verifiable witness signatures.*

Informal Corollary 2. *There exists an unconditionally secure protocol for witness signatures in the stateful hardware token model with malicious encapsulation, for any language in* NP*, using only two tokens, each of which implements K parallel oblivious transfers. This protocol realizes a-priori bounded verifiable witness signatures.*

This means that the given signature may only be verified a bounded number of times (and the tokens stop responding after that). The protocol simply comprises of having the signer issue two tokens that act as two isolated provers of a non-malleable MIP.

Informal Theorem 3. *An unconditionally secure protocol for witness signatures, cannot exist even in the stateful token model, if unbounded verifiability is required.*

Our impossibility result holds even in the weaker stateful token model where a malicious adversary is not allowed to encapsulate tokens, and even if honest parties can create stateful tokens implementing arbitrary functionalities (other than OT). To prove this, we use techniques from the work of Goyal et al [GIMS10] which in turn borrow from the literature on black-box separations. In particular, they rely on the notion of *inaccessible entropy* that was introduced in [HHRS07, HRVW09], to construct an algorithm that learns most of the entropy of any *stateless* token. Their treatment crucially relies on the fact that a stateless token can be modeled as black-box access to a function. However, this is not true in case of stateful tokens. In particular, given the same query a second time, a stateful token may change its output – and it is no longer clear how to construct an entropy learner in this setting.

Interestingly, we show that it is possible to extend the result of [GIMS10] (which works for a single stateless token) to an unbounded number of deterministic stateful tokens with bounded entropy. We then use such a learner for our impossibility result as follows. If an unbounded number of queries are allowed, then a set of bounded-entropy tokens end up revealing all their (combined) secrets to the learner. Very roughly, this brings us back to the plain model, where witness signatures are impossible. While this was an oversimplified overview, the actual argument requires much more effort to make it work, and forms the second main result of our paper.

Informal Theorem 4. *There exists a protocol for witness signatures in the stateful hardware token model with encapsulation, that uses two hardware tokens and realizes unbounded verifiable witness signatures, assuming the existence of one-way functions.*

This construction uses pseudo-random functions to generate fresh correlated pseudo-entropy within the tokens, for each verification step of the two-prover proof.

On our Model and the Optimality of our Construction. Goyal et al. [GIS+10] obtain general non-interactive unconditional UC secure computation with stateless hardware tokens, but their model requires making the assumption that every adversary knows the code of the token he creates.

However, we work in a stronger model than [GIS+10], where we allow a forger to build a *wrapper* around some honestly created token, to create a new token before passing it on to other parties. The outer token may relay queries to the honest token embedded inside it, and modify the response of the embedded token before relaying it to the outer world. This resembles, but is incomparable to, the model considered in [CGS08, MS08]. While [CGS08] work in the *stateless* token model to obtain *interactive* UC secure computation, Moran and Segev [MS08] work in a model with only uni-directional transfer of tokens.

As already pointed out, our construction is very efficient in terms of computation and communication complexities, because it relies on running an efficient non-malleable MIP. Our construction is also optimal in terms of number of tokens: any unconditional construction of witness signatures *requires* at least two tamper proof hardware tokens. We cannot hope to obtain an unconditional construction where a signer issues only a single hardware token, because then the single token would be solely responsible for giving an (unconditional) witness hiding proof for NP, which is impossible.

2 Preliminaries

In this section, we recall some definitions and introduce some notation for use in this paper. Let κ denote the statistical security parameter. We say $\{\mathcal{D}_{1,\kappa}\} \approx_c \{\mathcal{D}_{2,\kappa}\}$, if for every PPT distinguisher $\mathcal{A}$, $\Big| Pr\big[\mathcal{A}(1^\kappa, x \stackrel{\$}{\leftarrow} \mathcal{D}_{1,\kappa}) = 1\big] - Pr\big[\mathcal{A}(1^\kappa, x \stackrel{\$}{\leftarrow} \mathcal{D}_{2,\kappa}) = 1\big] \Big| \leq \mathsf{negl}(\kappa)$ where negl is a negligible function. Also, we say $\{\mathcal{D}_{1,\kappa}\} \approx_{\mathsf{stat}} \{\mathcal{D}_{2,\kappa}\}$, if the two distributions $\mathcal{D}_{1,\kappa}, \mathcal{D}_{2,\kappa}$ are statistically indistinguishable.

2.1 Non-malleable Codes

Definition 1. *(Coding scheme) A coding scheme consists of two functions: a randomized encoding function* $\mathsf{Enc} : \mathcal{M} \rightarrow \mathcal{C}$ *, and a deterministic decoding function* $\mathsf{Dec} : \mathcal{C} \rightarrow \mathcal{M} \cup \{\perp\}$ *such that, for each* $m \in \mathcal{M}$*,* $Pr(\mathsf{Dec}(\mathsf{Enc}(m)) = m) = 1$ *(over the randomness of the encoding algorithm).*

In this paper, we use non-malleable codes resilient to split-state tampering functions. In a split-state setting, the codeword for any message can be viewed as consisting of two parts L and R. Any tampering function $f = (f_1, f_2)$ in a 2-split state function family $\mathcal{F}$ takes as input the codeword (L, R) and outputs a tampered codeword $(\tilde{L} = f_1(L), \tilde{R} = f_2(R))$. We require that for all such tampering functions $f \in \mathcal{F}$, $\mathsf{Dec}(\tilde{L}, \tilde{R})$ is unrelated to $\mathsf{Dec}(L, R)$.

Definition 2 (One-one Non-malleable codes)**.** *[ADL14] Let* $\mathcal{F}$ *be some family of tampering functions. For each* $f \in \mathcal{F}$ *, and* $m \in \mathcal{M}$*, define the tampering-experiment*

$$Tamper_m^f = \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(m), \tilde{c} \leftarrow f(c), \tilde{m} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \mathsf{Output}\ \tilde{m} \end{array} \right\}$$

which is a random variable over the randomness of the encoding function Enc*. We say that the coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is* $\epsilon -$ *non-malleable w.r.t.* $\mathcal{F}$ *if for each* $f \in \mathcal{F}$*, there exists a distribution (corresponding to the simulator)* D_f *over* $\mathcal{M} \cup \{\perp, \mathsf{same}^*\}$*, such that, for all* $m \in \mathcal{M}$ *we have that the statistical distance between* Tamper_m^f *and*

$$\mathsf{Sim}_m^f := \left\{ \begin{array}{c} \tilde{m} \leftarrow D_f \\ \mathsf{Output} : m \textit{ if } \tilde{m} = \mathsf{same}^*, \textit{ and } \tilde{m} \textit{ otherwise} \end{array} \right\}$$

is at most ϵ*.*

Furthermore, any construction of non-malleable codes in the split-state model also gives a *secret sharing* scheme. The states (L, R) can be viewed as shares of a 2-out-of-2 secret sharing scheme.

Remark 1. *We require the non-malleable codes in question to satisfy a many-many non-malleability property. Informally, an adversary should not be able to tamper shares corresponding to even polynomially many messages, to achieve a related word in even one of polynomially many output messages. Next, we formally define one-many non-malleable codes which also extend directly to many-many non-malleable codes.*

Definition 3 (Many-Many Non-Malleable Codes)**.** *[CGL15] A coding scheme* $(\mathsf{Enc}; \mathsf{Dec})$ *with block length* n *and message length* k *is a non-malleable code with respect to a family of tampering functions* $\mathcal{F} \subset (\mathcal{F}_n)^t$ *and error* ϵ *if for every* $(f_1, \ldots f_t) \in \mathcal{F}$*, there exists a distribution* $D_{\mathbf{f}}$ *on* $\left(\{0,1\}^\kappa \cup \{\mathsf{same^*}_i\}_{i\in[u]}\right)^t$ *which is independent of the randomness in* Enc *such that for all vectors of messages* $(s_1, s_2, \ldots s_u), s_i \in \{0,1\}^\kappa$*, it holds that:* $|(\mathsf{Dec}(f_1(X)), \ldots, \mathsf{Dec}(f_t(X))) - \mathsf{replace}(\tilde{m} \leftarrow D_{\mathbf{f}}, s)| \leq \epsilon$*, where* $X = \mathsf{Enc}(s)$*. We refer to* t *as the tampering degree of the non-malleable code.*

The function $\mathsf{replace} : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ *is defined as follows. If the second input to* $\mathsf{replace}$ *is a single value* s*, replace all occurrences of* same^* *in the first input with* s *and output the result. If the second input to* $\mathsf{replace}$ *is a set* $(s_1, s_2, \ldots, s_n)$*, replace all occurrences of* same^*_i *in the first input with* s_i *for all* i *and output the result.*

Imported Theorem 1. *[CGL15] There exists a constant* $\gamma > 0$ *such that for all* $n > 0$ *and* $t \leq n^\gamma$*, there exists an efficient construction of many-many non-malleable codes in the 2-split state model with tampering degree* t*, relative rate* $n^{\Omega(1)}/n$*, and error* $2^{-n^{\Omega(1)}}$*.*

A non-malleable code is also a non-malleable secret sharing scheme. A non-malleable secret sharing scheme consists of the following three algorithms:

- $\mathsf{NM-SS}(\cdot)$: This algorithm is identical to the $\mathsf{Enc}(\cdot)$ algorithm; it computes a non-malleable encoding of the input, which can also be viewed as a 2-out of-2 non-malleable secret sharing.
- $\mathsf{NM-Reconstruct}(\cdot)$: This algorithm is identical to the $\mathsf{Dec}(\cdot)$ algorithm; it recovers the input by decoding the two non-malleable shares.
- $\mathsf{NM-Simulate}(1^\kappa)$: The simulation algorithm samples $\tilde{m} \leftarrow D_f$ and outputs $\mathsf{NM-SS}(\tilde{m})$.

We require split-state non-malleable codes that satisfy the following three properties. The first two properties are already satisfied by the construction of many-many non-malleable codes in Chattopadhyay et. al. [CGL15]. We modify their construction slightly to also satisfy the third.

- **One-many non-malleable.** We require the code to be one-many non-malleable for tampering degree $\theta(\kappa^2)$, in the 2-split state model.
- **Secret-Sharing with Efficient Reverse-Sampling.** Given a share L (or R respectively) and a message m, it is possible to efficiently reverse-sample uniformly from the set of possible shares R (or L respectively) such that (L, R) encode message m.
- **Symmetric Decoding.** For all $L, R \in \{0,1\}^n$ we require that $\mathsf{Dec}(L, R) = \mathsf{Dec}(R, L)$. This is because our split-state tampering function (f, g) is such that, for instance, f may tamper with the joint distribution (R_1, L_2) to output L_{out} and g may tamper with (L_1, R_2) to output R_{out}. Here (L_1, R_1) is an encoding of some message m_1 and (L_2, R_2) is an encoding of message m_2.

  Then, it is first possible to replace (L_2, R_2) with a simulated codeword, while keeping the resulting output distribution statistically close to the real output distribution, while also introducing a simulation error of ϵ. Now, symmetric decoding ensures that $\mathsf{Dec}(L_{\mathsf{out}}, R_{\mathsf{out}}) = \mathsf{Dec}(R_{\mathsf{out}}, L_{\mathsf{out}})$, in other words $\mathsf{Dec}(f(R_1, L_{\mathsf{sim}}), g(L_1, R_{\mathsf{sim}})) = \mathsf{Dec}(g(L_1, R_{\mathsf{sim}}), f(R_1, L_{\mathsf{sim}}))$. Now, it is possible to treat (g, f) as the new tampering function and replace (L_1, R_1) with a simulated encoding while still keeping the resulting output distribution statistically close to the real output distribution, such that the total simulation error is bounded by 2ϵ, where ϵ is the decoding error of the non-malleable codes in question.

  We suggest a minor modification to the construction in [CGL15], to obtain symmetric decoding. Here, we artificially append 0 at the end of one of the shares and 1 at the end of the other, and start the decoding (or non-malleable extraction) process with the one that ends with 0. Decoding is invalid if both shares end with the same bit. We give further details on this modification in Appendix A.

2.2 Σ-Protocols and Special Σ-protocols

Sigma(Σ)-protocols are proof systems that follow a specific commit-challenge-response structure. We define a sub-class of Σ-protocols, which we call *special* Σ-protocols, where we impose an additional restriction on the structure of the third message, i.e., the "response".

We now define special Σ protocols for languages $\mathcal{L} \in \mathsf{NP}$ with a corresponding witness relation R. We borrow some parts from the definition of (standard) Σ-protocols in [Lin14], and we underline the additional properties that we require of *special* Σ-protocols, that need not be satisfied by all Σ-protocols. The rest of the definition (that is not underlined), and the properties are identical to those of regular Σ protocols.

Definition 4 (Special Σ-protocols)**.** *Let* $(\mathsf{Com}, \mathsf{Decom})$ *denote a computationally hiding, statistically binding commitment scheme. A Σ protocol is a 3-round public-coin protocol* π *between a prover* $P(w, x)$ *and a verifier* $V(x)$ *where R(x,w) = 1, such that without loss of generality,* P *and* V *have polynomially many* (κ) *parallel repetitions of the following (here we only depict a single execution and not many parallel executions):*

1. $P(w, x)$ *does the following.*
   - *Sample* $\mathbf{y} = (y_1, y_2, \dots y_N) \leftarrow \mathcal{S}$ *where* $\mathcal{S}_x$ *is an efficiently samplable distribution,* $N = \mathsf{poly}(\kappa)$ *(usually* $N = \theta(\kappa^2)$*). Compute* $\{\mathbf{J}, \pi\} = f(w, y)$ *for a fixed function* f *of the witness* w *and the random variable* y*. Note here that for correctly generated* $\mathbf{y}$ *and* π*, the witness* w *can be obtained from* (J, π, y)*.*
   - *For all* $i \in [N]$*, send* $Z_i = \mathsf{Com}(y_i, r_i)$ *to* V*.*
2. V *does the following.*
   - *Send challenge string* $\mathsf{ch} \xleftarrow{\$} \{0,1\}^N$ *to* P*.*
3. P *does the following.*
   - If $\mathsf{ch} = 0$, $\mathsf{Decom}(Z_i)$ for all $i \in [N]$.
   - If $\mathsf{ch} = 1$, $\mathsf{Decom}(Z_i)$ for all $i \in J$, and send auxiliary string π to V.
4. *Based on his view,* V *either accepts or rejects.*

Let the entire transcript of the messages sent in all parallel repetitions in three rounds be denoted by (a, e, z)*. Then a Σ-protocol satisfies the following properties:*

- (Completeness). *If* P *and* V *follow the protocol correctly on input* x *and private input* w *to* P*, where* $R(x, w) = 1$*, then* V *always accepts.*
- (Special Soundness). *There exists a polynomial time algorithm* A *that given any* x *and a pair of accepting transcripts* (a, e, d) *and* (a', e', d') *such that* $a = a'$ *and* $e \neq e'$*, outputs* w *such that* $(x, w) \in R$*.*
- (Special Honest-verifier zero knowledge). *There exists a PPT simulator* $\mathcal{S}_\Sigma$ *such that* $\{\mathcal{S}_\Sigma(x, e)\}_{x \in L; e \in \{0,1\}^\kappa} \approx_c \{\langle P(x, w), V(x, e)\rangle\}$*, where* $\mathcal{S}_\Sigma(x, e)$ *denotes the output of the simulator* $\mathcal{S}_\Sigma$ *on input* x *and* e*, and* $\langle P(x, w), V(x, e)\rangle$ *denotes the output transcript of an execution between* P *and* V*, where* P *has input* (x, w)*,* V *has input* x *and* V*'s random tape (determining the query) equals* e*. Note that this is slightly stronger than honest-verifier ZK because we do not allow the simulator to choose* e*.*

Blum's Σ-protocol for Graph Hamiltonicity is a special Σ-protocol, while Schnorr's protocol proving knowledge of a discrete logarithm is not a special Σ-protocol, because the third message of P does not respect the special structure in Point 3.

2.3 Stateful Tokens with Encapsulation

In the information theoretic stateful (tamper-proof hardware) token model of Katz [Kat07], two (computationally unbounded) interactive algorithms A and B will interact with the following extra feature to the standard model. Each party at any time during the protocol can construct a turing machine T, put it inside a "token", and send the token T to the other party. The party receiving the token T will have oracle access to T and is allowed to make polynomially many but unbounded number of queries to the token. Additionally, the token has the ability to maintain "state" between queries/inputs to the circuit T. The token can contain a random tape programmed at the time of construction, but cannot flip fresh coins on its own.

