Block-Wise Non-Malleable Codes[*]

Nishanth Chandran[1], Vipul Goyal[2], Pratyay Mukherjee[†3], Omkant Pandey[4], and Jalaj Upadhyay[‡5]

1 **Microsoft Research, India**
nichandr@microsoft.com
2 **Microsoft Research, India**
vipul@microsoft.com
3 **University of California, Berkeley, USA**
pratyay85@berkeley.edu
4 **Drexel University, Philadelphia, USA**
omkant@drexel.edu
5 **Pennsylvania State University, State College, USA**
jalaj@psu.edu

Abstract

Non-malleable codes, introduced by Dziembowski, Pietrzak, and Wichs (ICS '10) provide the guarantee that if a codeword c of a message m, is modified by a tampering function f to c', then c' either decodes to m or to "something unrelated" to m. In recent literature, a lot of focus has been on explicitly constructing such codes against a large and natural class of tampering functions such as *split-state* model in which the tampering function operates on different parts of the codeword *independently*.

In this work, we consider a stronger adversarial model called *block-wise tampering* model, in which we allow tampering to depend on more than one block: if a codeword consists of two blocks $c = (c_1, c_2)$, then the first tampering function f_1 could produce a tampered part $c'_1 = f_1(c_1)$ and the second tampering function f_2 could produce $c'_2 = f_2(c_1, c_2)$ depending on *both* c_2 and c_1. The notion similarly extends to multiple blocks where tampering of block c_i could happen with the knowledge of all c_j for $j \leq i$. We argue this is a natural notion where, for example, the blocks are sent one by one and the adversary must send the tampered block before it gets the next block.

A little thought reveals that it is impossible to construct such codes that are non-malleable (in the standard sense) against such a powerful adversary: indeed, upon receiving the last block, an adversary could decode the entire codeword and then can tamper depending on the message. In light of this impossibility, we consider a natural relaxation called *non-malleable codes with replacement* which requires the adversary to produce not only related but also a valid codeword in order to succeed. Unfortunately, we show that even this relaxed definition is not achievable in the information-theoretic setting (i.e., when the tampering functions can be unbounded) which implies that we must turn our attention towards computationally bounded adversaries.

As our main result, we show how to construct a block-wise non-malleable code (BNMC) from sub-exponentially hard one-way permutations. We provide an interesting connection between BNMC and non-malleable commitments. We show that any BNMC can be converted into a non-malleable (w.r.t. opening) commitment scheme. Our techniques, quite surprisingly, give rise to a

[*] Part of this work was done while Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay were visiting Microsoft Research, India.

[†] Pratyay Mukherjee was supported in part from a DARPA/ARL SAFEWARE award, AFOSR Award FA9550-15-1-0274, and NSF CRII Award 1464397. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

[‡] Jalaj Upadhyay was supported by the NSF Award IIS-1447700. Part of this work was done while the author was at the University of Waterloo. The views expressed are those of the author and do not reflect the official policy or position of the National Science Foundation.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi; Article No. 31; pp. 31:1–31:14

non-malleable commitment scheme (secure against so-called synchronizing adversaries), in which *only* the committer sends messages. We believe this result to be of independent interest. In the other direction, we show that any non-interactive non-malleable (w.r.t. opening) commitment can be used to construct BNMC only with 2 blocks. Unfortunately, such commitment scheme exists only under highly non-standard assumptions (adaptive one-way functions) and hence can not substitute our main construction.

1998 ACM Subject Classification E.2 Public key cryptosystems

Keywords and phrases Non-malleable codes, Non-malleable commitments, Block-wise Tampering, Complexity-leveraging

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.31

1 Introduction

Non-malleable codes. Error correcting codes allow a message m to be encoded into a codeword c, such that m can always recovered even from a tampered codeword c', only if the tampering is done in a specific way. More formally, the class of tampering functions, $\mathcal{F}_{\mathsf{frac}}$, tolerated by traditional error correction codes are ones that erase or modify only a constant fraction of the codeword c. In particular, no guarantees are provided on the output of the decoding algorithm when the tampering function $f \notin \mathcal{F}_{\mathsf{frac}}$. A more relaxed notion, error detecting codes, allow the decoder to also output a special symbol $\perp$, when m is unrecoverable from c', but here too, the codes can not tolerate simple tampering functions $f \in \mathcal{F}_{\mathsf{const}}$ where $\mathcal{F}_{\mathsf{const}}$ contains all constant functions[1]. To address this shortcoming of error correction/detection codes, Dziembowski, Pietrzak, and Wichs [12], introduced a more flexible notion of *non-malleable codes* (NMC). Informally, an encoding scheme $\mathsf{Code} := (\mathsf{Enc}, \mathsf{Dec})$ is a NMC against a class of tampering functions, $\mathcal{F}$, if the following holds: the decoded message $m' = \mathsf{Dec}(c')$ is either equal to the original message m or is completely unrelated to m, when $c' = f(\mathsf{Enc}(m))$ for some $f \in \mathcal{F}$. In general, NMC cannot exist for the set of all tampering functions $\mathcal{F}_{\mathsf{all}}$. To see this, observe that a tampering function that simply runs the decode algorithm to retrieve m, and then encodes a message related to m, trivially defeats the requirement above. However, somewhat surprisingly, Dziembowski *et al.* [12] showed the (probabilistic) existence of a NMC against a function family, $\mathcal{F}_{\mathsf{almost}}$, that is only slightly smaller than the set of all functions. They also constructed an efficient NMC against the class of tampering functions, $\mathcal{F}_{\mathsf{bit}}$, that can tamper each bit of the codeword independently. NMC has found important applications in tamper-resilient cryptography [12, 21, 13, 14].

Split-state Tampering. Arguably, one of the strongest class of tampering functions for which explicit constructions of NMC are known, is in the so called *split-state model.* Informally, a split-state model with ℓ states has the following attributes: (i) the codeword is assumed to be partitioned into ℓ-disjoint blocks $(c_1, \cdots, c_\ell)$, and (ii) the class of tampering functions, $\mathcal{F}^{\ell}_{\mathsf{split}}$, consists of all the functions $(f_1, \cdots, f_\ell)$ where f_i operates *independently* on c_i[2]. Dziembowski *et al.* [12] gave a construction of a NMC against the tampering class $\mathcal{F}^2_{\mathsf{split}}$ in the random oracle

[1] In particular if f always outputs some valid codeword c', then it is impossible to detect the error. For some cryptographic application like protecting against memory tampering attack this is too restrictive.

