Non-Malleable Codes for Bounded Parallel-Time Tampering

Dana Dachman-Soled[1]*, Ilan Komargodski[2], and Rafael Pass[3]**

[1] University of Maryland
danadach@ece.umd.edu
[2] Hebrew University of Jerusalem and NTT Research
ilank@cs.huji.ac.il
[3] Cornell Tech
rafael@cs.cornell.edu

Abstract. Non-malleable codes allow one to encode data in such a way that once a codeword is being tampered with, the modified codeword is either an encoding of the original message, or a completely unrelated one. Since the introduction of this notion by Dziembowski, Pietrzak, and Wichs (ICS '10 and J. ACM '18), there has been a large body of works 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, no code which is non-malleable for *bounded* polynomial size attackers is 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 parallel time. This is an even larger class than all bounded polynomial size functions. In particular, this class includes 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 can obtain (sub-)exponentially small error.

* Supported in part by NSF grants #CNS-1933033, #CNS-1453045 (CAREER), and by financial assistance awards 70NANB15H328 and 70NANB19H126 from the U.S. Department of Commerce, National Institute of Standards and Technology.

** Supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, AFOSR Award FA9550-18-1-0267, and a JP Morgan Faculty Award. This material is based upon work supported by DARPA under Agreement No. HR00110C0086 and 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 DARPA, 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 [57,58,23,26] and additive combinatorics [3,2]) as well as in cryptography (e.g., non-malleable encryption and commitments [32,31,47]).

In the paper that introduced non-malleable codes, Dziembowski, Pietrzak, and Wichs [37,38], 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.[4] 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. [38] (see also [27] 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. [42] 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. [41] constructed non-malleable codes secure against space-bounded tampering. Ball et al. [7] constructed a non-malleable code secure against bounded depth circuits

[4] 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.

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 [25,8,5,11] (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 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. [8] 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. [6] 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 $\mathbf{P}$-certificates[5] 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 [67]), 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

[5] These are "succinct" one-message arguments for languages in $\mathbf{P}$, with proof length which is a fixed polynomial, independent of the time it takes to decide the language [28].

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 [77], a non-interactive non-malleable commitment [52,63,20,49], and a non-interactive SPS (super-polynomial-time simulatable) zero-knowledge protocol [14,20] (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 (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 [6] 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. [77] and we show how to further use results of Bitansky and Lin [20] and Lin et al. [63] to instantiate the required non-malleable commitment and zero-knowledge protocol. Thus, assuming the repeated squaring assumption [77] (i.e., there is no way to significantly speed-up repeated squarings in a hidden-order group), a keyless multi-collision resistant hash function [19] (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 $c \in \mathbb{N}$),[6] as well as other standard assumptions, we obtain the following theorem.

Theorem 2 (Informal). *Assume a keyless multi-collision resistant hash function, the repeated squaring assumption, an injective one-way function, and non-interactive witness-indistinguishable proofs,*[7] *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*.[8] In other words, under

[6] 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 satisfies by a random function.

[7] Non-interactive witness-indistinguishable proofs are known to exist based on various assumptions: trapdoor permutations and a particular derandomization-type assumption [13], cryptographic bilinear maps [48], or indistinguishability obfuscation and one-way permutations [21].

[8] 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

the same assumptions as in Theorems 1 and 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.

Related followup or concurrent work. We mention the following related followup or concurrent work [50,40,15,16]. Katz et al. [50] construct non-malleable non-interactive timed-commitments relying in the security proof on the algebraic group model [44] and on trusted setup. Ephraim et al. [40] focus on efficiency and applications; specifically, they give a more efficient construction than the one given in this work (which is proven secure in the auxiliary-input random oracle model) and further show how to use it to obtain desirable cryptographic protocols such as fair multiparty coin flipping. Lastly, Baum et al. [15,16] construct UC-secure time-lock puzzles while relying on a programmable random oracle, which they show to be necessary. Most recently, Ball et al. [10] (see [4, Theorem 32]) showed that the derandomization assumption that there is a language that can be computed in exponential deterministic time and requires exponential size nondeterministic circuits implies explicit codes for bounded polynomial size circuits (without any setup assumptions) with inverse polynomial security. The construction of [10] requires an encoding procedure that runs in time larger than the a priori polynomial upper bound on the size of the tampering circuit.

1.2 Related Work

Since the work of Dziembowski, Pietrzak, and Wichs [37,38] which introduced non-malleable codes, there has been a quite a significant amount of works on this subject in various different directions (for example, [1,53,3,2,23,34,22,25,59,60] 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 [37] presented a construction of non-malleable codes against bit-wise tampering functions. Also, Liu and Lysyanskaya [65] 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 [77] (following May's work [66] on timed-release cryptography), and they have been used quite extensively studied since. The most popular instantiation relies on the *repeated squaring assumption* that postulates that $\overline{T}$ repeated squarings mod N,

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.

where $N = pq$ is a product of two secret primes, require "roughly" $\overline{T}$ parallel time/depth. Bitansky et al. [18] 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 [35,78]). 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.