> $\mathcal{F}_{\mathsf{wrap}}$is parameterized by a polynomial p and an implicit security parameter κ. There are two main procedures:
>
> **Creation.** Upon receiving $(\mathsf{create}, \mathsf{sid}, P_i, P_j, M_{ij})$ from P_i, where P_j is another user in the system and M_{ij} is a (stateful) oracle machine, do:
>
> 1. For any call made by M_{ij} of the form $(\mathsf{run}, \mathsf{sid}, \mathsf{token}_{\mathsf{id}^*}, \mathsf{msg})$ to $\mathcal{F}_{\mathsf{wrap}}$, perform the following checks:
>   - If no tuple of the form $(\mathsf{sid}, \mathsf{token}_{\mathsf{id}^*}, P_\ell, *)$ is found for some user P_{ell}, output $\perp$ to P_i. *(Cannot encapsulate tokens that don't exist.)*
>   - If some tuple of the form $(\mathsf{run}, \mathsf{sid}, \mathsf{token}_{\mathsf{id}^{**}}, \mathsf{msg})$ has already been invoked, output $\perp$ to P_i. *(Cannot encapsulate more than one token.)*
>   - If $\mathsf{Owner}(\mathsf{Token}_{\mathsf{id}}) \neq P_i$, then output $\perp$ to P_i. *(Cannot encapsulate a token that you don't own.)*
>   - Finally, pick a unique identifier for the token, $\mathsf{Token}_{\mathsf{id}}$, and set $\mathsf{owner}(\mathsf{Token}_{\mathsf{id}^*}) = \mathsf{Token}_{\mathsf{id}}$.
> 2. Send $(\mathsf{create}, \mathsf{sid}, P_i, P_j, M_{ij})$ to P_j.
> 3. If there is no tuple of the form $(\mathsf{sid}, \mathsf{Token}_{\mathsf{id}}, P_i, *)$ stored, then store $(\mathsf{sid}, \mathsf{Token}_{\mathsf{id}}, P_i, M_{ij})$ and set $\mathsf{Owner}(\mathsf{Token}_{\mathsf{id}}) = P_j$.

Figure 1: The $\mathcal{F}_{\mathsf{wrap}}$ Functionality

Our model is a modification of the one in [Kat07]. The main modification we need is to allow for adversaries who may supply hardware tokens to other parties without knowing the code of the functionality implemented by the hardware token. We present a modified $\mathcal{F}_{\mathsf{wrap}}$ functionality in Figure 1. This formalizes the intuition that an honest user can create a hardware token by implementing any polynomial time functionality, but an adversary given the token T_F can do no more than observe its input/output characteristics.

$\mathcal{F}_{\mathsf{wrap}}$ models a hardware token sent by P_i to P_j as encapsulating a Turing Machine M_{ij}. However, similar to [CGS08], we allow M_{ij} to be a stateful Oracle machine instead, so as to allow an adversary to encapsulate other tokens. Note that M_{ij} is allowed to make black box calls to *a single* other token implementing M_{xy} (in order to model tokens created by an adversarial party). We assume that every token has a *single* calling procedure known as its *owner*. The owner can either be a party, or can be another token (in the case of adversarially generated tokens). This models the fact that an adversary that receives a token implementing M_{xy} can either keep this token to make calls later or incorporate the functionality of this token in a black-box manner into another (maliciously created) token, but cannot do both. Each token is uniquely identified by an identifier known as $\mathsf{Token}_{\mathsf{id}}$ and $\mathsf{Owner}(\mathsf{Token}_{\mathsf{id}})$ denotes the owner of this token.

K-OTM Tokens. Based on prior work [GKR08, GIS+10], we look at a simple stateful hardware token that implements K parallel oblivious transfer (OT) calls as a K-OTM token.

Functionality K-OTM

1. On input $(P_i, P_j, \mathsf{sid}, \mathsf{id}, (s_0^1, s_1^1), (s_0^2, s_1^2), \ldots (s_0^n, s_1^n))$ from party P_i, send $(P_i, P_j, \mathsf{sid}, \mathsf{id})$ to P_j and store the tuple $(P_i, P_j, \mathsf{sid}, \mathsf{id}, (s_0, s_1))$.
2. On receiving $(P_i, \mathsf{sid}, \mathsf{id}, c^1, c^2 \ldots c^n)$ from party P_j, if a tuple $(P_i, P_j, \mathsf{sid}, \mathsf{id}, (s_0^1, s_1^1), (s_0^2, s_1^2), \ldots (s_0^n, s_1^n))$ exists, return $(P_i, \mathsf{sid}, \mathsf{id}, s_{c^i}^i)$ to P_j and delete the tuple $(P_i, P_j, \mathsf{sid}, \mathsf{id}, (s_0^1, s_1^1), (s_0^2, s_1^2), \ldots (s_0^n, s_1^n))$. Else, do nothing.

Figure 2: The Ideal Functionality $\mathcal{F}_{\mathsf{K\text{-}OTM}}$for a K-OTM token

An (ℓ, K) OTM token is a generalization of this that implements the K-OTM functionality ℓ times sequentially. This token will be useful in the setting of bounded verifiable witness signatures. Finally, we note that no party can (physically) distinguish between $\mathcal{F}_{\mathsf{OTM}}$, $\mathcal{F}_{\mathsf{K\text{-}OTM}}$ and $\mathcal{F}_{\mathsf{wrap}}$ except via oracle access using run messages.

3 Non-malleable Two-Prover ZK Proofs

3.1 Two-Prover ZK from any Σ protocol

We recall the model and definition of a two-prover ZK proof, and then demonstrate the conversion of any special Σ-protocol according to Definition 4, into a two-prover zero knowledge proof.

3.1.1 Model

In the two-prover ZK setting there are three parties, namely two provers (P_1, P_2) and a verifier V. We do not require any party to be computationally bounded, i.e., all parties can be modeled as information theoretic adversaries.

The provers (P_1, P_2) obtain as input an NP instance x of a language L, along with a witness w for x. The verifier V obtains as input the NP instance x. The provers P_1, P_2 share a random tape of length $\mathsf{poly}(\kappa)$ for a fixed polynomial $\mathsf{poly}(\cdot)$. Moreover, P_1, P_2 are not allowed to interact with each other after the start of the protocol. At the end of the interaction, the verifier V outputs 0 or 1 (denoting an accepting versus rejecting transcript).

Definition 5. *A protocol τ is a two-prover ZK proof if it satisfies the following properties in the model above:*

- **Soundness.** *For honest verifier V, if $x \notin L$, $\Pr[V(\tau) = 1] = 0$.*
- **Completeness.** *For honest verifier V, if $x \in L$, $\Pr[V(\tau) = 1] = 1$.*
- **Zero-Knowledge.** *Let View_V denote the view of the verifier V. Then, for any possibly malicious unbounded verifier V, there exists a simulator Sim_V such that the output distributions $\mathsf{Sim}_V(1^\kappa)$ and View_V are identical.*
- **Proof of Knowledge**[3]**.** *For any two malicious unbounded provers P_1, P_2 that do not interact during the protocol, there exists a (possibly rewinding) extractor Ext_{P_1,P_2} which outputs a witness w such that: If $\Pr[V(\tau) = 1] = q$ over the randomness of the verifier, then $\Pr[w$ is a valid witness for $x] = \mathsf{poly}(q)$ for some polynomial $\mathsf{poly}(\cdot)$.*

Now, we prove the following theorem.

Theorem 1. *There exists a two-prover zero-knoweldge proof according to Definition 5 for any language L that admits a special Σ-protocol.*

[3]This property is not required in the definition, nevertheless most known constructions of two-prover proofs (including the ones we demonstrate in this paper) satisfy it.

3.1.2 Construction

Our two-prover ZK protocol consists of the following algorithms, based on ideas from Lapidot-Shamir [LS91][4]:
$\mathsf{Prove}(x, w)$: On input an NP instance x along with a witness w for x:

- For all $i \in [\kappa]$ (where each i corresponds to a parallel repetition of the two-prover proof), sample $y_i \xleftarrow{\$} \mathcal{S}_x$ (the efficiently samplable distribution for the Σ protocol according to Definition 4). Recall that each y_i is a vector of dimension N. Denote its k^{th} component by $y_{i,j}$.
- For all $i \in [\kappa]$ and $j \in [N]$, secret-share vector $y_{i,j}$ by setting $(a_{i,j}, b_{i,j}) = \mathsf{SS.Split}(y_{i,j})$. Denote by A_i the vector $(a_{i,1}, .., a_{i,N})$ and B_i the vector $(b_{i,1}, .., b_{i,N})$. For all $i \in [\kappa]$ sample auxiliary information J_i and the string π_i, using the witness w and y_i as in the sigma protocol. For all $i \in [n]$, send (A_i, B_i) and (π_i, J_i) to provers P_1 and P_2.
- The first and second prover algorithms are described in Figure 3 and Figure 4.

Constants: A_i, B_i, J_i, π_i for all $i \in [\kappa]$.
Input: Challenge string σ.
For all $i \in [\kappa]$, interpret the i^{th} bit of σ as σ_i. Then

1. If $\sigma_i = 0$, output A_i, B_i.
2. If $\sigma_i = 1$, output $\pi_i, a_{i,j\,(j \in J_i)}, b_{i,j\,(j \in J_i)}$.

Figure 3: Prover P_1

Constants: A_i, B_i for all $i \in [\kappa]$.
Input: Challenge string τ.
For all $i \in [\kappa]$, interpret the i^{th} bit of τ as τ_i. Then

1. If $\tau_i = 0$, output A_i.
2. If $\tau_i = 1$, output B_i.

Figure 4: Prover P_2

Verify : The verify algorithm samples two random strings $\sigma, \tau \xleftarrow{\$} \{0,1\}^\kappa$ and then performs the following checks for all $i \in [\kappa]$.

- If $\sigma_i = 0$, accept if and only if all the following checks pass.
  - The vector y_i is a valid transcript of the Σ protocol.
  - For all $j \in [N]$, the values $(a_{i,j}, b_{i,j}, y_{i,j})$ output by P_1 are such that $y_{i,j} = \mathsf{SS.Reconstruct}(a_{i,j}, b_{i,j})$.
  - For all $j \in [N]$ the values $a_{i,j}$ or $b_{i,j}$ output by P_2 match the corresponding values output by P_1.
- If $\sigma_i = 1$, accept if and only if all the following checks pass.
  - The values $(\pi_i, J_i, y_{i,j\,(j \in J_i)})$ are a valid transcript of the Σ protocol (that is, the underlying Σ protocol verifies).

[4] Lapidot-Shamir [LS91] provide a concrete instantiation of the protocol that follows the template we described. They construct a two-prover ZK proof for the Graph Hamiltonicity problem, and via Karp reduction, this automatically gives a two-prover ZK proof for any NP statement. However, we recall their construction while generalizing it to Σ protocols, which allows us to avoid going via the Karp reduction in order to prove any NP statement.

– For all $j \in J_i$, the values $(a_{i,j}, b_{i,j}, y_{i,j})$ output by P_1 are such that $y_{i,j} = \mathsf{SS.Reconstruct}(a_{i,j}, b_{i,j})$.
– For all $j \in J_i$, the values $a_{i,j}$ or $b_{i,j}$ output by P_2 match the corresponding values output by P_1.

Completeness follows directly from completeness of the special Σ-protocol. These algorithms satisfy the following other properties:

- Statistical Zero Knowledge: We describe the simulation strategy in Figure 5.

  We note that all special Σ-protocols according to Definition 4 must necessarily satisfy the following additional property: For all possible values of receiver message e, the distribution $(z|e)$ for z in the output of $\{\mathcal{S}_\Sigma(x, e)\}_{x \in L; e \in \{0,1\}^n}$ is statistically indistinguishable from $(z|e)$ for z in the output of $\{\langle P(x, w), V(x, e)\rangle\}$. This property requires statistical indistinguishability when restricted to the third message, instead of computational indistinguishability. Since the only computational assumption in any special Σ-protocol is made during the commit stage in the first step (while the third step only consists of openings), this property is satisfied by every special Σ-protocol. This directly implies that the distribution of the simulator's output in Figure 5 is statistically close to the real transcript.

Consider a pair of provers S, T proving a statement x on a tag tag. A query (σ, τ) to this pair is simulated in the following manner, for all $i \in [\kappa]$.

1. Pick a N-dimensional matrix R uniformly at random from the space of all shares of the secret sharing scheme.
2. If T is queried first
   – If $\tau_i = 0$, set $A_i = R$ and $B_i = \bot$. Output A_i.
   – If $\tau_i = 1$, set $A_i = \bot$ and $B_i = R$. Output B_i.
3. With one out of A_i, B_i thus fixed, pick the other component in the following way depending upon the query Σ to S_p.
   When S is queried with challenge query σ_i,
   – If $\sigma_i = 0$, sample random $y_i \xleftarrow{\$} \mathcal{S}_x$.
     If $A_i = \bot$, compute A_i such that each component $(a_{i,j}, b_{i,j}) = \mathsf{SS.Split}(y_{i,j})$.
     Otherwise, compute B_i such that each component $(a_{i,j}, b_{i,j}) = \mathsf{SS.Split}(y_{i,j})$.
   – If $\sigma_i = 1$, use the simulator of the underlying Σ protocol to sample $\pi_i, J_i, y_{i,j}$.
     If $A_i = \bot$, compute A_i such that each component $(a_{i,j}, b_{i,j}) = \mathsf{SS.Split}(y_{i,j})$.
     Otherwise, compute B_i such that each component $(a_{i,j}, b_{i,j}) = \mathsf{SS.Split}(y_{i,j})$.
4. If T is queried after S,
   – If $\tau_i = 0$, output A_i computed in Step 3.
   – If $\tau_i = 1$, output B_i computed in Step 3.

Figure 5: Simulation strategy for a pair of provers

- Statistical Soundness: Again, it follows from soundness of the Σ-protocol that a single execution of the resulting 2-prover ZK proof has constant soundness error.

  Ran Raz [Raz95] showed the following parallel repetition theorem for such games (and in particular, it applies to parallel repetition of a 2-prover ZK proof):

  Imported Theorem 2. *If the soundness error in a single execution is $(1 - \delta)$, then the probability that a pair of cheating provers (P_1, P_2) succeed in having the verifier accept when $x \notin L$, is at most $2^{-\Omega(\delta^3 n)}$, with constant factors in the exponent that depend (logarithmically) on the the space of the prover's answers.*

In general, statistical soundness is achieved when the number of repetitions is sufficiently larger than the logarithm of the size of the prover's answers.

- Proof of Knowledge: Again, the special soundness/ proof of knowledge property of Σ-protocol guarantees a simple PPT extractor for the 2-prover proof, which extracts a witness with probability polynomial in the verification probability. This extractor obtains the answers of the provers on four strings $(\sigma^1, \sigma^2, \tau^1, \tau^2)$ such that there exists an index i where $\sigma^1[i] \neq \sigma^2[i]$ and $\tau^1[i] \neq \tau^2[i]$, and uses these answers to reconstruct y_i, π_i; from which, by definition of Σ protocols, w can be obtained as $w = \pi_i(y_i)$.

3.2 Non-malleable Two-Prover ZK from any Σ-Protocol

In this section, we give our main construction: A non-malleable two-prover ZK protocol from any special Σ-protocol[5]. We first describe the model and define a non-malleable two-prover ZK proof, then we demonstrate a compiler from any Σ-protocol to a statistical non-malleable two-prover ZK proof.

3.2.1 Model

A non-malleable two-prover ZK protocol considers two-prover ZK in a setting where there is a *left* pair of provers (P_1, P_2) interacting with man-in-the-middle verifiers $V_{\mathsf{MIM},1}, V_{\mathsf{MIM},2}$. Furthermore, these man-in-the-middle verifiers are together controlled by two provers $(P_{\mathsf{MIM},1}, P_{\mathsf{MIM},2})$ that are themselves interacting with honest verifier V on the right.

Note that no two provers corresponding to any session are allowed to communicate between themselves once protocol execution starts. This holds true for the honest as well as MIM provers. Also note that in the single-use setting, no prover runs a protocol more than once. This means that the MIM verifiers are allowed to query any honest prover only once. Moreover, once the man-in-the-middle provers start their session on the right, they are only allowed to interact with a **disjoint subset** of provers on the left, for the same left session. That is, $P_{\mathsf{MIM},1}$ and $P_{\mathsf{MIM},2}$ cannot both be interacting with the same left prover *once their session on the right has started*. This is a natural requirement, since $P_{\mathsf{MIM},1}$ and $P_{\mathsf{MIM},2}$ are required to be isolated once the right session starts – so if they both talk to a single left prover, the prover will necessarily start separate sessions with them. Thus, in the synchronizing setting, we have the following possibilities:

- $P_{\mathsf{MIM},1}$ interacts with P_1 while $P_{\mathsf{MIM},2}$ interacts with P_2 (Figure 6), or,
- $P_{\mathsf{MIM},1}$ interacts with P_2 while $P_{\mathsf{MIM},2}$ interacts with P_1 (Figure 7), or,
- $P_{\mathsf{MIM},1}$ interacts with both P_1 and P_2 while $P_{\mathsf{MIM},2}$ interacts with neither (Figure 8), or,
- $P_{\mathsf{MIM},2}$ interacts with both P_1 and P_2 while $P_{\mathsf{MIM},1}$ interacts with neither (Figure 9).

