Non-Malleable Codes for Bounded Polynomial-Depth Tampering

Dana Dachman-Soled* Ilan Komargodski† Rafael Pass‡

Abstract

Non-malleable codes allow one to encode data in such a way that, after tampering, the modified codeword is guaranteed to decode to either the original message, or a completely unrelated one. Since the introduction of the notion by Dziembowski, Pietrzak, and Wichs (ICS '10 and J. ACM '18), a large body of work has focused on realizing such coding schemes secure against various classes of tampering functions. It is well known that there is no efficient non-malleable code secure against all polynomial-size tampering functions. Nevertheless, non-malleable codes in the plain model (i.e., no trusted setup) secure against *bounded* polynomial-size tampering are not known and obtaining such a code has been a major open problem.

We present the first construction of a non-malleable code secure against *all* polynomial-size tampering functions that have *bounded polynomial-depth*. This is an even larger class than all bounded polynomial-*size* functions and, in particular, we capture all functions in non-uniform **NC** (and much more). Our construction is in the plain model (i.e., no trusted setup) and relies on several cryptographic assumptions such as keyless hash functions, time-lock puzzles, as well as other standard assumptions. Additionally, our construction has several appealing properties: the complexity of encoding is independent of the class of tampering functions and we obtain sub-exponentially small error.

*University of Maryland. Email: danadach@ece.umd.edu. Supported in part by NSF grants #CNS-1933033, #CNS-1453045 (CAREER), and by financial assistance award 70NANB15H328 from the U.S. Department of Commerce, National Institute of Standards and Technology.

†Hebrew University of Jerusalem and NTT Research. Email: ilank@cs.huji.ac.il.

‡Cornell Tech. Email: rafael@cs.cornell.edu. Supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, AFOSR Award FA9550-18-1-0267, and by NSF Award DGE-1650441. This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2019-19-020700006. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

1 Introduction

A non-malleable code is a fascinating concept that (informally) allows one to encode messages such that it is impossible to modify the underlying message of a given codeword without decoding it first. More precisely, the operation applied to the codeword is called the *tampering function*, and the guarantee is that, with "high probability", decoding a tampered codeword results in either the original message or an unrelated one. We refer to the probability that the attacker succeeds in coming up with a tampered codeword of a related messages as its *distinguishing advantage*, and we typically require this advantage to be negligible (i.e., smaller than the inverse of any polynomial). Note that in contrast to standard error-correcting (or detecting) codes, non-malleable codes can achieve security against tampering functions that modify *every* part of a codeword.

Non-malleable codes have proven to be a fundamental concept, giving rise to many beautiful connections and results, both in complexity theory (e.g., two-source extractors [Li12, Li13, CGL15, CZ16] and additive combinatorics [ADL18, ADKO15]) as well as in cryptography (e.g., non-malleable encryption and commitments [CMTV15, CDTV16, GPR16]).

In the paper that introduced non-malleable codes, Dziembowski, Pietrzak, and Wichs [DPW10, DPW18], observed that it is impossible to construct a non-malleable code secure against arbitrary tampering functions, since a tampering function which first decodes the codeword and then re-encodes a related message breaks security. By the same principle, it is impossible to construct a code with polynomial-time decoding which is secure against all polynomial-time tampering functions.[1] Therefore, the class of tampering functions has to be limited in some way—either in terms of computational power or in the way the functions can access the codeword. One natural limitation is by restricting the available computational complexity resources (e.g., running time, space, etc).

Already in the original work of Dziembowski et al. [DPW18] (see also [CG16] for a followup), it was shown that (with high probability) a random function is a non-malleable code secure against all circuits of size (say) $2^{n/2}$, where n is the size of a codeword. However, the code is clearly inefficient. Faust et al. [FMVW16] gave an efficient version of that result, but it is still not an explicit construction: For any polynomial bound S, there is an efficiently samplable *family* of codes such that (with high probability) a random member of the family is a non-malleable code secure against all functions computable by a circuit of size S. Stated differently, the result can be seen as an explicit construction (i.e., a single code) assuming an untamperable *common reference string* (CRS) which is longer than the running time of the tampering function. In the random oracle model (which can be thought of as an exponential size common random string), Faust et al. [FHMV17] constructed non-malleable codes secure against space-bounded tampering. Ball et al. [BDKM16] constructed a non-malleable code secure against bounded depth circuits with constant fan-in (which includes $\mathbf{NC}^0$). Several works were able to get non-malleable codes secure against $\mathbf{AC}^0$ tampering functions [CL17, BDKM18, BDG$^+$18, BGW19] (actually even circuits of depth $O(\log n/\log\log n)$).

Arguably, the holy grail in this line of works is to construct an explicit non-malleable code which is secure against all tampering functions from the class of bounded polynomial-size circuits. Specifically, for a size bound S, we would like to get an efficient code which is non-malleable for all tampering functions that can be described by an arbitrary circuit of size S. Ideally, only decoding should require running-time greater than S and encoding should run in some a-priori

[1] Here is the attack: the tampering function can decode the codeword and if it contains some pre-defined message (say all 0s), then it replaces it with garbage (which might not even correspond to a valid codeword), and otherwise it does not change the input.

fixed polynomial-time, independent of S.

Does there exist an explicit construction (in the plain model) of an efficient non-malleable code which is secure against all bounded polynomial-size attackers?

Ball et al. [BDKM18] made an important step towards this goal by using computational assumptions. Concretely, using public-key encryption and non-interactive zero-knowledge (NIZK), they gave a generic way to construct non-malleable codes secure against tampering classes $\mathcal{F}$ using sufficiently strong average-case hardness for $\mathcal{F}$. This construction, however, still requires a CRS (for the public key of the encryption scheme and the CRS of the NIZK) albeit it is short (polynomial in the the security parameter and independent of the class $\mathcal{F}$).

In a recent follow-up work, Ball et al. [BDK$^+$19] managed to get rid of the CRS, but at the cost of (a) using non-standard assumptions, and (b) limiting the class of attacks and the level of security. In more detail, they showed a construction of an efficient non-malleable codes secure against all (uniform) tampering functions computable in an a-priori fixed polynomial-time. But:

- Their construction relies (amongst other assumptions) on sub-exponentially sound **P**-certificates[2] which is a very strong and non-standard assumption. In particular, the *only* known instantiation requires *assuming* soundness of a non-trivial argument system (Micali's CS proofs [Mic00]), which is true in the Random Oracle model.

- Their scheme is non-malleable only with respect to *uniform* polynomial-time tampering as opposed to the standard model of polynomial-*size* tampering. In other words, the tampering attacker is restricted to being a *uniform* polynomial-time algorithm, in contrast to the standard model of non-uniform polynomial-time attackers.

- Their scheme achieves only *a-priori bounded* inverse polynomial-distinguishing advantage, as opposed to achieving "full" security (i.e., negligble distinguishing advantage).

- Finally, both their encoding procedure, as well as the decoding procedure, run *longer* than the allowed tampering functions (i.e., the adversary can neither encode nor decode). In contrast, as mentionned, in principle encoding could be "efficient" in the sense that it is independent of the size/running-time of the tampering attacker.

To summarize, despite several beautiful steps towards resolving the above question, the answer is still largely unknown. Known partial solutions either require a CRS or strong and non-standard cryptographic assumptions that are only known to be instantiated in the Random Oracle model (and even then only achieve a weaker form of non-malleability).

1.1 Our Results

We give the first full affirmative answer to the aforementioned question. Specifically, we construct an efficient non-malleable code that is (computationally) secure against tampering functions computable by any bounded polynomial-size circuit. Our construction is in the plain model and relies on several generic and well-studied cryptographic building blocks: a time-lock puzzle [RSW96], a non-interactive non-malleable commitment [KS17, LPS17, BL18, KK19], and a non-interactive

[2] These are "succinct" one-message arguments for languages in **P**, with proof length which is a fixed polynomial, independent of the time it takes to decide the language [CLP13].

SPS (super-polynomial-time simulatable) zero-knowledge protocol [BP04, BL18] (all in the plain model). While we cannot use the aforementioned primitives in their most general form, we identify certain additional properties from them that will be needed in our construction; additionally, we note that particular known constructions of them satisfy the additional desired properties; see below and in Section 2 for more details.

Our construction actually captures an even larger class of tampering functions. Specifically, we give a non-malleable code secure against all tampering functions that can be computed by *arbitrary* (unbounded) polynomial-size circuit of *bounded polynomial-depth*. We emphasize that while the circuit depth of the tampering function is bounded a priori by some fixed polynomial in the security parameter, the size of the circuit is *unbounded* and can be any polynomial in the security parameter.

Theorem 1.1 (Informal Meta Theorem)**.** *Assume the existence of a "special-purpose" time-lock puzzle, one-message non-malleable commitment, and one-message SPS zero-knowledge protocol. For any* $\overline{T} \in \mathsf{poly}(\lambda)$*, there exists an explicit code where encoding takes time* $\mathsf{poly}(\lambda)$*, decoding takes time* $\mathsf{poly}(\overline{T}, \lambda)$*, and it is non-malleable against all tampering functions computable by a non-uniform arbitrary polynomial-size (in* λ*) circuit of depth* $\overline{T}$*.*

Our result is the first to handle all bounded polynomial-size tampering functions (and in fact much more). In particular, as a special case, we capture all tampering functions in non-uniform **NC** (while previously there was no construction even for $\mathbf{NC}^1$). We emphasize that our scheme is efficiently encodable, namely, encoding time depends only on the security parameter and not on the (depth) complexity of the decoder. Furthermore, our construction readily extends to withstand (sub-)exponential size tampering functions (of depth $\overline{T}$) without affecting the complexity of neither encoding nor decoding. Lastly, we note that the distinguishing advantage of any tampering function in our scheme can be made sub-exponentially small in λ at essentially no cost (since in any case we need to rely on sub-exponential hardness of the underlying building blocks).

In comparison, as mentioned, prior to this work, even dealing with just bounded polynomial-size tampering was not known. The only approach towards polynomial-size tampering [BDK$^+$19] captured only *uniform* polynomial-time tampering, but as mentioned above, even for this restricted class of tampering, their result has additional drawbacks: (1) it relies on a strong non-standard assumption (**P**-certificates) that we only know how to satisfy in the random oracle model, and (2) it only gives inverse-polynomial distinguishing advantage (as opposed to negligible distinguishing advantage).

We instantiate the time-lock puzzle using the construction of Rivest et al. [RSW96] and we show how to further use results of Bitansky and Lin [BL18] and Lin et al. [LPS17] to instantiate the required non-malleable commitment and zero-knowledge protocol. Thus, assuming the repeated squaring assumption [RSW96] (i.e., there is no way to significantly speed-up repeated squarings in a hidden-order group), a keyless multi-collision resistant hash function [BKP18] (i.e., a single function for which any PPT attacker with $\ell(\lambda)$ bits of non-uniform advice cannot find more than $\ell(\lambda)^c$ collisions for a constant $c \in \mathbb{N}$),[3] as well as other standard assumptions, we obtain the following theorem.

Theorem 1.2 (Informal)**.** *Assume a keyless multi-collision resistant hash function, the repeated squaring assumption, an injective one-way function, and non-interactive witness-indistinguishable*

[3] While keyless multi-collision resistance is a relatively new assumption, it is a natural and simple security property for keyless cryptographic hash functions, which in particular is satisfied by a random function.

proofs,[4] *all being sub-exponentially secure. Then, for any* $\overline{T} \in \mathsf{poly}(\lambda)$*, there exists an explicit code where encoding takes time* $\mathsf{poly}(\lambda)$*, decoding takes time* $\mathsf{poly}(\overline{T}, \lambda)$*, and it is non-malleable against all tampering functions computable by a non-uniform arbitrary polynomial-size (in* λ*) circuit of depth* $\overline{T}$*.*

We refer to Section 1.2 for more details about the above assumptions.

Non-malleable time-lock puzzle. Our non-malleable code construction is secure for all bounded polynomial-depth tampering functions and additionally it is efficiently encodable, meaning that encoding time is fixed as a function of the security parameter, but is otherwise independent of the time it takes to decode. We observe that the combination of these two properties actually implies a *time-lock puzzle* which is additionally *non-malleable*.[5] In other words, under the same assumptions as in Theorems 1.1 and 1.2, we get a *non-malleable* time-lock puzzle. We emphasize that the non-malleable time-lock puzzle that we obtain here is in the plain model, i.e., does not require any trusted setup.

In a followup work [EFKP20], the notion of non-malleable time-lock puzzles is formally defined and further studied, giving additional constructions and various applications. In particular, in [EFKP20] we give a more efficient construction than the one given in this work, which is proven secure in the (auxiliary-input) random oracle model. Additionally, in [EFKP20] we further construct time-lock puzzles which are concurrently non-malleability and show that they are useful in several desirable cryptographic protocols.

1.2 Related Work

Since the work of Dziembowski, Pietrzak, and Wichs [DPW10, DPW18] which introduced non-malleable codes, there has been a quite a significant amount of works on this subject in various different directions (for example, [AAG$^+$16, KLT16, ADL18, ADKO15, CGL15, DLSZ15, CGM$^+$16, CL17, Li17, Li18] to mention only a few in addition to the ones we mentioned earlier). Notably, various different classes of tampering functions were considered. The original work of [DPW10] presented a construction of non-malleable codes against bit-wise tampering functions. Also, Liu and Lysyanskaya [LL12] were the first to consider the class of split state tampering functions, where left and right halves of a codeword may be tampered arbitrarily, but independently. There has been a very long line of works on getting optimal constructions against such tampering functions (see the references above).

Next, we give more information about the building blocks used in our constructions and mention relevant related work.

Time-lock puzzles. These are puzzles that can be solved by "brute-force" in time $\overline{T}$, but cannot be solved significantly faster even using parallel processors. This concept was proposed by Rivest, Shamir, and Wagner [RSW96] (following May's work [May92] on timed-release cryptography), and they have been used quite extensively studied since. The most popular instantiation relies on the

[4] Non-interactive witness-indistinguishable proofs are known to exist based on various assumptions: trapdoor permutations and a particular derandomization-type assumption [BOV07], cryptographic bilinear maps [GOS06], or indistinguishability obfuscation and one-way permutations [BP15].

