A constant rate non-malleable code in the split-state model

Divesh Aggarwal
Department of Computer Science
and Center for Quantum Technologies
National University of Singapore
dcsdiva@nus.edu.sg

Maciej Obremski
Center for Quantum Technologies
National University of Singapore
obremski.math@gmail.com

***Abstract*—Non-malleable codes, introduced by Dziembowski, Pietrzak and Wichs in ICS 2010, have emerged in the last few years as a fundamental object at the intersection of cryptography and coding theory. Non-malleable codes provide a useful message integrity guarantee in situations where traditional error-correction (and even error-detection) is impossible; for example, when the attacker can completely overwrite the encoded message. Informally, a code is non-malleable if the message contained in a modified codeword is either the original message, or a completely "unrelated value".**

The family which received the most attention is the family of tampering functions in the so called (2-part) *split-state* model: here the message x is encoded into two shares L and R, and the attacker is allowed to arbitrarily tamper with each L and R individually.

In this work, we give a constant rate non-malleable code from the tampering family containing so called 2-lookahead functions and forgetful functions, and combined with the work of Dodis, Kazana and the authors from STOC 2015, this gives the first *constant rate* non-malleable code in the split-state model with *negligible error*.

The full version of this paper can be found here: https://eprint.iacr.org/2019/1299.

***Keywords*-constant rate; non-malleable code; split-state model**

I. Introduction

Non-malleable codes, introduced by Dziembowski, Pietrzak and Wichs [1], provide a useful message integrity guarantee in situations where traditional error-correction (and even error-detection) is impossible; for example, when the attacker can completely overwrite the encoded message. Informally, given a tampering family $\mathcal{F}$, an $\mathcal{F}$-non-malleable code (E, D) encodes a given message x into a codeword $y \leftarrow E(x)$ in a way that, if y is modified into $y' = f(y)$ by some $f \in \mathcal{F}$, then the message $x' = D(y')$ contained in the modified codeword y' is either the original message x, or a completely "unrelated value". In other words, non-malleable codes aim to handle a much larger class of tampering functions $\mathcal{F}$ than traditional error-correcting or error-detecting codes, at the expense of potentially allowing the attacker to replace a given message x by an unrelated message x' (and also necessarily allowing for a small "simulation error" ε). As shown by [1], this relaxation still makes non-malleable codes quite useful in a variety of situations where (a) the tampering capabilities of the attacker might be too strong for error-detection, and, yet (b) changing x to unrelated x' is not useful for the attack. For example, imagine x being a secret key for a signature scheme. In this case, tampering which keeps x the same corresponds to the traditional chosen message attack (covered by the traditional definition of secure signatures), while tampering which changes x to an unrelated value x' will clearly not help in forging signatures under the original (un-tampered) verification key, as the attacker can produce such signatures under x' by himself.

Split-State Model.: Although such codes do not exist if the family of "tampering functions" $\mathcal{F}$ is completely unrestricted [1], they are known to exist for many broad tampering families $\mathcal{F}$. One such natural family is the family of tampering functions in the so called split-state model. Here the k-bit message x is encoded into 2 shares y_1, y_2 of length n each, and the attacker is allowed to *arbitrarily* tamper with each y_i *individually*. The rate of such an encoding is naturally defined as $\tau = \frac{k}{2n}$.

Non-malleable codes in this model could be interpreted as "non-malleable secret-sharing schemes": even if *both* the shares are independently tampered with, the recovered message is either x or is unrelated to x. Non-malleable codes in the split-state model have received a lot of attention so far [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. In addition, some of the recent results [11], [12], [13], [14], [15], [16] have shown application of non-malleable codes in the split-state model to other important problems like non-malleable commitments and non-malleable secret sharing.

The known results can be summarized as follows. The first non-malleable code in the split-state model against an information-theoretic adversary was constructed in [3], who constructed a non-malleable code for 1-bit messages in the split-state model. Following that [4], [7], [17] gave the first information-theoretic construction supporting k-bit messages, but where the length of each share $n = O(k^5)$. There was a plausible conjecture stated in [4] about the non-malleability of the inner product function under which one would get a 2-part split-state code with constant rate, i.e., $n = O(k)$.

In [5], it was shown that the notion of non-malleable codes in the split-state model is closely related to the notion of

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00122

non-malleable two-source extractors and using this insight, and the alternating extraction protocol from [18], recent results [8], [9], [10] have obtained improved constructions of non-malleable codes in the split-state model. The most recent result [10] gives a construction with rate $\frac{c \cdot \log\log\log 1/\varepsilon}{\log\log 1/\varepsilon}$ for some constant c. This construction has a constant rate if ε is a constant, but the rate approaches 0 if ε is negligible in n, as is required for applications. In particular, if we choose $\varepsilon = 2^{-n^{\Omega(1)}}$, then the rate is $O(\frac{\log\log n}{\log n})$.

The authors, along with Dodis and Kazana [19] introduced the concept of non-malleable reductions and, under a plausible conjecture, gave a series of reductions that results in constant rate non-malleable codes in the split-state model [1].