Similarly, in an asynchronous setting, one or more of the left provers may already have been queried (and w.l.o.g. verified jointly by $P_{\mathsf{MIM},1}$ and $P_{\mathsf{MIM},2}$) before the session on the right starts. We can classify these into five categories, again depending on the interaction patterns.

- (Asynchronous). Either or both $P_{\mathsf{MIM},1}, P_{\mathsf{MIM},2}$, query both P_1, P_2 before the start of the right interaction. In this case, non-malleability follows from the zero knowledge and proof-of-knowledge properties of the Σ-protocol.
- Either or both $P_{\mathsf{MIM},1}, P_{\mathsf{MIM},2}$, query P_1 before the start of the right interaction. $P_{\mathsf{MIM},2}$ interacts with P_2 during the right interaction.
- Either or both $P_{\mathsf{MIM},1}, P_{\mathsf{MIM},2}$, query P_2 before the start of the right interaction. $P_{\mathsf{MIM},2}$ interacts with P_1 during the right interaction.
- Either or both $P_{\mathsf{MIM},1}, P_{\mathsf{MIM},2}$, query P_1 before the start of the right interaction. $P_{\mathsf{MIM},1}$ interacts with P_2 during the right interaction.
- Either or both $P_{\mathsf{MIM},1}, P_{\mathsf{MIM},2}$, query P_2 before the start of the right interaction. $P_{\mathsf{MIM},1}$ interacts with P_1 during the right interaction.

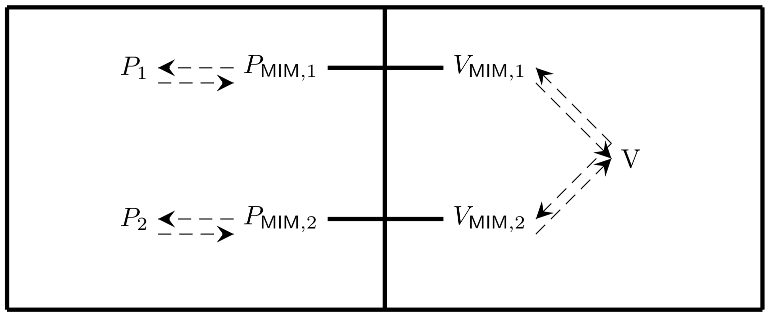

Figure 6: Synchronous Interaction 1: $P_{\mathsf{MIM},1}$ interacts with P_1 and $P_{\mathsf{MIM},2}$ interacts with P_2

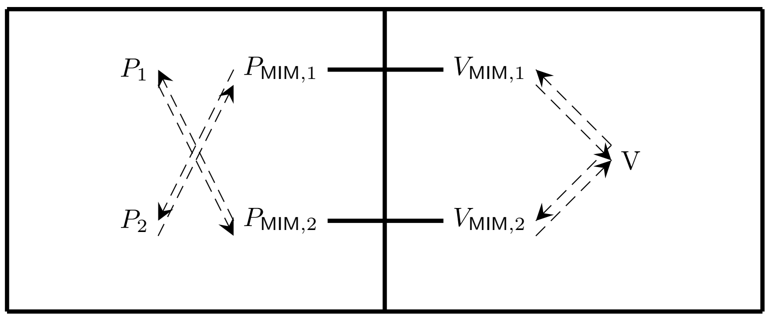

Figure 7: Synchronous Interaction 2: $P_{\mathsf{MIM},1}$ interacts with P_2 and $P_{\mathsf{MIM},2}$ interacts with P_1

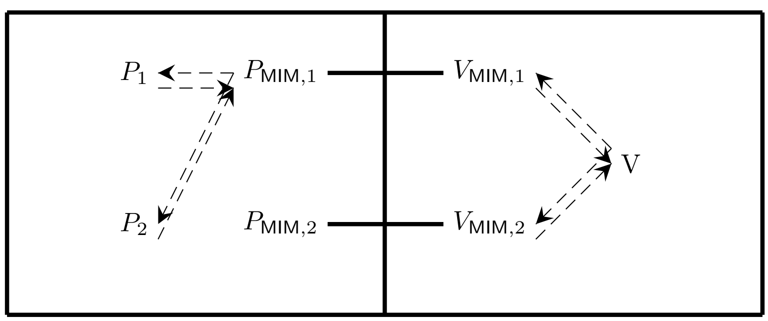

Figure 8: Synchronous Interaction 3: $P_{\mathsf{MIM},1}$ interacts with P_1 and P_2

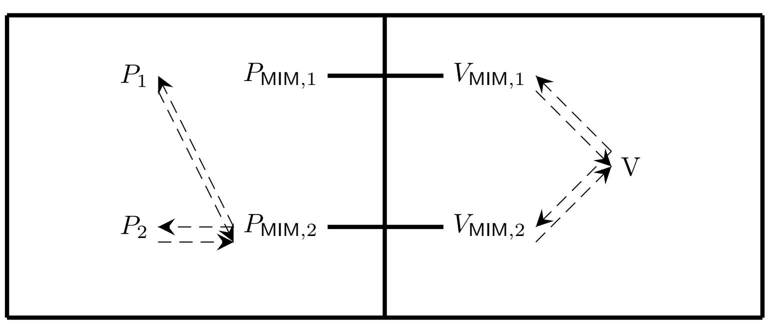

Figure 9: Synchronous Interaction 4: $P_{\mathsf{MIM},2}$ interacts with P_1 and P_2

We consider a natural tag-based malleation setting where provers $(P_{\mathsf{MIM},1}, P_{\mathsf{MIM},2})$ try to use provers (P_1, P_2) to prove a related statement (or possibly the same statement), but on a different tag.

Definition 6. *A protocol Π is a two-prover non-malleable ZK proof if it satisfies the following properties:*

- **Completeness.** *For any honest verifier V, if $x \in L$, $\Pr[V(\Pi) = 1] = 1$.*
- **Simulation-Extraction**[6]**.** *There exists a black-box (possibly rewinding) simulator-extractor* Sim-Ext *which outputs a witness w with black-box access to any malicious man-in-the-middle adversary* MIM *defined above – such that if $\Pr[V(\Pi) = 1] = q$ over the randomness of the verifier and the* MIM*, then* $\Pr[w$ *is a valid witness for* $x] = \mathsf{poly}(q)$ *for some polynomial* $\mathsf{poly}(\cdot)$.

Next, we state our main theorem. We prove it by giving a construction and a proof of security of a non-malleable two-prover ZK proof.

Theorem 2. *There exists a non-malleable two-prover zero-knowledge proof (of knowledge) according to Definition 6 for any language that admits a special Σ-protocol.*

Corollary 3. *There exists a non-malleable two-prover zero-knowledge proof (of knowledge) according to Definition 6 for any language in* NP.

We note that Corollary 3 can be obtained from Theorem 2 by obtaining a non-malleable two-prover zero-knowledge proof (of knowledge) according to Definition 6 for the Blum Hamiltonicity special Σ-protocol, and then using NP reductions.

3.2.2 Construction

The protocol consists of the following algorithms, where $\mathsf{NM}-\mathsf{SS}(\cdot)$ and $\mathsf{NM}-\mathsf{Reconstruct}(\cdot)$ denote the encoding and decoding algorithms of a many-many non-malleable scheme with tampering degree $\theta(\kappa^2)$[7]:

- $\mathsf{Prove}(x, w, \mathsf{tag})$: On input an NP instance x along with a witness w for x:
  - For all $i \in [\kappa]$ (where each i corresponds to a parallel repetition of the two-prover proof), sample $y_i \xleftarrow{\$} \mathcal{S}_x$ (the efficiently samplable distribution for the Σ protocol according to Definition 4). Recall that each y_i is a vector of dimension N. Denote its k^{th} component by $y_{i,j}$.
  - For all $i \in [\kappa]$ and $j \in [N]$, secret-share the vector $y_{i,j}$ by setting $(a_{i,j}, b_{i,j}) = \mathsf{NM}-\mathsf{SS}(\mathsf{tag}||y_{i,j})$. Denote by A_i the vector $(a_{i,1}, .., a_{i,N})$ and B_i the vector $(b_{i,1}, .., b_{i,N})$. For all $i \in [\kappa]$ sample auxiliary information J_i and the string π_i, using the witness w and y_i as in the sigma protocol. For all $i \in [n]$, send (A_i, B_i) and (π_i, J_i) to provers P_1 and P_2.
  - The first and second prover algorithms are described respectively in Figure 10 and Figure 11.

Constants: A_i, B_i, J_i, π_i for all $i \in [\kappa]$.
Input: Challenge string σ.
For all $i \in [\kappa]$, interpret the i^{th} bit of σ as σ_i. Then

1. If $\sigma_i = 0$, output A_i, B_i.
2. If $\sigma_i = 1$, output $\pi_i, a_{i,j_{(j \in J_i)}}, b_{i,j_{(j \in J_i)}}$.

Figure 10: Prover P_1

[5]Our construction can be modified by directly substituting $(\mathsf{tag}||\cdot)$ for $(\cdot)$ to yield non-malleable two-prover ZK against an MIM that is allowed to tamper any other way than forwarding messages from left to right and vice-versa.

[6]Note that this already implies stand-alone soundness, stand-alone ZK and stand-alone proof of knowledge properties.

[7]For simplicity we represent them as functionalities, but they can be implemented using hardware tokens.

Constants: A_i, B_i for all $i \in [\kappa]$.
Input: Challenge string τ.
For all $i \in [\kappa]$, interpret the i^{th} bit of τ as τ_i. Then

1. If $\tau_i = 0$, output A_i.
2. If $\tau_i = 1$, output B_i.

Figure 11: Prover P_2

- $\mathsf{Verify}(x, \mathsf{tag}, P_1, P_2)$: The verify algorithm samples two random strings $\sigma, \tau \xleftarrow{\$} \{0,1\}^\kappa$. It queries the prover P_1 on σ and prover P_2 on τ. It performs the following checks on the reponse for all $i \in [\kappa]$.
  - If $\sigma_i = 0$, accept if and only if all the following checks pass.
    * The vector y_i is a valid transcript of the Σ protocol.
    * For all $j \in [N]$, the values $(a_{i,j}, b_{i,j}, y_{i,j})$ output by P_1 are such that $(\mathsf{tag}||y_{i,j}) = \mathsf{NM} - \mathsf{Reconstruct}(a_{i,j}, b_{i,j})$.
    * For all $j \in [N]$, the values $a_{i,j}$ or $b_{i,j}$ output by P_2 match the values output by P_1.
  - If $\sigma_i = 1$, accept if and only if all the following checks pass.
    * The values $(\pi_i, J_i, y_{i,j\,(j \in J_i)})$ comprise valid transcript of Σ protocol.
    * For all $j \in J_i$, the values $(a_{i,j}, b_{i,j}, y_{i,j})$ output by P_1 are such that $(\mathsf{tag}||y_{i,j}) = \mathsf{NM} - \mathsf{Reconstruct}(a_{i,j}, b_{i,j})$.
    * For all $j \in J_i$, the values $a_{i,j}$ or $b_{i,j}$ output by P_2 match the values output by P_1.

3.3 Proof of Security of Non-malleable Two-Prover ZK

Now, we prove security of our construction.

3.3.1 Simulator-Extractor

We first describe a simulation strategy in Figure 12. This is similar to the stand-alone strategy except that for $i \in [\kappa]$ and $\sigma_i = 0$, the shares $a_{i,j}, b_{i,j}$ for $j \in [N]$ are generated as non-malleable secret shares of $y_{i,j}$. For $i \in [\kappa]$ and $\sigma_i = 1$, the shares $a_{i,j}, b_{i,j}$ for $j \in J_i$ are generated as non-malleable secret shares of $y_{i,j}$, the rest are generated as component-wise simulated shares.

Next, we describe the simulator-extractor $\mathcal{R}$ in Figure 13. Given black-box access to a pair of man-in-the-middle provers which generate a ZK proof for an instance x, after possibly interacting with several provers on the left with different tags; $\mathcal{R}$ extracts a witness to x with probability $\mathsf{poly}(p)$ for some polynomial $\mathsf{poly}(\cdot)$.

3.3.2 Proof Overview

At a high level, the simulator-extractor functions as follows. The simulator-extractor and the MIM obtain the instance x as input.

The MIM may begin by querying for proofs on some tag t. Then the simulator-extractor just sends across two provers P_1 and P_2. The MIM may now query both, one or none of the provers. If this is the case, the simulator-extractor simulates the response of the provers by using the simulation strategy of the two prover proof system. Then, MIM provers P_1' and P_2' may begin a proof session on some tag $t' \neq t$. At this point, if they have not already seen a proof, P_1' and P_2' may (in a disjoint fashion) initiate a session with either P_1 or P_2 or both. This lands our simulator extractor into one of several possible cases. These cases may, for example, have P_1' interacting with P_1 while P_2' interacts with P_2. Or, P_1' could be interacting with P_2 while P_2' interacts with P_1. Or, in another case, P_1' could be interacting with both P_1 and P_2 while P_2' operates stand-alone. In any case, P_1' and P_2' always interact with a disjoint subset of the left provers.

In most of the cases, our simulator-extractor queries the provers P_1' and P_2' on multiple strings and extract the witness from responses to these queries at some index ind. P_1' and P_2' may, in turn, translate these queries to other queries to the left provers P_1 and P_2. Since our simulator-extractor does not have a witness it cannot honestly answer these inner queries. The response to these queries are usually given

Consider a pair of provers P_1, P_2 proving a statement x on tag tag. A query (σ, τ) to this pair is simulated in the following manner, for all $i \in [\kappa]$.

1. Pick a N-dimensional matrix R uniformly at random from the space of all shares of the secret sharing scheme.
2. If P_2 is queried first
   - If $\tau_i = 0$, set $A_i = R$ and $B_i = \perp$. Output A_i.
   - If $\tau_i = 1$, set $A_i = \perp$ and $B_i = R$. Output B_i.
3. When P_1 is queried with challenge query σ_i.
   - If P_1 is queried first, set $A_i = \perp$ and $B_i = R$.
   - If $\sigma_i = 0$, sample random $y_i \stackrel{\$}{\leftarrow} \mathcal{S}_x$.
     If $A_i = \perp$, compute A_i such that each component $(a_{i,j}, b_{i,j}) = \mathsf{NM-SS}(\mathsf{tag}||y_{i,j})$.
     Otherwise, compute B_i such that each component $(a_{i,j}, b_{i,j}) = \mathsf{NM-SS}(\mathsf{tag}||y_{i,j})$.
   - If $\sigma_i = 1$, use the simulator of the underlying Σ protocol to sample $\pi_i, J_i, y_{i,j}{}_{j \in J_i}$.
     - If $A_i = \perp$, for $j \in J_i$, compute each component $a_{i,j}$ such that $(a_{i,j}, b_{i,j}) = \mathsf{NM-SS}(\mathsf{tag}||y_{i,j})$. For $j \notin J_i$, compute each component $a_{i,j}$ such that $(a_{i,j}, b_{i,j}) = \mathsf{NM-Simulate}(1^\kappa)$.
     - Otherwise, for $j \in J_i$, compute each component $b_{i,j}$ such that $(a_{i,j}, b_{i,j}) = \mathsf{NM-SS}(\mathsf{tag}||y_{i,j})$. For $j \notin J_i$, compute each component $b_{i,j}$ such that $(a_{i,j}, b_{i,j}) = \mathsf{NM-Simulate}(1^\kappa)$.
4. If P_2 is queried (again, possibly in a rewinding thread) after P_1,
   - If $\tau_i = 0$, output A_i computed in Step 3.
   - If $\tau_i = 1$, output B_i computed in Step 3.

Figure 12: Simulation strategy for a pair of provers

using simulation strategy of the proof, and we show, surprisingly, that this suffices. Indeed, the probability with which the simulator-extractor succeeds in these cases is at least some fixed polynomial function of the probability of verification of the proof in these cases. These arguments are mainly of combinatorial nature and crucially rely on zero knowledge property of the proof.

The most interesting case occurs when P_1' is in a session with P_1 while the corresponding P_2' is in a session with P_2. In this case, we rely on the security of many-many non malleable codes. This is done by querying P_1' on some string σ^1. When P_1' in turn queries P_1, we simulate the response of P_1 according to the simulation strategy. This fixes some entries of the matrices A_i, B_i for all indices $i \in [n]$, that we will use to simulate P_2. Since we do not have a witness, we cannot sample the remaining entries correctly.

We note that this can be taken care of by the simulator of the non malleable codes. Indeed, the output of prover P_2' at index ind on query τ^1 and τ^2, can be seen as a split state tampering function over the many-many non-malleable code-words. Therefore, substituting the real codewords with simulated codewords, we still extract with a probability which is ϵ-close to the probability of extraction in the real world. This finishes the overview. We now detail the hybrid experiments and the proofs of extraction.