[2] Note that the class $\mathcal{F}_{\mathsf{bit}}$ can be viewed as $\mathcal{F}^n_{\mathsf{split}}$, where n is the length of the codeword c.

model. Constructions of NMC against $\mathcal{F}^2_{\mathsf{split}}$ are now known both in the computational [21][3] and information-theoretic settings [2, 7, 11]. Recently, Chattopadhyay and Zuckerman [6] gave an explicit information-theoretic NMC against $\mathcal{F}^{10}_{\mathsf{split}}$ and Aggarwal *et al.* [1] showed how to construct explicit information-theoretic NMC against $\mathcal{F}^2_{\mathsf{split}}$.

Going beyond split-state: Block-wise Tampering. A severe restriction of the split-state model is that every block of the codeword can only be tampered *independently* of all other blocks. In particular f_i modifies c_i with absolutely no knowledge about c_j, for any $j \neq i$. In this work, we address this restriction by allowing modification of *each* block *depending* on more than one-block. In particular, each c_i can be modified in any arbitrary way based on the first i blocks $(c_1, \dots, c_i)$. Such a code is called *block-wise* NMC. More formally a code is called a *block-wise* NMC if it is a NMC against the class of tampering functions $\mathcal{F}^\ell_{\mathsf{block}}$: a set of functions $(f_1, \cdots, f_\ell) \in \mathcal{F}^\ell_{\mathsf{block}}$ if each f_i modify c_i to some c'_i depending on the first i-blocks. We also consider a stronger class of functions where the tampering can be done in *any* order. In particular f_i can modify any c_j depending on any i blocks. A natural scenario is a synchronous streaming model when the blocks are coming in one by one and the adversary on the channel sends across each modified blocks before the next block arrives.

NMC for $\mathcal{F}^\ell_{\mathsf{block}}$ is impossible. One can see that it is impossible to construct NMC against $\mathcal{F}^\ell_{\mathsf{block}}$ (for any ℓ): consider a tampering function, where the first $\ell-1$ functions, $(f_1, \dots, f_{\ell-1})$ are identity functions and the function f_ℓ (which gets the entire codeword as input) simply decodes the message and depending on the message, keeps it the same or overwrites it to something "invalid" (i.e., the modified codeword decodes to $\perp$). Note that, in this case the distribution of the (decoding of the) tampered codeword will indeed depend on the message, thereby violating non-malleability. In particular, such a tampering attack makes the decoder output $\perp$ with a probability distribution that depends on the input message. Therefore, we seek for a natural relaxation of the traditional definition of NMC such that it is achievable for the class $\mathcal{F}^\ell_{\mathsf{block}}$ and at the same time sufficient for interesting applications. In particular, we show that such relaxed NMC is sufficient to construct a simple non-malleable commitment scheme in a black-box manner. We note that the traditional application to tamper-resilient cryptography does not work with the relaxed version for obvious reason..

NMC with replacement (NMCwR). Essentially in the above attack the adversary breaks non-malleability by making the codeword "invalid". So, we take the most natural direction to relax the definition, in that the adversary is considered to be successful only if it produces some *valid and related* codeword via tampering. In particular, the adversary may selectively "destroy" a codeword depending upon the message we encode, however we show that in some sense, this is the "only attack" it can perform. Intuitively the guarantee provided by such an encoding scheme is that any adversary, by tampering with some encoded data can not produce a related encoded data without destroying it. However, formalizing such intuition turns out to be non-trivial. We take inspiration from the literature of non-malleable commitment w.r.t. replacement (introduced by Goyal [16]) and formalize such a relaxation by introducing an algorithm (possibly inefficient) called *replacer* which comes into play only when the tampered codeword is invalid, and in that case it replaces the $\perp$ by "anything" of his choice. Essentially, the idea is that if the invalidity depends on the input message (like

[3] In the computational setting, the functions f_i are assumed to run in polynomial time.

described in the above attack) then the replacer would rectify the output to remove such dependency. We call the new notion *non-malleable codes with replacement (NMCwR).*

1.1 Our results

In this paper we explore the properties, constructions and applications of NMCwR with respect to the class of block-wise tampering functions $\mathcal{F}_{\mathsf{block}}^{\ell}$. We call such code *block-wise non-malleable codes* (BNMC). Below we provide an overview of the results presented in this paper.

Information theoretic impossibility. Similar to the notion of continuous non-malleable codes [13](CNMC), any BNMC must possess a *uniqueness* property (a slightly different one than CNMC). For two blocks, uniqueness means that there can not exists two different valid codewords of the form (c_1, c_2) and (c_1, c_2') which decodes to different messages, i.e., for every valid code c_1, there is a unique decoding. If not, then an attack similar to the CNMC is possible without making the codeword invalid – the adversary can always tampers the first block to c_1 and depending on the message (since f_2 gets the entire codeword) tampers to one of c_2 or c_2' hence making the output distribution depend on the message. Consequently, just like CNMC, an information theoretic impossibility is evident with the only difference that in the setting of CNMC, the functions are unbounded, and, therefore (for two blocks) the function f_1 can derive the unique message corresponding to c_1 by brute-force and thus break the scheme. We show the following in the full version [4].

▶ **Theorem 1.** *It is impossible to construct an information-theoretic BNMC.*

Henceforth, in this paper we focus on constructing BNMC based on computational assumptions. We stress that even we are in the computationally bounded setting, we do not put any restriction on the efficiency of the replacer. In particular, the replacer is allowed to run in super-polynomial (or even exponential) time. In fact, later in this paper, we often encounter a replacer which runs in exponential time. Nonetheless, we must restrict the reduction to be probabilistic polynomial time (PPT). We are indeed able to overcome this technical hurdle by constructing such "efficient" reductions which can correctly simulate behavior of "highly inefficient" replacers.

Connection to Non-malleable Commitment. Since BNMC satisfies a definition weaker (that is NMC with replacement) than the traditional NMC, it is not possible to use such a code to build a tamper-resilient compiler as described in [12, 21] for obvious reason. In fact, it is nevertheless impossible to protect a system against memory tampering attack (see [8, 15, 18] for formal expositions on such attack) against any block-wise tampering. However we are able to show connections with non-malleable commitment with respect to opening (NMCom). To the best of our knowledge this is the first attempt to bridge these two non-malleability notions[4].