However, until this work, the problem of unconditionally constructing constant rate non-malleable codes in the split-state model (with ε negligible in the size of the codeword) remains open.

Our Result.: In this work, we give a constant rate non-malleable code in the split-state model.

Theorem 1.1 (Main Result): There exists an efficient, information-theoretically secure ε-non-malleable code in the split-state model with shares of size $O(k)$, where k is the length of the message, and $\varepsilon = 2^{-k^{\Omega(1)}}$.

Our result is achieved by giving a non-malleable code against the tampering family $\mathcal{G}$ containing 2-lookahead tampering functions and forgetful tampering functions. Combined with a non-malleable reduction from the 2-split tampering family to $\mathcal{G}$ gives a non-malleable code in the split-state model. For an overview of our construction and a discussion of our proof techniques, we refer the reader to Section II.

Other Related Work.: If we relax the number of states to more than 2, or we restrict the adversary to be computationally bounded, then there are known constructions of constant rate non-malleable codes with negligible error. In particular, some recent results [20], [21], [22], [23] obtain near optimal non-malleable codes in the t-split-state model where t is a constant greater than 2, and [24] gave a construction of a rate 1 non-malleable code against computationally bounded adversaries.

Other results that look at an (enhanced) split-state model are Faust et al. [25] which consider the model where the adversary can tamper continuously, and [26], that considers the model where the adversary, in addition to performing split-state tampering, is also allowed some limited interaction between the two states.

There have been some results that have obtained non-malleable codes against continuous tampering in the split-state model [27], [28].

[1]A previous version of [19] claimed a constant rate non-malleable codes in the split-state model. Unfortunately, Li [9] found a mistake in the proofs of one of the lemmas in the paper, and though the lemma is believable, currently the construction is secure only under a plausible conjecture.

In addition to the already-mentioned results, several recent works [29], [30], [31], [32], [33], [34], [35], [36], [37], [38] either used or built non-malleable codes for various families $\mathcal{F}$, but did not concentrate on the split-state model, which is our focus here.

The notion of non-malleability was introduced by Dolev, Dwork and Naor [39], and has found many applications in cryptography. Traditionally, non-malleability is defined in the computational setting, but recently non-malleability has been successfully defined and applied in the information-theoretic setting (generally resulting in somewhat simpler and cleaner definitions than their computational counterparts). For example, in addition to non-malleable codes studied in this work, the work of Dodis and Wichs [40] defined the notion of non-malleable extractors as a tool for building round-efficient privacy amplification protocols.

Finally, the study of non-malleable codes falls into a much larger cryptographic framework of providing countermeasures against various classes of tampering attacks. This work was pioneered by the early works of [41], [42], [43], and has since led to many subsequent models. We do not list all such tampering models, but we refer to [44], [2] for an excellent discussion of various such models.

Organization of the Paper.: In Section II, we provide an overview of our construction and our proof techniques. In Section III we give formal definitions of non-malleable reductions and their connection to non-malleable codes. In Section IV, we state the properties of the non-malleable code construction from [4] needed for our proofs. In Section V, we provide our construction in a series of steps. In particular, in Subsection V-A, we give a construction of non-malleable codes against 2-lookahead tampering. In Subsection V-B, we show how to extend this construction to obtain a non-malleable code against 2-lookahead and forgetful tampering. Finally, in Subsection V-C, we show how this yields a constant-rate non-malleable code in the split-state model. Full proofs can be found in full version of this paper.

II. Overview of the Construction and Techniques

A. Non-malleable reductions

In [19], the notion of non-malleable codes w.r.t. to a tampering family $\mathcal{F}$ was generalized to a more versatile notion of *non-malleable reductions* from $\mathcal{F}$ to $\mathcal{G}$. Intuitively, $(\mathcal{F}, \mathcal{G}, \varepsilon)$-non-malleable reduction allows one to encode a value x with $y \leftarrow E(x)$, so that the tampering of y by $y' = f(y)$ for $f \in \mathcal{F}$ gets "reduced" (by the decoding function $D(y') = x'$) to tampering *with x itself* via some (distribution over) $G \in \mathcal{G}$, that is $D(f(y)) \approx_\varepsilon G(x)$. For formal definitions and more details we refer to the Section III-A.

Notice that the notion of *non-malleable code* w.r.t. $\mathcal{F}$, is simply a reduction from $\mathcal{F}$ to the family of "trivial manipulation functions" consisting of identity function and

constant functions (see Def. 3.2 for formal definition). The utility of non-malleable reductions comes from the natural composition theorem that was shown in [19], which allows to construct a non-malleable code by gradually make our tampering families simpler and simpler, until we eventually end up with a family of trivial manipulation functions mentioned above.

B. Important tampering families

Let us briefly introduce few important function families.