3.3.3 Hybrid Experiments

In this section, we give a sequence of hybrids where we move from using the witness, to simulating without using a witness. We prove that extraction probability remains close between hybrids.

Hybrid_0 : This hybrid corresponds to the real game between honest prover(s) and MIM. The reduction has a witness w for x and follows honest strategy on behalf of the provers. It uses the PoK extractor of the

1. The man-in-the-middle (henceforth MIM) may open a left session on some tag t of his choice. Then $\mathcal{R}$ starts simulating two provers P_1, P_2 for MIM.

2. If the MIM queries any prover(s) P_1, P_2 before starting the session on the right, $\mathcal{R}$ simulates the answers to these queries as described in Figure 12.

3. The MIM provers P_1', P_2' start a session on tag t' with $\mathcal{R}$. If they have not done so already, MIM provers may open sessions with two left provers P_1, P_2.[a].

4. $\mathcal{R}$ now proceeds to extract a witness from (P_1', P_2') in the following way.

   (a) Sample four strings $\{\sigma^1, \sigma^2, \tau^1, \tau^2\} \xleftarrow{\$} \{0,1\}^{4n}$.

   (b) Find $\mathsf{ind} \in [n] : \sigma^1_{\mathsf{ind}} = 1, \sigma^2_{\mathsf{ind}} = 0$, $\tau^1_{\mathsf{ind}} \neq \tau^2_{\mathsf{ind}}$[b]. If no such ind exists, abort.

   (c) Input challenge σ^1 to prover P_1'. On input σ^1, P_1' may query some prover to the left. Simulate answers to these queries according to Figure 12.

   (d) Input challenge τ^1 to P_2'. Then, P_2' may query some prover on the left.

      - If P_2' queries P_1 such that its corresponding left prover P_2 **is in session with** P_1', use the following strategy to answer queries to P_1. If P_2' interacts with a left prover P_1 such that its corresponding left prover P_2 is in a session with P_1'. Then, extract only from P_1 by rewinding and querying it on σ^2. In both cases, simulate A_i and B_i as non-malleable shares of a random sample y_i, for all indices i.
      - For all other cases of left provers in sessions with P_2', simulate answers according to Figure 12.

   (e) Rewind prover P_2' and repeat previous step with input τ^2 to P_2'.

      When P_2' queries left provers, simulate answers consistent with the answers fixed in response to the queries of P_1' in the straight-line execution, according to Figure 12.

      The only exceptions is if P_2' queries a left prover P_2 such that its corresponding left prover P_1 is in a session with P_1'. Then use the simulation strategy of Figure 12 with the same randomness as was used in Step (d). In all other cases, sample fresh randomness to simulate a new response of P_1, P_2 to the queries of P_2' according to Figure 12.

5. At index ind, $\mathcal{R}$ has obtained outputs $A_{\mathsf{ind}}, B_{\mathsf{ind}}$ (for $\tau^1_{\mathsf{ind}} = 0$, $\tau^1_{\mathsf{ind}} = 1$ respectively) from P_2', and π_{ind} from P_1 (for $\sigma^1_{\mathsf{ind}} = 1$). It then:

   - Computes y_{ind} entry-wise as $t'||y_{\mathsf{ind},j,k} = \mathsf{Reconstruct}(a_{\mathsf{ind},j,k}, b_{\mathsf{ind},j,k})$.
   - Next, it computes $w = \pi_{\mathsf{ind}}(y_{\mathsf{ind}})$ and checks if w is a valid witness to x.
   - If the check passes, it outputs w. Otherwise it aborts.

[a] Note that once a prover has been queried, he cannot be queried again. Thus we will assume that the MIM provers do not re-query a prover if they have already queried that prover before starting his session.

[b] When $\sigma^1, \sigma^2, \tau^1, \tau^2$ are sampled uniformly at random, it is easy to see that such an index exists with overwhelming probability.

Figure 13: Algorithm for Simulator-Extractor $\mathcal{R}$.

two-prover proof system to extract a witness from the MIM. We detail this hybrid in Figure 14.

Hybrid_1 : If the MIM queries any prover(s) (in Step 2) before starting the session on the right, answers to these prover(s) are simulated using the strategy described in Figure 12. If one of the provers is queried in Step 2 and the other prover is queried at a later point, the latter query is answered consistently and honestly using the witness. This hybrid ensures that if both provers P_1, P_2 were queried in Step 2, the answers are simulated[8].

1. The MIM may open a session on a tags t of his choice. Then $\mathcal{R}$ starts simulating two-provers $P_{1,p}, P_{2,p}$ for MIM.
2. The MIM may query the provers before starting a session on the right. Then, $\mathcal{R}$ generates the answers to these queries honestly.
3. The MIM provers P_1', P_2' start a session on tag t' with $\mathcal{R}$. If they have not done so already, MIM provers may open sessions with two left provers P_1, P_2.
4. $\mathcal{R}$ now proceeds to extract a witness from (P_1', P_2') in the following way.
   (a) Sample four strings $\{\sigma^1, \sigma^2, \tau^1, \tau^2\} \xleftarrow{\$} \{0,1\}^{4n}$.
   (b) Find $\mathsf{ind} : \sigma^1_{\mathsf{ind}} = 1, \sigma^2_{\mathsf{ind}} = 0$, $\tau^1_{\mathsf{ind}} \neq \tau^2_{\mathsf{ind}}$. If no such ind exists, abort.
   (c) Input challenge string σ^1 to prover P_1'. On input σ^1, P_1' may query some prover(s) on the left. Generate answers to these queries honestly.
   (d) Input challenge τ^1 to prover P_2'. On input τ^1, P_2' may query some prover(s) on the left. Generate the answers to these queries honestly.
   (e) Rewind the prover P_2' and repeat previous step with input challenge string τ^2 to prover P_2'. Again P_2' may query some provers on the left. Generate the answers to these queries honestly.
   (f) If P_1' interacts with P_2 and P_2' interacts with P_1, then rewind P_1 and input challenge σ^2 to prover P_1'. On input σ^2, P_1' may query some prover(s) on the left. Generate the answers to these queries honestly.
5. Finally, at index ind, $\mathcal{R}$ obtains outputs $A_{\mathsf{ind}}, B_{\mathsf{ind}}$ (corresponding to $\tau^1_{\mathsf{ind}} = 0$ and $\tau^1_{\mathsf{ind}} = 1$ respectively) from P_2', and π_{ind} from P_1' (corresponding to $\sigma^1_{\mathsf{ind}} = 1$). It then:
   - Computes y_{ind} entry-wise as $t'||y_{\mathsf{ind},j,k} = \mathsf{Reconstruct}(a_{\mathsf{ind},j,k}, b_{\mathsf{ind},j,k})$.
   - Next, it computes $w = \pi_{\mathsf{ind}}(H_{\mathsf{ind}})$ and checks if w is a valid witness to x.
   - If the check passes, it outputs w. Otherwise it aborts.

Figure 14: Real World.

Then intuitively, the probability of extraction in this hybrid is close to that in Hybrid_0 by the perfect zero-knowledge property of the two-prover proof.

Hybrid_2 : In this hybrid, the reduction follows various extraction strategies depending upon the orientation of the provers left P_1, P_2 with respect to the MIM provers P_1', P_2'.

The rest of this proof assumes that the orientation of the MIM provers deterministically falls in one of the following set of exhaustive cases, and verifies with probability q. This proof directly extends to the setting where the MIM probabilistically chooses which of the cases to orient his provers in. In this setting the

[8]Note that it is impossible to tell when a prover is queried in Step 2, whether or not its counterpart will be queried in Step 2 itself. Thus, instead of restricting to indices for which both provers were queried in Step 2, we answer queries of all provers queried in Step 2 using the simulation strategy of Figure 12. This achieves the same effect.

probability of extraction from such an MIM is the minimum of the probability of extraction over all possible cases the provers could orient themselves in, which is a fixed polynomial in the probability of verification of the proof. We now enumerate the cases.

1. If P_1' is interacting with prover P_1 on the left (or P_2 on the left respectively) such that the counterpart P_2 (or P_1 respectively) has already been queried in step 2, $\mathcal{R}$ simulates the answers to all queries to such a P_1 (or P_2 respectively) using the strategy described in Figure 12.

   Intuitively, the probability of extraction in this hybrid is close to that in Hybrid_1 by the perfect zero-knowledge property of the two-prover proof.

2. If P_2' is interacting with prover P_1 (or P_2 respectively) such that its counterpart P_2 (or P_1 respectively) has already been queried in Step 2, then $\mathcal{R}$ simulates the answers to all queries made by the prover P_2' to such a prover P_1 (or P_2), using the strategy described in Figure 12. Since the prover P_2' may be rewound and may generate fresh queries upon rewinding, the simulator answers these queries consistent with the answer given to P_1', but possibly inconsistent with the answer given to P_2' in the main thread. This inconsistency is necessary, as giving answers that are consistent with each other in the main and rewinding threads, as well as with the answers given to P_1', will require the knowledge of a witness.

   Intuitively, the probability of extraction in this hybrid is close to that in Hybrid_1 by the perfect zero-knowledge property of the two-prover proof.

3. If both provers P_1, P_2 are interacting with the same man-in-the-middle verifier V which is further emulating either P_1' or P_2', $\mathcal{R}$ simulates the answers to all queries made by MIM in steps 3 and 4 to these provers, using the strategy described in Figure 12. Intuitively, the probability of extraction in this hybrid is close to that in Hybrid_1 by the perfect zero-knowledge property of the two-prover proof.

4. If a left prover P_1 is in a session with P_2' such that its corresponding left prover P_2 is in a session with P_1', then all queries in Step 4 to P_2 are simulated according to Figure 12.

   Intuitively, the probability of extraction in this hybrid is polynomially close to that in Hybrid_1 by the zero knowledge property of the two-prover proof.

5. If prover P_1 is in session with P_1' such that the corresponding prover P_2 is in session with P_2', the response to these provers is simulated according to Figure 12. This is the final case which represents the reduction algorithm interacting with the MIM, without access to a valid witness. Since the prover P2' may be rewound and may generate fresh queries upon rewinding, the simulator answers these queries consistent with the answer given to P1' (according to Figure 12), but inconsistent with the answer given to P2' in the main thread. This inconsistency is necessary, as the simulator cannot give answers that are consistent with each other in the main and rewinding threads, as well as consistent with the answers given to P1' without the knowledge of a witness.

   Intuitively, the probability of extraction in this hybrid is close to that in Hybrid_1 because of many-many non-malleability of a subset of the shares $a_{i,j}, b_{i,j}$ of the sample y_i.

3.3.4 Proofs of Extraction from the Hybrids

Roughly, we first prove that (in the real world) if the man-in-the-middle provers verify on a randomly chosen challenge input with probability q, then the rewinding execution (with another randomly chosen challenge) also verifies with probability at least $\mathsf{poly}(q)$. This observation will serve as a basic ingredient in our proofs, and we begin by proving this statement formally.

Lemma 1 (Extraction Lemma)**.** *Let X and Y denote two (possibly correlated) random variables from distribution $\mathcal{X}$ and $\mathcal{Y}$, with support $|\mathcal{X}|$ and $|\mathcal{Y}|$, and $U(X,Y)$ denote an event that depends on X, Y. We say that $U(X,Y) = 1$ if the event occurs, and $U(X,Y) = 0$ otherwise. Suppose $\Pr_{(X,Y)\sim(\mathcal{X},\mathcal{Y})}[U(X,Y) = 1] = p$. We say that a transcript $\mathbb{X}$ falls in the set* good *if $\Pr_{Y\sim\mathcal{Y}}[U(X,Y|X = \mathbb{X}) = 1] \geq p/2$. Then, $\Pr_{X\sim\mathcal{X}}[X \in$ good$] \geq p/2$.*

Proof. We prove the lemma by contradiction. Suppose $\Pr_{X\sim\mathcal{X}}[X \in \mathsf{good}] = c < \frac{p}{2}$. Then,

$$\Pr_{(X,Y)\sim(\mathcal{X}\mathcal{Y})}[U(X,Y)=1] = \Pr_{(X,Y)\sim(\mathcal{X},\mathcal{Y})}[U(X,Y)=1|X \in \mathsf{good}] \cdot \Pr_{X\sim\mathcal{X}}[X \in \mathsf{good}] + \Pr_{(X,Y)\sim(\mathcal{X},\mathcal{Y})}[U(X,Y)=1|X \notin \mathsf{good}] \cdot \Pr_{X\sim\mathcal{X}}[X \notin \mathsf{good}]$$

By definition of the set good, $\Pr_{(X,Y)\sim(\mathcal{X},\mathcal{Y})}[U(X,Y)=1|X \notin \mathsf{good}] < \frac{p}{2}$. Then, $p = \Pr[U(X,Y)=1] < 1 \cdot c + (1-c)\cdot p/2$. Then, if $c < \frac{p}{2}$, we will have that $p < \frac{p}{2} + \frac{p}{2}$, which is a contradiction. This proves our lemma. □

Jumping ahead, for most of our lemmas, U will be the event that the two-prover proof given by the MIM verifies on four challenge strings $\sigma^1, \sigma^2, \tau^1, \tau^2$ (and therefore witness extraction occurs). Looking ahead, X will correspond to a transcript that verifies in the main thread, and Y will denote the random coins of the provers.

In the following, we will use the phrase 'extraction occurs' to denote the event where the reduction $\mathcal{R}$ obtains a witness (possibly via rewinding) such that $R(x,w)=1$.

Lemma 2. *In* Hybrid_0*, if the man-in-the-middle's proof verifies with probability* q *(which is at least* $1/\mathsf{poly}(\kappa)$ *for some polynomial* $\mathsf{poly}(\kappa)$ *and size of instance being* κ*) over the randomness of the challenge query, then extraction occurs with probability at least* q^c *where* c *is a constant.*

Proof. Let q be the probability that the proof verifies at a randomly chosen challenge (σ,τ). Then the probability that the extraction succeeds in Hybrid_0 is the same as probability that the proof verifies at randomly chosen tuples of the form $(\sigma^1,\tau^1),(\sigma^2,\tau^1),(\sigma^1,\tau^2),(\sigma^2,\tau^2)$ such that there exists an index ind where $\sigma^1_{\mathsf{ind}} \neq \sigma^2_{\mathsf{ind}}$ and $\tau^1_{\mathsf{ind}} \neq \tau^2_{\mathsf{ind}}$. Lapidot-Shamir [LS91] showed that this probability is at least q^c for some constant c. □

In the following lemmas, we will demonstrate that the probability that the proof verifies at randomly chosen tuples of the form $(\sigma^1,\tau^1),(\sigma^2,\tau^1),(\sigma^1,\tau^2),(\sigma^2,\tau^2)$ remains close (upto a small gap) between hybrids. We will denote the event that the proof verifies at randomly chosen tuples of the form $(\sigma^1,\tau^1),(\sigma^2,\tau^1),(\sigma^1,\tau^2),(\sigma^2,\tau^2)$, by "extraction occurs" (since this implies the event that $\mathcal{R}$ successfully extracts a witness). Sometimes, we will also denote the weaker event that $\mathcal{R}$ successfully extracts a witness, by the same phrase "extraction occurs".

Lemma 3. $\Pr[$*extraction occurs in* $\mathsf{Hybrid}_0] = \Pr[$*extraction occurs in* $\mathsf{Hybrid}_1]$.

Proof. Note that Step 2 occurs *before* the sessions on the right were started. Therefore, when the adversary is rewound on the right sessions, the left execution is rewound to the beginning of Step 3. Therefore, the view of the man-in-the-middle is identical in both hybrids, because of the (stand-alone) zero-knowledge property of the two-prover proof system.

Otherwise, consider a MIM such that the probability of extraction from the MIM in Hybrid_0 and Hybrid_1 is unequal. Then, there is a non-uniform distinguisher $\mathcal{D}$ which violates the ZK property of the underlying two-prover proof. The distinguisher $\mathcal{D}$ interacts on the left with either a pair of honest provers, or with simulated provers. It obtains as non-uniform advice the witness w for instance x and runs the simulator-extractor $\mathcal{R}$, and executes Step 2 with the MIM by forwarding the proof from the left to the right. It executes the rest of the simulator-extractor $\mathcal{R}$ using witness w according to Hybrid_0. The distinguisher $\mathcal{D}$ outputs 1 when extraction occurs and 0 when it does not occur. Then, if $\Pr[\text{extraction occurs in } \mathsf{Hybrid}_0] \neq \Pr[\text{extraction occurs in } \mathsf{Hybrid}_1]$, then $\Pr[\mathcal{D}=1|\text{honest proof}] \neq \Pr[\mathcal{D}=1|\text{simulated proof}]$, therefore violating the ZK property of the underlying proof. □

Lemma 4. *If the provers are oriented according to Case 1,*
$\Pr[$*extraction occurs in* $\mathsf{Hybrid}_2] = \Pr[$*extraction occurs in* $\mathsf{Hybrid}_1]$.