[4] In a recent work, Agrawal et al. [3] showed how to use NMC to construct non-malleable string-commitment from non-malleable bit-commitment. In their work, NMC is used as a tool, and, no relations are shown between non-malleable commitments and NMC. Recently, another work by Goyal et al. [17] constructs round-optimal NMCom from split-state NMC, the full version of which appears after the first version of this work.

1. Given an ℓ-block BNMC we can construct (in a black-box way) a simple $(\ell-1)$ round commitment protocol which is non-malleable with respect to opening (against synchronizing adversaries) as follows: the committer sends the block c_i in the i-th round and sends the last block c_ℓ as the opening. The receiver sends only acknowledgements after receiving each message. The non-malleability essentially follows from the non-malleability of the underlying BNMC and the perfect binding follows from the uniqueness property described above. To best of our knowledge, this is the first NMCom protocol where the receiver is not required to send any message (e.g. challenge) except for acknowledgement.

▶ **Theorem 2.** *Suppose there is an ℓ-block BNMC. Then there is a $(\ell-1)$ round perfectly binding non-malleable commitment scheme with respect to opening against a synchronizing man-in-the-middle adversary.*

2. We also show that from any non-interactive NMCom one can easily construct an BNMC even for only $\ell = 2$ blocks (i.e. optimal for $\mathcal{F}^{\ell}_{\mathsf{block}}$). Unfortunately, the only assumptions under which we know how to construct such commitments are either in the (non-tamperable) CRS model [9] or under the highly non-standard assumption of *adaptive* one-way functions [22]. Evidently this construction can not substitute our main construction which is based on much more standard assumption like sub-exponentially hard OWP.

▶ **Theorem 3.** *Suppose there is a perfectly binding non-interactive non-malleable commitment scheme (w.r.t. opening) whose input is a k-bit message and output is an n-bit commitment. Then there is a 2-block BNMC.*

Combining Theorem 2 and Theorem 3, we can conclude that when $\ell = 2$ the NMCom and BNMC are equivalent. The details of these results are elaborated in the full version [4].

Constructing BNMC. As the main result we provide a construction of BNMC from a standard assumption in the plain model. Precisely, we show that, for any arbitrary constant $\varphi > 0$, how to construct a BNMC against $\mathcal{F}^{\ell}_{\mathsf{block}}$ for $\ell = O(\kappa^{2+\varphi})$ (where κ is the security parameter). The security (i.e. non-malleability) of the construction is based on "sub-exponentially" hard one-way permutations which says that there exists one-way permutations (OWP) which are "hard-to-invert" even against an adversary running in sub-exponential time, precisely in time $O(2^{\kappa_{\mathsf{s}}})$ such that $\kappa_{\mathsf{s}} = O(\kappa^{\epsilon}/2)$ for some $0 < \epsilon < 1$. In particular, our construction uses any perfectly binding commitment scheme that is computationally hiding against such sub-exponential adversary (and this primitive can be constructed from the above assumption). The key technical challenge, as remarked earlier, is that BNMC is *not* an interactive primitive that allows bi-directional communication. This limitation renders the previously proposed techniques for designing non-malleable protocols inherently unusable. This is because these previous techniques are based on having "challenge-response" rounds similar to the type also used in designing zero-knowledge protocols. Thus, techniques like rewinding the sender are not useful in this setting at all: since there are no receiver messages, one would end up with the same transcript every time. Thus, apriori, it seems unclear what advantage one could get by having multiple blocks. Our final construction is quite clean and in fact, also gives arguably one of the simplest known constructions of non-malleable commitments. We show the following theorem.

▶ **Theorem 4.** *Assume the existence of sub-exponentially hard one-way permutations. Then for any $\varphi > 0$ of our choice, and any message length $k \in \mathbb{N}$, there exists an explicit construction of ℓ-block* BNMC *of codeword length $n = O(k\kappa^{6+\varphi})$, where $\ell = O(\kappa^{2+\varphi})$.*

We give an overview of the construction used to prove Theorem 4 in Section 1.2.

Strong BNMC. Additionally, we also consider a strictly stronger model of tampering: assume any permutation $\pi : [\ell] \to [\ell]$ chosen by the adversary. Then each function f_i takes i blocks $(c_{\pi(1)}, \ldots, c_{\pi(i)})$ as input and modifies the $\pi(i)$-th block. We call this family of function strong block-wise and denote it by $\mathcal{F}^{\ell}_{\textsf{s-block}}$. We also provide a definition of strong BNMC which is essentially an explicit presentation of NMCwR for $\mathcal{F}^{\ell}_{\textsf{s-block}}$. We provide an unconditional generic transformation to construct strong BNMC from any BNMC which, along with the earlier results imply that any construction of BNMC can be transformed to a strong BNMC (with some blow up in the length of codeword). We show the following, the details of which appears in the full version [4].

▶ **Theorem 5.** *If there is an ℓ-block BNMC with codeword length n, then there is an (ℓ)-block SBNMC with codeword length $\Theta(\ell n)$.*

1.2 Overview of our techniques

We now give a brief overview of our main construction of BNMC along with intuition as to why it works. The detailed construction is provided in Sec. 3.

First fix a parameter μ (such that $\mu = O(\kappa^{2+\varphi})$ for any arbitrary constant $\varphi > 0$ of our choice where κ is the security parameter) such that we encode a message m using $\ell = (2\mu + 1)$-blocks of codeword for some parameter μ . At a very high level, our encoding is as follows. Let us first fix some index (or *tag*) for the encoder $i \in [\mu]$. The encoder then chooses a *perfectly binding* commitment scheme $\textsf{COM}$.