Non-malleable commitments. Non-malleable commitments, introduced by Dolev, Dwork and Naor [36], 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 [12,74,73,64,71,61,75,79,45,62,46,47,29,30,51,63,52,49].

We need a *non-interactive* (i.e., one-message) non-malleable commitment, of which relatively few constructions are known. Pandey et al. [71] 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 [68]) assumption. Lin et al. [63] 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 [20] 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,[9] or using a new assumption regarding sub-exponentially secure one-way functions admitting some strong form of hardness amplification. Most recently, Kalai and Khurana [49] 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 [20] and Lin et al. [63] both of which rely on time-lock puzzles. Various properties of their non-malleable commitments will be crucial for our construction.

[9] Actually, Bitansky and Lin [20] formulate an assumption about incompressible functions which is implied by keyless multi-collision resistant hash functions.

One-message SPS zero-knowledge. This is a one-message proof system for every language in $\mathbf{NP}$ in the plain model and without any setup assumptions that satisfies a relaxed notion of zero-knowledge referred to as super-polynomial-time simulation (SPS) zero-knowledge [72]. This concept was introduced by Barak and Pass [14] who also gave a construction assuming a keyless collision resistance hash function,[10] non-interactive commitments, and NIWI proofs, all with sub-exponential hardness. Their construction however satisfies soundness only against uniform attackers. Bitansky and Lin [20] showed how to overcome this limitation using keyless *multi*-collision resistant hash functions,[11] 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 [47,22]. In the reverse direction, some works used ideas from the (vast) literature on non-malleable commitments to get new non-malleable codes [24,70,6]. 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. [9] 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 [63] that we have already mentioned and the current work, this assumption was also used in several

[10] Actually, Barak and Pass [14] formulate an assumption regarding the existence of a language in $\mathbf{P}$ which is 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.

[11] See Footnote 9.

constructions of verifiable delay functions [76,80,39]. 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 *uniform* 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 [55,17,19,54] is a relaxation of collision resistance where the goal of the attacker is to find a collection of *many* inputs (rather than just two) to a random function in the family that collide. The keyless version, introduced by [19], 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., [8,6]), we follow the Naor-Yung [69] 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.