[5] Recall that time-lock puzzles are a cryptographic mechanism for sending messages "to the future", by allowing a sender to quickly generate a puzzle with an underlying message that remains hidden until a receiver spends a moderately large amount of time solving it. Non-malleability guarantees that not only the puzzle hides the underlying message, but actually it is hard to "maul" it into a puzzle with a different "related" message.

repeated squaring assumption that postulates that $\overline{T}$ repeated squarings mod N, where $N = pq$ is a product of two secret primes, require "roughly" $\overline{T}$ parallel time/depth. Bitansky et al. [BGJ$^+$16] gave a construction of a time-lock puzzle from (strong) assumptions related to program obfuscation.

Our construction requires a "weak" notion of (sub-exponential) security that guarantees that the puzzle cannot be solved by sub-exponential size attackers that have depth $\overline{T}^{\epsilon}$. Therefore, using the instantiation that relies on repeated squarings, we only need to assume that there are no huge improvements in the parallel complexity of repeated squaring algorithms even for very large attackers. It is worth mentioning that there are known algorithms for factoring that run in sub-exponential time. The best known algorithm has running time roughly $2^{n^{1/3}}$, where n is the input size (see [Dix81, Sho06]). In contrast, our assumption stipulates that there is no algorithm with running time $2^{n^{\epsilon}}$ for any $\epsilon > 0$ (for concreteness, think about $\epsilon = 0.001$). This is similar to the assumption being made in any construction that relies on sub-exponential factoring or discrete log.X

Non-malleable commitments. Non-malleable commitments, introduced by Dolev, Dwork and Naor [DDN91], guarantee *hiding* (the committed value is kept secret from the receiver), *binding* ("opening" can yield only a single value determined in the commit phase), and *non-malleability* (guaranteeing that it is hard to "maul" a commitment to a given value into a commitment to a related value). Non-malleable commitments are extremely well studied with huge body of works trying to pin down the exact round complexity and minimal assumptions needed to obtain them [Bar02, PR05b, PR05a, LPV08, PPV08, LP09, PW10, Wee10, Goy11, LP11, GLOV12, GPR16, COSV16, COSV17, Khu17, LPS17, KS17, KK19].

We need a *non-interactive* (i.e., one-message) non-malleable commitment, of which relatively few constructions are known. Pandey et al. [PPV08] formulated a concrete property of a random oracle and showed that it suffices for non-interactive non-malleable commitments. This is a non-standard and non-falsifiable (Naor [Nao03]) assumption. Lin et al. [LPS17] showed a construction that satisfies non-malleability against uniform attackers assuming a keyless collision resistant hash function, time-lock puzzles, non-interactive commitments, and NIWI proofs, all with sub-exponential hardness. Bitansky and Lin [BL18] were able to get non-malleability against all attackers (i.e., even non-uniform ones) by either replacing the keyless collision resistant hash function with a keyless *multi*-collision resistant hash function,[6] or using a new assumption regarding sub-exponentially secure one-way functions admitting some strong form of hardness amplification. Most recently, Kalai and Khurana [KK19] gave a construction of a non-interactive non-malleable commitment from sub-exponential hardness of factoring or discrete log, and sub-exponential *quantum* hardness of Learning With Errors (LWE) or Learning Parity with Noise (LPN).

We will use the construction of Bitansky and Lin [BL18] and Lin et al. [LPS17] both of which rely on time-lock puzzles. Various properties of their non-malleable commitments will be crucial for our construction.

One-message SPS zero-knowledge. This is a one-message proof system for every language in **NP** in the plain model and without any setup assumptions that satisfies a relaxed notion of zero-knowledge referred to a super-polynomial-time simulation (SPS) zero-knowledge [Pas03]. This concept was introduced by Barak and Pass [BP04] who also gave a construction assuming a keyless collision resistance hash function,[7] non-interactive commitments, and NIWI proofs, all with

[6] Actually, Bitansky and Lin [BL18] formulate an assumption about incompressible functions which is implied by keyless multi-collision resistant hash functions.

[7] Actually, Barak and Pass [BP04] formulate an assumption regarding the existence of a language in **P** which is

sub-exponential hardness. Their construction however satisfies soundness only against uniform attackers. Bitansky and Lin [BL18] showed how to overcome this limitation using keyless *multi*-collision resistant hash functions,[8] at the cost of obtaining a weaker soundness (allowing any attacker to output some bounded number of convincing proofs for false statements).

Non-malleable codes vs. commitments. (Non-interactive) non-malleable commitments and codes seem very similar. The only difference is that in the latter decoding should be efficient, while in the former it should be hard. There has been some evidence that the objects are not only syntactically related. For instance, non-malleable codes were used to construct non-malleable commitments [GPR16, CGM$^+$16]. In the reverse direction, some works used ideas from the (vast) literature on non-malleable commitments to get new non-malleable codes [CGL16, OPVV18, BDK$^+$19]. Our work continues the latter line of works and shows yet again that the notions are intimately related.

Lower bounds for non-malleability. We mentioned that there cannot be a non-malleable code secure against a class of tampering functions that includes the decoding procedures. In a very recent work, Ball et al. [BDKM20] gave various new lower bounds. The most related lower bound to this work is the one regarding (in)existence of non-malleable codes for $\mathbf{NC}^1$ ($\subseteq \mathbf{NC}$) in the standard model (a class that our construction captures). Their result introduces a notion of black-box reductions tailored for the setting of non-malleable codes and rules out such reductions for certain classes of tampering functions $\mathcal{F}$. Importantly, their impossibility results hold for constructions that rely only on the *minimal assumption* that there exists a distributional problem that is hard for the tampering class $\mathcal{F}$, but easy for $\mathbf{P}$. Our result bypasses the impossibility since we—in addition to an assumption of the above type (i.e. time-lock puzzles)—rely on standard cryptographic assumptions such as keyless multi-collision resistant hash functions, injective one-way functions, and non-interactive witness-indistinguishable proofs.

The magic of the repeated squaring assumption. In the past several years the repeated squaring assumption has played an important role in many works. In addition to the work about non-malleable commitments [LPS17] that we have already mentioned and the current work, this assumption was also used in several constructions of verifiable delay functions [Pie19, Wes19, EFKP19]. These functions are, roughly speaking, a publicly verifiable version of time-lock puzzles. The reason why this assumption has been so successful is that it brings a new dimension of hardness to the table, i.e., parallel-time, which is different from the type of hardness that standard cryptographic assumptions give.

(Multi-)collisions resistance. Collision resistant hash functions are (a family of) compressing functions where no efficient attacker can find a colliding pair in a random function from the family. The existence of such a family is a standard cryptographic assumption which is implied by many of the most classical assumptions such as factoring, discrete log, and more. A *keyless* collision resistant hash function is a *single* function where the above is hard for *(bounded) uniformity* attackers. Such functions exist in the random oracle model and may be heuristically instantiated using common constructions of cryptographic hash functions, such as SHA-3, where collisions are simply not known.

Multi-collision resistance [KNY19, BDRV18, BKP18, KNY18] is a relaxation of collision resistance where the goal of the attacker is to find a collection of *many* inputs (rather than just

hard to sample in slightly super-polynomial-time but easy to sample in a slightly larger super-polynomial-time. The existence of a keyless collision resistance hash function with sub-exponential hardness implies such a language.

[8] See Footnote 6.

two) to a random function in the family that collide. The keyless version, introduced by [BKP18], is again a single function but now the security guarantee can be formulated so that it holds for all efficient attackers, even non-uniform ones. Concretely, the security guarantee is that while an attacker of size s can find about s inputs that collide, it cannot find many more, say s^5 (i.e., multi-collisions cannot be compressed). Again, such functions exist in the random oracle model and may be heuristically instantiated using common constructions of cryptographic hash functions, such as SHA-3.

2 Technical Overview

At a very high level, as in several previous related works (e.g., [BDKM18, BDK+19]), we follow the Naor-Yung [NY90] paradigm that achieves CCA security of encryption by concatenating two instances of a CPA secure public key encryption scheme, followed by a (non-interactive) zero-knowledge proof of the equality of encrypted values. The novelty in this work stems from the way we instantiate and prove soundness of this approach in the context of non-malleable codes.

Concretely, the three main components in our construction are: a time-lock puzzle, a non-malleable commitment, and a one-message SPS zero-knowledge proof of consistency. As we will see later, these building blocks need to be instantiated in a very careful way to guarantee security. The construction $\mathsf{NMCode} = (\mathsf{NMCode.E}, \mathsf{NMCode.D})$ for a message space $\{0,1\}^\lambda$ and depth bound $\overline{T}$ is informally described in Algorithm 1.

```
NMCode.E($m$):
    1. Let $Z$ be a time-lock puzzle with hardness $\overline{T}$ and underlying message $m$.
    2. Let $c$ be a non-malleable commitment to $m$.
    3. Let $\pi$ be a zero-knowledge proof of consistency between $Z$ and $c$.
    4. Output $\hat{Z} := (Z, c, \pi)$.

NMCode.D($Z, c, \pi$):
    1. Verify the proof $\pi$.
    2. If verifies, solve the puzzle $Z$ and output the underlying message. Otherwise, output 0.
```

Algorithm 1: Our non-malleable code (*Informal*).

Let us provide some intuition and state some simple observations. Recall that a time-lock puzzle can be solved by "brute-force" in depth $\overline{T}$, but cannot be solved in depth $\ll \overline{T}$. However, time-lock puzzles may be malleable (in fact, the construction based on repeated squaring [RSW96] is easily malleable). Non-malleable commitments are, by definition, non-malleable but as opposed to time-lock puzzles, cannot be "brute-force" opened in polynomial time. Intuitively, adding the zero-knowledge proof of consistency in the above construction ties the hands of the attacker and achieves the desired properties of each of the primitives. The scheme inherits non-malleability from the non-malleable commitment while preserving the ability of solving the time-lock puzzle in polynomial time, which allows extraction of the underlying message and thereby decoding in polynomial time.

For efficiency, time-lock puzzles have a built-in trapdoor that allows one to generate puzzles *very* *fast* (while solving them requires considerable sequential effort). Thus, the running time of step 3 (generation of the zero-knowledge proof) takes fixed polynomial time (in the security parameter), independent of the depth bound $\overline{T}$. This is why $\mathsf{NMCode.E}$ has a fixed running time, polynomial in the security parameter, independent of $\overline{T}$. Negligible soundness of our construction, at a high level, is inherited from the security of the underlying primitives. Lastly, as we will explain shortly, we use the non-interactive non-malleable commitments of Lin et al. [LPS17] and Bitansky and Lin [BL18] both of which are based on time-lock puzzles (and keyless collision resistant hash functions or keyless multi-collision resistant hash functions, respectively) and so this will work nicely with our usage of the time-lock puzzle in our construction.

While the intuition described above is rather solid, proving that the above construction satisfies non-malleability turns out to be challenging. We explain the high-level approach next.

2.1 Overview of the Proof

We will first explain the high-level approach when considering only uniform tampering functions and later explain how to handle non-uniform ones.

Since we only handle uniform tampering functions (for now), it will suffice to rely (in addition to time-lock puzzles) on a non-malleable commitment for bounded uniformity tampering functions and a one-message SPS zero-knowledge proof which satisfies soundness for bounded uniformity attackers. For the commitment scheme we will use the one of Lin, Pass, and Soni [LPS17] and for the zero-knowledge we will use the one of Barak and Pass [BP04]. We remark again that while the scheme of Lin et al. [LPS17] is also based on a time-lock puzzle, it will be convenient to use it not only in terms of assumptions, but to actually use specific properties of the scheme that will help us carry out the proof.

The proof is by a hybrid argument where we start with the standard non-malleability game with a message m_0 and in the last hybrid we will play the non-malleability game with a message m_1. Recall that the non-malleability game (a.k.a. Man-In-Middle game) consists of two stages. In the first stage, the adversary gets a codeword and it tries to maul it into a code with a related message. Then, roughly, the distribution of the underlying message in that tampered codeword should be simulatable without knowing the message itself.

In a high level, here are the sequence of hybrids that we consider. We describe the changes incrementally, namely, each hybrid starts with the scheme from the previous hybrid and makes a modification.

- Hybrid 0: The original scheme.
- Hybrid 1: Instead of using the zero-knowledge prover, we use the simulator.
- Hybrid 2: Instead of committing to m, we commit to 0.
- Hybrid 3: Instead of decoding by solving the time-lock puzzle, we decode by extracting from the commitment.
- Hybrid 4: Instead of using m as the underlying message in the time-lock puzzle, use 0.

Showing that hybrids 0, 1, and 2 are indistinguishable is simple. Hybrids 0 and 1 are indistinguishable due to the zero-knowledge property, and hybrids 1 and 2 are indistinguishable due to the hiding of the commitment scheme. (We use a non-uniform simulator so that with an appropriate trapdoor we can generate simulated proofs in PPT.) The most challenging part is showing that hybrids 2 and 3 and hybrids 3 and 4 are indistinguishable.

Hybrids 2 and 3. The modification in this transition is from decoding via brute-force opening the time-lock puzzle, to decoding via extraction from the non-malleable commitment. To prove indistinguishability, we show that the distribution of the underlying value in the right commitment does not change throughout the hybrids when considering both methods of decoding.

A careful inspection of the schemes in each hybrid reveals that in order for the proof to go through, we need to satisfy two conditions simultaneously:

1. The extractor of the commitment scheme (whose size is S_{Ext}) cannot break zero-knowledge (which holds for all attackers of size at most S_{ZK}). That is,
$$S_{\mathsf{Ext}} \ll S_{\mathsf{ZK}}.$$
2. The simulator of the zero-knowledge scheme (whose size is S_{Sim}) cannot break non-malleability of the commitment (which holds for all attackers of size at most S_{NMCom}). That is,
$$S_{\mathsf{Sim}} \ll S_{\mathsf{NMCom}}.$$

It also holds that $S_{\mathsf{NMCom}} \ll S_{\mathsf{Ext}}$ since the commitment extractor can definitely break non-malleability (by extracting and re-committing to a related value). Therefore, the only way to satisfy the above two inequalities is if $S_{\mathsf{Sim}} \ll S_{\mathsf{ZK}}$, namely, a one-message zero-knowledge scheme where the simulator runs faster than the distinguisher![9] Unfortunately, no such scheme is known as in all known schemes the simulator needs to "break" the underlying cryptographic primitives and so it has to have more resources than the distinguishers.

