Witness Signatures and Non-Malleable Multi-Prover Zero-Knowledge Proofs

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

Abstract

Motivated by the goal of removing trusted setup assumptions from cryptography, we introduce the notion of witness signatures. This primitive allows any party with a valid witness to an NP statement to sign a message *on behalf* of that statement. We also require these signatures to be unforgeable: that is, producing a signature on a new message (even given several message, signature pairs) should be as hard as computing a witness to the NP statement itself. Witness signatures are closely related to previously well-studied notions such as non-malleable non-interactive zero knowledge arguments, and signatures of knowledge.

In this work, we formalize this notion and show that most natural definitions are impossible in the plain model without any setup assumptions. While still wanting to avoid a central trusted setup, we turn to the tamper proof hardware token model of Katz (Eurocrypt 2007). Interestingly, we show witness signatures in the hardware token model are closely related to what we call *non-malleable multi-prover zero-knowledge proofs* in the plain model (i.e. without hardware tokens). We initiate the study of non-malleable multi-prover zero-knowledge proofs, and, provide an *unconditional* construction of single round non-malleable two-prover zero-knowledge proofs. We then use this primitive to obtain an unconditional construction of witness signatures in the hardware token model.

Our construction makes a novel use of *non-malleable codes*. In particular, we crucially rely on the notion of *many-many* non-malleable codes introduced recently by Chattopadhyay, Goyal and Li (ECCC 2015). Our construction is unconditional, is extremely efficient (in terms of computation, number of tokens, and rounds of interaction with the token), and, only relies on elementary computations such as inner products.

Finally, this construction yields signatures which can only be verified a bounded number of times. Towards that end, we show how to extend it to get the unbounded (polynomial) verification property relying on the minimal additional assumption of one-way functions. We also show that obtaining unconditional unbounded-verifiable witness signatures under black-box extraction, is impossible even with access to an unbounded number of stateful tamper-proof hardware tokens- thereby giving a matching lower bound. This is done by relying on the techniques from the work of Goyal et al (Crypto 2012) (which in turn builds on techniques from the black-box separation literature). In particular, we rely on the notion of "inaccessible entropy" introduced in prior works.

[*]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

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. How should a scientist convince the government 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 what we call *witness signatures*, that offer a simple solution to this problem. Roughly, witness signatures allow any party with a witness to some NP statement x to sign a message, 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 [16] and witness PRFs [31]. 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. For example, this primitive is closely related to non-malleable non-interactive zero-knowledge arguments of knowledge (NIZKAoK). To construct witness signatures from (tag based) non-malleable NIZKAoKs, the signer can give a zero-knowledge argument of knowledge of the witness to the NP statement, non-malleably tagged by the message being signed. Another very similar primitive called 'signatures of knowledge' was explored by Chase and Lysyanskaya [6]. 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 [24] 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 multi-prover zero-knowledge (ZK) proofs*. Multi-prover zero knowledge proofs for NP allow two non-communicating provers to prove an NP statement x to a verifier, such that if $x \in L$, the proof verifies with overwhelming probability, and if $x \notin L$, then no cheating provers can generate a proof that verifies with non-negligible probability. The soundness of such a proof is guaranteed provided the two-provers do not communicate after the start of the protocol.

A non-malleable two-prover ZK proof can be seen as a natural extension of a two-prover ZK proof. Consider the following situation: In a left interaction there are two honest provers proving some statement x, who instead of interacting with a stand-alone verifier, interact with two verifiers. Both verifiers are part of a man-in-the-middle adversary, and act as two-provers proving some related statement x'. Indeed, the two man-in-the-middle provers cannot talk to each other after the right protocol starts (otherwise, soundness itself can be violated!). We formalize and construct (tag-based) non-malleable ZK proofs. This ensures that two man-in-the-middle verifiers cannot malleate an honestly generated proof for a statement $x \in L$ and tag tag, to a proof for the same statement on a different tag tag' unless they know a witness for x. To the best of our knowledge, this notion has not been explored prior to our work. Our construction of non-malleable two-prover zero knowledge, and the notion itself, can be seen as a primitive of independent interest.

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. This construction is unconditionally secure and allows for a polynomially bounded number of verifications. We also give an alternate construction based on one-way functions, that allows for an unbounded number of verifications. Finally, we prove a matching impossibility – showing that it is impossible to use even an unbounded number of stateful hardware tokens to obtain unconditionally secure witness signatures that verify an unbounded number of times.

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 [6] are those of ring signatures and delegatable anonymous credentials, and Chase and Lysyanskaya [6] 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).

1.1 Our Contributions

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.

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 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.

- A very efficient *unconditionally secure* protocol that uses only two tokens and 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). Our signature scheme is truly efficient and utilizes only basic operations like taking inner products and multiplications. To achieve this goal, we make a novel use of non-malleable codes introduced by [13, 7].

[1]However, the definitions of [6] 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.

- A matching lower bound, which proves that an unconditionally secure protocol for witness signatures, cannot exist even in the stateful token model, if unbounded verifiability is required. To do this, we use techniques from the work of Goyal et al [18] which in turn borrow from the literature on black-box separations. In particular, we rely on the notion of *inaccessible entropy* that was introduced in [21, 22].