Let $\textsf{COM}_{\kappa_{\textsf{s}}}(\cdot)$ and $\textsf{COM}_{\kappa}(\cdot)$ denote that $\textsf{COM}$ is computationally hidden with respect to security parameters $\kappa_{\textsf{s}}$ and κ respectively, where $\kappa_{\textsf{s}}$ is as mentioned above. The encoder then computes commitments to the message using $\textsf{COM}_{\kappa_{\textsf{s}}}$ and $\textsf{COM}_{\kappa}$. The first 2μ blocks of the encoding of m are blocks of all zeroes, except for block i and block $(2\mu - i)$ which are the commitments $\textsf{COM}_{\kappa}$ and $\textsf{COM}_{\kappa_{\textsf{s}}}$, respectively. The $(2\mu + 1)^{\text{th}}$ block of the encoding contains the openings to $\textsf{COM}_{\kappa_{\textsf{s}}}$ and $\textsf{COM}_{\kappa}$. The decoding algorithm checks if (i) all the openings are consistent with the commitments and (ii) the messages committed are equal. Now, for a moment, assume that adversary's index i' is not equal to i (this can be removed later on). Then if $i' < i$, then the adversary has to output its first commitment without seeing the first commitment in the input codeword (rather only seeing on the string of zeros). Thus, the first commitment in the output is independent of the first commitment in the input. Moreover, our definition (NMCwR) puts the additional restriction that the adversary has to output a valid codeword in order to succeed. Combining one can see that the output codeword, if valid, must contain a message independent of the message encoded in the input. On the other hand, if $i' > i$, then the second commitment of the adversary has to be independent of the second commitment in the input. In this case, we rely on complexity leveraging to prove non-malleability. Using this key-observation one can prove the non-malleability except in one case: when the index chosen by the adversary i' is equal to i. To prevent mauling in this case we use one-time signatures. The encoder signs the entire codeword using i as a public-key and thus leaving the adversary either to forge the signature or change the index. However, one problem still remains. To use i as a public-key we need it to be sufficiently long, in particular for a concrete instance of such OTS (we consider variant of Lamport [19]) the length needed to be $O(\kappa^{2+\varphi})$ for any arbitrary constant $\varphi > 0$ of our choice. But note that, we have $i \in [\mu]$ and $\ell = 2\mu + 1$. Trying to set the size of the index $|i| = \log(\mu)$ to even $\Omega(k)$ would result in an "inefficient" construction with $\ell = 2^{\Omega(k)}$ blocks which is not acceptable. We solve this problem by using a "well-known" technique from non-malleable commitment, so-called DDN-XOR trick. Through that, it is possible to use a long tag of size $t = O(\kappa^{2+\varphi})$ keeping the number of blocks also $O(\kappa^{2+\varphi})$ just by computing t shares (XOR's) of messages and and applying the above construction independently on the shares. So, our final construction would require a one-time signature which works with a public-key of bit-length $\mu = O(\kappa^{2+\varphi})$.

2 Definitions

We introduce the "relaxed" definition of non-malleable codes which is same as the NMC except there is a so-called *replacer* $\mathbf{R_f}$ which is an "all powerful" algorithm and comes into play only when the modified codeword is invalid (i.e. decodes to $\perp$). In that case, the replacer may replace the $\perp$ by any message in the message space or the symbol $\mathsf{same}^\star$. (The replacer can also keep the $\perp$ in case when it not harmful (i.e. does not depend on the input) e.g. when the tampering function always tampers to something invalid). Since the idea of replacer is similar in spirit with the notion of *non-malleable commitment with replacement* as introduced in [16] we call this relaxed version *non-malleable codes with replacement* (NMCwR in short). We present the formal definition below.

▶ **Definition 6** (Non-malleable codes with replacement)**.** Let $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ be an (k, n)-encoding scheme. Let $\mathcal{F}$ be some family of tampering functions. Then Code is called (k, n)-non-malleable code with replacement (NMCwR) if for every $f \in \mathcal{F}$ there exists an algorithm called the replacer $\mathbf{R_f}$ such that for any pair of messages $m_0, m_1 \in \{0,1\}^k$, $\mathsf{TampWR}^f_{m_0} \approx \mathsf{TampWR}^f_{m_1}$, where for any $m \in \{0,1\}^k$, TampWR^f_m is defined as

$$\mathsf{TampWR}^f_m \equiv \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(m); c' \leftarrow f(c); \\ \text{If } c' = c \text{ set } m' := \mathsf{same}^\star \text{ else } m' \leftarrow \mathsf{Dec}(c') \\ \text{If } m' = \perp \text{ then } m' := \mathbf{R_f}(c)\ ;\ \text{Output: } m' \end{array} \right\},$$

where the randomness is over the encoding function Enc.

▶ Remark. Here, and everywhere in this section, the indistinguishability depends on the setting (information theoretic or computational). However, we emphasize that even if we are in the computationally bounded scenario, where the adversary is PPT, we do not restrict the replacer to be a PPT algorithm. This assumption is justified because the replacer is required only to establish the meaningfulness of the definition without affecting the natural intuition. Intuitively the purpose of the replacer is to relax the traditional notion in a way such that the tampering function is allowed to distinguish the tampering experiments, albeit only by making the codeword invalid. Nonetheless in the computational setting all the other algorithms involved as well as the the tampering functions are required to be PPT.

Some intuitions. We first provide some intuition behind why the above definition is meaningful. For every adversary, there is guaranteed to exist another adversary which always tampers in the same way as the original adversary, except, when the original adversary were to output an invalid codeword. In that case, the new adversary may employ any other (PPT) strategy. However when the original adversary outputs an invalid codeword, (in many applications) it could be considered as aborting or failing in those cases. Hence, our new adversary could be seen as strictly more powerful than the original one. However as the definition guarantee, the new adversary actually obeys the standard non-malleable code guarantee. Thus, in many scenarios, we believe the above weaker notion may be sufficient. Indeed, as shown in [16], the corresponding weaker notion for non-malleable commitments (called non-malleability w.r.t. replacement) turns out to be sufficient for several applications including for obtaining constant round multi-party computation.

We now give the syntactic definition of *block-wise encoding scheme*.

▶ **Definition 7** (Block-wise encoding scheme). Let $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ be an (k, n)-encoding scheme. Then it is called an (ℓ, k, n)-*block-wise encoding scheme* if each string output by Enc is an ℓ-tuple: $(c_1, \ldots, c_\ell)$ where $|c_i| = n_i$, with $\sum_{i=1}^{\ell} n_i = n$. Also let $\nu_i = \sum_{j=1}^{i} n_i$.

Next we define a property of such block-wise encoding scheme called *reveal index*, that will be useful later on.