Proof. First, since there is no change in the simulation of any provers with which P'_2 is in session on the left, the view of P'_2 and therefore its output – remains the same in the main and rewinding executions, between Hybrid_1 and Hybrid_2.

If prover P'_1 is in a session with prover P_1 such that the corresponding prover P_2 was already queried before Step 2, then the answers to queries to P_1 are simulated as follows: Denote the queries made by P'_1 to

P_1, on being queried σ^1, by $\tilde{\sigma^1}$. Then the simulation strategy of Figure 12 answers such that the answer of P_1 is consistent with the answer of P_2. In this case, in the main execution as well as the rewinding execution, since P_1 and P_2 are never rewound, the answer given by P_2 is consistent with the answer given by P_1 in both Hybrid_0 and Hybrid_1. Thus, the output of P_1' matches both outputs of P_2' (in *both main and rewinding executions*) with identical probabilities in Hybrid_0 and Hybrid_1, in the real execution. Thus, extraction occurs with probability exactly q^c.

The same argument yields that extraction occurs with probability exactly q^c even when the prover P_1 was queried before Step 2, and P_2 interacts with P_1. □

Lemma 5. *If the provers are oriented according to Case 2,*
$\Pr[\textit{extraction occurs in } \mathsf{Hybrid}_2] \geq \mathsf{poly}(\Pr[\textit{extraction occurs in } \mathsf{Hybrid}_1])$ *for a fixed polynomial* $\mathsf{poly}(\cdot)$.

Proof. In this case, the simulation strategy of Figure 12 is such that both answers of P_2 (or P_1 respectively) are separately consistent with the answer of P_1 (or P_2 respectively). Without loss of generality, let the prover queried before Step 2 be P_1 and let the MIM prover P_2' be in session with P_2.

We fix a (randomly chosen) output μ of P_1 – this comes from an identical distribution in both Hybrid_1 and Hybrid_2. We recall that the MIM provers are queried on the verifier challenges only after P_1 has been queried by the MIM. The output of P_1' on challenge string $\sigma \in \{\sigma^1, \sigma^2\}$ is a deterministic function of the output μ of prover P_1 and the randomness of prover P_1'. We fix this randomness $r_{P_1'}$, thus fixing both possible outputs of P_1' on challenge strings (σ^1, σ^2). We denote the combined transcript of μ and the randomness $r_{P_1'}$, by T.

We define a good transcript T as one where, when the output of P_2 is sampled in Hybrid_1 (using the correct witness), then the probability that the proof of the MIM verifies on a fixed randomly chosen set of tuples $(\sigma^1, \tau^1), (\sigma^2, \tau^1), (\sigma^1, \tau^2), (\sigma^2, \tau^2)$ is at least $q^c/2$ over the randomness of prover P_2 and the MIM. This is the experiment of Hybrid_1, where the randomness and response of P_2 is chosen honestly using a witness, and consistent with the output μ. Then, via Lemma 1, the probability that a randomly chosen T is good is at least $q^c/2$.

Now, for a fixed T, we query the prover P_2' on challenge τ^1 while sampling the output of the prover P_2 honestly consistent with the output μ of P_1. Then, for a good T, the probability that the fixed response of P_1' on challenges (σ^1, σ^2) each verify with the response of P_2' on τ^1 is at least $q^c/2$.

However, in Hybrid_2, when P_2' is rewound to obtain its response to challenge τ^2, it is not possible for the simulator to sample the output of P_2 consistent with the output given corresponding to challenge τ^1 to P_2' and consistent with μ – as this would require a witness. Here, the simulator only samples the output of the prover P_2 honestly consistent with the output μ of the transcript (and possibly inconsistent with the response given when P_2 was queried by the MIM in response to τ^1). Even in this case, we know by the property of a good T, the probability that the (fixed) response of P_1' on challenges (σ^1, σ^2) each verify with the response of P_2' on τ^2 is at least $q^c/2$.

Finally, it is straightforward to see that the probability of verification on challenges $(\sigma^1, \tau^1), (\sigma^2, \tau^1), (\sigma^1, \tau^2)$, (σ^2, τ^2) for a fixed $(\sigma^1, \sigma^2, \tau^1, \tau^2)$ and a fixed response of P_1' on challenges (σ^1, σ^2), is at least the probability of separate verification on challenges $(\sigma^1, \tau^1), (\sigma^2, \tau^1)$ and $(\sigma^1, \tau^2), (\sigma^1, \tau^2)$, which is at least $q^{2c}/4$. Thus, the total probability of extraction in Hybrid_2 is at least the probability of sampling a good T output times the probability of verification on the set of tuples $(\sigma^1, \tau^1), (\sigma^2, \tau^1)$ and $(\sigma^1, \tau^2), (\sigma^2, \tau^2)$, which is $q^{3c}/8$. □

Lemma 6. *If provers are oriented according to Case 3,*
$\Pr[\textit{extraction occurs in } \mathsf{Hybrid}_2] \geq \mathsf{poly}(\Pr[\textit{extraction occurs in } \mathsf{Hybrid}_1])$ *for a fixed polynomial* $\mathsf{poly}(\cdot)$.

Proof. Again, a similar argument as that in Lemma 5 yields the probability that the proof of the MIM verifies at randomly chosen tuples of the form $(\sigma^1, \tau^1), (\sigma^2, \tau^1), (\sigma^1, \tau^2), (\sigma^2, \tau^2)$ remains close between Hybrid_1 and Hybrid_2. This proves that the probability of extracting a witness in Hybrid_1 is at least $q^{3c}/8$. □

Lemma 7. *If provers are oriented according to Case 4,*
$\Pr[\textit{extraction occurs in } \mathsf{Hybrid}_2] = \Pr[\textit{extraction occurs in } \mathsf{Hybrid}_1]$.

Proof. In this case, the reduction always extracts from the first prover P_1' only. Note that the view of prover P_1' are the answers of a prover P_2 which are random non-malleable shares of an efficiently sample-able y. This view can be kept perfectly identical between Hybrid_0 and Hybrid_1, by simply generating y and its shares at random and using honest prover P_2 strategy with these shares. Then, the probability of extraction from the outputs $(y'_{\mathsf{ind}}, J'_{\mathsf{ind}})$ of P_1 remains identical between Hybrid_0 and Hybrid_1. □

Lemma 8. *If provers are oriented according to Case 5,* $\Pr[\textit{extraction occurs in } \mathsf{Hybrid}_2] \geq \mathsf{poly}(\Pr[\textit{extraction occurs in } \mathsf{Hybrid}_1]) - \epsilon$*, for a fixed polynomial* $\mathsf{poly}(\cdot)$ *and* $\epsilon = \kappa \cdot 2^{-\mathsf{poly}(\kappa)}$.

Proof. In this case on being queried σ^1, suppose the prover P_1' queries the left prover P_1 on $\tilde{\sigma}^1$. For all indices $i \in [\kappa]$ where $\tilde{\sigma}_i^1 = 0$, the joint view of the MIM can be sampled perfectly, without a witness. For indices $i \in [\kappa]$ where $\tilde{\sigma}_i^1 = 1$, we rely on non-malleable codes to prove security.

We demonstrate that the prover P_2' reduces to a split-state tampering adversary, with the two states being the response of the prover P_2' on challenge index ind during the main and rewinding executions, and messages corresponding to all indices $i \in [\kappa]$ where $\tilde{\sigma}_i^1 = 1$.

In Hybrid_2, the reduction queries P_1' at σ^1 and P_2' at τ^1 and τ^2. The left provers P_1 and P_2 respond to these queries using a witness. Suppose P_1' on query σ^1, queries the left prover P_1 with $\tilde{\sigma}^1$. Fix the response μ of P_1. The output of P_1' on query σ^1 is a deterministic function of the output μ of prover P_1 and the randomness of prover P_1'. We fix this randomness $r_{P_1'}$, thus fixing the output of P_1' on challenge string σ^1. We done the combined transcript of μ and the randomness $r_{P_1'}$ by T.

We define a good transcript T as one where, when both outputs of P_2 (corresponding to challenges τ^1 and τ^2) are sampled in Hybrid_1 using the correct witness, then the probability that the response of MIM on $(\sigma^1, \tau^1, \tau^2)$ yields a witness is at least $q^c/2$ over the choice of randomness of the outputs of P_2 and the MIM. Then, via Lemma 1, such a transcript is sampled with probability at least $q^c/2$.

In Hybrid_2, the reduction $\mathcal{R}$ samples and fixes a random response μ of P_1 (in the main thread, on query $\tilde{\sigma}^1$) and then samples the response of P_2 using simulated code-words at all places not already revealed by this response μ. Then, the response of the MIM to the queries τ^1 and τ^2 induces a split-state functionality over these code-words.

Specifically, the reduction samples random J_i, and then uses witness w to obtain y_i for all $i \in [\kappa]$. This is the fixed response of P_1. Then for all $j \in J_i$, it sets each component $(a_{i,j}, b_{i,j}) = \mathsf{NM} - \mathsf{SS}(\mathsf{tag}||y_{i,j})$. For $j \notin J_i$, it obtains from the challenger of the many-many non-malleable codes, either the output of $\mathsf{NM} - \mathsf{Simulate}(\cdot)$, or $\mathsf{NM} - \mathsf{SS}(\mathsf{tag}||y_{i,j})$. Note that same^* never occurs for a valid MIM, since the tags in both executions are different. When the shares are simulated, we are in Hybrid_2, otherwise we are in Hybrid_1. The tampering function is the response of the prover $P_2' = (P_2'|\tau_1, P_2'|\tau_2)$; which obtains as input codewords for all indices (i, j) such that $\sigma_i = 1$ and $j \notin J_i$; and outputs values for index i' and all $j \in [N]$.

Then given non-uniform advice J'(which is possibly a function of J_i for all $i \in [\kappa]$), if the reconstruction of the output of P_2' yields a witness in Hybrid_1 with probability $q^c/2$, it also yields a witness in Hybrid_2, with probability at least $q^c/2 - \epsilon$, where the underlying non-malleable codes have simulation error at most ϵ.

Thus, the probability of extraction in Hybrid_2 is the probability of sampling a good transcript T times the probability of extraction from a good T, which is at least $q^{2c}/4 - \epsilon \cdot (q^c/2)$, which is negligibly far from $q^{2c}/4$ for small enough ϵ.

Finally, we point out that the split-state functionality induced by the adversary can tamper some left shares and some other (different, not corresponding) right shares together. Since we use symmetric non-malleable codes, left and right shares can be used interchangeably. In particular, we use a many-many non-malleable code with symmetric decoding, resilient against tampering degree κ^2 and simulation error at most 2^{κ}. Since there are κ fresh instances that the adversary can tamper with (corresponding to indices $i \in \kappa$ of the challenge queries σ and τ) to output the shares at index ind, the total simulation error is $\kappa \cdot 2^{-\mathsf{poly}(\kappa)}$. □

4 Witness Signatures

4.1 Definitions

Witness signatures allow any entity who knows a witness to an NP statement, to issue signatures on behalf of the statement. We now give a formal definition of witness signatures without setup.

Let R be an efficiently computable binary relation. For pairs (x, w) such that $R(x, w) = 1$ we call x the statement and w the witness. Let L be the language consisting of statements in R. Moreover, every language L in NP has a corresponding relation R. We let κ denote the statistical security parameter, such that $|x| = \kappa$. Each algorithm obtains 1^{κ} as input, and we omit this from our formal definitions.

Definition 7 (Witness Signatures)**.** *A witness signature scheme for some* NP *language L (with a corresponding witness relation R) consists of the following two PPT algorithms:*

- $\mathsf{Sign}(x, w, m; r)$*:* Sign *is an algorithm that takes as input a string x, a witness w, a message $m \in \{0,1\}^*$, randomness r and outputs a signature $\sigma_{m,x}$.*
- $\mathsf{Verify}(x, m, \sigma)$*:* Verify *is a deterministic algorithm takes as input a string x, a message $m \in \{0,1\}^*$ and a signature σ, and outputs 0 or 1.*

These algorithms satisfy the following properties:

- *Correctness. For any $m \in \{0,1\}^*$, $x \in L$ and w such that $R(x, w) = 1$, $\Pr[\mathsf{Verify}(x, m, \mathsf{Sign}(x, w, m; r)) = 1] = 1$.*
- *Witness Indistinguishability. For any instance x, given any two witnesses (w_1, w_2) such that $R(x, w_1) = 1$, $R(x, w_2) = 1$ and auxiliary information z, $\{z, \sigma_1 : \sigma_1 \leftarrow \mathsf{Sign}(x, w_1, m)\} \approx_{\mathsf{stat}} \{z, \sigma_2 : \sigma_2 \leftarrow \mathsf{Sign}(x, w_2, m)\}$, where $\approx_{\mathsf{stat}}$ denotes statistical indistinguishability.*
- *Unforgeability. (EUF-CMA) Consider a non-uniform PPT forger $\mathcal{F}$ which outputs an instance x^*. It then obtains access to a signing oracle $\mathcal{O}^{(x^*,\cdot)}$, which on input a message m outputs σ_{m,x^*} if $x^* \in L$ and $\perp$ otherwise. $\mathcal{F}$ can make a-priori unbounded adaptive queries to $\mathcal{O}^{(x^*,\cdot)}$ on messages $m_1, m_2, \ldots m_n$ for some $n = \mathsf{poly}(\kappa)$. $\mathcal{F}$ finally outputs a message, signature pair (m^*, σ_{m^*,x^*}) for the same instance x^* and message $m^* \notin \{m_1, m_2, \ldots m_n\}$.*

  Then, if $\Pr[\mathsf{Verify}(x^*, m^*, \sigma_{m^*,x^*})] = q$ over the space of randomness of $\mathcal{F}$, there must exist a black-box PPT reduction $\mathcal{E}$ and a constant c, such that $\mathcal{E}$ obtains input x^* and interacts with $\mathcal{F}$ in a black-box manner (we denote this interaction by $\mathcal{E}^{\mathcal{F}}(x^*)$) to output a string out*, where $\Pr[R(x, \mathsf{out}) = 1] \geq q^c$ (over the randomness of $\mathcal{F}$ and $\mathcal{E}$).*

Note that the oracle $\mathcal{O}$ need not be efficient. Also, we only consider statistical notions of indistinguishability in this paper, but we note that they can be generalized to perfect or computational indistinguishability. We formalize a generalization of this Definition 7, where the unforgeability constraint is strengthened to allow the adversary to request signatures on multiple different instances of his choice, and output a signature on one of the queried instances, or a different new instance. Our constructions are unconditionally secure even with respect to the stronger definition, but we only prove security with respect to Definition 7 because its more intuitive and easier to work with.

Definition 8 (Strong Witness Signatures)**.** *A strong witness signature is a witness signature with the unforgeability property strengthened as follows:*

Consider a non-uniform PPT forger $\mathcal{F}$ that obtains access to a signing oracle $\mathcal{O}^{(\cdot,\cdot,\cdot)}$, which on input a message m and instance x outputs witness signature $\sigma_{m,x}$ if $x \in L$ and $\perp$ otherwise. $\mathcal{F}$ can make a-priori unbounded adaptive queries $((m_1, x_1), (m_2, x_2) \ldots (m_n, x_n))$ to $\mathcal{O}^{(\cdot,\cdot,\cdot)}$, for some $n = \mathsf{poly}(\kappa)$. Finally, $\mathcal{F}$ outputs a message m^*, instance x^*, and signature σ_{m^*,x^*} such that $(m^*, x^*) \neq (m_i, x_i)$ for any $i \in [n]$.

If for all $i \in [n]$, $x_i \in L$; and $\Pr[\mathsf{Verify}(x^*, m^*, \sigma_{m^*,x^*})] = q$ over the space of randomness of $\mathcal{F}$, then there must exist a black-box PPT reduction $\mathcal{E}$ and a constant c, such that $\mathcal{E}$ interacts with $\mathcal{F}$ in a black-box manner (we denote this interaction by $\mathcal{E}^{\mathcal{F}(x^*)}$) to output a string (y, out), where $y \approx_{\mathsf{stat}} x^*$ and $\Pr[R(y, \mathsf{out}) = 1] \geq q^c$ (over the randomness of $\mathcal{F}$ and $\mathcal{E}$). Here $\approx_{\mathsf{stat}}$ refers to statistical indistinguishability.

Note that the instance y output by the reduction is not identical to x^*, in fact the distribution of the output instance y is only statistically indistinguishable from the distribution of the output x^* of $\mathcal{F}$.

Remark 2. *The unforgeability property implies that witness signatures for any language $\mathcal{L}$ must be statistically sound. That is, for $x \notin L$ there is no signature $(m^*, \sigma^*_{x,m})$ such that $\mathsf{Verify}(x, m^*, \sigma^*_{x,m}) = 1/\mathsf{poly}(\kappa)$ for some polynomial $\mathsf{poly}(\cdot)$ (otherwise a witness would be extracted with inverse-polynomial probability, which is impossible).*

Theorem 4. *Witness signatures for all of* NP *according to Definition 7 and Definition 8, are impossible unless* BPP = NP.