Our idea to make this go through is to introduce another axis of hardness which will allow us to satisfy both "inequalities" simultaneously—the axes of total size and *non-uniformity*. We will set the complexities of the above procedures as follows, where λ denotes the security parameter and where $0 < c_1 < c_2 < c_3 < 1$:

- S_{Ext} (extraction from the non-malleable commitment): in quasi-polynomial size.
- S_{ZK} (zero-knowledge security): for all sub-exponential size attackers.
- S_{Sim} (ZK simulator complexity): in fixed polynomial size and "short" (bounded polynomial size) non-uniform advice.
- S_{NMCom} (non-malleability): for all quasi-polynomial size attackers that have some bounded polynomial non-uniformity.

With this careful choice of parameters, it is evident that the commitment extractor cannot break zero-knowledge and also the zero-knowledge simulator cannot break non-malleability. It is also not too hard to instantiate the primitives with the above properties. The zero-knowledge scheme of Barak and Pass [BP04] readily satisfies the above properties if it is sub-exponentially hard. To get the required non-malleable commitment, rely on the scheme of Lin et al. [LPS17].

[9] This kind of zero-knowledge simulation is known as *strong* super-polynomial simulation. Recently, Khurana and Sahai [KS17] managed to obtain it in two rounds, but we need a non-interactive scheme.

Hybrids 3 and 4. In this hybrid, we change the time-lock puzzle's underlying value and we want to use its hiding property. While seemingly being a relatively simple hybrid, it turns out that some complications arise. Specifically, to reduce to the underlying security of the time-lock puzzle, we need to come up with a bounded time attacker while there are two procedures that we need to run which seem to be of *arbitrary* depth. Specifically, in the reduction we need to simulate the whole experiment and use the distinguisher to break the security of the time-lock puzzle. The two procedures that seem to require arbitrary large depth are:

- The distinguisher itself, denoted D from now on.
- The extraction procedure of the non-malleable commitment (which we should execute as part of decoding).

We have *no* control over the depth (or size) of the distinguisher D, except that it is of arbitrary polynomial size and depth. However, we do know that its input, the message underlying the tampered code, is of bounded length. So, we *modify* the distinguisher and write it as a *truth table* which has hardcoded all of D's outputs on every possible input. Call this distinguisher $\tilde{D}$. Observe that $\tilde{D}$ (1) has the same input-output functionality as that of D (and so it serves as a good distinguisher), and (2) while $\tilde{D}$'s size is now exponential in the security parameter, its depth is some fixed polynomial in the security parameter!

For the extraction procedure, we intuitively make a similar modification. We rely on the fact that the extractor can be readily implemented in low depth by finding the underlying message in brute-force and in parallel. Note that for this to go through, the size of the extraction procedure has to smaller than the hardness of the time-lock puzzle (and this can be achieved by making the time-lock puzzle sufficiently long and using sub-exponential security). So, after switching to this alternate extraction method, we can simulate the whole experiment in fixed polynomial depth and reduce to the security game of the underlying time-lock puzzle.

The non-uniform case. Extending to handle non-uniform tampering functions is challenging in the fully non-interactive setting and in the plain model. While it is relatively straight-forward to replace the non-malleable commitment scheme of Lin et al. [LPS17] (which is uniformly non-malleable) with the one of Bitansky and Lin [BL18], the challenge stems from finding an appropriate non-uniform analogue for the uniformly sound one-message zero-knowledge scheme of Barak and Pass [BP04]. Indeed, in the plain model and allowing only one message *there is no* non-uniformly sound zero-knowledge scheme (as accepting proofs for false statements just exist).

The closest candidate is the one of Bitansky and Lin [BL18] who constructed a non-uniformly *weakly* sound one-message zero-knowledge scheme. This notion captures all non-uniform attackers but the soundness guarantee is weak: every attacker *can* output some number of accepting proofs for false statements but not too many of those. Unfortunately, if we use this scheme directly in our construction instead of the current zero-knowledge scheme, the above proof outline fails. Specifically, when we switch to alternate decoding (which extracts from the commitment rather than breaks the time-lock puzzle), if the adversary uses such a maliciously crafted proof (which verifies), it can easily distinguish the two hybrids (as their outputs will be different). Another thing that makes the situation even harder is that the bad set of proofs is not global but actually attacker-dependent so we cannot just "black-list" some set of proofs in the decoding procedure.

To this end, we observe that in the security reduction, the attacker *is* fixed and so the set of "bad" proofs is *non-uniformly* known. Therefore, we can modify the alternate decoding procedure to check whether the tampered proof is one of some (polynomial size) non-uniformly hardcoded set of bad proofs—the ones that the given attacker can find. If it is one of these bad proofs, we output a fixed message, the one underlying the time-lock puzzle that corresponds to the false statement. In this way, we are guaranteed that even when switch to alternate decoding, for those maliciously crafted proofs, the attacker will not see any difference between the two hybrids.

3 Preliminaries

Unless stated otherwise, the logarithms in this paper are base 2. For a distribution $\mathcal{D}$ we denote by $x \leftarrow \mathcal{D}$ an element chosen from $\mathcal{D}$ uniformly at random. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. We denote by U_n the uniform distribution over n-bit strings. A function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every constant $c > 0$, there exists an integer N_c such that $\mathsf{negl}(n) < n^{-c}$ for all $n > N_c$.

Model of computation. We consider uniform and non-uniform algorithms and we distinguish between their size and parallel time. The amount of non-uniformity is usually denoted by κ, the parallel time by T, and the size by S. We think of those algorithms as (possibly probabilistic) Turing machines with multiple heads that can operate in parallel. A non-uniform algorithm $\mathcal{A}$ is described by a family of of algorithms $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, one per security parameter λ. Each $\mathcal{A}_\lambda$ corresponds to an algorithm that has input size $n(\lambda)$ for some function $n\colon \mathbb{N} \to \mathbb{N}$. We say that $\mathcal{A}$ is T-time, denoted $\mathsf{Time}[\mathcal{A}] = T(\lambda)$, if for every $\lambda \in \mathbb{N}$, the parallel running time of $\mathcal{A}_\lambda$ is at most $T(\lambda)$. We say that $\mathcal{A}$ is S-size, denoted $\mathsf{Size}[\mathcal{A}] = S(\lambda)$, if for every $\lambda \in \mathbb{N}$, the total work that the algorithm $\mathcal{A}_\lambda$ does is at most $S(\lambda)$. Lastly, the mount of non-uniformity κ is chosen such that $\kappa(\lambda)$ is an upper bound on the size of advice used per λ.

Indistinguishability. We recall the standard definition of computational indistinguishability.

Definition 3.1 (Computational indistinguishability)**.** Let $\mathcal{X} = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be two ensembles of random variables. Let $S = S(\cdot)$, and $\kappa = \kappa(\cdot)$ be functions. We say that $\mathcal{X}$ and $\mathcal{Y}$ are (S, κ)-indistinguishable if for all probabilistic S-size $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ with non-uniformity κ, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$|\Pr[\mathcal{A}(\mathcal{X}_\lambda) = 1] - \Pr[\mathcal{A}(\mathcal{Y}_\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

We will use the notation $\mathcal{X} \approx_S \mathcal{Y}$ for (S, S)-indistinguishability and $\mathcal{X} \approx \mathcal{Y}$ for (S, S)-indistinguishability for all polynomial functions S.

3.1 Non-Malleable Commitments

A tag-based non-interactive commitment scheme $\mathsf{NMCom} = (\mathsf{NMCom.C}, \mathsf{NMCom.O})$ has the following syntax:

1. For all $m, \mathsf{tag} \in \{0,1\}^*$, $(c, r) \leftarrow \mathsf{NMCom.C}(1^\lambda, m, \mathsf{tag})$ is the commitment/opening pair for the message m with tag tag.
2. $\mathsf{NMCom.O}(c, m, r, \mathsf{tag}) \to \{0, 1\}$, where 1 indicates that r is a valid opening of c to m under tag tag and 0 is returned otherwise

A commitments scheme must satisfy the following properties:

– Correctness: For every $\lambda \in \mathbb{N}$ and $m, \mathsf{tag} \in \{0,1\}^*$,

$$\Pr[\mathsf{NMCom.O}(c, m, r, \mathsf{tag}) = 1 \; ; \; c, r \leftarrow \mathsf{NMCom.C}(1^\lambda, m, \mathsf{tag})] = 1,$$

where the probability is taken over the randomness of $\mathsf{NMCom.C}$.

– (Perfect) binding: For any string c, strings $m, m' \in \{0,1\}^*$ of the same length, strings $\mathsf{tag}, \mathsf{tag}' \in \{0,1\}^*$ of the same length, and opening strings r, r',

$$\text{if } \mathsf{NMCom.O}(c, m, r, \mathsf{tag}) = \mathsf{NMCom.O}(c, m', r', \mathsf{tag}') = 1, \text{ then } m = m'.$$

– S-hiding: For all polynomials $n = n(\lambda)$ and $n' = n'(\lambda)$, all $\mathsf{tag} \in \{0,1\}^{n'}$ and $m, m' \in \{0,1\}^n$, it holds that

$$\{\mathsf{NMCom.C}(1^\lambda, m, \mathsf{tag})\}_{\lambda \in \mathbb{N}} \approx_S \{\mathsf{NMCom.C}(1^\lambda, m', \mathsf{tag})\}_{\lambda \in \mathbb{N}}.$$

A non-tag-based commitment scheme is the same as above but the tag is fixed to $\mathsf{tag} = \perp$. In such cases, we omit the last input to $\mathsf{NMCom.C}$ and $\mathsf{NMCom.O}$.

Definition 3.2 (Underlying committed value, val)**.** Define $\mathsf{val}\colon \{0,1\}^* \to \{0,1\}^*$ to be the function that gets as input a commitment and outputs the underlying committed value. That is, $\mathsf{val}(c) = m$ if there exists $r \in \{0,1\}^*$ such that $\mathsf{NMCom.O}(c, m, r) = 1$, and $\perp$ otherwise.

Definition 3.3 ((Over-)extractability)**.** We say a commitment scheme is S-extractable if there exists a (uniform) S-size procedure $\mathsf{NMCom.E}$ and a negligible function $\mathsf{negl}(\cdot)$, such that for all $c \in \{0,1\}^*$ it holds that $\Pr[\mathsf{NMCom.E}(c) \neq \mathsf{val}(c)] \leq \mathsf{negl}(|c|)$. The scheme is S-*over*-extractable if the above holds only for strings $c \in \{0,1\}^*$ for which $\mathsf{val}(c) \neq \perp$.

Man In The Middle Execution (MIM). Let $\mathsf{NMCom} = (\mathsf{NMCom.C}, \mathsf{NMCom.O})$ be a tag-based commitment scheme, and $\mathcal{A}$ an arbitrary adversary. For security parameter $\lambda \in \mathbb{N}$, consider the following interactions by $\mathcal{A}(1^\lambda)$:

- Left interaction: $\mathcal{A}(1^\lambda)$ interacts with the sender and receives commitment to a message m of length λ using identity tag as $c \leftarrow \mathsf{NMCom.C}(1^\lambda, m, \mathsf{tag})$.
- Right interaction: $\mathcal{A}(1^\lambda)$ interacts with the receiver and tries to commit to related message $\tilde{m}$ using identity $\tilde{\mathsf{tag}}$ of its choice. Specifically, for the commitment $\tilde{c}$ sent to the receiver, let $\tilde{m} = \mathsf{val}(\tilde{c})$. Furthermore, if $\tilde{\mathsf{tag}} = \mathsf{tag}$, then we set $\tilde{m} = \perp$.

Let $\mathsf{MIM}^{\mathcal{A}}_{\mathsf{NMCom}}(\lambda, m)$ denote the random variable that describes $\tilde{m}$ that $\mathcal{A}$ commits to in the right interaction along with its output in the MIM execution $\mathsf{MIM}^{\mathcal{A}}_{\mathsf{NMCom}}(\lambda, m)$ as described above.

Definition 3.4 ((S, κ)-non-malleability [LPS17])**.** A tag-based commitment scheme $\mathsf{NMCom} = (\mathsf{NMCom.C}, \mathsf{NMCom.O})$ is said to be S-non-malleable if for all S-size algorithms $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ with non-uniform advice of size κ, it holds that

$$\{\mathsf{MIM}^{\mathcal{A}}_{\mathsf{NMCom}}(\lambda, m_0)\}_{\lambda \in \mathbb{N}, m_0 \in \{0,1\}^m} \approx \{\mathsf{MIM}^{\mathcal{A}}_{\mathsf{NMCom}}(\lambda, m_1)\}_{\lambda \in \mathbb{N}, m_1 \in \{0,1\}^m}.$$

3.2 One-Message SPS Zero-Knowledge Proofs

We recall the definitions of a one-message SPS zero-knowledge (1ZK) systems, following Barak and Pass [BP04]. In a high-level, this is a non-interactive protocol (just one message from the prover to the verifier) which provides soundness for the verifier and zero-knowledge for the prover. We emphasize that we want a protocol in the plain model, namely, without any setup assumptions (such as a trusted common reference string).

More precisely, a one-message SPS zero-knowledge argument system (P, V) for an NP relation $R(x, w)$ with associated language L consists of two polynomial-time algorithms:

- $\pi \leftarrow P(x, w, 1^\lambda)$: Given an instance x, witness w, and security parameter 1^λ, P produces a proof π.
- $b \leftarrow V(x, \pi)$: Given a proof π for instance x, V outputs a bit b, where $b = 1$ indicates acceptance.

We require the following standard three properties from such a zero-knowledge system.

– Completeness: For every $\lambda \in \mathbb{N}$ and every $(x, w) \in R$,

$$\Pr_P[V(x,\pi) = 1 : \pi \leftarrow P(x, w, 1^\lambda)] = 1$$

– (S_P, κ)-soundness: For all S_P-size adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ with non-uniform advice of size κ, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\Pr_{(x,\pi)\leftarrow\mathcal{A}_\lambda}\begin{bmatrix} x \notin L \\ V(x, \pi, 1^\lambda) = 1\end{bmatrix} \leq \mathsf{negl}(\lambda).$$

– S_D-zero-knowledge: There exists an algorithm $\mathsf{Sim} = \{\mathsf{Sim}_\lambda\}_{\lambda\in\mathbb{N}}$, such that for every $(x, w) \in R$,

$$\{\pi \leftarrow P(x, w, 1^\lambda)\}_{\lambda\in\mathbb{N}} \approx_{S_D} \{\pi \leftarrow \mathsf{Sim}_\lambda(x)\}_{\lambda\in\mathbb{N}}.$$

The simulator Sim sequentially executes two algorithms: First, (1) $\mathsf{Sim}^{\mathsf{pre}}(1^\lambda)$ which runs in sub-exponential time (but fixed polynomial depth) and outputs a trapdoor td. Second, (2) $\mathsf{Sim}^{\mathsf{post}}(\mathsf{td}, x)$ which runs in uniform polynomial time and outputs a simulated proof.