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 [77] 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

```
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, π):
  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*).

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 many sequential resources). 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 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. [63] and Bitansky and Lin [20] 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 uniform tampering functions and a one-message SPS zero-knowledge proof which satisfies uniform soundness. For the commitment scheme we will use the one of Lin, Pass, and Soni [63] and for the zero-knowledge we will use the one of Barak and Pass [14]. We remark again that while the scheme of Lin et al. [63] 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, naturally, 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. 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![12] 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 depth. Namely, we think of algorithms as parallelizable machines and measure their complexity by counting their total size and parallel time (size and time/depth, in short). 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 and depth.
- S_{ZK} (zero-knowledge security): for all $2^{\lambda^{c_1}}$ size (and depth) attackers.
- S_{Sim} (ZK simulator complexity): in $2^{\lambda^{c_2}}$ size but fixed polynomial depth.
- S_{NMCom} (non-malleability): for all $2^{\lambda^{c_3}}$ size attackers with arbitrary polynomial depth.

With this 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 [14] readily satisfies the above properties if it is sub-exponentially hard. To get the required non-malleable commitment, we need to slightly adjust the scheme of Lin et al. [63] (as they did not consider such tampering functions), but the changes are relatively minor.

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

[12] This kind of zero-knowledge simulation is known as *strong* super-polynomial simulation. Recently, Khurana and Sahai [52] managed to obtain it in two rounds, but we need a non-interactive scheme.

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 there is another brute-force extractor that requires exponential size but only fixed polynomial time. 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, we switch to this alternate extractor. Now, 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. [63] (which is uniformly non-malleable) with the one of Bitansky and Lin [20], the challenge stems from finding an appropriate non-uniform analogue for the uniformly sound one-message zero-knowledge scheme of Barak and Pass [14]. 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 [20] 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

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.

Additional necessary but standard preliminaries appear in the full version [33]. There, the reader can find standard notation that we use and standard definitions related to non-malleable commitments (following Lin, Pass, and Soni [63]), one-message zero-knowledge proofs (following Barak and Pass [14]), and time-lock puzzles (following Rivest, Shamir, and Wagner [77]).

4 Definition of Non-Malleable Codes

In this section we give our definition of non-malleable codes. Our definition follows closely the definition of [8]. 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 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:*

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

where the randomness of the above experiment comes from the randomness of NMCode.E.

Definition 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

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

Medium non-malleability. We next define a different notion of non-malleability, referred to as *medium* non-malleability, which implies the one above (Definition 2) but is slightly easier to work with [6,56]. 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 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*

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

where the tampering experiment (whose randomness comes from the randomness of NMCode.E*) is defined as follows:*

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

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 3. *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 $\log t$ 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 4. *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* $(S^{\mathsf{NMCom}}, T^{\mathsf{NMCom}})$*-hiding,* $(S^{\mathsf{NMCom}}_{\mathsf{Ext}_1}, T^{\mathsf{NMCom}}_{\mathsf{Ext}_1})$*-extractable via* $\mathsf{NMCom.Ext}_1$ *and* $(S^{\mathsf{NMCom}}_{\mathsf{Ext}_2}, T^{\mathsf{NMCom}}_{\mathsf{Ext}_2})$*-extractable via* $\mathsf{NMCom.Ext}_2$*, and* $(S^{\mathsf{NMCom}}_{NM}, T^{\mathsf{NMCom}}_{NM})$*-non-malleable for all polynomial functions* T^{NMCom} *and* T^{NMCom}_{NM}*, and where* $S^{\mathsf{NMCom}}(\lambda) = 2^{\lambda^{\eta''}}$ *for an appropriate constant* η''*,* $S^{\mathsf{NMCom}}_{\mathsf{Ext}_1}(\lambda) = T^{\mathsf{NMCom}}_{\mathsf{Ext}_1}(\lambda) = 2^{\log^2 \lambda}$*,* $S^{\mathsf{NMCom}}_{\mathsf{Ext}_2}(\lambda) = 2^{2\lambda}$*,* $T^{\mathsf{NMCom}}_{\mathsf{Ext}_2}(\lambda) = \lambda^3$*, and* $S^{\mathsf{NMCom}}_{NM} = 2^\lambda$.

Theorem 5. *Assume that there is a keyless collision resistant hash function, the repeated squaring assumption, NIWI proof for all* $\mathbf{NP}$*, and injective one-way functions, all with sub-exponential hardness. Then, there exists a non-interactive commitment which is* $(S^{\mathsf{NMCom}}, T^{\mathsf{NMCom}})$*-hiding,* $(S^{\mathsf{NMCom}}_{\mathsf{Ext}_1}, T^{\mathsf{NMCom}}_{\mathsf{Ext}_1})$*-extractable via* $\mathsf{NMCom.Ext}_1$ *and* $(S^{\mathsf{NMCom}}_{\mathsf{Ext}_2}, T^{\mathsf{NMCom}}_{\mathsf{Ext}_2})$*-extractable via* $\mathsf{NMCom.Ext}_2$*, and* $(S^{\mathsf{NMCom}}_{NM}, T^{\mathsf{NMCom}}_{NM}, \kappa^{\mathsf{NMCom}}_{NM})$*-non-malleable for all polynomial functions* T^{NMCom} *and* T^{NMCom}_{NM}*, and where* $S^{\mathsf{NMCom}}(\lambda) = 2^{\lambda^{\eta''}}$ *for an appropriate constant* η''*,* $S^{\mathsf{NMCom}}_{\mathsf{Ext}_1}(\lambda) = T^{\mathsf{NMCom}}_{\mathsf{Ext}_1}(\lambda) = 2^{\log^2 \lambda}$*,* $S^{\mathsf{NMCom}}_{\mathsf{Ext}_2}(\lambda) = 2^{2\lambda}$*,* $T^{\mathsf{NMCom}}_{\mathsf{Ext}_2}(\lambda) = \lambda^3$*,* $S^{\mathsf{NMCom}}_{NM} = 2^{\lambda}$*, and* $\kappa^{\mathsf{NMCom}}_{NM} = 0$.

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 uniform attackers but we are using a keyless (plain) collision resistant hash.

We need a one-message non-malleable tag-based commitment scheme which is hiding for all (non-uniform) polynomial-size distinguishers, extractable either in size and time $2^{\log^2 \lambda}$ or in 2^{λ}-size and λ^3-time, and non-malleable for all exponential size and polynomial time tampering functions.