- t-*split-state model*. Is a family where the attacker can apply t arbitrarily correlated functions $h_1, \ldots, h_t$ to t separate parts of memory (but, of course, each h_i can only be applied to the i-th part individually).
- *forgetful* family. Memory is split into t parts. Adversary can apply any tampering function that depends only on $(t-1)$ parts. I.e. adversary has to 'forget' at least one part of the memory (it is up to him which will be forgotten), besides that, it is not restricted in any way.
- *lookahead manipulation* family. There are t parts of memory $(x_1, \ldots, x_t)$, adversary tampers with first part $x'_1 = f_1(x_1)$, then with next parts while knowing all previous parts: $x'_i = f_i(x_1, \ldots, x_i)$. In other words, x'_1 depends on x_1, x'_2 depends on both x_1 and x_2, and in general, x'_i depends on $x_1, \ldots, x_i$.
- $2-$*lookahead manipulation* family. Here t parts of memory (for t even) are split into two groups: $(x_1, \ldots, x_{t/2})$ and $(x_{t/2+1}, \ldots, x_t)$, each of the groups is tampered independently, within the groups adversary applies *lookahead manipulations*. That means that x'_i depends on $x_1, \ldots, x_i$ for $i = 1, \ldots, t/2$, and for $i = t/2+1, \ldots, t$ we get that x'_i depends on $x_{t/2+1}, \ldots, x_i$.

C. Reduction from $2-$split state model

Theorem 2.1: [Informal]. It was shown in [19] that there is an efficient non-malleable reduction from the $2-$*split state* tampering family to a union of the *forgetful* tampering family and the $2-$*lookahead* tampering family. Moreover this reduction has a constant rate, i.e. the size of the codeword is linear in the size of the message. (for a formal statement see Thm. 5.1).

By above theorem, to build explicit non-malleable code in the $2-$*split-state model* it suffices to build non-malleable code against sum of *forgetful* and *double lookahead* families. Moreover, if our code has a constant rate then induced code in $2-$*split state model* will have a constant rate.

D. Non-malleable code with rate zero but with additional properties

A key ingredient in our construction will be the non-malleable code construction from [4]. Even though, this construction has rate $\frac{1}{n^4} \to 0$ ([4] claimed a rate of $\frac{1}{n^6}$ but it was shown to be $\frac{1}{n^4}$ using the same construction in [17]), it has some additional properties that allow us to bootstrap it to obtain a constant rate code against 2-lookahead tampering. In particular, the non-malleable code from [4] has two additional properties that are crucial for our construction:

- *Leakage resilient storage.* The code in [4] is built on inner product function, which is a strong 2-source extractor. Thus it has excellent leakage resilience properties. Even if the adversary sees one state, and obtains a lot of independent leakage from the other state, he still won't be able to say anything about the message[2].
- *Detection of bijective tampering.* The adversary cannot hope to retain a lot of information about the codeword in the tampered codeword, and still be able to tamper successfully. If the two tampered states carries a lot of information about original states then we are guaranteed that either the tampered codeword decodes to the original message is preserved or the codeword is not valid (and decodes to an error message, $\perp$).

The two properties mentioned above together mean the following. A valid codeword encoding a valid message different from the original message can be produced only if the tampered codeword lost a significant fraction of the information about the original states. In fact, the tampered states carry so little[3] information about the original states, that the tampered states and some additional leakage of the codeword put together are still not enough to retrieve any information about the original message.

E. Non-malleable code (NMC) against lookahead tampering - the construction

For clarity, we will first discuss the construction of a non-malleable code against *lookahead* tampering alone (without resilience to *forgetful* tampering). The construction is described in Figure 1. The main ingredients are:

- (Enc, Dec) a non-malleable code from [4],
- Ext_2 and Ext_3 are inner product (strong 2-source) extractors with appropriate parameters,
- *Checks* is an appropriate 2-universal (collision resilient) hash function.

We would like to emphasize here that the reason this construction has a constant rate even though $(\mathsf{Enc}, \mathsf{Dec})$ was not constant rate is because we are using $\mathsf{Dec}(L, R)$ to only store "checks" to detect any tampering in X, Y, A, B and not to actually store the message.

F. A few tampering scenarios

We look at a few tampering scenarios to give the intuition behind the construction. We will write L', R' to denote states

[2]The leakage resilience is meant to be exactly as described here. We are not referring to *leakage resilient non-malleable code* as defined in [2] and [26].

[3]The information rate of the tampered codeword to original codeword is way more then $1/2$, but the rate of information required to retrieve the message is close to 1, the gap between the two is of a constant order.

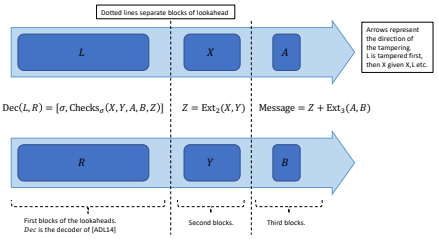

Figure 1. The decoding algorithm of NMC against lookahead tampering.

L and R after tampering. We remind that, since $\mathsf{Enc}, \mathsf{Dec}$ is a non-malleable code, adversary can only preserve the decoding $\mathsf{Dec}(L,R)$ or overwrite it completely or create an invalid codeword.