Non-uniform setting. In order to capture non-uniform attackers, Bitansky and Lin [BL18] introduced a relaxation of the soundness requirement and showed that one can obtain it for all non-uniform attackers. Their notion of soundness is called *weak soundness*. It allows each adversary to find some (small, bounded) number of accepting proofs for false statements, somewhat proportional to the size of its own description (and not much more than that). This relaxation by itself will not suffice for us since the simulator of [BL18] runs in super-polynomial time and depends on the statement being proven (a fact that will prevent our proof from going through). We would like to get more efficient, though non-uniform, simulation.

To this end, we consider languages where the statements $x = (x_1, x_2)$ are composed of two parts. The set of "bad" proofs in the weak soundness notion will consist of "bad" x_1's and arbitrary (unbounded number of) x_2. The gain is that we will guarantee the existence of a non-uniform polynomial size simulator per x_1. Bitansky and Lin [BL18] considered the same relaxation of soundness and improved simulation and called it φ-tuned soundness and simulation speedup. (In their notation φ is a projection function but we directly separate out the statement into x_1 and x_2.)

– (S_P, K)-weak-soundness: For any S_P-time probabilistic adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, there exists a negligible function $\mathsf{negl}(\cdot)$ and a collection of sets $\mathcal{Z} = \{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$, where $|\mathcal{Z}_\lambda| = K(|\mathcal{A}_\lambda|)$, such that for any $\lambda \in \mathbb{N}$,

$$\Pr_{\mathcal{A}_\lambda}\left[\begin{matrix} x \notin L(R) \wedge x_1 \notin \mathcal{Z}_\lambda \\ V(x, \pi, 1^\lambda) = 1 \end{matrix} : (x = (x_1, x_2), \pi) \leftarrow \mathcal{A}_\lambda\right] \leq \mathsf{negl}(\lambda).$$

– S_D-tuned zero-knowledge: For every x_1 we have an algorithm $\mathsf{Sim} = \{\mathsf{Sim}_\lambda\}_{\lambda\in\mathbb{N}}$, such that for every x_2 for which $(x = (x_1, x_2), w) \in R$,

$$\{\pi \leftarrow P(x, w, 1^\lambda)\}_{\lambda\in\mathbb{N}} \approx_{S_D} \{\pi \leftarrow \mathsf{Sim}_\lambda(x)\}_{\lambda\in\mathbb{N}}.$$

The simulator Sim sequentially executes two algorithms: First, (1) $\mathsf{Sim}^{\mathsf{pre}}(1^\lambda, x_1)$ which runs in sub-exponential time (but fixed polynomial depth) and outputs a trapdoor td. Second, (2) $\mathsf{Sim}^{\mathsf{post}}(\mathsf{td}, x)$ which runs in uniform polynomial time and outputs a simulated proof.

3.3 Time-Lock Puzzles

A time-lock puzzle [RSW96] is a a puzzle that can be generated given a solution, but given just the puzzle, it is moderately-hard to find the solution. A time-lock puzzle is initialized with some difficult parameter t and its security holds against a class of algorithms which we parametrize by their respective size and parallel time. The following definition follows Bitansky et al. [BGJ+16].

Definition 3.5 (Time-lock puzzles)**.** A time-lock puzzle (TL) is a tuple $\mathsf{TL} = (\mathsf{TL.Gen}, \mathsf{TL.Sol})$ with the following syntax

- $Z \leftarrow \mathsf{TL.Gen}(1^\lambda, t, s)$: A probabilistic algorithm that takes as input a security parameter, a difficulty parameter t and a solution $s \in \{0,1\}^\lambda$, where λ is a security parameter, and outputs a puzzle Z.
- $s \leftarrow \mathsf{TL.Sol}(Z)$: A deterministic algorithm that takes as input a puzzle Z and outputs a solution s.

We require the following three properties:

- Completeness: For every security parameter λ, difficulty parameter t, solution $s \in \{0,1\}^\lambda$, and puzzle Z in the support of $\mathsf{TL.Gen}(1^\lambda, t, s)$, $\mathsf{TL.Sol}(Z)$ outputs s.
- Efficiency:
  – $Z \leftarrow \mathsf{TL.Gen}(1^\lambda, t, s)$ runs in time $\mathsf{poly}(\log t, \lambda)$.
  – $s \leftarrow \mathsf{TL.Sol}(Z)$ can be computed in time $t \cdot \mathsf{poly}(\lambda)$ for Z in the support of $\mathsf{TL.Gen}(1^\lambda, t, \cdot)$
- (S, ϵ)-hardness: There exists a polynomial $\underline{t}(\cdot)$, such that for every $t(\cdot) \geq \underline{t}(\cdot)$, and every S-size t^ϵ-time adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$ and every pair $s_0, s_1 \in \{0,1\}^\lambda$, it holds that

$$\Pr\left[\begin{matrix} b \leftarrow \{0,1\},\ Z \leftarrow \mathsf{TL.Gen}(1^\lambda, t, s_b) \\ b' \leftarrow \mathcal{A}_\lambda(Z) \end{matrix} : b' = b\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

4 Definition of Non-Malleable Codes

In this section we give our definition of non-malleable codes. Our definition follows closely the definition of [BDKM18]. One difference though is that, rather than defining non-malleability for an abstract class of tampering functions, we define non-malleability directly for the class of tampering functions that we consider in this work .

Let Σ and Σ' be sets of strings. A *coding scheme* consists of two algorithms $\mathsf{NMCode} = (\mathsf{NMCode.E}, \mathsf{NMCode.D})$ such that $\mathsf{NMCode.E}\colon \Sigma \to \Sigma'$ and $\mathsf{NMCode.D}\colon \Sigma' \to \Sigma$. In words, $\mathsf{NMCode.E}$ ("encoding") maps messages to codewords and $\mathsf{NMCode.D}$ ("decoding") maps codewords to messages. The algorithm $\mathsf{NMCode.E}$ can be randomized and $\mathsf{NMCode.D}$ is assumed to be deterministic. For *correctness*, we require that for every message $m \in \Sigma$, it holds that

$$\Pr_{\mathsf{NMCode.E}}[\mathsf{NMCode.D}(\mathsf{NMCode.E}(m)) = m] = 1.$$

$\mathsf{NMCode.E}$ may also accept as an explicit input a security parameter in unary (in which case the syntax is $\mathsf{NMCode.E}(1^\lambda, m)$).

Non-malleability. Intuitively, this notion requires that given a codeword, as long as one cannot decode it, it is hard to generate a codeword with a different related underlying message. A function that takes a codeword and tries to generate a codeword for a related message out of it is called a *tampering function.* As mentioned, we have to limit the possible tampering functions in some way. Otherwise, a tampering function could decode a codeword and re-encode a related message.

Definition 4.1 (Tampering experiment)**.** For an algorithm $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, a security parameter $\lambda \in \mathbb{N}$, and a string $s \in \{0,1\}^\lambda$, define the tampering experiment:

$$\mathsf{Tamper}^{\mathsf{NMCode}}_{\mathcal{A},s}(\lambda) = \left\{ \begin{matrix} Z \leftarrow \mathsf{NMCode.E}(1^\lambda, s);\ \tilde{Z} = \mathcal{A}_\lambda(Z);\ \tilde{s} = \mathsf{NMCode.D}(\tilde{Z}) \\ \text{Output: } \tilde{s} \end{matrix} \right\},$$

where the randomness of the above experiment comes from the randomness of $\mathsf{NMCode.E}$.

Definition 4.2 ((S,T,κ)-non-malleability)**.** We say that a code NMCode is (S,T,κ)-non-malleable if for every S-size T-time algorithm $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ with κ bits of non-uniformity, there exists a (uniform) probabilistic polynomial-time simulator Sim such that

$$\{\mathsf{Tamper}^{\mathsf{NMCode}}_{\mathcal{A},s}(\lambda)\}_\lambda \approx \{\mathsf{Ideal}_{\mathsf{Sim},s}(\lambda)\}_\lambda,$$

where

$$\mathsf{Ideal}_{\mathsf{Sim},s}(\lambda) = \left\{ \begin{matrix} \tilde{s} \cup \{\text{same}\} \leftarrow \mathsf{Sim}^{\mathcal{A}_\lambda}(1^\lambda) \\ \text{Output: } s \text{ if output of } \mathsf{Sim} \text{ is same and otherwise } \tilde{s} \end{matrix} \right\}.$$

Medium non-malleability. We next define a different notion of non-malleability, referred to as *medium* non-malleability, which implies the one above (Definition 4.2) but is slightly easier to work with. The difference between the definitions is that the medium non-malleability experiment allows to output same^* only when some predicate g evaluated on an original codeword and a tampered one is satisfied. On the other hand, plain non-malleability (as defined above) does not impose restrictions on when the experiment is allowed to output same^*.

Definition 4.3 ((S, T, κ)-medium non-malleability)**.** We say that a code NMCode is (S, T, κ)-medium non-malleable if there exists a function g such that for every $s_0, s_1 \in \{0,1\}^\lambda$ and every S-size T-time algorithm $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ with κ bits of non-uniformity, it holds that

$$\{\mathsf{MedTamper}_{\mathcal{A},s_0}^{\mathsf{NMCode}}(\lambda)\}_{\lambda\in\mathbb{N}} \approx \{\mathsf{MedTamper}_{\mathcal{A},s_1}^{\mathsf{NMCode}}(\lambda)\}_{\lambda\in\mathbb{N}},$$

where the tampering experiment (whose randomness comes from the randomness of $\mathsf{NMCode.E}$) is defined as follows:

$$\mathsf{MedTamper}_{\mathcal{A},s}^{\mathsf{NMCode}}(\lambda) = \left\{ \begin{array}{c} Z \leftarrow \mathsf{NMCode.E}(1^\lambda, s);\ \tilde{Z} = \mathcal{A}_\lambda(Z);\ \tilde{s} = \mathsf{NMCode.D}(\tilde{Z}) \\ \text{Output: } \mathsf{same}^* \text{ if } g(Z, \tilde{Z}) = 1, \text{ and } \tilde{s} \text{ otherwise} \end{array} \right\},$$

and where $g(\cdot,\cdot)$ is a predicate such that for every $\mathcal{A}$ as above, $\lambda \in \mathbb{N}$, and $s \in \{0,1\}^\lambda$,

$$\Pr_{Z \leftarrow \mathsf{NMCode.D}(1^\lambda, s)}[g(Z, \mathcal{A}_\lambda(Z)) = 1 \ \wedge\ \mathsf{NMCode.D}(\mathcal{A}_\lambda(Z)) \neq s] \leq \mathsf{negl}(\lambda).$$

5 The Building Blocks

5.1 Time-Lock Puzzle

Theorem 5.1. *Assuming the sub-exponential hardness of the repeated squaring assumption, there exists a time-lock puzzle which is $(S^{\mathsf{TL}}, \epsilon)$-hard for a fixed $\epsilon \in (0,1)$ and where $S^{\mathsf{TL}} = 2^{3\lambda}$.*

We need a time-lock puzzle which, when instantiated with difficulty parameter t, is hard for machines that have parallel time at most t^ϵ for some fixed $\epsilon \in (0,1)$, even if their total size is $2^{3\lambda}$. We instantiate this primitive by relying on the repeated squaring assumption with sub-exponential hardness. The latter says that for some $\epsilon, \epsilon' \in (0,1)$ and any large enough t the following holds: any $2^{\lambda^{\epsilon'}}$-size t^ϵ-time algorithm cannot distinguish $(g, N, t, g^{2^t} \bmod N)$ from (g, N, t, g') for uniform $g, g' \in Z_{p\cdot q}^*$, where p and q are two random λ-bit primes. Note that it is common to assume the above assumption even for $((1-\epsilon)\cdot t)$-time algorithms—our assumption is much weaker.

To generate a puzzle Z with difficulty t and a message m, one does the following (we assume here for simplicity that m is short enough but it is easy to extend this): Sample an RSA modulus $N = pq$ to be a product of two random $\mathsf{poly}(\lambda)$-bit primes (with some large enough polynomial; see below), and computes $Z = (g, N, t, m + g^{2^t} \bmod N)$, where g is a randomly chosen element in Z_N^*. Note that using p and q it is possible to compute $g^{2^t} \bmod N$ in fixed polynomial time in λ (and in $\log t$ which is absorbed by the $\mathsf{poly}(\lambda)$ term) by first computing $a = 2^t \bmod \phi(N)$ (where $\phi(N) = (p-1)(q-1)$) and then computing $Z = g^a \bmod N$.

Assuming the sub-exponential hardness of the repeated squaring assumption, we want a time-lock puzzle whose guarantee is that the underlying value is completely hidden as long as the attacker has size less than $2^{3\lambda}$ size and t^ϵ time. To achieve this, the bit-length of p and q needs to be large enough. That is, we need to instantiate our primes with say $\tilde{\lambda} = (3\lambda)^{1/\epsilon}$ bits which would give security for attackers of size $2^{3\lambda}$ and t^ϵ time.