Proof. We prove our impossibility for the weaker definition of witness signatures, Definition 7. This suffices to show that both definitions are impossible unless $\mathsf{BPP} = \mathsf{NP}$. Intuitively, a reduction $\mathcal{E}$, which takes as input an instance x and interacts with a forger $\mathcal{F}$ in a black-box way, must as a first step, issue signatures to $\mathcal{F}$ on behalf of instance x. Since the signatures are single-message (one-shot), rewinding the forger does not help the reduction $\mathcal{E}$ in violating soundness. This means $\mathcal{E}$ will either have to decide whether or not $x \in L$ (solving decidability of NP-hard problems), or violate the soundness property. Since $\mathcal{E}$ is non-rewinding, if it violates the soundness property, then such an $\mathcal{E}$ can be used by a forger $\mathcal{F}$ in a black-box manner, to violate soundness.

Formally, given witness signatures for a language L, we construct a polynomial time algorithm $\mathcal{A}$ that on input x decides whether $x \in L$. $\mathcal{A}$ acts as a forger which requests a signature on 0 on behalf of instance x. If the signature verifies, $\mathcal{A}$ runs an exhaustive search to compute a witness w for x (if it exists), and output $\mathsf{Sign}(m, x, w)$. If the signature does not verify, it aborts. Of course, an efficient $\mathcal{A}$ cannot do this – but we will now describe how to emulate such an $\mathcal{A}$ efficiently.

Consider an efficient forger strategy $\mathcal{F}$ which takes as input a signature on 0 on behalf of instance x, and then aborts. Note that the view of the (efficient) black-box reduction $\mathcal{E}$ while interacting with strategy $\mathcal{A}$ is indistinguishable from its view when interacting with $\mathcal{F}$, unless $\mathcal{E}$ itself can compute a valid signature for $x \in L$, which is impossible if it cannot even decide whether or not $x \in L$ (i.e., $\mathsf{BPP} \neq \mathsf{NP}$). This is because there is the output of both $\mathcal{A}$ and $\mathcal{F}$ is identical until the reduction $\mathcal{E}$ outputs a (single message) signature that verifies with some probability. Now, if any $\mathcal{E}$ interacting with $\mathcal{A}$ outputs a witness w for x, then the same $\mathcal{E}$ must output a witness w for x when interacting with $\mathcal{F}$. This implies that $\mathcal{E}$ can compute a witness for x by itself, which is impossible unless $\mathsf{BPP} = \mathsf{NP}$. Finally, we note that this impossibility result holds even for a variant of Definition 7 that only requires computational indistinguishability. ☐

Remark 3. *Due to the impossibility in Theorem 4, it would make sense to consider even weaker variants of Definition 7. For instance, we could allow the reduction $\mathcal{E}$ non black-box access to the code of $\mathcal{F}$, or allow it to run in quasi-polynomial time. However, in order to bypass the impossibility we turn our attention to hardware token model and give efficient, unconditionally secure constructions with black-box extraction.*

Next, we will define witness signatures in the stateful tamper-proof hardware token model. In fact, our constructions are feasible in the K-OTM model of Figure 2 but are secure even in the stronger malicious $\mathcal{F}_{\mathsf{wrap}}$ model of Figure 1.

Definition 9 (Witness Signatures in the Hardware Model)**.** *A witness signature scheme in the stateful token model, for some NP language L (with a corresponding witness relation R) consists of the following two PPT algorithms:*

- $\mathsf{Sign}(P_1, P_2, x, w, m; r)$*:* Sign *takes as input identifiers of party P_1 (the signer) and party P_2 (the receiver), a string x, a witness w, a message $m \in \{0,1\}^*$, randomness r, picks a unique identifier* id, sid_i*, and outputs $\big(P_1, P_2, \mathsf{sid}_i, \mathsf{id}, (s_0^i, s_1^i)\big)$ to the i^{th} invocation of $\mathcal{F}_{\mathsf{OTM}}$, for $i \in \mathsf{poly}(\kappa)$. It also outputs* $\mathsf{id}, \mathsf{sid}_i$.
- $\mathsf{Verify}(x, m, (P_i, P_j, \mathsf{sid}, \mathsf{id}))$*:* Verify *takes as input identifiers of party P_1 (the signer) and party P_2 (the receiver), a string x, a message $m \in \{0,1\}^*$ and* $(\mathsf{id}, \mathsf{sid})$ *corresponding to a set of hardware tokens. It can send* run *messages to the $\mathcal{F}_{\mathsf{OTM}}$ or $\mathcal{F}_{\mathsf{wrap}}$ functionality (this denotes making an oracle query to a token it possesses), and finally outputs* 0 *or* 1.

These algorithms satisfy the following properties:

- *Correctness. For any message $m \in \{0,1\}^*$, for all (possibly adversarially chosen) $x \in L$ and w such that $R(x, w) = 1$, $\Pr[\mathsf{Verify}(x, m, \mathsf{Sign}(x, w, m; r)) = 1] = 1$.*
- *Witness Indistinguishability. For any instance x, given two witnesses (w_1, w_2) such that $R(x, w_1) = 1$, $R(x, w_2) = 1$ and auxiliary information z, $\{z, \sigma_1 : \sigma_1 \leftarrow \mathsf{Sign}(x, w_1, m); r\} \approx_{\mathsf{stat}} \{z, \sigma_2 : \sigma_2 \leftarrow \mathsf{Sign}(x, w_2, m; r)\}$, where $\approx_{\mathsf{stat}}$ denotes statistical indistinguishability.*
- *Unforgeability. (EUF-CMA) Consider a non-uniform PPT forger $\mathcal{F}$ which outputs an instance x^*. It then obtains access to a signing oracle $\mathcal{O}^{(x^*,\cdot)}$ which on input a message m outputs σ_{m,x^*} if $x^* \in L$ and*

$\perp$ otherwise. $\mathcal{F}$ can make a-priori unbounded adaptive queries to $\mathcal{O}^{(x^*,\cdot)}$ on messages $m_1, m_2, \ldots m_n$ for some $n = \mathsf{poly}(\kappa)$. In the malicious hardware token model, $\mathcal{F}$ has access to $\mathcal{F}_{\mathsf{wrap}}$ of Figure 1, and finally outputs a message, signature pair (m^*, σ_{m^*,x^*}) for the same instance x^* and message $m^* \notin \{m_1, m_2, \ldots m_n\}$.

Then, if $\Pr[\mathsf{Verify}(x^*, m^*, \sigma_{m^*,x^*})] = q$ over the space of randomness of $\mathcal{F}$, there must exist a black-box PPT reduction $\mathcal{E}$ and a constant c, such that $\mathcal{E}$ obtains input x^* and interacts with $\mathcal{F}$ in a black-box manner (we denote this interaction by $\mathcal{E}^{\mathcal{F}(x^*)}$) to output a string out, where $\Pr[R(x, \mathsf{out}) = 1] \geq q^c$ (over the randomness of $\mathcal{F}$ and $\mathcal{E}$).

This definition can be generalized to obtain strong witness signatures in the stateful hardware token model, exactly like Definition 7 was generalized to Definition 8 above.

Remark 4. *The model that we consider in this paper restricts the adversary to only use a single hardware token within each token it tries to construct (Refer Figure 1). That is, in the physical world, we restrict the adversary from encapsulating multiple honest party tokens inside a single token it constructs. We believe this restriction is quite reasonable and can be enforced, for example, by checking and limiting the physical size of the token that the adversary sends. A generalization of our model leads to proofs becoming significantly more messy and lengthier, and, is left to the future work.*

4.2 Single-Use Witness Signatures from Non-Malleable MIPs

In this section, we give a construction (according to Definition 9) of witness signatures in the stateful tamper-proof hardware token model, making black-box use of unconditional non-malleable two-prover proofs. We prove security in a setting where any adversarial token can encapsulate an honest token. In fact, our construction also satisfies the strengthened (adaptive) definition – it also yields *strong witness signatures* in the hardware token model.

To prove security, we show a reduction that interacts with any forger (that forges on some instance x of an NP language L) in a black-box manner; and makes black-box use of the simulator-extractor of the non-malleable two-prover proofs; to extract a witness w for x.

Theorem 5. *In the stateful hardware token model, it is possible to realize unconditionally secure witness signatures according to Definition 9, which can be verified an a-priori bounded number of times, with a total of two tokens sent from the signer to the verifier.*

Proof. We show that there exists a polynomial time (in the size of the instance and the running time of the forger) reduction $\mathcal{E}$, such that if the forger $\mathcal{F}$ (that may encapsulate tokens) forges and outputs a signature (m', S', T'), on a message that was never queried, and that verifies with probability $q > 1/\mathsf{poly}(\cdot)$ then $\mathcal{E}$ extracts out a witness by interacting with this forger with probability at least q^c for some constant c.

The witness signature is a non-malleable two-prover proof on instance x, where two tokens to play the role of both provers of the non-malleable two-prover proof. Let $\mathsf{Non\text{-}Malleable.Prove}(\cdot)$ and $\mathsf{Non\text{-}Malleable.Verify}(\cdot)$ denote the proving and verification algorithms of a non-malleable two-prover proof. We describe the algorithms below:

1. $\mathsf{Sign}(x, w, m)$: The signer on input the NP instance x and witness w generates a signature on message m by invoking $\mathsf{Non\text{-}Malleable.Prove}(x, w, m)$, where m acts as the tag of the non-malleable two-prover proof. It constructs two *stateful* single-use tokens S, T which have programmed in them, the algorithms corresponding to provers P_1 and P_2 respectively.

2. $\mathsf{Verify}(x, m, \sigma)$: To verify a signature σ comprised of two tokens S, T, the verifier runs the underlying algorithm $\mathsf{Non\text{-}Malleable.Verify}(x, m, S, T)$ on the two tokens acting as two non-communicating provers.

3. $\mathsf{Reduction}(\mathcal{F})$: This reduction (denoted by $\mathcal{R}_{\mathsf{sign}}$) extracts a witness w using the underlying simulator-extractor $\mathcal{R}$ of Figure 13.

   However, this setting needs a more careful analysis. In particular, a forger is allowed to see an unbounded number of signatures before issuing his own signature. He can also create tokens where each token encapsulates another honestly-generated token inside.

For all signatures that the forger inputs verification queries to, *before* creating his tokens, $\mathcal{R}_{\mathsf{sign}}$ uses the simulator of the non-malleable two-prover proof. Extraction occurs with the same probability because of the zero knowledge property of the two-prover proof.

Finally, the witness can be extracted for all cases of token encapsulation via the simulator-extractor strategy for the corresponding cases in the non-malleable two-prover proof system, according to Figure 13.

An exception is the case when the forger constructs two tokens S', T', such that S' encapsulates either P_i or Q_i, and T' encapsulates either P_j or Q_j for $i \neq j$; where (P_i, Q_i) and (P_j, Q_j) are tokens for two signatures on different messages m_i, m_j. In this case, $\mathcal{R}_{\mathsf{sign}}$ extracts the witness w by running the simulator for both (P_i, Q_i) and (P_j, Q_j) independently, and then the extractor $\mathcal{R}$. Since the extraction strategy is uniform across both hybrids, there exists a polynomial $\mathsf{poly}(\cdot)$ such that extraction occurs with probability $\mathsf{poly}(q)$ if q is the forger's verification probability.

This construction can straightforwardly be seen to satisfy the correctness and unforgeability properties of Definition 9, based on the correctness and tag-based non-malleability properties of the underlying non-malleable two-prover proof. Furthermore, the standalone zero knowledge property of the underlying non-malleable MIP, implies perfect witness indistinguishability of this construction, in a black-box manner. □

Bounded-Use Witness Signatures from Non-Malleable Two-Prover ZK Proofs. Single-use witness signatures can be extended to the n-verification setting by hardwiring sufficient randomness for n sequential executions of the non-malleable two-prover proof. Then, each verification is invoked with fresh randomness. However, the number of verifications n must be fixed before the start of the protocol.

4.3 Unbounded-Use Witness Signatures from Non-Malleable MIPs and One-way functions

Construction of Unbounded-Use Signatures In this section, we describe how to modify the construction of single-use witness signatures to obtain unbounded-use witness signatures. We use a psuedorandom function (PRF) with the same random key K hidden in both tokens. This PRF is invoked on a stateful counter to generate shared pseudorandomness in sync for the tokens. Our construction is as follows:

$\mathsf{Sign}(x, w, m)$: The signer samples a PRF key K, then outputs stateful tokens S and T which are constructed as described in Figure 15 and Figure 16. Both tokens are initialized with $\mathsf{count} = 0$.

Constants: PRF key K, counter count, instance x, witness w and message m.
Input: Challenge string σ.

1. Compute $R = \mathsf{PRF.Eval}(K, \mathsf{count})$.
2. Use randomness R and invoke the algorithm P_1 of $\mathsf{Non\text{-}Malleable.Prove}(x, w, m)$.
3. Set $\mathsf{count} = \mathsf{count} + 1$.

Figure 15: Token S

$\mathsf{Verify}(x, m, \sigma)$: In order to verify a signature σ comprised of two tokens S, T, the verifier runs the underlying algorithm $\mathsf{Non\text{-}Malleable.Verify}(x, m, S, T)$ on the two-tokens acting as two non-communicating provers. This can be done an unbounded number of times.

Correctness and soundness follow directly from the properties of the single use construction. We prove security formally by giving a reduction that extracts a witness from any forger.

Theorem 6. *In the stateful hardware token model, assuming one-way functions exist, it is possible to realize witness signatures according to Definition 9, which can be verified an unbounded number of times, with a total of two tokens sent from the signer to the verifier.*

Constants: PRF key K, counter count and message m.
Input: Challenge string τ.

1. Compute $R = \mathsf{PRF.Eval}(K, \mathsf{count})$.
2. Use randomness R and invoke the algorithm P_2 of $\mathsf{Non\text{-}Malleable.Prove}(x, w, m)$.
3. Set $\mathsf{count} = \mathsf{count} + 1$.

Figure 16: Token T

Proof. We show that there exists a polynomial time (in the size of the instance and the running time of the forger) reduction $\mathcal{E}$, such that if the forger $\mathcal{F}$ (that may encapsulate tokens) forges and outputs a signature (m', S', T'), on a message that was never queried, and that verifies with probability $q > 1/\mathsf{poly}(\cdot)$ then $\mathcal{E}$ extracts out a witness by interacting with this forger with probability at least q^c for some constant c.

We now describe how the reduction $\mathcal{E}$ works. $\mathcal{E}$ invokes $\mathcal{F}$ on x. $\mathcal{F}$ may query for signatures on messages m_p. For each fresh signature query p, the reduction maintains a counter count_p. Starting at $\ell = 1$, for each fresh verification query $q_{\ell,p}$ on a signature on m_p, the simulator samples fresh randomness $r_{\ell,p}$. Given the tokens of the verifier, it uses the extraction strategy of Figure 13 to extract a witness w for x (using fresh randomness $r_{\ell,p}$ to simulate the answers to each token query during the extraction phase).

We argue that this extractor succeeds in extracting a witness. This can be shown in a sequence of hybrids as follows. The first hybrid corresponds to the game where signer is given honestly generated unbounded-use signatures and it outputs a forgery. In the second hybrid, the only change is that the simulator does not compute the randomness $r_{\ell,p}$ as a PRF applied to the counter variable count_p, instead it generates fresh randomness for every verification query to the signature tokens. Indistinguishability between these two hybrids follows directly from the pseudo-randomness property of the PRF. In fact, since the forger's signature must verify with nearly the same probability (as otherwise one can use such a forger to break the pseudo-randomness property of the PRF), therefore extraction also occurs in this hybrid with the nearly same probability as the previous hybrid.

At this point, we follow the same sequence of hybrids as the bounded-use setting and argue indistinguishability between them to show that the reduction succeeds in extracting a witness from any forger with significant probability. □

5 Impossibility of Unconditional Unbounded-Use Witness Signatures

In this section, we prove that witness signatures for all of NP with unbounded verification, are impossible to construct unconditionally in the hardware token model. *We show that even the weaker notion of witness signatures in Definition 7 – where the forger must output a challenge instance x^* ahead of time, and then make all future queries on the same instance x^* – is impossible in the stateful hardware token model.* This is done by extending the notion of inaccessible entropy, first introduced in [HRVW09] and later used in [GIMS10] in the stateless token model. The stateful token model with unbounded queries is a strict generalization of the stateless token model, and hence this impossibility also extends to stateless tokens.

Recall that a witness signature, by definition, must be non-interactive. In the stateful token model, this translates to the signer creating polynomially many tokens and sending them to the verifier. Next, the verifier can make unbounded verification queries to these tokens. In this section, our main result is the following theorem in the stateful token model.