Scenario 1*::* The adversary preserves checks encoded with non-malleable code from [4], i.e. $\mathsf{Dec}(L',R') = \mathsf{Dec}(L,R)$.
In this case any tampering with X, Y, A, B will be detected via the checks in $\mathsf{Dec}(L',R') = \mathsf{Dec}(L,R)$.
There is a technical issue with making this formal: the adversary tampers with X, A after seeing L and with Y, B after seeing R, but we choose the lengths of the states appropriately so that we can model everything as a small leakage from L and R and the secrecy of checks is preserved i.e. $X, X', Y, Y', A, A', B, B'$ together do not reveal any information about the random seed σ, and thus the probability that checks of original and tampered parts of the codeword will collide is negligible.

Scenario 2*::* The adversary overwrote checks i.e. $\mathsf{Dec}(L',R') \neq \mathsf{Dec}(L,R)$ is a valid codeword.
In this case, by the NMC properties, we get that after the decoding, the modified seed σ' and the corresponding check value c' are independent of the original values. However, we can not rule out the possibility that the adversary knows both σ' and c' (e.g., if he completely overwrote L, R by something unrelated). Now we have two sub-scenarios:

Scenario 2.1*::* Adversary lost some information about X or Y.
In this case, either X can not be fully recovered from L', X', or Y cannot be fully recovered from R', Y'. Then by the strong 2-source extractor property of Ext_2, the adversary has lost all information about Z and, as a consequence, the tampered codeword is uncorelated with the message.

Scenario 2.2*::* Adversary preserved information about X and Y.
We know that σ' and c' are controlled by the adversary but completely independent of the original checks. We also know that X' and Y' must have high min-entropy, or else they wouldn't carry information about X, Y. In order to argue tamper detection in this case, we need to go into the details of the definition of our check. Our check function consists of two checks, one is a collision resilient hash function on $X\|Y\|A\|B$ using half of σ as a seed. The other half of σ say σ_2 is used for a check on Z,

$$\textit{Check-Z}_{\sigma_2}(Z) := c_2 = Z_1 \oplus \sigma_2,$$

where Z_1 is an appropriate length prefix of Z.
In this scenario, the check on Z comes into play. We have that X and Y are sampled independently and uniform at random and $Z := \mathsf{Ext}_2(X,Y)$. We can argue that X', Y' are high-entropic and independent even given L, R (and hence given L', R'), and thus $Z' = \mathsf{Ext}_2(X', Y')$ is close to uniform and independent of the message given L', R'. Notice that the check for Z (or Z') has the property that for any fixing of σ_2' and c_2', the probability that for U uniform $U \oplus \sigma_2' = c_2'$ is negligible. Since, as we just discussed, Z' is close to uniform and independent of σ_2' and c_2' the probability that $Z' \oplus \sigma_2' = c_2'$ is negligible, and hence the tampering is detected by the decoding algorithm with overwhelming probability.

Scenario 2.2′*::* Imagine we are in the scenario 2.2 mentioned above, but we did not have the last parts A, B in our encoding function, i.e. the message is simply Z instead of $Z \oplus \mathsf{Ext}_3(A,B)$.
Notice that in the previous scenario, we used that X' and Y' are independent of the message and since they have high entropy, $Z' = \mathsf{Ext}_2(X', Y')$ is close to uniform and independent of the message. We did not show (and did not need) independence of Z and Z', since Z was sampled uniformly and independently from the message. Now, however, if the message is simply Z, then we cannot say any more that X' and Y' are independent or that $\mathsf{Ext}_2(X', Y')$ is uniform and independent of the message.

Our encoding algorithm appends random variables A, B, respectively as the last part in both lookaheads to ensure that X, Y are independent of the message m.

G. Why do we need additional properties of [4]?

In Scenarios 2.1 and 2.2 we need to argue that all information about original $\mathsf{Dec}(L,R)$ has been lost. This is not quite as trivial as it seems at the first glance. $\mathsf{Dec}(L',R')$ might be independent of $\mathsf{Dec}(L,R)$ but L', R', X', Y', A', B' together might carry some information about the original checks. Imagine, for example, that $\mathsf{Dec}(L',R')$ is fixed and independent of $\mathsf{Dec}(L,R)$. We have to exclude the possibility that the rest of the codeword (X', Y', A', B') fulfills the checks from $\mathsf{Dec}(L',R')$ if and only if the first bit of $\mathsf{Dec}(L,R)$ is equal to $\mathsf{Dec}(L',R')$. If we didn't rule out this possibility, then $\mathsf{Dec}(L',R')$ is not independent of $\mathsf{Dec}(L,R)$ conditioned on the codeword being valid. This is

not only a technical issue, but given that $\mathsf{Dec}(L, R)$ encodes a check on Z, this would raise a concern about the security of the whole scheme.

As we discussed earlier, [4] has the property that if $\mathsf{Dec}(L, R) \neq \mathsf{Dec}(L', R')$ then L', R' form a valid codeword if and only if it doesn't carry much information about L, R, i.e. the tampering function on L and R are very far from bijective. Then we simply consider X', Y', A', B' as an extra leakage[4] and we can show that L', R', X', Y', A', B' together do not carry enough information about L, R and thus are independent of the original checks.