▶ **Definition 8** (Reveal Index). Let $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ be an (ℓ, k, n)-block-wise encoding scheme. Then Code is said to have reveal index η if $\eta - 1 \in [\ell]$ is the largest index for which the following condition holds: For all pair of messages $m_0, m_1 \in \{0,1\}^k$ if $(c_1^{(0)}, \ldots, c_\ell^{(0)}) \leftarrow \mathsf{Enc}(m_0)$ and $(c_1^{(1)}, \ldots, c_\ell^{(1)}) \leftarrow \mathsf{Enc}(m_1)$ then $(c_1^{(1)}, \ldots, c_{\eta-1}^{(1)}) \approx (c_1^{(1)}, \ldots, c_{\eta-1}^{(1)})$.

▶ Remark. This definition formalizes the fact that, for any encoding scheme, there is an index η which reveals some information about the encoded message for the first time in the sequence and the sequence $(c_1, \ldots, c_{\eta-1})$ before that does not reveal anything about the encoded message. Obviously $\eta \leq \ell$ for any block-wise encoding scheme.

Finally, we present our main definition of a *block-wise non-malleable encoding scheme* which is essentially an explicit presentation of NMCwR for the class $\mathcal{F}_{\mathsf{block}}^{\ell}$.

▶ **Definition 9** (Block-wise non-malleable codes). Let $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ be an (ℓ, k, n)-block-wise encoding scheme. Let $\mathbf{f} = (f_1, \ldots, f_\ell)$ be any tuple of functions specified as follows: $\forall i \in [\ell], f_i : \{0,1\}^{\nu_i} \to \{0,1\}^{n_i}$. Then Code is called an (ℓ, k, n)-*block-wise non-malleable code* (BNMC in short) if, for any such tuple $\mathbf{f}$, there exists a replacer $\mathbf{R_f}$, such that, for any pair of messages $(m_0, m_1) \in \{0,1\}^k$, the following holds: $\mathsf{BLTamp}_{m_0}^{\mathbf{f}} \approx \mathsf{BLTamp}_{m_1}^{\mathbf{f}}$. where $\mathsf{BLTamp}_m^{\mathbf{f}}$ for any $m \in \{0,1\}^k$ is defined as:

$$\mathsf{BLTamp}_m^{\mathbf{f}} = \left\{ \begin{array}{c} \mathbf{c} = (c_1, \ldots, c_\ell) \leftarrow \mathsf{Enc}(m); \\ \forall i \in [\ell] : c'_i = f_i(c_1, \cdots, c_i);\ \text{Let } \mathbf{c}' = (c'_1, \ldots, c'_\ell); \\ \text{If } \mathbf{c}' = \mathbf{c} \text{ then set } m' := \mathsf{same}^\star; \text{Else decode } m' \leftarrow \mathsf{Dec}(c'_1, \ldots, c'_\ell); \\ \text{If } m' = \perp \text{ then } m' \leftarrow \mathbf{R_f}(c_1, \ldots, c_\ell);\ \text{Output } m' \end{array} \right\}.$$

▶ Remark. Our notion of block-wise non-malleable codes is identical to the notion of *look-ahead non-malleable codes* defined in the concurrent and independent work of Aggarwal *et al.* [1]. We choose to use the term block-wise as it is more appropriate in our setting.

3 Our Construction

In this section, we provide our main construction of a BNMC based on *sub-exponentially hard one-way permutations*. We construct the encoding scheme in three steps:

(i) In Sec 3.1 we begin by constructing a weaker BNMC that we call *tag-based block-wise non-malleable encoding scheme* (TBNMC). In such a code, every codeword has a *tag* associated with it and the tampering function must change the tag of a codeword in order to successfully maul a codeword. In other words, we allow an adversary to create a related codeword only when the tag remains the same. The tag used here is an index of the block and hence is only of size $\log(\kappa)$.

(ii) Then in Sec. 3.2 we use a technique, commonly known as the DDN-XOR trick [10], to construct a tag-based BNMC with tags of length *poly*(κ).

(iii) Finally in Sec. 3.3 we construct an BNMC which achieves Def. 9, by using the public key of a *one-time signature scheme* as the tag of the above code, and by signing the entire codeword using the corresponding signing key.

3.1 Tag-based non-malleability

In this section we diverge from our original definition and construct an encoding scheme which meets a weaker definition of non-malleability, called tag-based non-malleability.

We define the tag to be always the first block of any codeword. A tag-based BNMC (TBNMC for short) is defined exactly as the same way as BNMC with the only difference that whenever the tag of the tampered codeword is equal to the tag of the original codeword, then the tampering experiment outputs $\mathsf{same}^{\star}$ even if there is any other modification. Clearly this is strictly weaker than BNMC. Please see the full version [4] for a formal definition.

Now we construct an encoding scheme which satisfies this weaker definition based on sub-exponentially hard OWP. The proof uses complexity leveraging which essentially forces us to assume sub-exponential hardness as opposed to standard (super-poly) hardness.

We assume that sub-exponentially hard OWP exist that are considered to be hard to break even if the adversary is allowed to run in sub-exponential time, namely in $O(2^{\kappa_{\mathsf{s}}})$ such that $\kappa_{\mathsf{s}} = \kappa^{\epsilon}/2$ (recall that κ is the security parameter) for some constant $\epsilon \in (0, 1)$. The proof crucially relies on this as it uses one level of complexity leveraging. In particular, while reducing to such OWP, we assume that the adversary (the reduction in this case) is unable to break the one-way permutation (the hiding of a commitment scheme in this case) even when it is allowed to run in time $O(2^{\kappa_{\mathsf{s}}})$ (but in time $o(2^{\kappa})$).