5.2 Non-Malleable Commitment

Theorem 5.2. *Assume that there is a keyless multi-collision resistant hash function, the repeated squaring assumption, NIWI proof for all* **NP***, and injective one-way functions, all with sub-exponential hardness. Then, there exists a non-interactive commitment which is S^{NMCom}-hiding,*

$S_{\mathsf{Ext}}^{\mathsf{NMCom}}$*-extractable via* $\mathsf{NMCom.Ext}$*, and* $(S_{NM}^{\mathsf{NMCom}}, \kappa_{NM}^{\mathsf{NMCom}})$*-non-malleable for all polynomial functions* $\kappa_{NM}^{\mathsf{NMCom}}(\lambda)$*, and where* $S^{\mathsf{NMCom}}(\lambda) = S_{NM}^{\mathsf{NMCom}}(\lambda) = 2^{\log^2 \lambda}$ *and* $S_{\mathsf{Ext}}^{\mathsf{NMCom}}(\lambda) = 2^{\log^3 \lambda}$*.*

Theorem 5.3. *Assume that there is a keyless collision resistant hash function, the repeated squaring assumption, NIWI proof for all* **NP***, and injective one-way functions, all with sub-exponential hardness. Then, there exists a non-interactive commitment which is* S^{NMCom}*-hiding,* $S_{\mathsf{Ext}}^{\mathsf{NMCom}}$*-extractable via* $\mathsf{NMCom.Ext}$*, and* $(S_{NM}^{\mathsf{NMCom}}, \kappa_{NM}^{\mathsf{NMCom}})$*-non-malleable, where* $S^{\mathsf{NMCom}}(\lambda) = S_{NM}^{\mathsf{NMCom}}(\lambda) = 2^{\log^2 \lambda}$*,* $S_{\mathsf{Ext}}^{\mathsf{NMCom}}(\lambda) = 2^{\log^3 \lambda}$*, and* $\kappa_{NM}^{\mathsf{NMCom}}(\lambda)$ *is a fixed polynomial.*

The difference between the two theorems are that in the former we obtain non-malleability for non-uniform attackers but using a keyless multi-collision resistant hash, while in the latter we obtain non-malleability only for attackers with *bounded* non-uniformity of fixed polynomial size but we are using a keyless (plain) collision resistant hash.

We need a one-message non-malleable tag-based commitment scheme (Definition 3.1) which is hiding for all (non-uniform) polynomial-size distinguishers and, extractable in quasi-polynomial size, and non-malleable for (smaller) quasi-polynomial size tempering functions. We briefly explain how these commitments are obtained.

The bounded uniformity scheme. To get the scheme satisfying the properties listed in Theorem 5.3 we use the scheme of Lin et al. [LPS17]. Let us review their scheme and explain why and how it satisfies the above properties. In a high-level, they use two types of commitment scheme, each with a different "axis" of hardness. From sub-exponentially secure injective one-way functions, they obtain a sub-exponentially secure commitment scheme Com^s. By instantiating Com^s with different security parameters, one can obtain a family of γ commitment schemes $\{\mathsf{Com}_i^s\}_{i\in[\gamma]}$ such that Com_{i+1}^s is harder than Com_i^s for all $1 \leq i \leq \gamma - 1$ in the *axis of size*. Namely, using size which is sufficient to extract from Com_i^s it is still hard to break Com_{i+1}^s. The extraction procedure is essentially a brute force algorithm that "tries all option".

A similar trick is performed using time-lock puzzles. They are used to obtain a family of γ commitment schemes $\{\mathsf{Com}_i^t\}_{i\in[\gamma]}$ such that Com_{i+1}^t is harder than Com_i^t for all $1 \leq i \leq \gamma - 1$ in the *axis of (parallel) time*. Namely, in time which is sufficient to extract from Com_i^t it is still hard to break Com_{i+1}^t. The extraction procedure is highly sequential and requires very small total size. In particular, in size which is sufficient to extract from any Com^t it is still hard to break any Com^s.

To construct a non-malleable commitment scheme NMCom, their key idea is to combine a Com^s and Com^t scheme with opposite strength. That is,

$$\mathsf{NMCom}(1^\lambda, m, \mathsf{tag}) = \mathsf{Com}_{\mathsf{tag}}^s(1^\lambda, s) \| \mathsf{Com}_{\gamma-\mathsf{tag}}^t(1^\lambda, s \oplus m) \text{ , where } s \leftarrow \{0,1\}^{|m|}.$$

The hiding and non-malleability proofs are the same as in [LPS17]. Hiding is immediate from hiding of the two underlying commitments Non-malleability holds by considering two cases. First, if the left tag i is smaller than the right tag j, the Com_j^t commitment on the right remains hiding for attackers of size and time enough for extracting from both Com_i^t and Com_j^s. Therefore the right committed value remains hidden, while the right is extracted. Otherwise, if the left tag i is larger than the right commitment j, then the Com_i^s commitment on the left remains hiding for attackers of size and time enough for extracting from both Com_j^s and $\mathsf{Com}_{\gamma-j}^t$. Thus, the left committed value remains hidden, while the right is extracted.

Of course, the above construction is not the final construction of [LPS17] as it supports only a small number of tags (while our goal is to support an exponential number of tags). To get around this they present a tag-amplification technique that is based on a tree-like structure and the way they avoid blow-up in the commitment size is by using a (keyless) collision resistant hash function (which causes the final construction to be non-malleable only with respect to bounded-uniformity attackers). We refer to [LPS17] for the precise details.

The non-uniform scheme. To get the scheme satisfying the properties listed in Theorem 5.2 we use the scheme of Bitansky and Lin [BL18] (which in turns is based on the scheme of [LPS17]). Here, they present a new tag-amplification technique, inspired by a interactive tag-amplification technique of Khurana and Sahai [KS17], where they make it non-interactive using their one-message zero-knowledge protocol (which is based on keyless multi-collision resistant hash functions).

5.3 One-Message Zero-Knowledge

Theorem 5.4. *Assume the existence of a one-way permutation, a NIWI proof systems for all NP, a keyless multi-collision resistant hash function, all sub-exponentially secure. Then, there exists a one-message SPS zero-knowledge argument system satisfying* (S_P, K)*-weak-soundness and* S_D*-tuned zero-knowledge for all polynomials* $S_P(\lambda)$*, and where* $K \in \mathsf{poly}(\lambda)$ *is a fixed polynomials, and* $S_D(\lambda) = 2^{\lambda^\eta}$ *for some constant* $\eta \in (0, 1)$*.*

Theorem 5.5. *Assume the existence of a one-way permutation, a NIWI proof systems for all NP, a collision resistant hash function secure against uniform polynomial-time algorithms, all sub-exponentially secure. Then, there exists a one-message SPS zero-knowledge argument system satisfying* (S_P, κ)*-soundness and* S_D*-zero-knowledge for all polynomial* $S_P(\lambda)$*, and where* $\kappa(\lambda) = 0$ *and* $S_D(\lambda) = 2^{\lambda^\eta}$ *for some constant* $\eta \in (0, 1)$*.*

The difference between the two theorems are that in the former we obtain weak-soundness for non-uniform attackers but using a keyless multi-collision resistant hash, while in the latter we obtain (plain) soundness only for uniform attackers but we are using a keyless (plain) collision resistant hash.

Barak and Pass [BP04] showed that a one-message zero-knowledge system exists assuming a collection of sub-exponentially hard primitives: a one-way permutation, a NIWI for all NP, and a keyless collision resistant hash function. Intuitively, their construction follows the Feige-Lapidot-Shamir paradigm [FLS90] where the protocol consists of a commitment to 0 and a WI argument for the statement that either the prover knows a witness for the given instance, or it used a commitment to a special (hard to guess) value. The special value which is hard to guess is, intuitively, a collision in an appropriately chosen hash function and this is why soundness only applies to uniform malicious provers. The simulator can either find such a collision by running in sub-exponential size or be PPT and have such a collision hardcoded as non-uniform advice. Their construction gives Theorem 5.5.

Bitansky and Lin [BL18] constructed a one-message zero-knowledge argument system by replacing the uniform hash function used by Barak and Pass with a *keyless* multi collision resistant hash function [BKP18]. Intuitively, their trapdoor is a collision that depends on (part of) the statement being proven. Their construction gives Theorem 5.4.

6 The Non-Malleable Code

In this section, we present a construction of a non-malleable code that satisfies non-malleability against all (non-uniform) polynomial size attackers that have bounded polynomial depth. In other words, the only way to maul a codeword is by having high depth.

Our construction relies on several building blocks on which we elaborate next.

1. A time-lock puzzle $\mathsf{TL} = (\mathsf{TL.Gen}, \mathsf{TL.Sol})$ as in Definition 3.5 and Theorem 5.1. This construction, for all large enough difficulty parameters t, allows to generate puzzles which are hard for any (non-uniform) machine whose parallel time/depth is at most t^{ϵ}, even it has size $2^{3\lambda}$.

2. A one-message SPS zero-knowledge argument system $\mathsf{ZK} = (\mathsf{ZK.P}, \mathsf{ZK.V})$ as in Definition 3.2 and Theorem 5.4. This construction is weakly sound w.r.t. all (non-uniform) polynomial-size attackers and satisfies zero-knowledge w.r.t. sub-exponential size adversaries.

3. A one-message non-malleable tag-based commitment scheme $\mathsf{NMCom} = (\mathsf{NMCom.C}, \mathsf{NMCom.O})$ as in Definition 3.1 and Theorem 5.2. This scheme is non-malleable for quasi-polynomial size tampering functions ($2^{\log^2 \lambda}$ say) and extractable in slightly higher quasi-polynomial size ($2^{\log^3 \lambda}$ say).

4. $\mathsf{Sig} = (\mathsf{Sig.G}, \mathsf{Sig.S}, \mathsf{Sig.V})$. A one-time signature scheme, unforgeable for polynomial-size attackers.

We show that assuming the existence of the above primitives, there is a code which is non-malleable for *all* polynomial-size attackers that run in bounded polynomial depth. We denote the latter $\overline{T}$. Our main result is summarized in the following theorem.

Theorem 6.1. *Assume a time-lock puzzle* TL*, a one-message SPS zero knowledge system* ZK*, a one-message non-malleable commitment scheme* NMCom*, and a one-time signature scheme* Sig*, as above. Then, there exist constants $\alpha, \beta, \gamma \in \mathbb{N}$ such that for any large enough polynomial $\overline{T}$, there is a code* $\mathsf{NMCode} = (\mathsf{NMCode.E}, \mathsf{NMCode.D})$ *(described below in Algorithms 2, 3, and 4) with the following properties:*

1. *The input of* $\mathsf{NMCode.E}$ *is a message from $\{0,1\}^{\lambda}$ and it outputs a codeword in $\{0,1\}^{\lambda^{\alpha}}$.*

2. *The running time of* $\mathsf{NMCode.E}$ *is λ^{β} and the running time of* $\mathsf{NMCode.D}$ *is $(\overline{T} \cdot \lambda)^{\gamma}$.*

3. *It is $(S, \overline{T})$-non-malleable for all polynomials $S(\lambda)$.*

The construction. Fix $\overline{T}$, the upper bound on the depth of the tampering function. The high level idea of the construction is to combine the hardness for parallel machines that comes from a time-lock puzzle together with non-malleability that comes from a non-malleable commitment. Specifically, the way we combine them is so that an encoding of a message m consists of a time-lock puzzle for m, a non-malleable commitment for m, and a zero-knowledge proof that ties them together by asserting that they have the same underlying message. The construction is described formally in Algorithms 2, 3, and 4

Sub-exponential security. The theorem extends to show that the resulting non-malleable code cannot be mauled in depth better than $\overline{T}$ even if the total size of the solver is (sub-)exponential in λ. For that, we need to make all of our underlying building blocks sub-exponentially secure (in particular, they have to remain secure in the presence of an exponential size adversary). We focus on the polynomial regime for simplicity.

```
Algorithm $\mathsf{NMCode.E}(1^\lambda, m)$ for $m \in \{0,1\}^\lambda$:

1. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.G}(1^\lambda)$.
2. $Z \leftarrow \mathsf{TL.Gen}(1^\lambda, \overline{T}^{2/\epsilon}, m;\ r_{\mathsf{TL}})$ with uniformly random $r_{\mathsf{TL}}$.
3. $(c, r_{\mathsf{NMCom}}) \leftarrow \mathsf{NMCom.C}(1^\lambda, m, tag = \mathsf{vk})$.
4. Compute a $\mathsf{ZK}$ proof $\pi \leftarrow \mathsf{ZK.P}(\cdot, \cdot, 1^\lambda)$ for the relation $\mathcal{R}_{\mathsf{u}}$ from Algorithm 4 using $(Z, c, \mathsf{vk})$
   as the instance and $(r_{\mathsf{TL}}, r_{\mathsf{NMCom}}, m)$ as the witness.
5. $\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sk}, (Z, c, \pi))$.
6. Output $\hat{Z} = (\mathsf{vk}, Z, c, \pi, \sigma)$.
```

Algorithm 2: The encoding procedure $\mathsf{NMCode.E}$.

```
Algorithm $\mathsf{NMCode.D}(\mathsf{vk}, Z, c, \pi, \sigma)$:

1. Verify the signature $\sigma$:
       $\mathsf{Sig.V}(\mathsf{vk}, (Z, c, \pi), \sigma) \stackrel{?}{=} 1$.
2. Verify the proof $\pi$:
       $\mathsf{ZK.V}((\mathsf{vk}, Z, c), \pi) \stackrel{?}{=} 1$.
3. If both accept, output $\mathsf{TL.Sol}(Z)$. Otherwise, output $0^\lambda$.
```

Algorithm 3: The decoding procedure $\mathsf{NMCode.D}$.

Organization. The proof of Theorem 6.1 consists of two parts: (1) efficiency analysis showing that the encoding and decoding procedures can be implemented with the required complexities and (2) showing that the code is non-malleable. Part (1) is proven in Section 6.1 and Part (2) is proven in Section 6.2.

6.1 Efficiency Analysis

Fix a security parameter $\lambda \in \mathbb{N}$ and a message $m \in \{0,1\}^\lambda$. The encoding (i.e., the output of $\mathsf{NMCode.E}(1^\lambda, m)$ consists of a verification key of a signature scheme, a time-lock puzzle, a non-malleable commitment scheme, a zero-knowledge proof, and a signature. All of these are of fixed polynomial size in λ.

The procedure $\mathsf{NMCode.E}$, on input $(1^\lambda, s)$, runs in time $\mathsf{poly}(\log \overline{T}, \lambda)$. Indeed, steps 1,3, and 5 are independent of $\overline{T}$ and take $\mathsf{poly}(\lambda)$ time. Step 2, by definition of time-lock puzzles, takes time $\mathsf{poly}(\log \overline{T}, \lambda)$. Finally, step 4 takes time $\mathsf{poly}(\log \overline{T}, \lambda)$ due to the running time of the verification procedure of the underlying language. The procedure $\mathsf{NMCode.D}$ can be computed in time $\overline{T}^{2/\epsilon} \cdot \mathsf{poly}(\lambda)$. Indeed, verifying the proof and the signature both take fixed polynomial time $\mathsf{poly}(\lambda)$ and the last step takes time $\overline{T}^{2/\epsilon} \cdot \mathsf{poly}(\lambda)$, by definition.