Another place where we use an additional properties of [4] is in Scenario 1 where we need to show that $X, X', Y, Y', A, A', B, B'$ doesn't carry any information about $\mathsf{Dec}(L, R)$. here the argument is much more straight forward and follows from the *leakage resilient storage* property of the code.

H. Last step: resilience to forgetful tampering

Finally in Section V-B, we show how to add resilience to *forgetful* tampering. We need to modify our construction so that forgetting about any of the states leads to forgetting the message. This means that the message can not be retrieved using only 5 of the states (i.e. that the construction is 6 out of 6 secret sharing). First notice:

1) *Forgetting A or B:* by the property of inner product forgetting A or B immediately leads to losing any information about $\mathsf{Ext}_3(A, B)$ and thus we lose information about the original message.
2) *Forgetting X or Y:* Losing information about Z immediately leads to losing any information about message. However, situation is more complicated, since L, R encode the check on Z. Thus, not all information about Z is lost and some partial information about the message can be retrieved.
3) *Forgetting L or R:* forgetting any of the two is inconsequential, we can fully retrieve the message simply by calculating $\mathsf{Ext}_2(X, Y) + \mathsf{Ext}_3(A, B)$.

Point 2:: This problem can be easily resolved, notice that L, R carry only information about the prefix of Z. Thus the only information about the encoded message the adversary can retrieve is the prefix of that message. To fix this, we simply require that the prefix of the message is 0^{2t} where $2t$ is the length of the check, which essentially means that the message is encoded only on the suffix i.e $\mathsf{Ext}_2(X, Y) + \mathsf{Ext}_3(A, B) = 0^{2t}||message$.

Point 3:: Notice that now, after the above fix, forgetting X or Y leads to forgetting the whole message. Using this fact, we can easily fix the issue from point 3. We will split X (and Y) into 2 states $X = X_1 + X_2$ (and $Y = Y_1 + Y_2$). Notice that forgetting any of the four X_1, X_2, Y_1, Y_2 leads to forgetting the message. We will exploit this fact simply by extending states holding L and R to store $L\|X_1$ and $R\|Y_1$ respectively, while states previously storing X and Y will only store X_2 and Y_2 respectively (see Figure 2 for the diagram of modified decoder function).

[4]Where X', A' is bounded leakage from L and Y', B' is a bounded leakage from R.

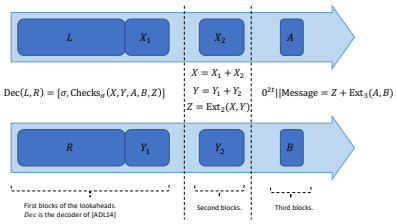

Figure 2. The decoding algorithm of NMC against lookahead and forgetful tampering .

After the above modifications.: Notice that:

1) *Forgetting A or B:* we still immediately lose the message.
2) *Forgetting X_2 or Y_2:* we lose the information about Z which immediately leads to losing any information about the message.
3) *Forgetting $L\|X_1$ or $R\|Y_1$:* forgetting any of the two means forgetting X_1 or Y_1 which has the same consequences as forgetting X_2 or Y_2.

III. Definitions and Formal Theorem Statements

A. Non-malleable Codes and Reductions

Definitions. In [19], the notion of non-malleable codes w.r.t. to a tampering family $\mathcal{F}$ (see [1]) was generalized to a more versatile notion of *non-malleable reductions* from $\mathcal{F}$ to $\mathcal{G}$. The following definitions are taken from [19].

Definition 3.1 (non-malleable reduction): Let $\mathcal{F} \subset A^A$ and $\mathcal{G} \subset B^B$ be some classes of functions (which we call manipulation *functions). We will write:*

$$(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon)$$

and say $\mathcal{F}$ reduces to *$\mathcal{G}$, if there exist an efficient randomized* encoding *function $E : B \to A$, and an efficient deterministic* decoding *function $D : A \to B$, such that (a) for all $x \in B$, we have $D(E(x)) = x$, and (b) for all $f \in \mathcal{F}$, there exists G such that for all $x \in B$,*

$$\Delta\Big(D(f(E(x)))\ ;\ G(x)\Big) \leq \varepsilon, \tag{1}$$

where G is a distribution *over $\mathcal{G}$, and $G(x)$ denotes the distribution $g(x)$, where $g \leftarrow G$.*

The pair (E, D) *is called* $(\mathcal{F}, \mathcal{G}, \varepsilon)$-non-malleable reduction.

Intuitively, $(\mathcal{F}, \mathcal{G}, \varepsilon)$-non-malleable reduction allows one to encode a value x by $y \leftarrow E(x)$, so that tampering with y by $y' = f(y)$ for $f \in \mathcal{F}$ gets "reduced" (by the decoding function $D(y') = x'$) to tampering *with* x *itself* via some (distribution over) $g \in \mathcal{G}$.

In particular, the notion of *non-malleable code* w.r.t. $\mathcal{F}$, is simply a reduction from $\mathcal{F}$ to the family of "trivial manipulation functions" NM_k defined below.

Definition 3.2: Let NM_k *denote the set of* trivial manipulation functions *on* k*-bit strings, which consists of the identity function* $I(x) = x$ *and all constant functions* $f_c(x) = c$*, where* $c \in \{0,1\}^k$.