We use a *non-interactive commitment*, Com, that is *perfectly binding*. We write $\mathsf{Com}_{\kappa_{\mathsf{s}}}$ and Com_{κ} to denote the commitment scheme has *computational hiding* with the security parameters κ_{s} and κ, respectively. In particular, Com_{κ} is a computationally hiding commitment scheme even against an adversary running in $O(2^{\kappa_{\mathsf{s}}})$ time. Suppose that such commitment scheme, on input some bit-string of length $k \in \mathbb{N}$, outputs commitments of length $\mathsf{p}(\kappa, k)$ where $\mathsf{p}(\cdot) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a fixed polynomial (determined by the specifications of the commitment scheme) in security parameter. We stress that such commitments can be constructed from *sub-exponentially hard one-way permutations.*

First we give a brief overview of the construction. Let $\mu \in \mathbb{N}$ be a parameter. We will now construct a TBNMC with ℓ blocks where $\ell = 2\mu + 2$. For now, assume ℓ to be a even number. Now for any tag $\mathsf{tg} \in [\mu]$ we construct the encoding scheme as follows: we put strings of 0 in all the blocks except the four "special" blocks: the first block is set to tg, the $(\mathsf{tg} + 1)$-th block is set to the "bigger" commitment $\mathsf{Com}_{\kappa}(m)$, the $(\ell - \mathsf{tg})$-th block is set to the "smaller" commitment $\mathsf{Com}_{\kappa_{\mathsf{s}}}(m)$ and the ℓ-th (and final) block is set to the openings of the commitments. Now, for odd ℓ, one can just append one dummy block (string of 0's) right before the final block. So, without loss of generality we would assume ℓ to be even in this section. The detail construction is presented in Fig. 1. Note that here the blocks are of different length. However, it is easy to convert the code with equal block-length by padding additional zeros. We keep it without such padding for simplicity. Also, note that, from the computational hiding property of the commitment scheme, it follows that the construction has reveal index $\ell = 2\mu + 2$ for any PPT adversary.

The following theorem states that the construction is a TBNMC. The proof can be found in the full version [4].

▶ **Theorem 10.** *Let $\mu \in \mathbb{N}$ be some parameter. Assume that sub-exponentially hard one-way-permutations exists. Then, for any tag $\mathsf{tg} \in [\mu]$ and any $k \in \mathbb{N}$, the encoding scheme*

$\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ *described in Fig. 1 is a* $(\mathsf{tg}, \ell, k, n)$-TBNMC *against all* PPT *adversary such that* $n = O(k + \mu \cdot \mathsf{p})$ *and* $\ell = 2\mu + 2$.

3.2 Non-malleability amplification

In this section we extend our construction to an efficient construction which can support larger tags. This extension is similar to a well-known phenomenon, namely *non-malleability amplification* [20], in the non-malleable commitment literature using the DDN-XOR trick [10].

3.2.1 One-many non-malleability

Towards that, we first show that the construction given in Fig. 1 already satisfies a stronger notion, which we call *one-many* tag-based non-malleability (OMTBC). This definition (we refer to the full version [4] for a formal definition), informally states that an adversary that is able to tamper a single codeword of m, cannot even come up with a set of codewords such that one of them is related to m. In particular, each function f_i in the tuple $\mathbf{f} = (f_1, \ldots, f_\ell)$ has much larger range than the domain and produces many c_i's together with the knowledge of the first i blocks of the input codeword.[5] Our next theorem shows that our construction (Fig. 1) achieves this stronger definition.

▶ **Theorem 11.** *Let* $\mu, t \in \mathbb{N}$ *be some parameter. Assume that sub-exponentially hard one-way-permutations exists. Then, for any tag* $\mathsf{tg} \in [\mu]$ *and any* $k \in \mathbb{N}$ *the* $(\mathsf{tg}, \ell, k, n)$-TBC $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ *described in Fig. 1 is an* $(t, \mathsf{tg}, \ell, k, n)$*-one-many tag-based BNMC against all* PPT *adversary such that* $n = O(k + \mu \cdot \mathsf{p})$ *and* $\ell = 2\mu + 2$.

3.2.2 Using DDN-XOR trick

In this section we use the DDN-XOR trick to construct an "efficient" TBNMC with "large" tags. Let us start with some intuitions. The construction uses any OMTBC (called "inner code" in the following) with "small" tag in a black-box way. The basic idea is as follows: let the "big" tag TG be t-bit long. Then compute t shares of message m just using XOR's i.e. $(m_1, \ldots, m_t)$ which is nothing but a t-out-of-t secret sharing. Then encode each m_j with the inner code using $j \| \mathsf{TG}[j]$ (which is of $O(\log(t))$-size) as tag. Finally put the encodings in increasing order of j (from 1 to t). The first block of the final codeword is, by definition the tag TG. the second block would consist of the first t blocks of inner codes in order and so on. The key-intuitions why the construction works are as follows. In order to break the tag-based non-malleability of the final encoding (called "outer code" within this sub-section), the adversary must produce a valid codeword with different"big" tag $\widetilde{\mathsf{TG}} \neq \mathsf{TG}$. In that case, evidently, there must exist at least one index $j \in [t]$ where the "small" tags differ $\widetilde{\mathsf{tg}}_j \neq \mathsf{tg}_j$. Moreover notice that, the adversary can't copy tg_j to any other position than j as that would result in an invalid codeword. Therefore $\mathsf{tg}_j = j \| \mathsf{TG}[j]$ is different from *all* the "small tags" of the tampered inner codewords. Then we reduce to the one-many non-malleability of the inner code in first such position (say $j^\star$). In particular, if the adversary tampers with the $j^\star$-th inner code, then by one-many non-malleability of the "inner code" no tampering

[5] We note that Chattopadhyay et al. [5] introduced the notion of one-many non-malleable code which is in turn built on continuous non-malleable code [13](CNMC). It is important not to confuse this notion with CNMC where the adversary chooses each subsequent tampering function after observing the result of the previous tamperings.

Parameters: Let Com_{κ_s} takes a k-bit message as input and u_s-bit randomness to produce a v_s-bit commitment and Com_κ takes a message of the same length, but randomness of u-bit to produce a v-bit commitment[a]. Let $\mathsf{tg} \in [\mu]$ be the tag of the encoding scheme for some $\mu \in \mathbb{N}$. We define a $(\mathsf{tg}, \ell, k, n)$-block-wise encoding scheme where $\ell = 2\mu + 2$ and $n = k + u_s + u + \mu(v_s + v) + \lfloor \log \mu \rfloor + 1$ as follows:

Encoding $\mathsf{TEnc}(m)$**:** The encoder gets a message $m \in \{0,1\}^k$ as input and do as follows:

1. Initialize: Choose randomnesses $r_s \xleftarrow{\$} \{0,1\}^{u_s}$ and $r \xleftarrow{\$} \{0,1\}^u$ for commitment scheme. Set the first block $c_1 := \mathsf{tg}$.
2. Stage-1: For all $i \in \{2, \ldots, \mu+1\}$, define the i-th block of codeword c_i as follows:
$$c_i := \begin{cases} 0^v & i \neq \mathsf{tg}+1 \\ \mathsf{Com}_\kappa(m, r) & i = \mathsf{tg}+1 \end{cases}$$
3. Stage-2: For all $i \in \{\mu+2, \ldots, 2\mu+1\}$, define the i-th block of codeword c_i as follows:
$$c_i := \begin{cases} 0^{v_s} & i \neq 2\mu+2-\mathsf{tg} \\ \mathsf{Com}_{\kappa_s}(m, r_s) & i = 2\mu+2-\mathsf{tg} \end{cases}$$
4. Final stage: Define the last block as the decommitments i.e. the message and the randomnesses in the order of commitments are sent: $c_{2\mu+1} := (m, r, r_s)$.