- A very efficient protocol that uses two tokens and realizes unbounded verifiable witness signatures, assuming the existence of one-way functions.

On the Optimality of our Construction. Goyal et al. [20] obtain general non-interactive unconditional UC secure computation in the hardware token model. This general positive result also applies to the witness signature functionality, thereby yielding a feasibility result in the hardware token model. However this gives an extremely inefficient solution literally involving the transfer of millions of tokens. To understand the number of tokens involved, [20] would require several tokens for each gate of the circuit which implements the next message function of the Ishai et al [23] UC secure two-party computation protocol. On the other hand, our techniques yield truly efficient witness signature schemes requiring only elementary computation and sending only two tokens.

We remark here, that our construction is optimal in terms of number of tokens: any unconditional construction *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 solely be responsible for giving an (unconditional) witness hiding proof for NP, which is impossible.

Non-malleable Two-Prover Zero-Knowledge Proofs. We introduce the notion of non-malleable two-prover ZK proofs, and give an efficient, information theoretic secure construction. Our construction makes novel use of split-state non-malleable codes in such a way that the two states in the split-state functionality are defined by the main thread and the rewinding execution of the adversary.

As a separate technical contribution, we also show that any sigma protocol (with some natural additional properties) can be converted to a two-prover proof in a similar vein as the Lapidot-Shamir [25] construction.

1.2 Our Techniques

We start by observing that in the stateful hardware token model, a signer who wishes to output a witness signature on some message m, can send a token which executes a non-malleable zero knowledge proof of knowledge with a receiver. This is what we leverage to obtain a witness signature scheme.

Unconditional ZK from a Two-Prover Proof. As mentioned before, the signer can program two stateful tokens to participate in a two-prover proof system. Two-prover ZK proofs were first studied by Lapidot-Shamir [25] and are known to be unconditionally sound and perfectly zero-knowledge. The verifier would execute the protocol independently with each of the two tokens. Assuming that the tokens issued by a possibly malicious prover cannot communicate with each other when queried in isolation, this gives the soundness property required by witness signatures. This perspective, of using tokens as independent provers, is not completely new and has been explored before in [19]. However, this would only give us (unconditional) soundness and (perfect) zero knowledge, whereas to obtain unforgeability – we require non-malleability of the underlying proof.

Non-malleable Two-prover Proofs via Split-State Non-malleable Codes. Here, we demonstrate how we obtain a non-malleable two-prover proof. At the heart of our techniques lies a method of using split-state non-malleable codes, in the multi-prover model to achieve non-malleability in protocols. A split-state non-malleable code encodes some message M into two parts L, R, such that if the adversary separately tampers with L or with R, he cannot construct a codeword that is related to M.

To give a better intuition of our use of these codes, we start by discussing an unconditional commitment scheme in the two-prover model. The sender, in order to commit to a value X, secret shares X using an XOR encoding into shares A, B. He inputs the values A, B into both the provers, and sends these provers as his commitment. On receiving the commitment, the verifier queries the second prover only, and depending upon the challenge string, the prover outputs either A or B.

To decommit, the first prover outputs both values A, B to the verifier. It is easy to see that the construction described above behaves like a $1/2$-binding commitment to X. Such commitments are an integral part of two-prover proofs. It is feasible to port Σ protocols (which follow a commit, challenge, response structure) to a single round two-prover proof by using the second token to verify the commitment, and the first token to directly open a response to the challenge.

However, if instead of creating the shares of X using an ordinary XOR secret sharing scheme, we used a split-state non-malleable encoding scheme (which incidentally, also satisfies secret-sharing properties), then we could argue that an adversarial man-in-the-middle verifier which only obtains one of these shares from the second prover, can only tamper one of them at a time, and thus the resulting commitment would be non-malleable. It appears that porting Σ protocols to a two-prover proof while using this new non-malleable commitment scheme, should directly yield non-malleable two-prover ZK proofs.

Unfortunately, this idea does not directly work. This is because for most Σ protocols (including the Blum Hamiltonicity protocol used in Lapidot-Shamir [25], the prover may be required to open only a part of his committed message in the first round, depending upon the challenge query. Opening a part of the entire committed message is possible if a XOR encoding (ordinary secret sharing) were used, but not if a non-malleable encoding was used to commit to the entire first-round message at once.

This can easily be overcome if we non-malleably encode each index separately. However, proving security of this construction requires a reduction to many-many non-malleable codes. Fortunately, such codes were recently constructed by Chattopadhyay, Goyal and Li [7]. We make a minor modification to their construction to obtain many-many non-malleable codes with symmetric decryption, and the resulting symmetric many-many non-malleable codes suffice for our purposes. Then, in the construction outlined above, we observe that any man-in-the-middle adversary can only tamper with one of the split-state shares at a time (since he can only query a prover for *one* of the shares), while the other one is information theoretically hidden from him.

We show the existence of a simulator-extractor which builds on the standalone 2-prover ZK simulator. We perform a meticulous case analysis, and observe that in the most interesting case, the second man-in-the-middle (MIM) cheating prover behaves like a split-state tampering function on the shares output by the honest prover, in the real and rewinding executions. Then, we can argue non-malleability by a clever reduction to the security of the underlying many-many non-malleable codes.