We say that a pair (E, D) *defines an* $(\mathcal{F}, k, \varepsilon)$-non-malleable code*, if it defines a* $(\mathcal{F}, \mathsf{NM}_k, \varepsilon)$*-non-malleable reduction.*

The utility of non-malleable reductions comes from the following natural composition theorem that was shown in [19], which allows to gradually make our tampering families simpler and simpler, until we eventually end up with a non-malleable code (corresponding to the trivial family NM_k).

Theorem 3.1 (Composition): If $(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon_1)$ and $(\mathcal{G} \Rightarrow \mathcal{H}, \varepsilon_2)$, then $(\mathcal{F} \Rightarrow \mathcal{H}, \varepsilon_1 + \varepsilon_2)$.

We will also need the following trivial observation.

Observation 3.1 (Union): Let (E, D) be an $(\mathcal{F}, \mathcal{H}, \varepsilon)$ and a $(\mathcal{G}, \mathcal{H}, \varepsilon')$ non-malleable reduction . Then (E, D) is an $(\mathcal{F} \cup \mathcal{G}, \mathcal{H}, \max(\varepsilon, \varepsilon'))$ non-malleable reduction .

USEFUL TAMPERING FAMILIES. We define several natural tampering families we will use in this work. For this, we first introduce the following "direct product" operator on tampering families:

Definition 3.3: Given tampering families $\mathcal{F} \subset A^A$ *and* $\mathcal{G} \subset B^B$*, let* $\mathcal{F} \times \mathcal{G}$ *denote the class of functions* h *from* $(A \times B)^{A \times B}$ *such that*

$$h(x) = h_1(x_1) \| h_2(x_2)$$

for some $h_1 \in \mathcal{F}$ *and* $h_2 \in \mathcal{G}$ *and* $x = x_1 \| x_2$*, where* $x_1 \in A, x_2 \in B$.

We also let $\mathcal{F}^1 := \mathcal{F}$*, and, for* $t \geq 1$*,* $\mathcal{F}^{t+1} := \mathcal{F}^t \times \mathcal{F}$.

We can now define the following tampering families:

- $\mathcal{S}_n = (\{0,1\}^n)^{\{0,1\}^n}$ denote the class of *all* manipulation functions on n-bit strings.
- Given $t > 1$, $\mathcal{S}_n^t$ denotes the tampering family in the *t-split-state model*, where the attacker can apply t arbitrarily correlated functions $h_1, \ldots, h_t$ to t separate, n-bit parts of memory (but, of course, each h_i can only be applied to the i-th part individually).
- $\mathcal{FOR}^t_{n_1, n_2, \ldots, n_t}$ denotes *forgetful* family. It is applied to t parts of memory of length n_i but the output value can depend only on $(t-1)$ parts. More precisely: Let $x \in \{0,1\}^n$ be a bit vector and $x_i \in \{0,1\}^{n_i}$ denote i-th block of n bits. For any $h \in \mathcal{FOR}^t_{n_1, n_2, \ldots, n_t}$ there exist a subset $S \subset \{1, 2, \ldots, t\}$ of size $(t-1)$ such that $h(x)$ can be evaluated from x_S. Besides that, it is not restricted in any way.
- Finally, $\mathcal{LA}^{\leftarrow t}_{n_1, \ldots, n_t}$, where $n = n_1 + \cdots + n_t$ denotes the class of *lookahead manipulation functions* l that can be rewritten as $l = (l_1, \ldots, l_t)$, for $l_i : \{0,1\}^{n_1 + \cdots + n_i} \to \{0,1\}^{n_i}$, and where

$$l(x) = l_1(x_1) \| \ldots \| l_t(x_1, \ldots, x_t)$$

for $x_i \in \{0,1\}^{n_i}$. In other words, if $l(x_1, \ldots, x_t) = y_1, \ldots, y_t$, then y_1 depends on x_1, and y_2 depends on both x_1 and x_2, and in general, y_i depends on $x_1, \ldots, x_i$.

IV. THE NON-MALLEABLE CODE CONSTRUCTION FROM [4], [7], [17]

We will need the construction of non-malleable codes in the split-state model from [4], [7], [17]. We need a little more than just the non-malleability property of the construction. The following theorem states and proves the precise property of the code that we require for our construction. The proof appears in the full version and is a rather straightforward modification of the proofs in [4], [7], [17]. The reader can safely skip this section and return to it when referenced.

Theorem 4.1: There exists an efficient construction $(\mathsf{Enc}, \mathsf{Dec})$ of an ε-non-malleable code in the split-state model from $\{0,1\}^{7t}$ to $\mathbb{F}_p^n \times \mathbb{F}_p^n$ with $\varepsilon = 2^{-\Omega(t)}$, $n = O(t^5)$ and $p \leq 2^{O(t)}$ is a prime[5]. Furthermore, for any functions $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$, and $g : \mathbb{F}_p^n \to \mathbb{F}_p^n$, the space $\mathbb{F}_p^n \times \mathbb{F}_p^n$ can be partitioned into $\mathbb{F}_p^n \times \mathcal{R}_0$, $\mathcal{L}_0 \times (\mathbb{F}_p^n \setminus \mathcal{R}_0)$, $(\mathcal{L}_{\textit{same},i} \times \mathcal{R}_{\textit{same},i})_{1 \leq i \leq q}$, $(\mathcal{L}_{\perp,i} \times \mathcal{R}_{\perp,i})_{1 \leq i \leq r}, \mathsf{Rem}$ such that the following hold.