Decoding $\mathsf{TDec}(\mathbf{c})$**:** On receiving a codeword $\mathbf{c}$ parse it as $\mathbf{c} = (c_1, \ldots, c_{2\mu+2})$ such that $|c_1| = \lfloor \mu \rfloor + 1$, for $i \in \{2, \ldots, \mu+1\}$, $|c_i| = v$, for $i \in \{\mu+2, \ldots, 2\mu+1\}$, $|c_i| = v_s$ and for $i = 2\mu+2$, $|c_i| = k + u_s + u$. Then do as follows:

1. Correctness of Structure: First check if the structure is correct: that is if $c_1 \neq 0$ and there are exactly two indexes $i_1 \in \{2, \ldots, \mu+1\}$, $i_2 \in \{\mu+2, 2\mu+1\}$ such that:
   a. $c_{i_1} \neq 0^v$ and $c_{i_2} \neq 0^{v_s}$.
   b. for all other indexes $i \in \{2, \ldots, \mu+1\} \backslash \{i_1\}$, $c_i = 0^v$ and $i \in \{\mu+2, \ldots, 2\mu+1\} \backslash \{i_2\}$, $c_i = 0^{v_s}$.
   c. $i_1 + i_2 = 2\mu + 1$.

   if any of them fails, then the structure of the tampered codeword is incorrect and therefore output $\perp$, else go to the next step.
2. Consistency of commitment: Parse $c_{2\mu+2}$ as $(m, r, r_s) := c_{2\mu+2}$ such that $|m| = k$, $|r| = u$ and $|r_s| = u_s$. Then check the validity of the commitment-decommitment pair $(c_{i_1}, (m, r))$ and $(c_{i_2}, (m, r_s))$, if any of them are invalid output $\perp$, otherwise output the committed message m.

[a] We assume $|v_s|, |v| = \mathit{poly}(\kappa)$

Figure 1 The construction of $(\mathsf{tg}, \ell, k, n)$-TBNMC for tag size $\log \kappa$.

function would not be able to succeed in producing any valid inner codeword that encodes a value which is "related" to the $j^\star$-th original share. Clearly, this implies the entire tampered outer codeword would have no information about $j^\star$-th share which makes the encoded massage (if valid) completely unrelated to the original message by the property of secret sharing.

For any tag $\mathsf{TG} \in \{0,1\}^t$ we construct a $(\mathsf{TG}, \ell', k', n')$-TBNMC $\mathsf{LCode} = (\mathsf{LEnc}, \mathsf{LDec})$ from a $(t, \mathsf{tg}, \ell, k, n)$-OMTBC $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ for any $\mathsf{tg} \in \{0,1\}^\alpha$ such that $t = 2^{\alpha-1} - 1$, $\ell' = \ell + 1$, $k' = k$ and $n' = nt$ as follows.

- **Encode** $\mathsf{LEnc}(m)$:
  1. SECRET-SHARING: On receiving an input message $m \in \{0,1\}^{k'}$, first choose $(t-1)$ random k'-bit strings $(m_1, \ldots, m_{t-1})$ and then compute $m_t = m \oplus m_1 \oplus \cdots \oplus m_{t-1}$. Note that the tuple $(m_1, \ldots, m_t)$ represents a (t,t)-secret sharing of m.
  2. ENCODE USING SMALLER TAG: Then for each $j \in t$, let the j-th "smaller" tag be $\mathsf{tg}_j = \mathtt{BIT}(j) \| \mathsf{TG}[j]$. Then compute the encoding of m_j as: $(c_{1,j}, \ldots, c_{\ell,j}) \leftarrow \mathsf{TEnc}_{\mathsf{tg}_j}(m_j)$.
  3. CONSTRUCTING BLOCKS: Define the tag-block $c_0 := \mathsf{TG}$. For all $i \in [\ell]$ define the i-th block as $c_i := (c_{i,1}, \ldots, c_{i,t})$. Output the codeword $\mathbf{c} = (c_0, \ldots, c_\ell)$.
- **Decode** $\mathsf{LDec}(\mathbf{c})$:
  1. PARSING: On receiving a codeword $\mathbf{c}$, parse it as $(c_0, \ldots, c_\ell) := \mathbf{c}$ such that $|c_0| = t$ and for all $i \in [\ell]$ $|c_i| = tn_i$. Then, for all $i \in [\ell]$ parse c_i as $(c_{i,1}, \ldots, c_{i,t})$ such that for all $j \in [t]$, $|c_{i,j}| = n_i$.
  2. CHECKING TAG CONSISTENCY: Check if the "bigger" tag is consistent with the "smaller" tag: $c_0 = c_{1,1}[\alpha] \| c_{1,2}[\alpha] \| \cdots \| c_{1,t}[\alpha]$. Also check if the positions of the smaller tags are correct: $\forall\ j \in [t]$, $c_{1,j}[1 \ldots (\alpha-1)] = \mathtt{BIT}(j)$. If any of these fail output $\perp$, otherwise go to the next step.
  3. DECODING WITH SMALLER TAG: For each $j \in [t]$ decode each value $v_j \leftarrow \mathsf{TDec}_{\mathsf{tg}_j}(c_{1,j}, \ldots, c_{\ell,j})$. If any of them is $\perp$ then output $\perp$. Otherwise, parse each v_j as m_j and finally output $m = m_1 \oplus \cdots \oplus m_t$.