Theorem 7. *Unconditional unbounded-verification secure witness signatures, where the signer issues polynomially many independent tokens and each verification entails polynomially many rounds of interaction between the verifier and the tokens, are impossible in the stateful token model.*

Overview of the proof. We start by showing that witness signatures imply witness hiding arguments of knowledge with black-box extraction (or, equivalently, proofs of knowledge in the information theoretic setting).

Next, we prove that if unbounded verifications are allowed in the (stateful) token model, then there exists a curious extension to any verification protocol between the verifier and the tokens, that can estimate each of the signing token's response to every query with high probability, by running only a polynomial number of verification protocols with the token. (However since we are in the information-theoretic setting, this extension has unbounded computational power.) This is done via an extension of the *canonical entropy learner* of [GIMS10], to the setting of stateful tokens with unbounded verification. The essence of the proof is that the tokens can have only bounded entropy. Therefore, if queries are chosen wisely, it is possible to access all the entropy (and therefore predict the token's response to all challenges) with high probability with queries that are polynomial in the total entropy of the token.

Finally given such a curious verifier, we show that it is impossible to simultaneously maintain the witness hiding and proof of knowledge properties of a witness signature scheme. This technique for this part is inspired from similar techniques used to prove the impossibility of constant-round public coin zero-knowledge proofs in [GK90, HRVW09]. [GK90] observe that a public-coin verifier can be converted to one which resets for each query by re-sampling from its own randomness. Informally, rewinding such a verifier is useless. Then, a simulator for such a verifier can be used by a malicious prover to violate soundness.

We use a similar argument, but from the malicious verifier's side. The verifier first runs the canonical entropy learner in unbounded verifications, such that it can predict the prover's answers to any challenges for some execution. Next, the verifier constructs a cheating prover that answers any query by using the answers predicted by the learner. Informally, rewinding such a prover is useless. This is because the knowledge extractor could potentially use the values predicted by the entropy learner, to generate prover messages for the rewinding queries. Then, a knowledge extractor that extracts a witness from such a prover can be used by a malicious verifier to obtain a witness from the interaction. However, proving that the knowledge extractor extracts a witness from this cheating prover, requires a careful technical argument about the view of a verifier interacting with this prover. This proves that unconditionally secure witness signatures cannot exist in the stateful token model.

Now, we give the formal proof of Theorem 7. We begin by observing that a witness signature on an NP statement with any message is in fact, a (non-malleable) witness hiding argument of knowledge for the same statement.

Claim 1. *Witness signatures in the token model according to Definition 9 for all of* NP *imply witness hiding arguments of knowledge with black-box extraction, for all of* NP *in the token model.*

Proof. In order to give a witness hiding argument of knowledge for some statement $x \in L$, on input a witness w for x, the prover picks some message m and sends $m, \sigma = \mathsf{WitnessSignature.Sign}(x, w, m)$ to the verifier. The verifier outputs $b = \mathsf{WitnessSignature.Verify}(x, \sigma, m)$. Soundness and completeness follow directly from the soundness and completeness of the witness signature.

Next, we show that this is an argument of knowledge, with a black-box extractor $\mathcal{E}$ that interacts with the prover to output a witness w for $x \in L$. The extractor algorithm is as follows. On input the prover, it runs the black-box witness signature reduction $\mathsf{WitnessSignature.Reduce}$ on input the code of the prover as the forger, and outputs the witness extracted by $\mathcal{R}$. It is straightforward to see that the probability of extraction is the same as the success probability of the reduction.

Finally, we show that the proof $(m, \sigma) = \mathsf{WitnessSignature.Sign}(x, w, m)$ is witness hiding. This is true if for any verifier $\mathcal{V}$ that can output some witness $w'(x)$ for $x \in L$ with significant probability after seeing a proof, there exists an extractor that interacts with $\mathcal{V}$ in a black-box way and outputs a witness $w''(x)$ for $x \in L$ such that the distributions $w'(x)$ and $w''(x)$ are identical. In other words, the extractor outputs the same witness that was output by the verifier after seeing a proof. If an instance x has two or more witnesses, then witness signatures are also witness hiding by virtue of being witness indistinguishable.

Suppose instance x has only a single witness w, then we construct an extractor for any $\mathcal{V}$ as follows. The extractor first constructs a forger $\mathcal{F}_\mathcal{V}$ that uses $\mathcal{V}$ in a black-box way. $\mathcal{F}_\mathcal{V}$ requests a signature on an arbitrary message m_1, and forwards this internally as a proof to the verifier $\mathcal{V}$. Suppose $\mathcal{V}$ outputs a witness $w'(x)$ for x with probability $c \geq 1/\mathsf{poly}(\kappa)$ for some polynomial $\mathsf{poly}(\cdot)$, on input the signature on message m_1. The forger $\mathcal{F}_\mathcal{V}$ picks another message $m_2 \neq m_1$ and uses $\mathsf{WitnessSignature.Sign}(x, w', m_2)$ to generate a new signature on message m_2. It is straightforward to see that this forgery verifies with probability at least c. The extractor then executes the witness signature reduction algorithm $\mathsf{WitnessSignature.Reduce}$ on this forger $\mathcal{F}_{\mathcal{V}}$. By definition, the reduction outputs a witness w'' for x with probability $\mathsf{poly}(c)$. Since x has only one witness, the distributions w'' and w' are identical and we are done. □

Now, we prove that unconditional unbounded-verifiable witness hiding arguments of knowledge are impossible in the stateful token model. In the next lemma, we construct a learner which learns the output of some token to any first query of the protocol. Next, we will generalize this to a learner which learns the response of the token to entire partial protocol transcripts.

Claim 2. *Consider n stateful tokens $T_1, T_2, \ldots T_n$ each with entropy $H(T_i)$ for $i \in [n]$, which are used for unbounded verifications. Then, there exists a canonical entropy learner $\mathcal{L}$ that runs an expected $(\Sigma_{i\in[n]} H(T_i))/\epsilon$ protocol executions with the token T, such that there exists a fresh execution where $\mathcal{L}$ can predict the response of each token to any first-query (for a protocol) with probability at least $(1-\epsilon)$ for arbitrarily small constant ϵ.*

Proof. At each execution p (starting at $p = 1$), $\mathcal{L}$ computes the lexicographically smallest first-query-set $x_{i,p}$ for $i \in [n]$, such that the response of the tokens to this set $x_{i,p}$ is ϵ-unpredictable conditioned all the query-answers obtained from protocol executions so far. It then records the queries $x_{i,p}$ and the answers obtained. The remaining queries of the protocol are randomly sent and answers are not noted.

Denote the total set of protocol executions made by Q, and the size of this set is $|Q|$. Denote the set of entire query-answers learned in the p^{th} protocol by Q_p. Denote a partial set of query-answers learned till execution $(p-1)$ by $Q^{(p-1)}$, where $Q^{(0)} = \perp$.

Then, conditioned on $Q^{(p-1)}$, query-set $\cup_{i\in[n]} x_{i,p}$ (which is a first-query-set) is ϵ-unpredictable, when there does not exist any set of answers $a_1, a_2, \ldots a_n$ such that $\Pr[T_1(x_{1,p}) = a_1 \wedge T_2(x_{2,p}) = a_2 \wedge \ldots T_n(x_{n,p}) = a_n] > (1-\epsilon)$.

That is, the Shannon entropy $H(\cup_{i\in[n]} T_i(x_{i,p}) | Q^{(p-1)}) \geq \epsilon \cdot log(1/\epsilon) + (1-\epsilon) \cdot log(1/1-\epsilon) > \epsilon$, for $\epsilon < \frac{1}{2}$. This can be proved following [GIMS10]. Moreover, if $|Q| > p$, then there is a sequence of at least p (ϵ-unpredictable) first-query-sets to the tokens conditioned on previous queries. That is, $H(\cup_{i\in[n]} T_i(x_{i,p}) | Q^{(p-1)} \ \wedge \ |Q| \geq p) > \epsilon$.

For a position p, we note that the information about $\cup_{i\in[n]} T_i(x_{i,p})$ is encoded in $Q^{(p-1)}$. Then, the following holds.

$$
\begin{aligned}
\sum_{i\in[n]} H(T_i) &= H(\cup_{i\in[n]} T_i(x_{i,1}), \cup_{i\in[n]} T_i(x_{i,2}) \ldots) \\
&\geq \sum_p H(\cup_{i\in[n]} T_i(x_{i,p}) | Q^{(p-1)}) \\
&\geq \sum_p H(\cup_{i\in[n]} T_i(x_{i,p}) | Q^{(p-1)} \ \wedge \ \Pr[|Q| \geq p) \cdot (\Pr[|Q| \geq p) \\
&\geq \epsilon \cdot \sum_p \Pr[|Q| \geq p \\
&\geq \epsilon \cdot E[|Q|]
\end{aligned}
$$

Then, $E[|Q|] \leq {}^{(\Sigma_{i\in[n]} H(T_i))}/_{\epsilon}$ □

Now, we prove a stronger version of this claim where we try to learn the entire transcript generated by the prover (and not just a single message).

Lemma 9. *Consider n stateful tokens $T_1, T_2, \ldots T_n$ each with entropy $H(T_i)$ for $i \in [n]$, which are used for unbounded verification protocols of polynomial (r) rounds each. Then, there exists a canonical entropy learner $\mathcal{L}$ that runs $(\Sigma_{i\in[n]} H(T_i)/\epsilon)$ executions with the tokens, such that there exists a fresh execution where $\mathcal{L}$ can predict the response of the prover to any partial transcript of the verifier (consisting of r query-sets corresponding to the r rounds of interaction) with probability $(1-\epsilon)$ for arbitrarily small ϵ.*

Proof. At each execution p (starting at $p = 1$), $\mathcal{L}$ computes the lexicographically smallest partial transcript for the verifier $\tau_{i,p}$ for $i \in [n]$, such that the response of the tokens to this set $\tau_{i,p}$ is ϵ-unpredictable conditioned all the query-answers obtained from protocol executions so far. It then records the queries $x_{i,p}$ and the answers obtained. The remaining queries of the protocol are randomly sent and answers are not noted.

Denote the total set of protocol executions made by Q, and the size of this set as $|Q|$. Denote the set of entire query-answers learned in the p^{th} protocol by Q_p. Denote a partial set of query-answers learned till execution $(p-1)$ by $Q^{(p-1)}$, where $Q^{(0)} = \perp$.

Then, conditioned on $Q^{(p-1)}$, a partial verifier transcript $\cup_{i\in[n]}\tau_{i,p}$ is ϵ-unpredictable, when there does not exist any set of answers $a_{i,j}$ for $i \in [n]$ and $j \in [r]$ such that $\Pr[T_1(\tau_{1,1,p}) = a_{1,1} \ \wedge \ T_2(\tau_{2,1,p}) = a_{2,1} \ \wedge \ \ldots T_n(\tau_{n,1,p}) = a_{n,1} \ \wedge \ T_1(\tau_{1,2,p}) = a_{1,2} \ \wedge \ T_2(\tau_{2,2,p}) = a_{2,2} \ \wedge \ \ldots T_n(\tau_{n,2,p}) = a_{n,2} \ldots T_1(\tau_{1,r,p}) = a_{1,r} \ \wedge \ T_2(\tau_{2,r,p}) = a_{2,r} \ \wedge \ \ldots T_n(\tau_{n,r,p}) = a_{n,r}] > (1-\epsilon)$.

That is, the Shannon entropy $H(\cup_{i\in[n]}T_i(\tau_{i,p})|Q^{(p-1)}) \geq \epsilon \cdot log(1/\epsilon) + (1-\epsilon) \cdot log(1/1-\epsilon) > \epsilon$, for $\epsilon < \frac{1}{2}$. This can be proved following [GIMS10]. Moreover, if $|Q| > p$, then there is a sequence of at least p (ϵ-unpredictable) first-query-sets to the tokens conditioned on previous queries. That is, $H(\cup_{i\in[n]}T_i(x_{i,p})|Q^{(p-1)} \ \wedge \ |Q| \geq p) > \epsilon$. For a position p, we note that the information about $\cup_{i\in[n]}T_i(x_{i,p})$ is encoded in $Q^{(p-1)}$. Then, the following holds.

$$\begin{aligned}
\sum_{i\in[n]} H(T_i) &= H(\cup_{i\in[n]}T_i(\tau_{i,1}), \cup_{i\in[n]}T_i(\tau_{i,2})\ldots) \\
&\geq \sum_p H(\cup_{i\in[n]}T_i(\tau_{i,p})|Q^{(p-1)}) \\
&\geq \sum_p H(\cup_{i\in[n]}T_i(\tau_{i,p})|Q^{(p-1)} \ \wedge \ \Pr[|Q| \geq p) \cdot (\Pr[|Q| \geq p) \\
&\geq \epsilon \cdot \sum_p \Pr[|Q| \geq p \\
&\geq \epsilon \cdot E[|Q|]
\end{aligned}$$

Then, $E[|Q|] \leq {(\sum_{i\in[n]} H(T_i))}/{\epsilon}$ □

Lemma 10. *Consider witness hiding arguments of knowledge with black-box extraction in the stateful token model, where the prover sends polynomially many tokens as his proof and the verifier has polynomially many (say r) rounds of interaction with each token.*

Suppose the i^{th} token has entropy $H(T_i)$, for $i \in [n]$. Then there exists a curious extension of the verifier such that for any ϵ, with a total of $\mathsf{poly}\left(\sum_{i\in[n]}(H(T_i)/\epsilon)\right)$ *verification queries to the tokens, the verifier can extract a witness w from the tokens with non-negligible probability.*

Proof. Consider the following cheating verifier strategy $\mathcal{V}(x)$ that uses the PoK extractor $\mathcal{E}(x)$.

- Internally simulate $\mathcal{E}(x)$ step by step.
- Suppose $\mathcal{E}(x)$ makes at most $\mathcal{T}$ queries to the tokens (which may include rewinding the prover tokens several polynomially many times), and a single verification involves r rounds of interaction with the tokens. Without loss of generality, we may assume that the extractor queries all n tokens on a query set $x_j^{(n)} = x_{1,j}, x_{2,j}, \ldots x_{n,j}$ (if not, replace the tokens that are not queried by random queries) corresponding to the j^{th} round, conditioned on a previous partial transcript $x_{[j-1]}$.
- Run $\mathcal{L}$ with input the current partial transcript, to predict the output set for this round $a_j^{(n)} = a_{1,j}, a_{2,j}, \ldots a_{n,j}$ of the tokens to the new query $x_j^{(n)}$ (note that $a_j^{(n)}$ is correct with probability at least $(1-\epsilon)$). If the current partial transcript is not one of the predicted transcripts, $\mathcal{L}$ will abort. This happens with probability at most $(1 - r\epsilon)$.

Our goal will be to prove that this verifier strategy successfully extracts a witness for x. $\mathcal{V}$ makes unbounded polynomial queries to the tokens and uses the canonical entropy learner $\mathcal{L}$ defined in Lemma 9.

Then, there exists a protocol execution for which $\mathcal{L}$ can predict the answers to any query with probability $(1-\epsilon)$. At each step, $\mathcal{V}$ runs $\mathcal{L}$ with input the current partial transcript, uses $\mathcal{L}$ to predict the output $a_j^{(n)}$ to any token query $x_j^{(n)}$, and returns the value $a_j^{(n)}$ to $\mathcal{E}$ as the output of the prover token.

Via a simple union bound, we observe that with probability $(1-\mathcal{T}r\epsilon)$, the view of the extractor while interacting with $\mathcal{V}$, is identical to that of the extractor interacting with an honest prover. Moreover, in an interaction with the honest prover, suppose extraction occurs with probability c (since verification occurs with probability 1). Thus, extraction occurs with probability at least $c(1-\mathcal{T}r\epsilon)$. Then, a valid witness w is extracted by $\mathcal{E}$, with probability at least $c(1-\mathcal{T}r\epsilon)$ even when $\mathcal{E}$ interacts with $\mathcal{V}$. By setting $\epsilon = 1/\mathcal{T}r$, we have that the witness is extracted with probability at least c. This violates the witness hiding property and proves our lemma. □

6 Acknowledgement

We would like to thank Eshan Chattopadhyay and Xin Li for helpful discussions regarding the construction and properties of many-many non-malleable codes.