- $\mathcal{L}_0 = \{\ell \in \mathbb{F}_p^n : |f^{-1}(f(\ell))| \geq p^{0.45n}$, and $\mathcal{R}_0 = \{r \in \mathbb{F}_p^n : |g^{-1}(g(r))| \geq p^{0.45n}$, i.e., $\mathcal{L}_0$ is the subset of $\mathbb{F}_p^n$ on which f is "far from" being a bijective function, and $\mathcal{R}_0$ is the subset of $\mathbb{F}_p^n$ on which g is "far from" being a bijective function.
- For all $m \in \{0,1\}^{7t}$, for all i, $\Pr[\mathsf{Dec}(\mathsf{Enc}(m)) = m \mid \mathsf{Enc}(m) \in \mathcal{L}_{\textit{same},i} \times \mathcal{R}_{\textit{same},i}] = 1$, and $|\mathcal{L}_{\textit{same},i} \times \mathcal{R}_{\textit{same},i}| \geq p^{1.9n}$.
- For all $m \in \{0,1\}^{7t}$, for all i, $\Pr[\mathsf{Dec}(\mathsf{Enc}(m)) = \perp \mid \mathsf{Enc}(m) \in \mathcal{L}_{\perp,i} \times \mathcal{R}_{\perp,i}] = 1 - \varepsilon$, and $|\mathcal{L}_{\perp,i} \times \mathcal{R}_{\perp,i}| \geq p^{1.9n}$.
- For all $m \in \{0,1\}^{7t}$, $\Pr[\mathsf{Enc}(m) \in \mathsf{Rem}] \leq \varepsilon$.
- The decoding function $\mathsf{Dec}(\ell, r) := h(Ext(\ell, r))$ is a deterministic function of the inner product two-source extractor function from $\mathbb{F}_p^n \times \mathbb{F}_p^n$ to $\mathbb{F}_p$.

[5]The constant 7 in this Theorem statement are chosen to match those required in our results. There is some freedom in the choice of parameters in [4], [7], [17], and so the result of this theorem follows for an appropriate choice of t.

V. Our constructions and the main result

It was shown in [19] that

Theorem 5.1: For any q, there is an $n = O(q)$ such that

$$(S_n^2 \Rightarrow \mathcal{LA}^{\leftarrow 3}_{q,q,q} \times \mathcal{LA}^{\leftarrow 3}_{q,q,q} \cup \mathcal{FOR}^{6}_{q,q,q,q,q,q}, 2^{-\Omega(q)}) \ .$$

So, now we construct non-malleable codes for the tampering family $\mathcal{LA}^{\leftarrow 3}_{q,q,q} \times \mathcal{LA}^{\leftarrow 3}_{q,q,q} \cup \mathcal{FOR}^{6}_{q,q,q,q,q,q}$. In Section V-A, we give a non-malleable code against 2-lookahead tampering family, and in Section V-B, we show how to extend it to include the forgetful tampering family.

A. A non-malleable code against 2-lookahead tampering

Theorem 5.2: There exists a $2^{-k^{\Omega(1)}}$-non-malleable code for k-bit messages against the tampering family $\mathcal{LA}^{\leftarrow 3}_{O(k),O(k),O(k)} \times \mathcal{LA}^{\leftarrow 3}_{O(k),O(k),O(k)}$.

Construction.: Our construction (E, D) depicted in Figure 1 that achieves the above result is as follows[6].

Encode: Given $m \in \{0,1\}^k$, we do the following.

- Let Ext_3 be the inner product function from $\mathbb{F}_{2^k}^5 \times \mathbb{F}_{2^k}^5 \to \mathbb{F}_{2^k}$. Let A, B be chosen uniformly at random from $\{0,1\}^{5k}$.
- Let Ext_2 be the inner product function from $\mathbb{F}_{2^k}^{25} \times \mathbb{F}_{2^k}^{25} \to \mathbb{F}_{2^k}$. Sample $X, Y \in \{0,1\}^{25k}$ uniformly at random, conditioned on $z := \mathsf{Ext}_2(X,Y) = m \oplus \mathsf{Ext}_3(A,B)$.
- Let σ_1, σ_2 be $2t$-bit strings sampled uniformly at random for $t = \Theta(k^{1/5})$.
- Let $C_1, C_2 : \{0,1\}^{2t} \times \{0,1\}^{25k} \to \{0,1\}^t$, and $C_3, C_4 : \{0,1\}^{2t} \times \{0,1\}^{5k} \to \{0,1\}^t$ be $2^{-t/2}$-almost universal hash functions[7]. Also, let $z = z_1 \| z_2$ where $|z_1| = 2t$.
- Let $s = \sigma_1, \sigma_2, c_1 := C_1(\sigma_1, X) \| C_2(\sigma_1, Y) \| C_3(\sigma_1, A) \| C_4(\sigma_1, B)$, $c_2 := z_1 \oplus \sigma_2$.
- Let $L, R := \mathsf{Enc}(s)$, where $(\mathsf{Enc}, \mathsf{Dec})$ be a non-malleable code in the split state model given by Theorem 4.1 from $\{0,1\}^{10t}$ to $\mathbb{F}_p^n \times \mathbb{F}_p^n$ where $n = \lceil \frac{100k}{\log p} \rceil$.
- Output (L, X, A) as the first part of the codeword, and (R, Y, B) as the second part.