Although this is the most technically challenging case to study, there are other ways in which the MIM provers could orient themselves. In particular, both MIM provers could be talking to various possible disjoint subsets of the two left provers. For example, a single man-in-the-middle prover could be acting as a verifier to both left provers, while the second man-in-the-middle interacts with none of them. Proving security in all these cases requires different ideas – for a detailed analysis, refer to Section 4.

Impossibility of Unconditional Unbounded-Verifiable Witness Signatures with Hardware Tokens. Our starting point is the result of Goyal et. al. [19], who used the notion of accessible entropy from [21, 22] and constructed an algorithm that learns most of the entropy of any stateless token. Their treatment crucially relied 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. Then it is unclear if one can have entropy learners in this setting.

Interestingly, we show that it is possible to extend the result of [19] (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.

Unbounded Verification-Secure Witness Signatures from One-way Functions. Given the previous two results, we now use PRFs to generate fresh entropy within the tokens, for each execution of the two-prover proof. The two tokens share a PRF key and generate entropy in sync for the two-prover proof. This suffices to allow an unbounded polynomial number of verifications.

1.3 Organization

The rest of this paper is organized as follows: In Section 2, we recall some definitions and primitives used in the rest of the paper. In Section 3, we give the formal definition of witness signatures and the plain-model impossibility for a black-box reduction. In Section 4, we give the construction and proof of security of our two-prover non-malleable proof. In Section 5, we show how to use these results to obtain witness signatures. Then, in Section 6, we show that it is impossible unconditional unbounded-verifiable witness signatures with a black-box proof of security, using stateful hardware tokens.

1.4 Related Work

Signatures of Knowledge. As already observed above, witness signatures are related to some existing primitives such as signatures of knowledge and non-malleable NIZKAoKs. Signatures of knowledge were introduced in 2006 by Chase and Lysyanskaya [6]. This primitive allows a signer to sign on behalf of an NP statement, while additionally ensuring that the signature is zero knowledge. This primitive is also closely related to non-malleable non-interactive zero-knowledge arguments of knowledge (NIZKAoK) [29, 30]. To construct witness signatures from (tag based) non-malleable NIZKAoKs, the signer can give a zero-knowledge argument of knowledge of the witness to the NP statement, non-malleably tagged by the message being signed. However, all of these primitives work in the presence of a common reference string, whereas witness signatures aim to remove reliance on a central trusted setup.

Non-Malleable Codes. There has been a vast amount of work on non-malleable codes. We give a summary of known constructions. Since the introduction of non-malleable codes by Dziembowski, Pietrzak and Wichs [13], the most well studied model is the split-state model introduced above. By a recent line of work [12, 2, 9, 8, 1], we now have almost optimal constructions of non-malleable codes in the C split-state model, for any $C \geq 2$. In the global-tampering (not split-state) model, Agrawal et al. [3] constructed efficient non-malleable codes with rate $1 - o(1)$ against a class of tampering functions slightly more general than the family of permutations.

A different but slightly related model is that of continuous non-malleable codes, which were introduced and constructed by Faust et al. [14]. Liu and Lysyanskaya [27] constructed efficient constant rate non-malleable codes in the split-state model against computationally bounded adversaries under strong cryptographic assumptions. Faust et al. [15] constructed almost optimal non-malleable codes against the class of polynomial sized circuits in the CRS framework. [5, 4, 10, 14] considered non-malleable codes in other models.

2 Preliminaries

In this section, we recall some definitions and introduce some notation for use in the rest of the paper. Let κ denote the statistical security parameter. First, we recall the stateful tamper-proof hardware token model.

2.1 Stateful Token Model

In the information theoretic stateful (tamper-proof hardware) token model, 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.

2.2 Non-malleable Codes

Here, we recall the definition of non-malleable codes.

Definition 1. *(Coding scheme) A coding scheme consists of two functions: a randomized encoding function* $\mathsf{Enc}:\mathcal{M} \to \mathcal{C}$ *, and a deterministic decoding function* $\mathsf{Dec}:\mathcal{C} \to \mathcal{M} \cup \{\perp\}$ *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 work, we require non-malleable codes resilient to split-state tampering functions [2]. Informally, the codeword for any message consists 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 a code word (L, R) and outputs a tampered codeword $(\tilde{L} = f_1(L), \tilde{R} = f_2(R))$.

Furthermore, any construction of non-malleable codes in the split-state model satisfies an additional *secret sharing* property. Any codeword (L, R) is such that L and R can be viewed as two 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. We formally define this notion next.*

Definition 2 (Many-Many Non-Malleable Codes)**.** *[7] 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 random variable* $D_{\mathbf{f}}$ *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:* $|(\mathsf{Dec}(f_1(X)), \ldots, \mathsf{Dec}(f_t(X))) - \mathsf{replace}(D_{\mathbf{f}}, s)| \leq \epsilon$*, where* $X = \mathsf{Enc}(s)$*. We refer to* t *as the tampering degree of the non-malleable code.*

The function $\mathsf{replace}: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *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.*

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 from the distribution D_f and outputs the result.