References

[ADKO15] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 459–468, 2015. 1

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 774–783. ACM, 2014. 1, 6

[AGM+15] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 538–557, 2015. 1

[BCFW09] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. Foundations of non-malleable hash and one-way functions. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 524–541. Springer, 2009. 1

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 113–131. ACM, 1988. 1

[CCFP11] Hervé Chabanne, Gérard D. Cohen, Jean-Pierre Flori, and Alain Patey. Non-malleable codes from the wire-tap channel. In *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*, pages 55–59. IEEE, 2011. 1

[CCP12] Hervé Chabanne, Gérard D. Cohen, and Alain Patey. Secure network coding and non-malleable codes: Protection against linear tampering. In *Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012*, pages 2546–2550. IEEE, 2012. 1

[CG14] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 440–464, 2014. 1

[CGL15] E. Chattopadhyay, V. Goyal, and X. Li. Non-Malleable Extractors and Codes, with their Many Tampered Extensions. *ArXiv e-prints*, May 2015. 1, 4, 7, 35, 36, 37

[CGS08] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In Smart [Sma08], pages 545–562. 6, 9

[CKM10] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: Built-in tamper resilience. *IACR Cryptology ePrint Archive*, 2010:503, 2010. 1

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96. Springer, 2006. 2, 3

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 494–503. ACM, 2002. 1

[Coh15] Gil Cohen. Local correlation breakers and applications to three-source extractors and mergers. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:38, 2015. 37

[CZ14] Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 306–315, 2014. 1

[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, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 542–552. ACM, 1991. 1

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. 1

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 434–452. Tsinghua University Press, 2010. 1

[DW09] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In Mitzenmacher [Mit09], pages 601–610. 1

[FMNV14] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 465–488, 2014. 1

[FMVW14] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 111–128, 2014. 1

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476. ACM, 2013. 2

[GIMS10] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. Interactive locking, zero-knowledge pcps, and unconditional cryptography. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2010. 5, 28, 29, 30, 31

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2010. 6, 9

[GK90] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 1990. 29

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008. 9

[HHRS07] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In *FOCS*, pages 669–679. IEEE Computer Society, 2007. 5

[HRVW09] Iftach Haitner, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Inaccessible entropy. In Mitzenmacher [Mit09], pages 611–620. 5, 28, 29

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128. Springer, 2007. 2, 9

[Lin14] Yehuda Lindell. An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. *IACR Cryptology ePrint Archive*, 2014:710, 2014. 8

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2012. 1

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In Mitzenmacher [Mit09], pages 179–188. 1

[LS91] Dror Lapidot and Adi Shamir. A one-round, two-prover, zero-knowledge protocol for NP. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 213–224. Springer, 1991. 1, 4, 11, 21

[Mit09] Michael Mitzenmacher, editor. *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*. ACM, 2009. 33, 34

[MS08] Tal Moran and Gil Segev. David and goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In Smart [Sma08], pages 527–544. 6

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 427–437. ACM, 1990. 1

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 232–241. ACM, 2004. 1

[Raz95] Ran Raz. A parallel repetition theorem. In Frank Thomson Leighton and Allan Borodin, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 447–456. ACM, 1995. 1, 12

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 543–553, 1999. 1, 2

[SCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 566–598, 2001. 2

[Sma08] Nigel P. Smart, editor. *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*. Springer, 2008. 33, 34

[Zha16] Mark Zhandry. How to avoid obfuscation using witness prfs. In *Proceedings of TCC 2016*, 2016. 2

A Many-Many Non-Malleable Codes with Symmetric Decoding

For our construction of a non-malleable two prover interactive proof, we require the following properties out of a split-state non-malleable code.

1. The code should support a tampering degree N (specified by the sigma protocol for the language $\mathcal{L}$ and the instance x) for a message space $\{0,1\}^{|\mathsf{tag}|+|y_i|}$. Here, y_i represents a coordinate element in a vector sampled from $\mathcal{S}_x$ and N is the dimension of the vector (typically $\theta(\kappa^2)$).

2. Given a left share L (or right share R), it should be possible to efficiently sample uniformly the right share R (or left share L) such that (L, R) encode any target message m.

3. For all strings $L, R \in \{0,1\}^n$, $\mathsf{Dec}(L, R) = \mathsf{Dec}(R, L)$ should be true. This is because of the following reason. In our construction for non-malleable proofs, when the prover P_2' is in a session with P_2 such that P_1' opens a session with P_1, our reduction queries P_2' on two strings τ_1 and τ_2. These queries induce a split state tampering function on the non-malleable encodings of $\mathsf{tag}||y_{i,j}$ for $i \in [\kappa]$ and $j \in [N]$. However, the tampering function tampers with a mixture of left and right shares. For example, let $(f_1, g_1), .., (f_N, g_N)$ be the tampering functions tampering codes (L_i, R_i) for $i \in [u]$. Then, for all $j \in [N]$, f_j for every i may tamper any one of the two shares (L_i, R_i) while g_j tampers the other. This is because the prover P_2' on query τ_1 and τ_2 queries P_2 on τ_1' and τ_2' which on each query releases a collection of left and right shares of encodings.

   In such a situation if the decoding is symmetric (or commutative) it is possible to simulate the tampering experiment. Consider $N = 1$ and $u = 2$ in the example above, that is, f_1 tampers with the joint distribution (R_1, L_2) and g_1 tampers with the joint distribution (L_1, R_2), then we can replace L_2, R_2 with an encoding of 0 and the resulting distribution should be statistically close to the distribution of the tampering experiment. Since $\mathsf{Dec}(f_1(R_1, L_2), g_1(L_1, R_2)) = \mathsf{Dec}(g_1(L_1, R_2), f_1(R_1, L_2))$, we can again replace (L_1, R_1) with an encoding of 0 and the output is still close to the output of the tampering experiment.

We note that the construction in [CGL15] already satisfies the first two properties. We now show how to modify the construction in [CGL15] so that it satisfies all the three properties. Let the scheme in [CGL15] be denoted by $(\mathsf{Enc}, \mathsf{Dec})$, the resulting scheme $(\mathsf{Enc}', \mathsf{Dec}')$ is constructed as described below. Let us say that each code share output by the scheme $(\mathsf{Enc}, \mathsf{Dec})$ be in $\{0,1\}^n$ and message space be $\{0,1\}^m$.

- $\mathsf{Enc}' : \{0,1\}^m \to \{0,1\}^{2n+2}$. On input s, it samples $\mathsf{Enc}(s) \to (L, R)$ and outputs $(L||0, R||1)$.

- $\mathsf{Dec}'\{0,1\}^{n+1} \times \{0,1\}^{n+1} \rightarrow \{0,1\}^m$. On input (L', R'), it parses $L' = L||b_1$ and $R' = R||b_2$ where $b_1, b_2 \in \{0,1\}$. If $b_1 = b_2$ it outputs $\perp$. Else, if $b_1 = 0$ it outputs $\mathsf{Dec}(L, R)$ otherwise it outputs $\mathsf{Dec}(R, L)$.

Note that decoding is a commutative operation now. It is possible to show that if the underlying code supports a tampering degree t, then the modified code also supports a tampering degree t. The code is secure against all split state functions $(f_1, g_1), .., (f_t, g_t)$ that tampers any polynomial U codewords (L_i, R_i) for $i \in [U]$ in the following manner.

1. There exists a set $S \subseteq [U]$ such that for all $j \in [t]$, f_j takes as input $\{L_i\}_{i \in S}$,$\{R_i\}_{i \in [U] \setminus S}$ and g_j takes as input $\{L_i\}_{i \in S}$,$\{R_i\}_{i \in [U] \setminus S}$.
2. For all $j \in [t]$, f_j outputs a string of the form $L'_j||0$ and g_j outputs a string of the form $R'_j||1$.

In the following section, we sketch about why this modification is secure.

A.1 Proof Overview

Notation: For any two distributions, $\mathcal{D}_1, \mathcal{D}_2$ by $|\mathcal{D}_1 - \mathcal{D}_2|$ we denote the statistical distance between the two distributions. We say that the source X is an (n, k) source if it has a support on $\{0,1\}^n$ and has min-entropy $H_\infty(X) = k$.

We first define a $(2, t)$ non-malleable extractor. Our definitions are borrowed from [CGL15].

Definition 10. *A function $nmExt : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^m$ is a seedless (2, t)-non-malleable extractor at min-entropy k and error ϵ if it satisfies the following property: If X and Y are independent (n, k) sources and $\mathcal{A}_1 = (f_1, g_1), .., \mathcal{A}_t = (f_t, g_t)$ are t arbitrary 2-split state tampering functions, then there exists a random variable $D_{\mathbf{f},\mathbf{g}}$ on $(\{0,1\}^m \cup \{same^*\})$ which is independent of X and Y such that*

$$|(nmExt(X,Y), nmExt(\mathcal{A}_1(X,Y)), ..., nmExt(\mathcal{A}_t(X,Y))) - (U_m, copy^{(t)}(D_{\mathbf{f},\mathbf{g}}, U_m))| < \epsilon$$

where both U_m's refer to the same uniform m-bit string.

Here we define following functions.

$$copy(x, y) = \left\{ \begin{array}{cl} x & \text{if } x \neq same^* \\ same^* & \text{otherwise} \end{array} \right\}$$

Similarly we define $copy^{(t)}((x_1, .., x_t), (y_1, .., y_t)) = (copy(x_1, y_1), .., copy(x_t, y_t))$. Then once we have a non-malleable extractor it can be compiled to a non-malleable code as described in [CGL15].

Imported Theorem 3. *[CGL15] Let $nmExt : (\{0,1\}^n \times \{0,1\}^n) \rightarrow \{0,1\}^m$ be a polynomial time computable seedless $(2, t)$ non-malleable extractor for min-entropy n with error ϵ. Then there exists a one-many non-malleable code with an efficient decoder in the 2-split-state model with tampering degree t, block length $= 2n$, relative rate $m/2n$, and error $= \epsilon 2^{mt+1}$.*

The one-many non-malleable codes in the 2-split-state model is define in the following way: For any message $s \in \{0,1\}^m$, the encoder $\mathsf{Enc}(s)$ outputs a uniformly random string from the set $nmExt^{-1}(s) \in \{0,1\}^{2n}$. For any codeword $c \in \{0,1\}^{2n}$, the decoder Dec outputs $nmExt(c)$. Thus for the encoder to be efficient, one need to sample almost uniform from $nmExt^{-1}(s)$. [CGL15] constructs a non malleable extractor for independent $(n, n - n^\gamma)$ sources which has an explicit sampling algorithm that allows for sampling uniformly from $nmExt^{-1}(s)$ for any s.

Let γ be a small enough constant and C a large enough one. Let $t = n^{\gamma/C}$. [CGL15] construct an explicit function $nmExt : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^m$, $m = n^{\Omega(1)}$ which satisfies the following property: If X and Y be independent $(n, n - n^\gamma)$ sources on $\{0,1\}^n$, and $\mathcal{A}_1 = (f_1, g_1), .., \mathcal{A}_t = (f_t, g_t)$ are arbitrary 2-split sate tampering functions such that for any $i \in [t]$, at least one of f_i or g_i has no fixed points, the following holds:

$$|(nmExt(X,Y), nmExt(\mathcal{A}_1(X,Y)), ..., nmExt(\mathcal{A}_t(X,Y))) - (U_m, nmExt(\mathcal{A}_1(X,Y)), ..., nmExt(\mathcal{A}_t(X,Y)))| \leq \epsilon$$

where $\epsilon = 2^{-n^{\Omega(1)}}$.

By a convex combination argument, they show that if $nmExt$ satisfies the property above, then it is indeed a seedless $(2,t)$-non-malleable extractor.

We now define a new function $nmExt' : \{0,1\}^{n+1} \times \{0,1\}^{n+1} \rightarrow \{0,1\}^m$ (which is essentially our decoder for the non -malleable code) for $(n+1, n-n^{\gamma})$ sources as follows: $nmExt'(x', y')$: Takes as input strings $x' = x||1$ and $y' = y||0$ (in any order; x is sampled from X and y is sampled from Y) where x and y are in $\{0,1\}^n$, and computes $nmExt(x,y)$. If the strings do not satisfy the format it outputs $\perp$. We call the distribution $X' = X||1$ and $Y' = Y||0$. Hence, $nmExt'$ is a commutative function on x', y'. We can prove that this function is a non-malleable extractor for sources $X' = X||0$ and $Y' = Y||1$ such that X and Y are independent $(n, n-n^{\gamma})$ sources (for the constant γ). The tampering adversaries $\mathcal{A}'_i = (f'_i, g'_i)$ are such that there exists a bit b_i so that f_i outputs in $\{0,1\}^n||b_i$ and g_i outputs in $\{0,1\}^n||\bar{b_i}$.

In order to prove that the resulting function is a non-malleable extractor, we need to check the following weaker property. By a convex combination argument, it is possible to show that this property suffices to prove that this function is a non-malleable extractor for the class of tampering adversaries and distributions described above. The property is that for $t' = n^{\gamma/C}$,

$$|(nmExt'(X', Y'), nmExt'(\mathcal{A}'_1(X', Y')), \ldots, nmExt(\mathcal{A}'_{t'}(X', Y')))-$$

$$(U_m, nmExt'(\mathcal{A}'_1(X', Y')), \ldots, nmExt'(\mathcal{A}'_{t'}(X', Y')))| \leq \epsilon$$

where $\epsilon = 2^{-n^{\Omega(1)}}$.
Here, we define the class of valid adversary as any split-state tampering function $\mathcal{A}'_i = (f'_i, g'_i)$ which additionally satisfies the property that at least one of f'_i or g'_i has no fixed points, and that for some $b_i \in \{0,1\}$, f'_i outputs in $\{0,1\}^n||b_i$ and g'_i outputs in $\{0,1\}^n||\bar{b_i}$.

Then, using this non-malleable extractor we can construct a non- malleable code secure against split state adversaries of the following kind. The adversaries consists of a family of split state functions $(f_1, g_1), .., (f_{t'}, g_{t'})$ that tamper any polynomial u codewords (L_i, R_i) for $i \in [u]$.

1. There exists a set $S \subseteq [u]$ such that for all $j \in [t']$, f_j takes as input $\{L_i\}_{i \in S}$,$\{R_i\}_{i \in [u] \setminus S}$ and g_j takes as input $\{L_i\}_{i \in S}$,$\{R_i\}_{i \in [u] \setminus S}$.
2. For all $j \in [t']$, f_j outputs a string of the form $L'_j||b_j$ and g_j outputs a string of the form $R'_j||\bar{b_j}$ where $b_j \in B$.

Note that because encoding operation is efficient in the underlying coding scheme, it is also efficient for the modified coding scheme.

This (weaker) property described about $nmExt'$ translates to the following property on the underlying extractor $nmExt$ for sources X and Y i.e.

$$|(nmExt(X,Y), nmExt(\mathcal{A}_1(X,Y)), .., nmExt(\mathcal{A}_{t'}(X,Y)))-(U_m, nmExt(\mathcal{A}_1(X,Y)), .., nmExt(\mathcal{A}_{t'}(X,Y)))| \leq \epsilon$$

where $\epsilon = 2^{-n^{\Omega(1)}}$. Here each $\mathcal{A}_i$ for $i \in [t']$ comprises of two arbitrary functions (f_i, g_i) and can be either of the two types. $\mathcal{A}_i$ on input (X,Y):

1. Outputs $(f_i(X), g_i(Y))$ (in which case at least one of f_i or g_i has no fixed points)
2. Outputs $(g_i(Y), f_i(X))$

We note that the underlying construction in [CGL15] already satisfies this property. Hence, $nmExt'$ gives us our desired non-malleable code with symmetric decoding operation. We now give an overview of why this property is satisfied by [CGL15].

The $(2,t')$ non-malleable extractor in [CGL15] works by computing 'flip-flop' alternating extraction (introduced in [Coh15]) between X and Y, ℓ number of times (ℓ is a function of t', n and m) and outputs this extracted value. The key idea of why the extraction $nmExt(X,Y)$ still looks independent and uniform in the presence of output of $nmExt(f(Y), g(X))$ is that at i^{th} step when extraction is done from X using a seed S_i, the corresponding seed S'_i in evaluation of $nmExt(f(Y), g(X))$ is a deterministic function of X (fixing all prior seeds) and independent of S_i (which is a deterministic function of Y). Since the size of the seed S'_i is small in comparison with n, X still has enough entropy left even when S'_i is released. Moreover, X and

Y are still independent even when S_i and S'_i are released. Hence, extraction at i^{th} step in $nmExt(X, Y)$ looks uniform and independent from corresponding extraction in $nmExt(f(Y), g(X))$ because of almost full entropy in S_i and the property of 'flip-flop' extraction used in the construction.

Although the resulting code $(\mathsf{Enc}', \mathsf{Dec}')$ is not secure against arbitrary family of split state functions, it is secure against split state functions (f, g) such that there exists a bit $b \in \{0, 1\}$ so that f tampers one share and outputs a string of the form $x||b$ while g tampers the other share and outputs a string of the form $y||\bar{b}$. A code secure against such adversaries suffices for our construction of the non-malleable proof. This is because for any index ind, the reduction expects either a left or right share of the encoded values depending upon the bits τ^1_{ind} and τ^2_{ind}. Here τ^1 and τ^2 is the query made by the reduction to the MIM in the case when P_1 and P_2 are in a session with P'_1 and P'_2 respectively.