Relation $\mathcal{R}_{\mathsf{u}}\left((Z, c, \mathsf{vk}), (r_{\mathsf{TL}}, r_{\mathsf{NMCom}}, m)\right)$:

- Instance: a puzzle generated by $\mathsf{TL.Gen}(1^\lambda, \overline{T}^{2/\epsilon}, m)$, a commitment c, and a verification key vk.
- Witness: a string $r_{\mathsf{TL}} \in \{0,1\}^*$, a string $r_{\mathsf{NMCom}} \in \{0,1\}^*$, and a string $m \in \{0,1\}^\lambda$.
- Statement: $\mathsf{TL.Gen}(1^\lambda, \overline{T}^{2/\epsilon}, m; r_{\mathsf{TL}}) = Z$ and $\mathsf{NMCom.O}(c, m, r_{\mathsf{NMCom}}, tag = \mathsf{vk}) = 1$.

Algorithm 4: The Relation $\mathcal{R}_{\mathsf{u}}$.

6.2 Proof of Non-Malleability

In what follows, we prove that the coding scheme from Algorithms 2 and 3 is medium-non-malleable for all polynomial-size S and bounded polynomial-time $\overline{T}$ tampering functions. Let $g(Z, Z')$ be the procedure defined in Algorithm 5.

$g(Z, Z')$:

1. Parse Z as $(\mathsf{vk}, Z, c, \pi, \sigma)$ and Z' as $(\mathsf{vk}', Z', c', \pi', \sigma')$.
2. If $\mathsf{vk} = \mathsf{vk}'$ and σ' verifies (that is, $\mathsf{Sig.V}(\mathsf{vk}', (Z, c', \pi'), \sigma')$=1), output 1. Otherwise output 0.

Running time: The procedure g has fixed polynomial size in its input size.

Algorithm 5: The procedure g.

Claim 6.2. *For every non-uniform polynomial-size tampering function $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$, every difficulty parameter t, and every $m \in \{0,1\}^\lambda$, it holds that*

$$\Pr_{\hat{Z} \leftarrow \mathsf{NMCode.E}(1^\lambda, m)}\left[g(\hat{Z}, \mathcal{A}_\lambda(\hat{Z})) = 1 \ \wedge\ \mathsf{NMCode.D}(\mathcal{A}_\lambda(\hat{Z})) \neq m\right] \leq \mathsf{negl}(\lambda).$$

Proof. Let $\hat{Z} = (\mathsf{vk}, Z, c, \pi, \sigma)$ and $\mathcal{A}_\lambda(\hat{Z}) = \hat{Z}' = (\mathsf{vk}', Z', c', \pi', \sigma')$. If $g(\hat{Z}, \hat{Z}') = 1$, then $\mathsf{vk} = \mathsf{vk}'$ and $\mathsf{Sig.V}(\mathsf{vk}', Z', c', \pi'), \sigma')$=1. Also, recall that Z is a puzzle with underlying message m. Thus, if $\mathsf{NMCode.D}(\hat{Z}') \neq m$, it means that $(Z, c, \pi) \neq (Z', c', \pi')$. Thus, $\mathcal{A}_\lambda$ can be used to create (in polynomial time) a valid signature σ' w.r.t. verification key vk for a new statement which is a contradiction to the security of the one-time signature. ∎

We next show that w.r.t. the above g (Algorithm 5), for any polynomial-size algorithm $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ such that $\mathsf{Time}[\mathcal{A}] \leq \overline{T}$ and any $m_0, m_1 \in \{0,1\}^\lambda$, it holds that

$$\{\mathsf{MedTamper}^{\mathsf{NMCode}}_{\mathcal{A}, m_0}(\lambda)\}_{\lambda \in \mathbb{N}} \approx \{\mathsf{MedTamper}^{\mathsf{NMCode}}_{\mathcal{A}, m_1}(\lambda)\}_{\lambda \in \mathbb{N}},$$

where

$$\mathsf{MedTamper}^{\mathsf{NMCode}}_{\mathcal{A}, m}(\lambda) = \left\{ \begin{array}{c} \hat{Z} \leftarrow \mathsf{NMCode.E}(1^\lambda, m); \quad \tilde{m} = \mathsf{NMCode.D}(\mathcal{A}_\lambda(\hat{Z})) \\ \text{Output: } \mathsf{same}^* \text{ if } g(Z, \mathcal{A}_\lambda(Z)) = 1, \text{ and } \tilde{m} \text{ otherwise} \end{array} \right\}.$$

We do so by defining a sequence of hybrid experiments where we slowly change how $\mathsf{NMCode.E}$ and $\mathsf{NMCode.D}$ work and showing that every two consecutive hybrids are indistinguishable. For consistency of notation with what follows, we denote the non-malleable code from Algorithms 2 and 3 used in the original scheme by $\mathsf{NMCode}_0 = (\mathsf{NMCode}_0.\mathsf{E}, \mathsf{NMCode}_0.\mathsf{D})$, where $\mathsf{NMCode}_0.\mathsf{E} \equiv \mathsf{NMCode.E}$ and $\mathsf{NMCode}_0.\mathsf{D} \equiv \mathsf{NMCode.D}$. The first experiment that we define corresponds to the experiment $\{\mathsf{MedTamper}^{\mathsf{NMCode}_0}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in`\mathbb{N}}$ and the last one corresponds to an experiment where we encode m_1. From that point, one can "reverse" the sequence of experiment to reach the experiment $\{\mathsf{MedTamper}^{\mathsf{NMCode}_0}_{\mathcal{A},m_1}(\lambda)\}_{\lambda\in\mathbb{N}}$. We omit this part to avoid repetition.

Throughout the following sequence of hybrids, we treat $\mathcal{A}$ and m_0, m_1 as fixed.

Experiment $\mathcal{H}_0(\lambda)$. This is the original experiment, where we encode m_0 under NMCode_0 (see Algorithms 2 and 3) and execute the experiment $\{\mathsf{MedTamper}^{\mathsf{NMCode}_0}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}$.

Experiment $\mathcal{H}_1(\lambda)$. This experiment is the same as Experiment $\mathcal{H}_0(\lambda)$ except that we use the simulator of the ZK proof to generate π. Recall that the instance is $x = (x_1 = Z, x_2 = (c, \sigma))$. This gives rise to the scheme $\mathsf{NMCode}_1 = (\mathsf{NMCode}_1.\mathsf{E}, \mathsf{NMCode}_0.\mathsf{D})$, where $\mathsf{NMCode}_1.\mathsf{E}$ is describer in Algorithm 6. Using this scheme we execute the experiment $\{\mathsf{MedTamper}^{\mathsf{NMCode}_1}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}$. By the zero-knowledge property of ZK, this hybrid is indistinguishable from $\mathcal{H}_0(\lambda)$.

Algorithm $\mathsf{NMCode}_1.\mathsf{E}(m)$ for $m \in \{0,1\}^\lambda$:

1. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.G}(1^\lambda)$.
2. $Z \leftarrow \mathsf{TL.Gen}(1^\lambda, \overline{T}^{2/\epsilon}, m)$.
3. $(c, r) \leftarrow \mathsf{NMCom.C}(1^\lambda, m, tag = \mathsf{vk})$.
4. Use the (uniform sub-exponential size) simulator Sim to simulate a proof π for the relation $\mathcal{R}_\mathsf{u}$ using (Z, c, vk) as the instance.
5. $\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sk}, (Z, c, \pi))$.
6. Output $\hat{Z} = (\mathsf{vk}, Z, c, \pi, \sigma)$.

Algorithm 6: The encoding procedure $\mathsf{NMCode}_1.\mathsf{E}$ used in $\mathcal{H}_1(\lambda)$.

Claim 6.3. *It holds that*

$$\{\mathsf{MedTamper}^{\mathsf{NMCode}_0}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}} \approx \{\mathsf{MedTamper}^{\mathsf{NMCode}_1}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}.$$

Proof. Let $\hat{Z}_0 \leftarrow \mathsf{NMCode}_0.\mathsf{E}(1^\lambda, m_0)$ and $\hat{Z}_1 \leftarrow \mathsf{NMCode}_1.\mathsf{E}(1^\lambda, m_0)$. Next, since $\mathcal{A}$, g, and sampling $\hat{Z}_0$ and $\hat{Z}_1$ except the proof can all be done in polynomial size, the zero-knowledge property of ZK (see Section 3.2) guarantees that

$$\left|\Pr[g(\hat{Z}_0, \mathcal{A}_\lambda(\hat{Z}_0)) = 1] - \Pr[g(\hat{Z}_1, \mathcal{A}_\lambda(\hat{Z}_1)) = 1]\right| \leq \mathsf{negl}(\lambda). \tag{1}$$

Second, conditioning on the event that $g(\hat{Z}_0, \mathcal{A}_\lambda(\hat{Z}_0)) = 0$ together with the fact that $\mathsf{NMCode}_0.\mathsf{D}$ is polynomial size, by the zero-knowledge property,

$$\{\mathsf{NMCode}_0.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_0))\}_{\lambda\in\mathbb{N}} \approx \{\mathsf{NMCode}_1.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_1))\}_{\lambda\in\mathbb{N}}. \tag{2}$$

Combining Equations (1) and (2), the claim follows. ■

Experiment $\mathcal{H}_2(\lambda)$. This experiment is the same as Experiment $\mathcal{H}_1(\lambda)$ except that instead of committing to m_0 with a non-malleable commitment, we commit to 0^λ. This gives rise to the scheme $\mathsf{NMCode}_2 = (\mathsf{NMCode}_2.\mathsf{E}, \mathsf{NMCode}_0.\mathsf{D})$ which is described in Algorithm 7. Using this scheme we execute the experiment $\{\mathsf{MedTamper}^{\mathsf{NMCode}_2}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}$. By the hiding property of NMCom, this hybrid is indistinguishable from $\mathcal{H}_1(\lambda)$.

Algorithm $\mathsf{NMCode}_2.\mathsf{E}(m)$ for $m \in \{0,1\}^\lambda$:

1. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.G}(1^\lambda)$.
2. $Z \leftarrow \mathsf{TL.Gen}(1^\lambda, \overline{T}^{2/\epsilon}, m)$.
3. $(c, r) \leftarrow \mathsf{NMCom.C}(1^\lambda, 0^\lambda, tag = \mathsf{vk})$.
4. Use the (uniform sub-exponential size) simulator Sim to simulate a proof π for the relation $\mathcal{R}_\mathsf{u}$ using (Z, c, vk) as the instance.
5. $\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sk}, (Z, c, \pi))$.
6. Output $\hat{Z} = (\mathsf{vk}, Z, c, \pi, \sigma)$.

Algorithm 7: The encoding procedure $\mathsf{NMCode}_2.\mathsf{E}$ used in $\mathcal{H}_2(\lambda)$.

Claim 6.4. *It holds that*

$$\{\mathsf{MedTamper}^{\mathsf{NMCode}_1}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}} \approx \{\mathsf{MedTamper}^{\mathsf{NMCode}_2}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}.$$

Proof. Assume that there is a distinguisher between the two distributions that succeeds with noticeable probability. It must succeed with noticeable probability on some fixed Z in the support of $\mathsf{TL.Gen}(1^\lambda, \overline{T}^{2/\epsilon}, m)$. Now, when Z is fixed, we can simulate the rest of $\hat{Z}_1 \leftarrow \mathsf{NMCode}_1.\mathsf{E}(1^\lambda, m_0)$ and $\hat{Z}_2 \leftarrow \mathsf{NMCode}_2.\mathsf{E}(1^\lambda, m_0)$ in polynomial time using a non-uniform simulator $\mathsf{Sim}^{\mathsf{post}}$ that will have the trapdoor corresponding to Z hardwired (the trapdoor is essentially the output of $\mathsf{Sim}^{\mathsf{pre}}(Z)$). Furthermore, $\mathcal{A}$ and g are PPT and so the hiding property of the commitment scheme, which holds for non-uniform distinguishers of size $2^{\log^2\lambda}$, guarantees that

$$\left|\Pr[g(\hat{Z}_1, \mathcal{A}_\lambda(\hat{Z}_1)) = 1] - \Pr[g(\hat{Z}_2, \mathcal{A}_\lambda(\hat{Z}_2)) = 1]\right| \leq \mathsf{negl}(\lambda). \tag{3}$$

Second, conditioning on the event that $g(Z_1, \mathcal{A}_\lambda(Z_1)) = 0$ together with the fact that $\mathsf{NMCode}_0.\mathsf{D}$ is efficient (i.e., polynomial size), again by the hiding property of the commitment scheme,

$$\mathsf{NMCode}_0.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_1)) \approx \mathsf{NMCode}_0.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_2)). \tag{4}$$

Combining Equations (3) and (4), the claim follows. ■

Experiment $\mathcal{H}_3(\lambda)$. This experiment is the same as Experiment $\mathcal{H}_2(\lambda)$ except that we use an alternate decoding procedure. The alternate decoding procedure does not solve the time-lock puzzle in order to decode the secret m, but rather it "breaks" the commitment scheme and extracts m from it using $\mathsf{NMCom.Ext}$ *unless* the (tampered) proof corresponds to a time-lock puzzle instance coming from some fixed set of "bad" puzzles.

Recall that our instances for the ZK system have two parts $x = (x_1 = Z, x_2 = (c, \sigma))$. Therefore, by weak-soundness of ZK, every algorithm can come up with some small bounded number (depending on its non-uniformity size) of Z's that could be successfully verified yet they could come with a commitment which is not consistent and therefore comprise a false statement. If we encounter any of these puzzles, we will output a hard coded value instead of trying to extract the value from the commitment. We define an adversary $\mathcal{B}$ which runs the tampering function $\mathcal{A}$ on an honestly generated codeword ($\mathsf{NMCode}_0.\mathsf{E}$, uniform polynomial-size procedure). We look at the set of "bad" puzzles corresponding to $\mathcal{B}$.