Formally we show the following theorem. The proof can be found in the full version [4].

▶ **Theorem 12.** *Let* $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ *be a* $(t, \mathsf{tg}, \ell, k, n)$*-OMTBC for any tag* $\mathsf{tg} \in \{0,1\}^\alpha$*,* $t = 2^{\alpha-1} - 1$ *and* $k \in \mathbb{N}$*. Then for any tag* $\mathsf{TG} \in \{0,1\}^t$ *the above construction* $\mathsf{LCode} = (\mathsf{LEnc}, \mathsf{LDec})$ *is a* $(\mathsf{TG}, \ell', k', n')$*-TBNMC for* $\ell' = \ell + 1$*,* $k' = k$ *and* $n' = nt$*.*

3.3 The full construction by removing tags

Finally we present a transformation to remove tags using one-time signature scheme and a tag-based code with "large tag" (will be referred to as "inner code" in this section). This is similar to a standard trick [10] used in the area of non-malleable commitment for the same purpose. The main idea is to sign the entire codeword and set the public-key as the tag. This forces the tampering function either to keep the tag same and forge the signature in order to tamper, otherwise change the tag by producing its own key-pairs and then tamper. But the "inner code" guarantees that whenever the tag is changed, the tampering would result in an "unrelated" codeword.

Let $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ be an $(\mathsf{tg}, \ell, k, n)$-TBNMC for any tag $\mathsf{tg} \in \{0,1\}^t$. Let $\mathsf{OTSig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-time signature scheme with public key $pk \in \{0,1\}^t$ which takes any $k_m = n - t$-bit message to produce a n_s-bit signature. Then we construct an $(\ell, k, n + n_\mathsf{s})$-BNMC $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ as follows:

- **Encode** $\mathsf{Enc}(m)$:
  1. Generate signature keys: On input message $m \in \{0,1\}^k$ first run the key-generation algorithm of the signature scheme OTSig to generate a key pair: $(pk, sk) \leftarrow \mathsf{KGen}(1^\kappa)$.
  2. Encode with tag: Run the tag-based encoding scheme with pk as the tag on the input message m to produce the codeword $(\tilde{c}_1, \ldots, \tilde{c}_\ell) \leftarrow \mathsf{TEnc}(m)$. Note that $\tilde{c}_1 = pk$.
  3. Sign the codeword: Sign the codeword (except the tag) $(\tilde{c}_2, \ldots, \tilde{c}_\ell)$ to compute the signature $\sigma \leftarrow \mathsf{Sign}(sk, (\tilde{c}_2, \ldots, \tilde{c}_\ell))$.
  4. Output: Set for all $i \in [\ell - 1]$, $c_i = \tilde{c}_i$ and $c_\ell = \tilde{c}_\ell \| \sigma$. Output the codeword $\mathbf{c} = (c_1, \ldots, c_\ell)$
- **Decode** $\mathsf{Dec}(c_1, \ldots, c_\ell)$:
  1. Parse: On input the codeword $(c_1, \ldots, c_\ell)$, set $\forall i \in [\ell - 1]$, $\tilde{c}_i := c_i$ and parse c_ℓ as $(\tilde{c}_\ell \| \sigma) := c_\ell$ such that $|\tilde{c}_\ell| = n_\ell$ and $|\sigma| = n_\mathsf{s}$.
  2. Verify signature: Then verify the signature $d \leftarrow \mathsf{Verify}\,(\tilde{c}_1, (\tilde{c}_2, \ldots, \tilde{c}_\ell), \sigma)$. If $d = 0$ (i.e. verification fails) then output $\perp$. Otherwise go to the next step.
  3. Decode with tag: Decode the codeword as $\widetilde{m} \leftarrow \mathsf{TDec}(\tilde{c}_1, \ldots, \tilde{c}_\ell)$. Output $\widetilde{m}$.

Formally, we have the following theorem (see the full version [4] for a formal proof).

▶ **Theorem 13.** *Let* $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ *be a* $(\mathsf{tg}, \ell, k, n)$*-TBNMC for any tag* $\mathsf{tg} \in \{0,1\}^t$ *and* $\mathsf{OTSig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a one-time signature scheme with public key* $pk \in \{0,1\}^t$ *which takes any* $k_m = n - t$*-bit message to produce a* n_s*-bit signature. Then the above construction* $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ *is a* (ℓ', k', n')*-BNMC for* $\ell' = \ell$*,* $k' = k$ *and* $n' = n + n_\mathsf{s}$*.*

Theorem 4 now follows by combining Theorem 11, Theorem 12 and Theorem 13 and instantiating with appropriate parameters.

References

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

2 Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 774–783. ACM, 2014.

3 Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *Advances in Cryptology – CRYPTO 2015 – 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 538–557, 2015. doi:10.1007/978-3-662-47989-6_26.

4 Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. *IACR Cryptology ePrint Archive*, 2015:129, 2015. URL: http://eprint.iacr.org/2015/129.

5 Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. *To Appear in* STOC *(full version available at arXiv:1505.00107)*, 2016.

6 Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 306–315. IEEE, 2014.

7 Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *Theory of Cryptography*, pages 440–464. Springer, 2014.

8 Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. In *ASIACRYPT (2)*, pages 140–160, 2013.

9 Ivan Damgard and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 426–437. ACM, 2003.

10 Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM review*, 45(4):727–784, 2003.

11 Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In *CRYPTO (2)*, pages 239–257, 2013.

12 Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452, 2010.

13 Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *Theory of Cryptography*, pages 465–488. Springer, 2014.

14 Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In *Advances in Cryptology – EUROCRYPT 2014*, pages 111–128. Springer, 2014.

15 Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *Theory of Cryptography*, pages 258–277. Springer, 2004.

16 Vipul Goyal. Constant round non-malleable protocols using one way functions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 695–704. ACM, 2011.

17 Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. *IACR Cryptology ePrint Archive*, 2015:1178, To appear at STOC 2016. URL: `http://eprint.iacr.org/2015/1178`.

18 Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In *CRYPTO*, pages 373–390, 2011.

19 Leslie Lamport. Constructing digital signatures from a one-way function. In *Technical Report SRI-CSL-98*. SRI International Computer Science Laboratory, 1979.

20 Huijia Lin and Rafael Pass. Non-malleability amplification. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 189–198. ACM, 2009.

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

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