Imported Theorem 1. *[7] 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* $2^{-n^{\Omega(1)}}$*.*

Specifically, we need 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. [7]. 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 [7], 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.

2.3 Sigma (Σ) Protocols

We recall the definition of Σ protocols for any language $\mathcal{L} \in \mathsf{NP}$ with corresponding relation R. We borrow the definition partially from [26].

Definition 3. *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 parallel repetitions of the following:*

1. *$P(w,x)$ does the following.*
   - *Sample $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_N) \in \mathcal{S}$ 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 as $\pi_i(y_i)$ for any $i \in [\kappa]$.*
   - *For all $i \in [N]$, send $Z_i = \mathsf{Com}(y_i, r_i)$ to V.*
2. *V does the following.*
   - *Send challenge string $ch \xleftarrow{\$} \{0,1\}$ to P.*
3. *P does the following.*
   - *If $ch = 0$, $\mathsf{Decom}(Z_i)$ for all $i \in [N]$.*
   - *If $ch = 1$, $\mathsf{Decom}(Z_i)$ 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:

1. (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.*
2. (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 $(x, w) \in R$.*
3. (Honest-verifier zero knowledge). *There exists a PPT simulator $\mathcal{S}_\Sigma$ such that*

$$\{\mathcal{S}_\Sigma(x, e)\}_{x \in L; e \in \{0,1\}^n} \equiv \{\langle P(x,w), V(x,e) \rangle\}$$

   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.

4. *Additionally, we require that the third message output by the simulator of the proof is identically distributed as in the real proof.*

Remark 2. *Following Lapidot-Shamir [25], we can transform any Σ protocol in this format to a two-prover perfect zero knowledge proof. Although Lapidot-Shamir demonstrated this only for the special case of Graph Hamiltonicity, their argument directly generalizes to any Σ-protocol for a language L (as we demonstrate in the next section). Moreover, there exists a fixed constant c such that n parallel repetitions of this proof have soundness $2^{-c \cdot n}$ [25].*

3 Definitions

3.1 Witness Signatures

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.

Definition 4 (Witness Signatures)**.** *A witness signature scheme for some NP language L (with a corresponding witness relation R) consists of the two PPT algorithms:*

- $\mathsf{Sign}(x, w, m; r)$*:* Sign *(is a non-interactive algorithm that) takes as input an unbounded-length 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 *takes as input an unbounded-length 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 message $m \in \{0,1\}^*$, for all (possibly adversarially chosen) $x \in L$ and w such that $R(x, w)$ holds,*
  $\Pr[\mathsf{Verify}(x, m, \mathsf{Sign}(x, w, m)) = 1] = 1$
- *Witness Indistinguishability. For any instance x, given two witnesses w_1, w_2 for $x \in L$ and auxiliary information z, the distributions $\{z, \sigma_1 : \sigma_1 \leftarrow \mathsf{Sign}(x, w_1, m)\}$ and $\{z, \sigma_2 : \sigma_2 \leftarrow \mathsf{Sign}(x, w_2, m)\}$ are identical.*
- *Unforgeability. (EUF-CMA) Consider any non-uniform PPT forger $\mathcal{F}$ which adaptively (dynamically) and interactively obtains signatures $(\sigma_1, \sigma_2, \dots \sigma_n)$ for some instance x (where $|x| = \kappa$) on his choice messages $(m_1, m_2, \dots m_n)$, where $n = \mathsf{poly}(\kappa)$. Next, $\mathcal{F}$ adaptively chooses message $m^* \notin \{m_1, m_2, \dots m_n\}$ and outputs $(m^*, \sigma_{m^*,x})$.*

  Then if for any $x \in L$, $(m^*, \sigma_{m^*,x})$ successfully verifies with probability q over the space of randomness of the verifier, there exists a PPT reduction $\mathcal{E}$ and a constant c, such that $\mathcal{E}$ takes input x and interacts with any such forger $\mathcal{F}$ in a black-box manner (we denote this interaction by $\mathcal{E}^{\mathcal{F}}(x)$) to output a string out. *The string* out *is such that if $\Pr[\mathsf{Verify}(x, m^*, \sigma_{m^*,x}) = 1] = q$ (over the randomness of the forger and the verifier), then $\Pr[R(x, \mathsf{out}) = 1] \geq q^c$ for a constant c (over the randomness of the reduction.)*

Remark 3. *The unforgeability condition implies that witness signatures for any language $\mathcal{L}$ must be perfectly 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 have to be extracted with non-negligible probability, which is impossible).*

Theorem 1. *Witness signatures for all of* NP *according to Definition 4, are impossible unless* $\mathsf{BPP} = \mathsf{NP}$.

Proof. 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 in violating soundness. This means $\mathcal{E}$ will have to either decide whether or not $x \in L$ (solving decidability of NP-hard problems), or violate the perfect soundness property.

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, and if it 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, moreover it is impossible for any such strategy $\mathcal{A}$ to generate a signature on $x \notin L$.