The uniform scheme. To get the scheme satisfying the properties listed in Theorem 5 we use the scheme of Lin et al. [63]. 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 $\{\mathsf{Com}^s_i\}_{i \in [\gamma]}$ such that Com^s_{i+1} is harder than Com^s_i for all $1 \leq i \leq \gamma - 1$ in the *axis of size*. Namely, using size which is sufficient to extract from Com^s_i it is still hard to break Com^s_{i+1}. Also, the extraction procedure is essentially a brute force algorithm that "tries all option" and so it is highly parallelizable and requires fixed parallel time (depth).

A similar trick is performed using time-lock puzzles. They are used to obtain a family of γ commitment schemes $\{\mathsf{Com}^t_i\}_{i \in [\gamma]}$ such that Com^t_{i+1} is harder than Com^t_i for all $1 \leq i \leq \gamma - 1$ in the *axis of time*. Namely, in time which is sufficient to extract from Com^t_i it is still hard to break Com^t_{i+1}. 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,

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

The hiding and non-malleability proofs are the same as in [63]. Hiding is immediate from hiding of the two underlying commitments, and we sketch the main idea behind the proof of non-malleability next. Non-malleability holds by considering two cases. First, if the left tag i is smaller than the right tag j, the Com^t_j commitment on the right remains hiding for attackers of size and time enough for extracting from both Com^t_i and Com^s_j. 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^s_i commitment on the left remains hiding for attackers of size and time enough for extracting from both Com^s_j and $\mathsf{Com}^t_{\gamma-j}$. Thus, the left committed value remains hidden, while the right is extracted.

Lastly, we explain how to implement the two extraction procedures that we need. Recall that we need one extraction procedure that works in quasi-linear size and time and another procedure that works in exponential size and fixed polynomial time. The former is implemented by extracting from the right commitment (by breaking the time-lock puzzles) and the latter is implemented by breaking the left commitment (by checking in parallel all possible openings).

Of course, the above construction is not the final construction of [63] 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 uniform attackers). We refer to [63] for the precise details.

The non-uniform scheme. To get the scheme satisfying the properties listed in Theorem 4 we use the scheme of Bitansky and Lin [20] (which in turns is based on the scheme of [63]). Here, they present a new tag-amplification technique, inspired by a interactive tag-amplification technique of Khurana and Sahai [52], where they make it non-interactive using their one-message zero-knowledge protocol (which is based on keyless multi-collision resistant hash functions).

For our purposes, the details of this transformation are not very relevant—the only thing that is important is the structure of thier final commitment. Indeed, it consists of the same time or space hard commitments of [63] (along with various proofs). These are extractable in the same manner, either in quasi-linear time or in exponential size and polynomial time.

5.3 One-Message Zero-Knowledge

Theorem 6. *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 $(S_D, S_{\mathsf{Sim}}, T_{\mathsf{Sim}})$-zero-knowledge for all polynomials $S_P(\lambda)$, and where $K \in \mathsf{poly}(\lambda)$ is a fixed polynomial, $S_D(\lambda) = 2^{\lambda^\eta}$ and $S_{\mathsf{Sim}}(\lambda) = 2^{\lambda^{\eta'}}$ for some constants $\eta, \eta' \in (0,1)$, and $T_{\mathsf{Sim}}(\lambda) = \lambda^2$.*

Theorem 7. *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* $(S_D, S_{\mathsf{Sim}}, T_{\mathsf{Sim}})$*-zero-knowledge for all polynomial* $S_P(\lambda)$*, and where* $\kappa(\lambda) = 0$*,* $S_D(\lambda) = 2^{\lambda^{\eta}}$ *and* $S_{\mathsf{Sim}}(\lambda) = 2^{\lambda^{\eta'}}$ *for some constants* $\eta, \eta' \in (0,1)$*, and* $T_{\mathsf{Sim}}(\lambda) = \lambda^2$.

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 [14] 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 [43] 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 is a parallel machine that can find such a collision by brute force using a super-polynomial size procedure which has only fixed polynomial time (it tries all possibilities in parallel). Their construction gives Theorem 7.

Bitansky and Lin [20] 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 [19]. Their construction gives Theorem 6.

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 (Theorem 3) $\mathsf{TL} = (\mathsf{TL.Gen}, \mathsf{TL.Sol})$ which, 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 $2^{3\lambda}$.
   More precisely, for a difficulty parameter t, it is $(S^{\mathsf{TL}}, \epsilon)$-hard for a fixed $\epsilon \in (0,1)$ and for $S^{\mathsf{TL}}(\lambda) = 2^{3\lambda}$.
2. A one-message SPS zero-knowledge argument system (Theorem 6) $\mathsf{ZK} = (\mathsf{ZK.P}, \mathsf{ZK.V})$ which is weakly sound w.r.t. all (non-uniform) polynomial-size attackers, there is a (uniform) simulator that requires sub-exponential size and fixed polynomial time, and zero-knowledge holds w.r.t. sub-exponential size adversaries.

More precisely, it is $(S_P^{\mathsf{ZK}}, K^{\mathsf{ZK}})$-sound and $(S_D^{\mathsf{ZK}}, S_{\mathsf{Sim}}^{\mathsf{ZK}}, T_{\mathsf{Sim}}^{\mathsf{ZK}})$-zero-knowledge for all polynomial functions S_P^{ZK} and where $K^{\mathsf{ZK}} \in \mathsf{poly}(\lambda)$ is a fixed polynomial, $S_D^{\mathsf{ZK}}(\lambda) = 2^{\lambda^\eta}$, $S_{\mathsf{Sim}}^{\mathsf{ZK}}(\lambda) = 2^{\lambda^{\eta'}}$, and $T_{\mathsf{Sim}}^{\mathsf{ZK}}(\lambda) = \lambda^2$.

3. A one-message non-malleable tag-based commitment scheme (Theorem 4) $\mathsf{NMCom} = (\mathsf{NMCom.C}, \mathsf{NMCom.O})$ which is hiding for all (non-uniform) polynomial-size distinguishers, extractable either in size and time $2^{\log^2 \lambda}$ or in 2^λ size and λ^3 time, and non-malleable for all exponential size and polynomial time tampering functions.
   More precisely, it is $(S^{\mathsf{NMCom}}, T^{\mathsf{NMCom}})$-hiding, $(S_{\mathsf{Ext}_1}^{\mathsf{NMCom}}, T_{\mathsf{Ext}_1}^{\mathsf{NMCom}})$-extractable via $\mathsf{NMCom.Ext}_1$ and $(S_{\mathsf{Ext}_2}^{\mathsf{NMCom}}, T_{\mathsf{Ext}_2}^{\mathsf{NMCom}})$-extractable via $\mathsf{NMCom.Ext}_2$, and $(S_{NM}^{\mathsf{NMCom}}, T_{NM}^{\mathsf{NMCom}})$-non-malleable for all polynomial functions T^{NMCom} and T_{NM}^{NMCom}, and where $S^{\mathsf{NMCom}}(\lambda) = 2^{\lambda^{\eta''}}$ where $\eta'' > \eta'$, $S_{\mathsf{Ext}_1}^{\mathsf{NMCom}}(\lambda) = T_{\mathsf{Ext}_1}^{\mathsf{NMCom}}(\lambda) = 2^{\log^2 \lambda}$, $S_{\mathsf{Ext}_2}^{\mathsf{NMCom}}(\lambda) = 2^{2\lambda}$, $T_{\mathsf{Ext}_2}^{\mathsf{NMCom}}(\lambda) = \lambda^3$, and $S_{NM}^{\mathsf{NMCom}} = 2^\lambda$.
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 8. *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 and asserts 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