More precisely, the adversary $\mathcal{B}$ can find a set $\mathcal{Z}'$ (that depends on the $\mathcal{B}$ and the hybrid experiment) of size at most $K \triangleq K^{\mathsf{ZK}}(|\mathcal{B}_\lambda| + O(1)) \in \mathsf{poly}(\lambda)$ of puzzles Z that might end up being falsely verified. We denote by $\mathcal{Z}$ the augmented set of puzzles together with their underlying value. Namely, $\mathcal{Z}$ is a set that consists of tuples of the form $(Z, \tilde{m})$, where $\tilde{m}$ is the message underlying Z.

This gives rise to the scheme $\mathsf{NMCode}_3 = (\mathsf{NMCode}_2.\mathsf{E}, \mathsf{NMCode}_1.\mathsf{D})$ which is described in Algorithm 8. Using this scheme we execute the experiment $\{\mathsf{MedTamper}_{\mathcal{A},m_0}^{\mathsf{NMCode}_3}(\lambda)\}_{\lambda\in\mathbb{N}}$.

```
Algorithm $\mathsf{NMCode}_1.\mathsf{D}(\mathsf{vk}, Z, c, \pi, \sigma)$:

  1. Verify the signature $\sigma$:
         $\mathsf{Sig.V}(\mathsf{vk}, (Z, c, \pi), \sigma) \overset{?}{=} 1.$
  2. Verify the ZK proof $\pi$:
         $\mathsf{ZK.V}((\mathsf{vk}, Z, c), \pi) \overset{?}{=} 1.$
  3. If either test from Steps 1 or 2 does not pass or $c = \bot$, output $0^\lambda$ and terminate.
  4. If $Z$ is a puzzle which is in $\mathcal{Z}$, output the corresponding message $\tilde{m}$ and terminate.
  5. Otherwise (both tests pass, $c \neq \bot$, and $Z \notin \mathcal{Z}'$), use the extractor $\mathsf{NMCom.Ext}(c)$ to get the
     underlying value $\tilde{m}$. Output $\tilde{m}$ (if extraction fails, $\tilde{m} = \bot$).
```

Algorithm 8: The decoding procedures $\mathsf{NMCode}_1.\mathsf{D}$ used in $\mathcal{H}_3(\lambda)$.

Claim 6.5. *It holds that*

$$\{\mathsf{MedTamper}_{\mathcal{A},m}^{\mathsf{NMCode}_2}(\lambda)\}_{\lambda\in\mathbb{N}} \approx \{\mathsf{MedTamper}_{\mathcal{A},m}^{\mathsf{NMCode}_3}(\lambda)\}_{\lambda\in\mathbb{N}}.$$

Proof. By definition of the $\mathsf{MedTamper}$ experiment, letting $Z_3 \leftarrow \mathsf{NMCode}_3.\mathsf{E}(1^\lambda, m)$, it is enough to show that

$$P_3 \triangleq \Pr[\mathsf{NMCode}_0.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_3)) \neq \mathsf{NMCode}_1.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_3)) \ \wedge \ g(\hat{Z}_3, \mathcal{A}_\lambda(\hat{Z}_3)) = 0] \leq \mathsf{negl}(\lambda). \quad (5)$$

Define the following three quantities where $\hat{Z}_0 \leftarrow \mathsf{NMCode}_0.\mathsf{E}(1^\lambda, m)$, $\hat{Z}_1 \leftarrow \mathsf{NMCode}_1.\mathsf{E}(1^\lambda, m)$, and $\hat{Z}_2 \leftarrow \mathsf{NMCode}_2.\mathsf{E}(1^\lambda, m)$:

$$\begin{aligned}
P_0 &\triangleq \Pr[\mathsf{NMCode}_0.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_0)) \neq \mathsf{NMCode}_1.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_0)) \ \wedge \ g(\hat{Z}_0, \mathcal{A}_\lambda(\hat{Z}_0)) = 0], \\
P_1 &\triangleq \Pr[\mathsf{NMCode}_0.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_1)) \neq \mathsf{NMCode}_1.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_1)) \ \wedge \ g(\hat{Z}_1, \mathcal{A}_\lambda(\hat{Z}_1)) = 0], \\
P_2 &\triangleq \Pr[\mathsf{NMCode}_0.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_2)) \neq \mathsf{NMCode}_1.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_2)) \ \wedge \ g(\hat{Z}_2, \mathcal{A}_\lambda(\hat{Z}_2)) = 0].
\end{aligned}$$

Note that $\mathsf{NMCode}_3.\mathsf{E} \equiv \mathsf{NMCode}_2.\mathsf{E}$ and so $P_2 = P_3$. Thus, to prove Equation (5), it is enough to show that $P_2 \leq \mathsf{negl}(\lambda)$. For this, we would like to show that P_0, $|P_0 - P_1|$ and $|P_1 - P_2|$ are all bounded by a negligible function in λ.

Claim 6.6. *$P_0 \leq \mathsf{negl}(\lambda)$.*

Proof. If $\mathsf{NMCode}_0.\mathsf{D}(\hat{Z}_0) \neq \mathsf{NMCode}_1.\mathsf{D}(\hat{Z}_0)$, then it must be that the ZK proof π (1) verifies yet the statement being proven is false and yet (2) the corresponding puzzle $Z \notin \mathcal{Z}'$. If this happens with noticeable probability in λ, we can use it to break the *weak*-soundness of the ZK by outputting an accepting proof for a false statement corresponding to a puzzle which is not in the set $\mathcal{Z}'$. To do this, we need to run $\mathcal{B}$ which in turn just executes the tampering function $\mathcal{A}_\lambda$ on an honestly generated codeword ($\mathsf{NMCode}_0.\mathsf{E}$). This is a contradiction to weak-soundness for $\mathcal{B}$. ■

Claim 6.7. *$|P_0 - P_1| \leq \mathsf{negl}(\lambda)$.*

Proof. We argue that if the difference is noticeable, then we can break the zero-knowledge property of ZK. Indeed, we can verify that the statement proven is true or false faster than the security of zero-knowledge. Verifying that $\mathsf{TL.Sol}(Z) = m$ can be done in polynomial-time (using the randomness used to generate Z). Extracting the committed value by running $\mathsf{NMCom.E}$ takes size $S^{\mathsf{NMCom}}_{\mathsf{Ext}}(\lambda) = 2^{\log^3 \lambda}$ which is much smaller than the size required to break the zero-knowledge property (the latter is $S^{\mathsf{ZK}}_D(\lambda) = 2^{\lambda^\eta}$). ■

Claim 6.8. *$|P_1 - P_2| \leq \mathsf{negl}(\lambda)$.*

Proof. Assume towards contradiction that $|P_1 - P_2| \geq 1/p(\lambda)$ for a polynomial $p(\cdot)$. We show that we can break the non-malleability of the commitment scheme. First, observe that in both terms P_1 and in P_2, the underlying experiment samples a time-lock puzzle of the message m. Therefore, there must be some fixed Z in the support of $\mathsf{TL.Gen}(1^\lambda, \overline{T}^{2/\epsilon}, m)$ for which the two quantities differ. From now on, we fix this Z and also fix the corresponding trapdoor td to ZK (that can be generated via $\mathsf{Sim}^{\mathsf{pre}}(Z)$). Next, we need to show an attacker-distinguisher pair $(D, \mathcal{B})$ for the Man-In-the-Middle execution such that $\mathcal{B}$ is an algorithm whose size is $\ll 2^{\log^2 \lambda}$ and whose non-uniformity is bounded by a fixed polynomial and D can distinguish (in polynomial-time) $\mathsf{MIM}^{\mathcal{A}}_{\mathsf{NMCom}}(\lambda, m)$ from $\mathsf{MIM}^{\mathcal{A}}_{\mathsf{NMCom}}(\lambda, 0^\lambda)$ for a fixed message $m \in \{0,1\}^\lambda \setminus \{0^\lambda\}$. The algorithm $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ is going to have Z and td hardwired and it is defined as follows.

On input 1^λ, algorithm $\mathcal{B}_0$ does:

1. Generate $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.G}(1^\lambda)$.

2. Send $tag = \mathsf{vk}$ to the challenger as the desired tag and outputs sk to $\mathcal{B}_1$.

On input $(\mathsf{sk}, \mathsf{vk}, c)$, algorithm $\mathcal{B}_1$ has hardwired the time-lock puzzle Z and corresponding trapdoor td of polynomial size for the ZK proof system, and it does:

1. Use the simulator $\mathsf{Sim}^{\mathsf{post}}$ along with a trapdoor td to simulate a proof π for the statement (Z, c, vk), and sign $\sigma \leftarrow \mathsf{Sig.S}(\mathsf{sk}, (Z, c, \pi))$. Let $\hat{Z} = (\mathsf{vk}, Z, c, \pi, \sigma)$.
2. Compute $\mathcal{A}_\lambda(\hat{Z})$ which we interpret as $\hat{Z}' = (\mathsf{vk}', Z', c', \pi', \sigma')$.
3. If either $g(\hat{Z}, \hat{Z}') = 1$, σ' does not verify, π' does not verify, or $\pi' \in \mathcal{Z}'$, output $\perp$. Otherwise, output (c', Z').

The distinguisher D receives either the message m' underlying c' or $\perp$, as well as $\mathsf{out} = (c', Z')$. The distinguisher D outputs 0 if either its input c' is $\perp$ or if $\mathsf{TL.Sol}(Z') = m'$. It outputs 1 otherwise. It holds that

$$\Pr_{c \leftarrow \mathsf{NMCom.C}(m, tag=\mathsf{vk})}[D(m', \mathsf{out}) = 1] = P_1(\lambda)$$

and

$$\Pr_{c \leftarrow \mathsf{NMCom.C}(0^\lambda, tag=\mathsf{vk})}[D(m', \mathsf{out}) = 1] = P_2(\lambda).$$

Thus,

$$\left| \Pr_{c \leftarrow \mathsf{NMCom.C}(m, tag=\mathsf{vk})}[D(m', \mathsf{out}) = 1] - \Pr_{c \leftarrow \mathsf{NMCom.C}(0^\lambda, tag=\mathsf{vk})}[D(m', \mathsf{out}) = 1] \right| \geq \frac{1}{p(\lambda)}$$

which is a contradiction to the non-malleability of NMCom. All of the steps of $\mathcal{B}$ can be implemented in PPT (with polynomial size non-uniformity), while non-malleability of the commitment scheme holds for all non-uniform PPT attackers. Lastly, D runs in (uniform) polynomial time directly by definition. ∎

Experiment $\mathcal{H}_4(\lambda)$. This experiment is the same as Experiment $\mathcal{H}_3(\lambda)$ except that we modify the alternate decoding procedure to use another extractor $\mathsf{NMCom.Ext}'$ instead of $\mathsf{NMCom.Ext}$. The new extractor is a depth/parallel-time efficient version of $\mathsf{NMCom.Ext}$. That is, we break the commitment scheme by brute-forcing all options for an opening in parallel. For concreteness, this can be implemented in (say) 2^λ size and fixed polynomial depth. Namely, we execute the experiment $\{\mathsf{MedTamper}_{\mathcal{A},m_1}^{\mathsf{NMCode}_4}(\lambda)\}_{\lambda \in \mathbb{N}}$. This gives rise to the scheme $\mathsf{NMCode}_4 = (\mathsf{NMCode}_2.\mathsf{E}, \mathsf{NMCode}_2.\mathsf{D})$ which is described in Algorithm 9. Using this scheme we execute the experiment $\{\mathsf{MedTamper}_{\mathcal{A},m_0}^{\mathsf{NMCode}_4}(\lambda)\}_{\lambda \in \mathbb{N}}$.

Claim 6.9. *It holds that* $\{\mathsf{MedTamper}_{\mathcal{A},m_0}^{\mathsf{NMCode}_3}(\lambda)\}_{\lambda \in \mathbb{N}}$ *and* $\{\mathsf{MedTamper}_{\mathcal{A},m_0}^{\mathsf{NMCode}_4}(\lambda)\}_{\lambda \in \mathbb{N}}$ *are identically distributed.*

Proof. The difference between the two experiments is the way that decoding extracts the message underlying the commitment scheme. Specifically, in the experiment that corresponds to NMCode_3 it works by running $\mathsf{NMCom.Ext}$ while in NMCode_4 it works by running $\mathsf{NMCom.Ext}'$. Both extraction procedures output the same exact value on every input and so since the output of the experiment contains only the output of this procedure, the two experiments are identically distributed. ∎

Algorithm $\mathsf{NMCode}_2.\mathsf{D}(\mathsf{vk}, Z, c, \pi, \sigma)$:

1. Verify the signature σ:
$$\mathsf{Sig.V}(\mathsf{vk}, (Z, c, \pi), \sigma) \overset{?}{=} 1.$$

2. Verify the ZK proof π:
$$\mathsf{ZK.V}((\mathsf{vk}, Z, c), \pi) \overset{?}{=} 1.$$

3. If either test from Steps 1 or 2 does not pass or $c = \bot$, output 0^λ and terminate.

4. If π is a proof which is in $\mathcal{Z}$, output the corresponding message $\tilde{m}$ and terminate.

5. Otherwise (both tests pass, $c \neq \bot$, and $\pi \notin \mathcal{Z}'$), use the extractor $\mathsf{NMCom.Ext}'(c)$ to get the underlying value $\tilde{m}$. Output $\tilde{m}$ (if extraction fails, $\tilde{m} = \bot$).

Algorithm 9: The decoding procedures $\mathsf{NMCode}_2.\mathsf{D}$ used in $\mathcal{H}_4(\lambda)$.

Experiment $\mathcal{H}_5(\lambda)$. This experiment is the same as Experiment $\mathcal{H}_4(\lambda)$ except that we use m_1 as the underlying message for $\mathsf{TL.Gen}$ (rather than m_0), namely, we execute the experiment $\{\mathsf{MedTamper}^{\mathsf{NMCode}_4}_{\mathcal{A},m_1}(\lambda)\}_{\lambda\in\mathbb{N}}$.