Now consider some hypothetical efficient forger strategy $\mathcal{F}$ which can output a forgery on any x (whether or not it is in L) iff the input signature verifies. The view of the black-box reduction $\mathcal{E}$ while interacting with this strategy $\mathcal{A}$ is indistinguishable from the view of $\mathcal{E}$ interacting with $\mathcal{F}$. This is because there is no output from either of $\mathcal{A}$ or $\mathcal{F}$ until the reduction outputs a (single message) signature that verifies with some probability.

Thus, $\mathcal{A}$ runs the extraction algorithm $\mathcal{E}(x)$, requests a signature on 0, and waits for the next message (the witness signature) from $\mathcal{E}$. Since the signature is non-interactive, (after possibly polynomially many rewindings), $\mathcal{E}$ eventually issues a signature on x. By the perfect soundness property, it cannot issue a valid signature unless $x \in L$. Therefore, $\mathcal{A}$ can verify the signature issued by $\mathcal{E}$ such that $x \in L$ if it verifies and $x \notin L$ otherwise. □

Remark 4. *Due to the impossibility in Theorem 1, it would make sense to consider weaker variants of Definition 4. 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.*

3.2 Witness Signatures in the Stateful Hardware Token Model

Definition 5 (Witness Signatures in the Stateful Hardware Token Model)**.** *A witness signature scheme in the stateful token model, for some* NP *language L (with a corresponding witness relation R) consists of the two PPT algorithms:*

- $\mathsf{Sign}(x, w, m; r)$*:* Sign *takes as input an unbounded-length string x, a witness w, a message $m \in \{0,1\}^*$, randomness r and outputs a set of stateful hardware tokens $[T_1, T_2, \ldots T_{\mathsf{poly}(\kappa)}]_{m,x}$ as a signature $\sigma_{m,x}$.*
- $\mathsf{Verify}(x, m, \sigma)$*:* Verify *takes as input an unbounded-length string x, a message $m \in \{0,1\}^*$ and a signature set of tokens $[T_1, T_2, \ldots T_{\mathsf{poly}(\kappa)}]$, and outputs 0 or 1.*

These algorithms satisfy the following properties:

- *Correctness. For any message $m \in \{0,1\}^*$, for any $x \in L$ and w such that $R(x, w)$ holds,* $\Pr[\mathsf{Verify}(x, m, \mathsf{Sign}(x, w, m)) = 1] = 1$
- *Witness Indistinguishability. Given any two witnesses w_1, w_2 for some instance $x \in L$ and auxiliary information z, $\{z, \sigma_1 : \sigma_1 \leftarrow \mathsf{Sign}(x, w_1, m; r)\} \approx_c \{z, \sigma_2 : \sigma_2 \leftarrow \mathsf{Sign}(x, w_2, m; r)\}$.*
- *Unforgeability. Consider any non-uniform PPT forger $\mathcal{F}$ which (adaptively) obtains signatures $(\sigma_1, \sigma_2, \ldots \sigma_n)$ for some instance x (where $|x| = \kappa$) on his choice messages $(m_1, m_2, \ldots m_n)$, where $n = \mathsf{poly}(\kappa)$. Next, $\mathcal{F}$ outputs a message signature pair for a message that was never queried, $(m^*, \sigma_{m^*,x})$ that successfully verifies with probability p over the space of randomness of the verifier.*

  In the hardware token model, each token issued by such a forger $\mathcal{F}$ could be secretly encapsulating upto one token generated by some honest signer, such that the forger's token queries the signer's token after *it has received an external query from the verifier. For example, the forger might construct tokens that encapsulate honest signer tokens and relay messages between the verifier and the encapsulated tokens.*

  Then, there should a PPT reduction $\mathcal{E}$ and a constant c, such that $\mathcal{E}$ takes input x and interacts with any such forger $\mathcal{F}$ in a black-box manner (we denote this interaction by $\mathcal{E}^{\mathcal{F}}(x)$) to output a string out*. The string* out *is such that if $\Pr[\mathsf{Verify}(x, m^*, \sigma_{m^*,x}) = 1] = p > 1/\mathsf{poly}(\mid x \mid)$ for some polynomial $\mathsf{poly}(\cdot)$, then $\Pr[R(x, \mathsf{out}) = 1] \geq p^c$ for a constant c, over the randomness of the reduction.*

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. That is, in the physical world, we restrict the adversary from putting 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 Non-malleable Two-Prover ZK Proofs

4.1 Two-Prover ZK from any Σ protocol

We first recall the model and definition of a two-prover ZK proof, then demonstrate the conversion of any Σ-protocol into a perfect zero knowledge proof system in the two-prover model following the Lapidot-Shamir technique [25].

4.1.1 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 6. *A protocol τ is a two-prover 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**[2]**.** *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 $\Pr[w \text{ is a valid witness for } x] = \mathsf{poly}(q)$ for some polynomial $\mathsf{poly}(\cdot)$.*

4.1.2 Construction

The Lapidot-Shamir two-prover ZK protocol [25] consists of the following algorithms[3]:

$\mathsf{Prove}(x, w)$: 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 3). 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.

[2]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]Lapidot-Shamir [25] 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.

- The first and second prover algorithms are described in Figure 4 and Figure 5.