```
Algorithm 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 ZK proof $\pi \leftarrow \mathsf{ZK.P}(\cdot, \cdot, 1^\lambda)$ for the relation $\mathcal{R}_{\mathsf{u}}$ from Algorithm 4 using
   $(\mathsf{vk}, Z, c)$ as the instance and $(r_{\mathsf{TL}}, m, r_{\mathsf{NMCom}})$ 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 NMCode.E.

```
Algorithm NMCode.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 proof $\pi$:
       $\mathsf{ZK.V}((\mathsf{vk}, Z, c), \pi) \overset{?}{=} 1$.
3. If both accept, output $\mathsf{TL.Sol}(Z)$. Otherwise, output $0^\lambda$.
```

Algorithm 3: The decoding procedure NMCode.D.

size of the solver is 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.

Organization. The proof of Theorem 8 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 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 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 $\mathcal{R}_{\mathsf{u}}\left((\mathsf{vk}, Z, c), (r_{\mathsf{TL}}, m, r_{\mathsf{NMCom}})\right)$:

- Instance: a verification key vk, a puzzle generated by $\mathsf{TL.Gen}(1^\lambda, \overline{T}^{2/\epsilon}, m)$, and a commitment c.
- Witness: a string $t_{\mathsf{TL}} \in \{0,1\}^*$, a string $m \in \{0,1\}^\lambda$, and a string $r_{\mathsf{NMCom}} \in \{0,1\}^*$.
- 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 (and time) in its input size.

Algorithm 5: The procedure g.

Claim 9 *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

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

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 $\{\mathsf{MedTamper}^{\mathsf{NMCode}_0}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\text{`}\mathbb{N}}$ 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. Some of the proofs are deferred to the full version [33].

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. 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 simulator $\mathsf{Sim}$ to simulate a proof for the relation $\mathcal{R}_\mathsf{u}$ using $(\mathsf{vk}, Z, c)$ 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 10 *It holds that*

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

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 NMCode_2.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 simulator Sim to simulate a proof for the relation $\mathcal{R}_\mathsf{u}$ using $(\mathsf{vk}, Z, c)$ 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 11 *It holds that*

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

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 $\mathsf{NMCom.Ext}_1$ *unless* the (tampered) proof is one of a fix set of "bad" proofs.

Concretely, recall that by weak-soundness of ZK, every algorithm (and in particular $\mathcal{A}$) can come up with some small bounded number of proofs that are verified yet are for false statements. If the proof on the right size is one of those proofs, we will output a hard coded value instead of trying to extract the value from the commitment.