Claim 6.10. *It holds that*

$$\{\mathsf{MedTamper}^{\mathsf{NMCode}_4}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}} \approx \{\mathsf{MedTamper}^{\mathsf{NMCode}_4}_{\mathcal{A},m_1}(\lambda)\}_{\lambda\in\mathbb{N}}.$$

Proof. Let $Z_0 \leftarrow \mathsf{NMCode}_3.\mathsf{E}(m_0)$ and $Z_1 \leftarrow \mathsf{NMCode}_3.\mathsf{E}(m_1)$. We first argue that

$$|\Pr[g(Z_0, \mathcal{A}_\lambda(Z_0)) = 1] - \Pr[g(Z_1, \mathcal{A}_\lambda(Z_1)) = 1]| \leq \mathsf{negl}(\lambda).$$

This proof uses the fact that $\mathcal{A}$ has bounded depth. Towards contradiction, if the above difference is noticeable, then we can use $\mathsf{NMCode}_3.\mathsf{E}$, g, and $\mathcal{A}$ to create a polynomial size and depth $\overline{T}+\mathsf{poly}(\lambda)$ machine that gets as input a time-lock puzzle Z that corresponds to either m_0 or to m_1 (each with probability 1/2), computes the appropriate non-malleable encoding $\hat{Z}$, and then decide whether $\mathsf{TL.Sol}(Z)$ is m_0 or m_1 with noticeable probability. This is a contradiction to the assumption on the underlying time-lock puzzle which stays that no machine with the above size and depth $(\overline{T}^{2/\epsilon})^\epsilon = \overline{T}^2$ (which is much larger than $\overline{T} + \mathsf{poly}(\lambda)$ for large enough $\overline{T}$) can solve it. (Recall that the whole experiment, including the ZK simulator and the extractor, can be implemented in fixed polynomial depth.)

Next, we complete the proof by showing that

$$\mathsf{NMCode}_2.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_0)) \approx \mathsf{NMCode}_2.\mathsf{D}(\mathcal{A}_\lambda(\hat{Z}_1)).$$

Assume towards contradiction that there is a distinguisher D between the above two distributions which are the message underlying the tampered codeword encoded using $\mathsf{NMCode}_2.\mathsf{E}$. Let $\tilde{D}$ be a distinguisher that has the same truth table as D but is implemented in (at most) 2^λ size and $\leq \lambda$ time. This can be done by a procedure that has hardwired the whole truth table and it uses the input, which is of size λ, to access an entry. As before, we can fix D and this distinguisher to create the non-malleable codeword, which we can then use by simulating $\mathsf{NMCode}_2.\mathsf{E}$ and $\mathsf{NMCode}_2.\mathsf{D}$

to simulate the codeword and its decoding, and then to decide what was the underlying message. This requires an algorithm of (at most) fixed polynomial size and fixed polynomial time in λ, except $\mathsf{NMCom.Ext}'$ which is at most exponential size and fixed polynomial depth, and $\tilde{D}$ which is of 2^{λ} size and of λ depth. Thus, we obtain a procedure of size $S = 2^{\lambda} + 2^{\lambda} + \mathsf{poly}(\lambda)$ and time $T \in \mathsf{poly}(\lambda)$ (for some fixed polynomial poly) that has noticeable correlation with $\mathsf{TL.Sol}(Z)$. The latter, by assumption, cannot be done by a $\overline{T}$-time procedure, even if its size is $2^{3\lambda}$, which implies our contradiction by choosing $\overline{T}$ large enough so that $T < \overline{T}$.

■

■

7 The Case of Uniform Tampering

In Section 6 we gave a construction of a non-malleable code secure against all tampering functions that can be described as non-uniform polynomial size algorithm with *bounded* polynomial depth. In this section we focus on the natural class of tampering functions that consists of *uniform* polynomial size algorithm with bounded polynomial parallel running time. This is the class that was considered in the work of Ball et al. [BDK+19].

The construction is essentially the same as the one for non-uniform tampering functions and the main differences are in how we instantiate the building blocks and how the security proof goes through. Let us precisely list the building blocks with which we use the scheme from Section 6 (Algorithms 2, 3, and 4). We note that the time-lock puzzle and the signature scheme that we use (Items 1 and 4 below) are the same as the one we used in Section 6.

1. A time-lock puzzle $\mathsf{TL} = (\mathsf{TL.Gen}, \mathsf{TL.Sol})$ as in Definition 3.5 and Theorem 5.1. The construction, for all large enough difficulty parameters t, allows to generate puzzles which are hard for any (non-uniform) machine whose parallel time is at most t^{ϵ}, even it has size $2^{3\lambda}$.

2. A one-message zero-knowledge argument system $\mathsf{ZK} = (\mathsf{ZK.P}, \mathsf{ZK.V})$ as in Definition 3.2 and Theorem 5.5). This construction is sound w.r.t. all polynomial-size attackers with polynomially bounded non-uniformity, there is a (non-uniform) PPT simulator that satisfies zero-knowledge w.r.t. sub-exponential size adversaries.

3. A one-message non-malleable tag-based commitment scheme $\mathsf{NMCom} = (\mathsf{NMCom.C}, \mathsf{NMCom.O})$ as in Definition 3.1 and Theorem 5.3. This scheme which is hiding for all (non-uniform) polynomial-size distinguishers, extractable in quasi-polynomial ($2^{\log^2 \lambda}$), and non-malleable for all quasi-polynomial ($2^{\log^3 \lambda}$) tampering functions that have bounded (larger than the non-uniformity needed for the ZK simulator) polynomial size non-uniformity.

4. $\mathsf{Sig} = (\mathsf{Sig.G}, \mathsf{Sig.S}, \mathsf{Sig.V})$. A one-time signature scheme, unforgeable for polynomial-size attackers.

Overview of the proof. The proof works by defining a sequence of hybrid experiments, where in the first the Man-In-the-Middle game is played with a message m_0 and in the last with a message m_1. The sequence of experiments is analogous to the one described in Section 6 except that we do not need worry about "weak-soundness" of the ZK scheme and so some transitions follow due to slightly different reasons.

- $\mathcal{H}_0(\lambda)$: This is the original experiment.
- $\mathcal{H}_1(\lambda)$: This is the same experiment, except that we use the simulator to generate proofs. Experiments $\mathcal{H}_0(\lambda)$ and $\mathcal{H}_1(\lambda)$ are indistinguishable due to the zero-knowledge property of ZK.
- $\mathcal{H}_2(\lambda)$: This is the same experiment, except that we commit to 0 and not to m_0. Experiments $\mathcal{H}_1(\lambda)$ and $\mathcal{H}_2(\lambda)$ are indistinguishable due to the hiding property of NMCom (which holds for all non-uniform PPT attackers).
- $\mathcal{H}_3(\lambda)$: This is the same experiment, except that we switch to alternate decoding where, rather than solving the time-lock puzzle, the commitment is being extracted via the extractor $\mathsf{NMCom.Ext}$. Indistinguishability of $\mathcal{H}_2(\lambda)$ and $\mathcal{H}_3(\lambda)$ is proven by defining the same three quantities P_0, P_1, and P_2 such that P_2 corresponds to $|\Pr[\mathcal{H}_2(\lambda) = 1] - \Pr[\mathcal{H}_3(\lambda) = 1]|$.
  - $|P_0| \leq \mathsf{negl}(\lambda)$: This follows by (polynomially bounded non-uniformity) soundness of ZK. (Recall that soundness against uniform attacker suffices since our attacker is uniform.)
  - $|P_1 - P_0| \leq \mathsf{negl}(\lambda)$: This follows by observing that by the choice of the hardness parameters of the scheme, one can extract from the commitment (in quasi-polynomial size) without breaking zero-knowledge (which holds for sub-exponential attackers).
  - $|P_2 - P_1| \leq \mathsf{negl}(\lambda)$: This follows by observing that if this is false, then we can use the experiment to break (polynomially bounded non-uniformity) non-malleability. Since the ZK simulator (which is of fixed polynomial size and thus has bounded non-uniformity) cannot break non-malleability (which holds for all polynomial size attackers that have bounded but larger non-uniformity) this implies a contradiction.
- $\mathcal{H}_4(\lambda)$: This is the same experiment, except that we modify the alternate decoding to use a low depth/parallel-time extraction procedure $\mathsf{NMCom.Ext}'$. Since the extraction procedures have the same functionality, experiments $\mathcal{H}_3(\lambda)$ and $\mathcal{H}_4(\lambda)$ are identical.
- $\mathcal{H}_5(\lambda)$: This is the same experiment, except that we use time-lock puzzle to encode the other message m_1. Experiments $\mathcal{H}_5(\lambda)$ and $\mathcal{H}_4(\lambda)$ are indistinguishable due to the security of the time-lock puzzle (and using the fact that the attacker as well as the extractor have bounded polynomial depth).

Acknowledgements

We are grateful to Naomi Sirkin for carefully reading the paper and providing useful feedback. We also thank the CRYPTO 2021 reviewers for their useful remarks.

References

[AAG+16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In *Theory of Cryptography - 13th International Conference, TCC*, pages 393–417, 2016.

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

[ADL18] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. *SIAM J. Comput.*, 47(2):524–546, 2018.

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *43rd Symposium on Foundations of Computer Science FOCS*, pages 345–355, 2002.

[BDG$^+$18] Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 826–837, 2018.

[BDK$^+$19] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, Huijia Lin, and Tal Malkin. Non-malleable codes against bounded polynomial time tampering. In *Advances in Cryptology - EUROCRYPT*, pages 501–530, 2019.

[BDKM16] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In *Advances in Cryptology - EUROCRYPT*, pages 881–908, 2016.

[BDKM18] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes from average-case hardness: AC^0, decision trees, and streaming space-bounded tampering. In *Advances in Cryptology - EUROCRYPT*, pages 618–650, 2018.

[BDKM20] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Limits to non-malleability. In *11th Innovations in Theoretical Computer Science Conference, ITCS*, pages 80:1–80:32, 2020.

[BDRV18] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Multi-collision resistant hash functions and their applications. In *Advances in Cryptology - EUROCRYPT 2018*, pages 133–161, 2018.

[BGJ$^+$16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, ITCS*, pages 345–356, 2016.

[BGW19] Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. In *Advances in Cryptology - CRYPTO*, pages 413–434, 2019.

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 671–684, 2018.

[BL18] Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In *Theory of Cryptography - 16th International Conference, TCC*, pages 209–234, 2018.

[BOV07] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

[BP04] Boaz Barak and Rafael Pass. On the possibility of one-message weak zero-knowledge. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, pages 121–132, 2004.

[BP15] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC*, pages 401–427, 2015.

[CDTV16] Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. In *Theory of Cryptography - 13th International Conference, TCC*, pages 306–335, 2016.

[CG16] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. *IEEE Trans. Information Theory*, 62(3):1097–1118, 2016.

[CGL15] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:75, 2015.

[CGL16] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 285–298, 2016.

[CGM$^+$16] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP*, pages 31:1–31:14, 2016.

[CL17] Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1171–1184, 2017.

[CLP13] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from P-certificates. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 50–59, 2013.

[CMTV15] Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC*, pages 532–560, 2015.

[COSV16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In *Advances in Cryptology - CRYPTO*, pages 270–299, 2016.

[COSV17] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In *Advances in Cryptology - CRYPTO*, pages 127–157, 2017.

[CZ16] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 670–683, 2016.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, STOC*, pages 542–552, 1991.

[Dix81] John D Dixon. Asymptotically fast factorization of integers. *Mathematics of computation*, 36(153):255–260, 1981.

[DLSZ15] Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC*, pages 427–450, 2015.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *Innovations in Computer Science - ICS*, pages 434–452, 2010.

[DPW18] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *J. ACM*, 65(4):20:1–20:32, 2018.

[EFKP19] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. *IACR Cryptology ePrint Archive*, 2019:619, 2019.

[EFKP20] Naomi Epraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Non-malleable time-lock puzzles and applications, 2020.

[FHMV17] Sebastian Faust, Kristina Hostáková, Pratyay Mukherjee, and Daniele Venturi. Non-malleable codes for space-bounded tampering. In *Advances in Cryptology - CRYPTO*, pages 95–126, 2017.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, FOCS*, pages 308–317, 1990.

[FMVW16] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key derivation for poly-size tampering circuits. *IEEE Trans. Information Theory*, 62(12):7179–7194, 2016.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 51–60, 2012.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In *Advances in Cryptology - CRYPTO*, pages 97–111, 2006.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 695–704, 2011.

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1128–1141, 2016.

[Khu17] Dakshita Khurana. Round optimal concurrent non-malleability from polynomial hardness. In *Theory of Cryptography - 15th International Conference, TCC*, pages 139–171, 2017.

[KK19] Yael Tauman Kalai and Dakshita Khurana. Non-interactive non-malleability from quantum supremacy. In *Advances in Cryptology - CRYPTO*, pages 552–582, 2019.

[KLT16] Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Practical non-malleable codes from l-more extractable hash functions. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1317–1328, 2016.

[KNY18] Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. In *Advances in Cryptology - EUROCRYPT 2018*, pages 162–194, 2018.

[KNY19] Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *J. ACM*, 66(5):34:1–34:28, 2019.

[KS17] Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 564–575, 2017.

[Li12] Xin Li. Non-malleable extractors, two-source extractors and privacy amplification. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 688–697, 2012.

[Li13] Xin Li. New independent source extractors with exponential improvement. In *Symposium on Theory of Computing Conference, STOC*, pages 783–792, 2013.

[Li17] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1144–1156. ACM, 2017.

[Li18] Xin Li. Non-malleable extractors and non-malleable codes: Partially optimal constructions. *arXiv preprint arXiv:1804.04005*, 2018.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *Advances in Cryptology - CRYPTO*, pages 517–532, 2012.

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

[LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 705–714, 2011.

[LPS17] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 576–587. IEEE Computer Society, 2017.

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC*, pages 571–588, 2008.

[May92] Timothy May. Timed-release crypto, 1992.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Advances in Cryptology - CRYPTO*, pages 96–109, 2003.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, STOC*, pages 427–437, 1990.

[OPVV18] Rafail Ostrovsky, Giuseppe Persiano, Daniele Venturi, and Ivan Visconti. Continuously non-malleable codes in the split-state model from minimal assumptions. In *Advances in Cryptology - CRYPTO*, pages 608–639, 2018.

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

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS*, pages 60:1–60:15, 2019.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *Advances in Cryptology - CRYPTO*, pages 57–74, 2008.

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

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

[PW10] Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from subexponential one-way functions. In *Advances in Cryptology - EUROCRYPT*, pages 638–655, 2010.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto, 1996. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

[Sho06] Victor Shoup. *A computational introduction to number theory and algebra.* Cambridge University Press, 2006.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 531–540, 2010.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In *Advances in Cryptology - EUROCRYPT*, pages 379–407, 2019.