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 1: 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 2: 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).
  - 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.

It is easy to verify completeness. These algorithms satisfy the following other properties:

- Perfect Zero Knowledge: We describe the simulation strategy in Figure 3. It is straightforward to see that the simulated view is identical to the real view.
- Statistical Soundness: This protocol is statistically sound and has a soundness error less than $2^{-n/9}$. The proof follows analogously to Theorem 6 in Lapidot-Shamir [25] and we omit details here.
- Proof of Knowledge: Again, following Lapidot-Shamir [25] there is a simple PPT extractor 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 $w = \pi_i(y_i)$.

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 $B_i = \perp$. Output A_i.
   - If $\tau_i = 1$, set $A_i = \perp$ 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 $A_i = \perp$, 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 $A_i = \perp$, 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 3: Simulation strategy for a pair of provers

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

In this section, we construct tag-based non-malleable two-prover ZK[4]. We first describe the model and define a non-malleable two-prover ZK proof, then we demonstrate how to convert any Σ-protocol into a perfect non-malleable zero-knowledge proof system in the two-prover model.

4.2.1 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 only 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. That is, $P_{\mathsf{MIM},1}$ can be interacting with P_1 while $P_{\mathsf{MIM},2}$ interacts with P_2, or, $P_{\mathsf{MIM},1}$ can be interacting with P_2 while $P_{\mathsf{MIM},1}$ interacts with P_2, or, $P_{\mathsf{MIM},1}$ can be interacting with both P_1 and P_2 while $P_{\mathsf{MIM},2}$ interacts with neither. However, they cannot both be interacting with the same prover *once their session on the right has started.*

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

[4]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.

- **Completeness.** *For any honest verifier V, if $x \in L$, $\Pr[V(\Pi) = 1] = 1$.*
- **Simulation-Extraction**[5]**.** *There exists a black-box (possibly rewinding) simulator-extractor* $\mathsf{Sim\text{-}Ext}$ *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 $\mathsf{poly}(\cdot)$.*

4.2.2 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 $\theta(\kappa^2)$:

- $\mathsf{Prove}(x, w, \mathsf{tag})$: 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 3). 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 4 and Figure 5.

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 4: 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 5: 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.

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

– 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.

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

Here, we prove the following theorem about our construction.

Theorem 2. *The construction above is a non-malleable two-prover zero-knowledge proof (of knowledge) according to Definition 7.*

4.3.1 Simulator-Extractor

We first describe a simulation strategy in Figure 6. 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.

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 $y_i \xleftarrow{\$} \mathcal{S}_x$.
     If $A_i = \perp$, compute A_i such that each component $(a_{i,j}, b_{i,j}) = \mathsf{NM} - \mathsf{SS}(\mathsf{tag}||y_{i,j})$.
     Otherwise, compute B_i such that each component $(a_{i,j}, b_{i,j}) = \mathsf{NM} - \mathsf{SS}(\mathsf{tag}||y_{i,j})$.
   - 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 $(a_{i,j}, b_{i,j}) = \mathsf{NM} - \mathsf{SS}(\mathsf{tag}||y_{i,j})$. For $j \notin J_i$, compute each component $a_{i,j}$ such that $(a_{i,j}, b_{i,j}) = \mathsf{NM} - \mathsf{Simulate}(1^\kappa)$.
     – Otherwise, for $j \in J_i$, compute each component $b_{i,j}$ such that $(a_{i,j}, b_{i,j}) = \mathsf{NM} - \mathsf{SS}(\mathsf{tag}||y_{i,j})$. For $j \notin J_i$, compute each component $b_{i,j}$ such that $(a_{i,j}, b_{i,j}) = \mathsf{NM} - \mathsf{Simulate}(1^\kappa)$.
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 6: Simulation strategy for a pair of provers

Next, we describe the simulator-extractor $\mathcal{R}$ in Figure 7. 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 $\mathsf{poly}(\cdot)$.

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 6.

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 6.

   (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 6.

   (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 6.
   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 6 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 6.

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

4.3.2 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 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.

4.3.3 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 two-prover proof system to extract a witness from the MIM. We detail this hybrid in Figure 8.

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 6. 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[6].

[6]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 6. This achieves the same effect.

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 8: 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 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 6.

   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 6. 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 6. 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 6.

   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 6. 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 6), 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.

4.3.4 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, $\Pr_{X\sim\mathcal{X}}[X\in \mathsf{good}]\geq p/2$.*

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

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

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 [25] 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 6 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,*
$\Pr[$*extraction occurs in* $\mathsf{Hybrid}_2] \geq \mathsf{poly}(\Pr[$*extraction occurs in* $\mathsf{Hybrid}_1])$ *for a fixed polynomial* $\mathsf{poly}(\cdot)$.

Proof. In this case, the simulation strategy of Figure 6 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 $(\sigma^1, \tau^1), (\sigma^2, \tau^1), (\sigma^1, \tau^2), (\sigma^2, \tau^2)$ 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 $q^{3c}/8$. $\square$

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 $q^{3c}/8$. $\square$

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. $\square$

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 $\tilde{\sigma}^1_i = 0$, the joint view of the MIM can be sampled perfectly, without a witness. For indices $i \in [\kappa]$ where $\tilde{\sigma}^1_i = 1$, 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 $\tilde{\sigma}^1_i = 1$.