More precisely, the adversary $\mathcal{A}$ can find a set $\mathcal{Z}'$ (that depends on the adversary and the hybrid) of size at most $K \triangleq K^{\mathsf{ZK}}(|\mathcal{A}_\lambda| + O(1)) \in \mathsf{poly}(\lambda)$ of proofs that are verified yet are for false statements. We denote by $\mathcal{Z}$ the augmented set of proofs (for false statements) together with the instance and with the values underlying the time-lock puzzle in each such statements. Namely, $\mathcal{Z}$ is a set that consists of tuples of the form $(\pi, \mathsf{vk}, Z, c, \tilde{m})$, where π is a proof from $\mathcal{Z}'$ for the instance (vk, Z, c) and $\tilde{m}$ 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 $\{\mathsf{MedTamper}^{\mathsf{NMCode}_3}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}$.

```
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 $\mathsf{ZK}$ 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 $\pi$ 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 $\mathsf{NMCom.Ext}_1(c)$
   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 12 *It holds that*

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

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 the extractor $\mathsf{NMCom.Ext}_2$ instead of $\mathsf{NMCom.Ext}_1$. Namely, we execute the experiment $\{\mathsf{MedTamper}^{\mathsf{NMCode}_4}_{\mathcal{A},m_1}(\lambda)\}_{\lambda\in\mathbb{N}}$. 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 $\{\mathsf{MedTamper}^{\mathsf{NMCode}_4}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}$.

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

1. Verify the signature $\sigma$:
       $\mathsf{Sig.V}(\mathsf{vk}, (Z, c, \pi), \sigma) \overset{?}{=} 1.$
2. Verify the $\mathsf{ZK}$ 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 $\pi$ 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 $\mathsf{NMCom.Ext}_2(c)$
   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 $\mathcal{H}_4(\lambda)$.

Claim 13 *It holds that* $\{\mathsf{MedTamper}^{\mathsf{NMCode}_3}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}$ *and* $\{\mathsf{MedTamper}^{\mathsf{NMCode}_4}_{\mathcal{A},m_0}(\lambda)\}_{\lambda\in\mathbb{N}}$ *are identically distributed.*

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 14 *It holds that*

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

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. [6].

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 (Theorem 3) $\mathsf{TL} = (\mathsf{TL.Gen}, \mathsf{TL.Sol})$ which, 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 $2^{3\lambda}$.
   More precisely, for a difficulty parameter t, it is $(S^{\mathsf{TL}}, \epsilon)$-hard for a fixed $\epsilon \in (0,1)$ and for $S^{\mathsf{TL}}(\lambda) = 2^{3\lambda}$.
2. A one-message zero-knowledge argument system (Theorem 7) $\mathsf{ZK} = (\mathsf{ZK.P}, \mathsf{ZK.V})$ which is sound w.r.t. all uniform polynomial-size attackers, there is a (uniform) simulator that requires sub-exponential size and fixed polynomial time, and zero-knowledge holds w.r.t. sub-exponential size adversaries.
   More precisely, it is $(S^{\mathsf{ZK}}_P, \kappa^{\mathsf{ZK}})$-sound and $(S^{\mathsf{ZK}}_D, S^{\mathsf{ZK}}_{\mathsf{Sim}}, T^{\mathsf{ZK}}_{\mathsf{Sim}})$-zero-knowledge for all polynomial functions S^{ZK}_P and where $\kappa^{\mathsf{ZK}} = 0$, $S^{\mathsf{ZK}}_D(\lambda) = 2^{\lambda^\eta}$, $S^{\mathsf{ZK}}_{\mathsf{Sim}}(\lambda) = 2^{\lambda^{\eta'}}$, and $T^{\mathsf{ZK}}_{\mathsf{Sim}}(\lambda) = \lambda^2$.
3. A one-message non-malleable tag-based commitment scheme (Theorem 5) $\mathsf{NMCom} = (\mathsf{NMCom.C}, \mathsf{NMCom.O})$ which is hiding for all (non-uniform) polynomial-size distinguishers, extractable either in size and time $2^{\log^2}\lambda$ or in 2^λ size and λ^3 time, and non-malleable for all *uniform* exponential size and polynomial time tampering functions.
   More precisely, it is $(S^{\mathsf{NMCom}}, T^{\mathsf{NMCom}})$-hiding, $(S^{\mathsf{NMCom}}_{\mathsf{Ext}_1}, T^{\mathsf{NMCom}}_{\mathsf{Ext}_1})$-extractable via $\mathsf{NMCom.Ext}_1$ and $(S^{\mathsf{NMCom}}_{\mathsf{Ext}_2}, T^{\mathsf{NMCom}}_{\mathsf{Ext}_2})$-extractable via $\mathsf{NMCom.Ext}_2$, and $(S^{\mathsf{NMCom}}_{NM}, T^{\mathsf{NMCom}}_{NM}, \kappa^{\mathsf{NMCom}}_{NM})$-non-malleable for all polynomial functions T^{NMCom} and T^{NMCom}_{NM}, and where $S^{\mathsf{NMCom}}(\lambda) = 2^{\lambda^{\eta''}}$ for $\eta'' > \eta'$, $S^{\mathsf{NMCom}}_{\mathsf{Ext}_1}(\lambda) = T^{\mathsf{NMCom}}_{\mathsf{Ext}_1}(\lambda) = 2^{\log^2\lambda}$, $S^{\mathsf{NMCom}}_{\mathsf{Ext}_2}(\lambda) = 2^{2\lambda}$, $T^{\mathsf{NMCom}}_{\mathsf{Ext}_2}(\lambda) = \lambda^3$, $S^{\mathsf{NMCom}}_{NM} = 2^\lambda$, and $\kappa^{\mathsf{NMCom}}_{NM} = 0$.
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. We refer to the full version [33] for details.

References

1. Aggarwal, D., Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Optimal computational split-state non-malleable codes. In: TCC. pp. 393–417 (2016)
2. Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: Non-malleable reductions and applications. In: STOC. pp. 459–468 (2015)
3. Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. SIAM J. Comput. **47**(2), 524–546 (2018)
4. Ball, M.: On Resilience to Computable Tampering. Ph.D. thesis, Columbia University (2021), https://academiccommons.columbia.edu/doi/10.7916/d8-debr-bw49
5. Ball, M., Dachman-Soled, D., Guo, S., Malkin, T., Tan, L.: Non-malleable codes for small-depth circuits. In: FOCS. pp. 826–837 (2018)
6. Ball, M., Dachman-Soled, D., Kulkarni, M., Lin, H., Malkin, T.: Non-malleable codes against bounded polynomial time tampering. In: Advances in Cryptology - EUROCRYPT. pp. 501–530 (2019)
7. Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes for bounded depth, bounded fan-in circuits. In: Advances in Cryptology - EUROCRYPT. pp. 881–908 (2016)
8. Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes from average-case hardness: AC^0, decision trees, and streaming space-bounded tampering. In: Advances in Cryptology - EUROCRYPT. pp. 618–650 (2018)
9. Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Limits to non-malleability. In: ITCS. pp. 80:1–80:32 (2020)
10. Ball, M., Dachman-Soled, D., Loss, J.: Explicit non-malleable codes for polynomial size circuit tampering, (unpublished manuscript)
11. Ball, M., Guo, S., Wichs, D.: Non-malleable codes for decision trees. In: Advances in Cryptology - CRYPTO. pp. 413–434 (2019)
12. Barak, B.: Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: FOCS. pp. 345–355 (2002)
13. Barak, B., Ong, S.J., Vadhan, S.P.: Derandomization in cryptography. SIAM J. Comput. **37**(2), 380–400 (2007)
14. Barak, B., Pass, R.: On the possibility of one-message weak zero-knowledge. In: TCC. pp. 121–132 (2004)
15. Baum, C., David, B., Dowsley, R., Nielsen, J.B., Oechsner, S.: Craft: Composable randomness and almost fairness from time. Cryptology ePrint Archive, Report 2020/784 (2020)
16. Baum, C., David, B., Dowsley, R., Nielsen, J.B., Oechsner, S.: TARDIS: A foundation of time-lock puzzles in UC. In: Advances in Cryptology - EUROCRYPT. pp. 429–459 (2021)

17. Berman, I., Degwekar, A., Rothblum, R.D., Vasudevan, P.N.: Multi-collision resistant hash functions and their applications. In: Advances in Cryptology - EUROCRYPT. pp. 133–161 (2018)
18. Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. In: ITCS. pp. 345–356 (2016)
19. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: a paradigm for keyless hash functions. In: STOC. pp. 671–684 (2018)
20. Bitansky, N., Lin, H.: One-message zero knowledge and non-malleable commitments. In: TCC. pp. 209–234 (2018)
21. Bitansky, N., Paneth, O.: Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In: TCC. pp. 401–427 (2015)
22. Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. In: ICALP. pp. 31:1–31:14 (2016)
23. Chattopadhyay, E., Goyal, V., Li, X.: Non-malleable extractors and codes, with their many tampered extensions. Electronic Colloquium on Computational Complexity (ECCC) **22**, 75 (2015)
24. Chattopadhyay, E., Goyal, V., Li, X.: Non-malleable extractors and codes, with their many tampered extensions. In: STOC. pp. 285–298 (2016)
25. Chattopadhyay, E., Li, X.: Non-malleable codes and extractors for small-depth circuits, and affine functions. In: STOC. pp. 1171–1184 (2017)
26. Chattopadhyay, E., Zuckerman, D.: Explicit two-source extractors and resilient functions. In: STOC. pp. 670–683 (2016)
27. Cheraghchi, M., Guruswami, V.: Capacity of non-malleable codes. IEEE Trans. Information Theory **62**(3), 1097–1118 (2016)
28. Chung, K., Lin, H., Pass, R.: Constant-round concurrent zero knowledge from P-certificates. In: FOCS. pp. 50–59 (2013)
29. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Concurrent non-malleable commitments (and more) in 3 rounds. In: Advances in Cryptology - CRYPTO. pp. 270–299 (2016)
30. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Four-round concurrent non-malleable commitments from one-way functions. In: Advances in Cryptology - CRYPTO. pp. 127–157 (2017)
31. Coretti, S., Dodis, Y., Tackmann, B., Venturi, D.: Non-malleable encryption: Simpler, shorter, stronger. In: TCC. pp. 306–335 (2016)
32. Coretti, S., Maurer, U., Tackmann, B., Venturi, D.: From single-bit to multi-bit public-key encryption via non-malleable codes. In: TCC. pp. 532–560 (2015)
33. Dachman-Soled, D., Komargodski, I., Pass, R.: Non-malleable codes for bounded polynomial depth tampering. IACR Cryptol. ePrint Arch. **2020**, 776 (2020)
34. Dachman-Soled, D., Liu, F., Shi, E., Zhou, H.: Locally decodable and updatable non-malleable codes and their applications. In: TCC. pp. 427–450 (2015)
35. Dixon, J.D.: Asymptotically fast factorization of integers. Mathematics of computation **36**(153), 255–260 (1981)
36. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: STOC. pp. 542–552 (1991)
37. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: ICS. pp. 434–452 (2010)
38. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. J. ACM **65**(4), 20:1–20:32 (2018)
39. Ephraim, N., Freitag, C., Komargodski, I., Pass, R.: Continuous verifiable delay functions. In: Advances in Cryptology - EUROCRYPT. pp. 125–154 (2020)