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 $\kappa \cdot 2^{-\mathsf{poly}(\kappa)}$. □

5 Witness Signatures from Stateful Hardware Tokens

5.1 Single-Use Witness Signatures from Non-Malleable MIPs

In this section, we give a construction (according to Definition 5) 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.

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 3. *In the stateful hardware token model, it is possible to realize unconditionally secure witness signatures according to Definition 5, 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 7.

   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.

   An exception is the case when the forger constructs two tokens S', T', such that S' encapsulates S_i and T' encapsulates T_j for $i, j \in \{1, 2\} \times \{1, 2\}$ and $i \neq j$; where (S_1, T_1) and (S_2, T_2) are tokens for two signatures on different tags t_1, t_2. In this case, since the simulation strategy is identical for both Case 2 and Case 3, $\mathcal{R}_{\mathsf{sign}}$ extracts the witness w by running the simulator for both (S_1, T_1) and (S_2, T_2) 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.

   Finally, the witness can be extracted for all other cases of token encapsulation via the simulator-extractor strategy for the corresponding cases in the non-malleable two-prover proof system, according to Figure 7.

It is straightforward to see that this construction satisfies the correctness and unforgeability properties of Definition 5, based on the correctness and security properties of the underlying non-malleable two-prover proof. Furthermore, the standalone zero knowledge property of the underlying Lapidot-Shamir two-prover proof, implies perfect witness indistinguishability. □

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.

5.2 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 9 and Figure 10. Both tokens are initialized with $\mathsf{count} = 0$.

$\mathsf{Verify}(x, m, \sigma)$: 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 4. *In the stateful hardware token model, assuming one-way functions exist, it is possible to realize witness signatures according to Definition 5, 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, 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 9: Token S

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 10: 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 7 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. □

6 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. This is done via extending the notion of inaccessible entropy, first introduced in [22] and later used in [19] 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.

Stateful Token Model. In the information theoretic stateful (tamper-proof hardware) token model, two (computationally unbounded) interactive algorithms A and B interact with the following extra feature to the standard model. At any time during the protocol parties can construct a circuit T, put it inside a "token", and send the token to the other party. The party receiving the token gets oracle access to T and can make polynomially many but unbounded queries to the token. Additionally, the token has the ability to maintain "state" between queries/inputs to the circuit T.

Moreover, witness signatures 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 5. *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 [19], 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 [17, 22]. [17] 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 5. 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 5 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 [19]. 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, $E[|Q|] \leq {}^{(\sum_{i\in[n]} H(T_i))}/_{\epsilon}$ □

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 [19]. 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. □

References

[1] Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: 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. pp. 459–468 (2015), http://doi.acm.org/10.1145/2746539.2746544 7

[2] Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. In: Shmoys, D.B. (ed.) Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014. pp. 774–783. ACM (2014), http://doi.acm.org/10.1145/2591796.2591804 7, 8

[3] Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: 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. pp. 538–557 (2015), http://dx.doi.org/10.1007/978-3-662-47989-6_26 7

[4] Chabanne, H., Cohen, G.D., Flori, J., Patey, A.: Non-malleable codes from the wire-tap channel. In: 2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011. pp. 55–59. IEEE (2011), http://dx.doi.org/10.1109/ITW.2011.6089565 7

[5] Chabanne, H., Cohen, G.D., Patey, A.: 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. pp. 2546–2550. IEEE (2012), http://dx.doi.org/10.1109/ISIT.2012.6283976 7

[6] Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4117, pp. 78–96. Springer (2006), http://dx.doi.org/10.1007/11818175_5 3, 4, 7

[7] Chattopadhyay, E., Goyal, V., Li, X.: Non-Malleable Extractors and Codes, with their Many Tampered Extensions. ArXiv e-prints (May 2015) 4, 6, 8, 9, 32, 33, 34

[8] Chattopadhyay, E., Zuckerman, D.: 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. pp. 306–315 (2014), http://dx.doi.org/10.1109/FOCS.2014.40 7

[9] Cheraghchi, M., Guruswami, V.: 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. pp. 440–464 (2014), http://dx.doi.org/10.1007/978-3-642-54242-8_19 7

[10] Choi, S.G., Kiayias, A., Malkin, T.: Bitr: Built-in tamper resilience. IACR Cryptology ePrint Archive 2010, 503 (2010), http://eprint.iacr.org/2010/503 7

[11] Cohen, G.: Local correlation breakers and applications to three-source extractors and mergers. Electronic Colloquium on Computational Complexity (ECCC) 22, 38 (2015), http://eccc.hpi-web.de/report/2015/038 34

[12] Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. pp. 239–257 (2013), http://dx.doi.org/10.1007/978-3-642-40084-1_14 7

[13] Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: Yao, A.C. (ed.) Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings. pp. 434–452. Tsinghua University Press (2010), http://conference.itcs.tsinghua.edu.cn/ICS2010/content/papers/34.html 4, 7

[14] Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. In: Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings. pp. 465–488 (2014), http://dx.doi.org/10.1007/978-3-642-54242-8_20 7

[15] Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: 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. pp. 111–128 (2014), http://dx.doi.org/10.1007/978-3-642-55220-5_7 7

[16] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013. pp. 467–476. ACM (2013), http://doi.acm.org/10.1145/2488608.2488667 3

[17] Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. In: Paterson, M. (ed.) Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, July 16-20, 1990, Proceedings. Lecture Notes in Computer Science, vol. 443, pp. 268–282. Springer (1990), http://dx.doi.org/10.1007/BFb0032038 26

[18] Goyal, V., Ishai, Y., Mahmoody, M., Sahai, A.: Interactive locking, zero-knowledge pcps, and unconditional cryptography. In: Rabin [28], pp. 173–190, http://dx.doi.org/10.1007/978-3-642-14623-7_10 5

[19] Goyal, V., Ishai, Y., Mahmoody, M., Sahai, A.: Interactive locking, zero-knowledge pcps, and unconditional cryptography. In: Rabin [28], pp. 173–190, http://dx.doi.org/10.1007/978-3-642-14623-7 5, 6, 25, 26, 27, 28

[20] Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings. Lecture Notes in Computer Science, vol. 5978, pp. 308–326. Springer (2010), http://dx.doi.org/10.1007/978-3-642-11799-2_19 5

[21] Haitner, I., Hoch, J.J., Reingold, O., Segev, G.: Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In: FOCS. pp. 669–679. IEEE Computer Society (2007) 5, 6

[22] Haitner, I., Reingold, O., Vadhan, S.P., Wee, H.: Inaccessible entropy. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009. pp. 611–620. ACM (2009), http://doi.acm.org/10.1145/1536414.1536497 5, 6, 25, 26

[23] Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5157, pp. 572–591. Springer (2008), http://dx.doi.org/10.1007/978-3-540-85174-5_32 5

[24] Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4515, pp. 115–128. Springer (2007), http://dx.doi.org/10.1007/978-3-540-72540-4_7 3

[25] Lapidot, D., Shamir, A.: A one-round, two-prover, zero-knowledge protocol for NP. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 213–224. Springer (1991), http://dx.doi.org/10.1007/3-540-46766-1_16 5, 6, 10, 12, 13, 21

[26] Lindell, Y.: An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. IACR Cryptology ePrint Archive 2014, 710 (2014), http://eprint.iacr.org/2014/710 9

[27] Liu, F., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7417, pp. 517–532. Springer (2012), http://dx.doi.org/10.1007/978-3-642-32009-5_30 7

[28] Rabin, T. (ed.): Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings, Lecture Notes in Computer Science, vol. 6223. Springer (2010), http://dx.doi.org/10.1007/978-3-642-14623-7 31

[29] Sahai, A.: 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. pp. 543–553 (1999), http://dx.doi.org/10.1109/SFFCS.1999.814628 7

[30] Santis, A.D., Crescenzo, G.D., Ostrovsky, R., Persiano, G., Sahai, A.: 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. pp. 566–598 (2001), http://dx.doi.org/10.1007/3-540-44647-8_33 7

[31] Zhandry, M.: How to avoid obfuscation using witness prfs. In: Proceedings of TCC 2016 (2016) 3

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. It should satisfy the property that given a left share L (or right share R) it is 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 $L, R \in \{0,1\}^n$ it should happen that $\mathsf{Dec}(L, R) = \mathsf{Dec}(R, L)$. 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, what needs to be observed is that the tampering function tampers with a mixture of left and right shares. For example, let us say $(f_1, g_1), .., (f_N, g_N)$ be the tampering functions tampering codes (L_i, R_i) for $i \in [n]$. Then, we allow 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. For example, consider $n = 2$ and $N = 1$. Suppose f_1 tampers with R_1, L_2 and g_1 tampers with 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 [7] already satisfies the first two properties. We now show how to modify the construction in [7] so that it satisfies all the three properties. Let the scheme in [7] 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 $(L||0, R||1)$.
- $\mathsf{Dec}'\{0,1\}^{n+1} \times \{0,1\}^{n+1} \to \{0,1\}^m$. 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. One can show that if the underlying code supports a tampering degree t 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 [7].

Definition 8. *A function $nmExt : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ 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 [7].

Theorem 6. *Let $nmExt : (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^m$ 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 $nmExt^{-1}(s)$. [7] 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 $t = n^{\gamma/C}$. [7] construct an explicit function $nmExt : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, $m = n^{\Omega(1)}$ 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 $nmExt' : \{0,1\}^{n+1} \times \{0,1\}^{n+1} \to \{0,1\}^m$ (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 one can show that proving 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 $t' = n^{\gamma/C}$,

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

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

where $\epsilon = 2^{-n^{\Omega(1)}}$.
Here, we define the class of valid adversary as $\mathcal{A}'_i = (f'_i, g'_i)$ 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 tampers 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 [7] already satisfies this property. Hence, $nmExt'$ gives us our desired non-malleable code with symmetric decoding operation. We describe a high level idea. The $(2, t')$ non-malleable extractor in [7] works by computing 'flip-flop' alternating extraction (introduced in [11]) 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.