40. Ephraim, N., Freitag, C., Komargodski, I., Pass, R.: Non-malleable time-lock puzzles and applications. IACR Cryptol. ePrint Arch. **2020**, 779 (2020)
41. Faust, S., Hostáková, K., Mukherjee, P., Venturi, D.: Non-malleable codes for space-bounded tampering. In: Advances in Cryptology - CRYPTO. pp. 95–126 (2017)
42. Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: Efficient non-malleable codes and key derivation for poly-size tampering circuits. IEEE Trans. Information Theory **62**(12), 7179–7194 (2016)
43. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: FOCS. pp. 308–317 (1990)
44. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Advances in Cryptology - CRYPTO 2018. pp. 33–62 (2018)
45. Goyal, V.: Constant round non-malleable protocols using one way functions. In: Fortnow, L., Vadhan, S.P. (eds.) STOC. pp. 695–704 (2011)
46. Goyal, V., Lee, C., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: A black-box approach. In: FOCS. pp. 51–60 (2012)
47. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: STOC. pp. 1128–1141 (2016)
48. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Advances in Cryptology - CRYPTO. pp. 97–111 (2006)
49. Kalai, Y.T., Khurana, D.: Non-interactive non-malleability from quantum supremacy. In: Advances in Cryptology - CRYPTO. pp. 552–582 (2019)
50. Katz, J., Loss, J., Xu, J.: On the security of time-lock puzzles and timed commitments. In: TCC. pp. 390–413 (2020)
51. Khurana, D.: Round optimal concurrent non-malleability from polynomial hardness. In: TCC. pp. 139–171 (2017)
52. Khurana, D., Sahai, A.: How to achieve non-malleability in one or two rounds. In: FOCS. pp. 564–575 (2017)
53. Kiayias, A., Liu, F., Tselekounis, Y.: Practical non-malleable codes from l-more extractable hash functions. In: CCS. pp. 1317–1328 (2016)
54. Komargodski, I., Naor, M., Yogev, E.: Collision resistant hashing for paranoids: Dealing with multiple collisions. In: Advances in Cryptology - EUROCRYPT 2018. pp. 162–194 (2018)
55. Komargodski, I., Naor, M., Yogev, E.: White-box vs. black-box complexity of search problems: Ramsey and graph property testing. J. ACM **66**(5), 34:1–34:28 (2019)
56. Kulkarni, M.R.: Extending the Applicability of Non-Malleable Codes. Ph.D. thesis, The University of Maryland (2019), https://drum.lib.umd.edu/bitstream/handle/1903/25179/Kulkarni_umd_0117E_20306.pdf?sequence=2
57. Li, X.: Non-malleable extractors, two-source extractors and privacy amplification. In: 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS. pp. 688–697 (2012)
58. Li, X.: New independent source extractors with exponential improvement. In: STOC. pp. 783–792 (2013)
59. Li, X.: Improved non-malleable extractors, non-malleable codes and independent source extractors. In: STOC. pp. 1144–1156. ACM (2017)
60. Li, X.: Non-malleable extractors and non-malleable codes: Partially optimal constructions. arXiv preprint arXiv:1804.04005 (2018)
61. Lin, H., Pass, R.: Non-malleability amplification. In: STOC. pp. 189–198 (2009)
62. Lin, H., Pass, R.: Constant-round non-malleable commitments from any one-way function. In: STOC. pp. 705–714 (2011)

63. Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In: FOCS. pp. 576–587 (2017)
64. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent non-malleable commitments from any one-way function. In: TCC. pp. 571–588 (2008)
65. Liu, F., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Advances in Cryptology - CRYPTO. pp. 517–532 (2012)
66. May, T.: Timed-release crypto (1992)
67. Micali, S.: Computationally sound proofs. SIAM Journal on Computing **30**(4), 1253–1298 (2000)
68. Naor, M.: On cryptographic assumptions and challenges. In: Advances in Cryptology - CRYPTO. pp. 96–109 (2003)
69. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC. pp. 427–437 (1990)
70. Ostrovsky, R., Persiano, G., Venturi, D., Visconti, I.: Continuously non-malleable codes in the split-state model from minimal assumptions. In: Advances in Cryptology - CRYPTO. pp. 608–639 (2018)
71. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: Advances in Cryptology - CRYPTO. pp. 57–74 (2008)
72. Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: Advances in Cryptology - EUROCRYPT. pp. 160–176 (2003)
73. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: FOCS. pp. 563–572 (2005)
74. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: STOC. pp. 533–542 (2005)
75. Pass, R., Wee, H.: Constant-round non-malleable commitments from sub-exponential one-way functions. In: Advances in Cryptology - EUROCRYPT. pp. 638–655 (2010)
76. Pietrzak, K.: Simple verifiable delay functions. In: ITCS. pp. 60:1–60:15 (2019)
77. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto (1996), technical report, Massachusetts Institute of Technology, Cambridge, MA, USA
78. Shoup, V.: A computational introduction to number theory and algebra. Cambridge University Press (2006)
79. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: FOCS. pp. 531–540 (2010)
80. Wesolowski, B.: Efficient verifiable delay functions. In: Advances in Cryptology - EUROCRYPT. pp. 379–407 (2019)