For the rest of the paper, we denote $C_1(\sigma_1, X) \| C_2(\sigma_1, Y) \| C_3(\sigma_1, A) \| C_4(\sigma_1, B)$ by $C(\sigma_1, X \| Y \| A \| B)$.[8]

Decode: Given $(L, X, A), (R, Y, B)$ we do the following.

- Compute $s = \mathsf{Dec}(L, R)$, and $z = Ext_2(X, Y)$.
- If $s = \perp$, output $\perp$, else let $s = \sigma_1, \sigma_2, c_1, c_2$.
- If $z_1 \neq c_2 \oplus \sigma_2$, where z_1 is the first $2t$ bits of z, or $c_1 \neq C(\sigma_1, X \| Y \| A \| B)$, output $\perp$.
- Else output $z \oplus Ext_3(A, B)$.

Overview of the proof.: Given a message $m \in \{0,1\}^k$, let $E(z) = (L, X, A), (R, Y, B)$. Let $f_1, g_1 : \mathbb{F}_p^n \to \mathbb{F}_p^n$, $f_2, g_2 : \mathbb{F}_p^n \times \{0,1\}^{25k} \to \{0,1\}^{25k}$, and $f_3, g_3 : \mathbb{F}_p^n \times \{0,1\}^{30k} \to \{0,1\}^{5k}$ be arbitrarily chosen functions, and let $L' = f_1(L)$, $R' = g_1(R)$, $X' = f_2(L, X)$, $Y' = g_2(R, Y)$, $A' = f_3(L, X, A)$, $B' = g_3(R, Y, B)$. Also, let $z', z_1', z_2', \sigma_1', \sigma_2', c_1', c_2'$ be the corresponding tampered values.

As is the case with almost all proofs for non-malleable code constructions, our proof proceeds by first partitioning the ambient space $\mathbb{F}_p^n \times \{0,1\}^{30k} \times \mathbb{F}_p^n \times \{0,1\}^{30k}$ depending on the functions $f_1, g_1, f_2, g_2, f_3, g_3$. We then argue that for each partition, as long as the partition is large enough, conditioned on the random variables L, X, A, R, Y, B being restricted to be in that partition, we can show that either the codeword remains unchanged after tampering, or $D((L', X', A'), (R', Y', B')) = \perp$ with high probability, or the tampered codeword is almost independent of the message m, (i.e., it reveals no information about the message m).

We first consider the partition where $\mathsf{Dec}(L', R') = \mathsf{Dec}(L, R)$. In this case, notice that if X, Y, A, B are changed then with high probability, $C(\sigma_1, X \| Y \| A \| B) \neq C(\sigma_1, X \| Y \| A \| B)$, and so the decoding algorithm outputs $\perp$ with high probability. On the other hand, if X, Y, A, B are unchanged, then the decoder outputs *same*. For the formal proof, we need to deal with the dependence between various random variables, and the detailed proof can be found in the full version.

We next consider the partition where $\mathbf{H}_\infty(L') + \mathbf{H}_\infty(R') \gg n \log p$, and $\mathsf{Dec}(L', R') \neq \mathsf{Dec}(L, R)$. In this case, by Theorem 4.1, we have that $\mathsf{Dec}(L', R') = \perp$ with high probability.

This leaves us with the partitions where one of $\mathbf{H}_\infty(L|L')$ or $\mathbf{H}_\infty(R|R')$ (say $\mathbf{H}_\infty(L|L')$) is at least $0.45n \log p$. Notice that here we are using the fact that for an appropriate choice of partitions, we have that $\mathbf{H}_\infty(L') + \mathbf{H}_\infty(L|L') \approx n \log p$ for L chosen uniformly from that partition. This in particular means that $\mathbf{H}_\infty(L|L', X', A') \geq 45k - 25k - 5k = 15k$. Thus, again using the observation that $\mathsf{Dec}(L, R)$ is a deterministic function of a strong two-source extractor $h(\mathsf{Ext}(L, R))$, we have that $\mathsf{Dec}(L, R)$ is independent of $L', R', X', Y', A', B', X, Y, A, B$. At this point, we can fix

[6] We note here, that the construction is efficient. Please notice that since Ext_2 and Ext_3 are just inner product extractors they are efficiently invertible, in particular for any output z it is possible to efficiently sample X, Y uniformly random fulfilling $\mathsf{Ext}_2(X, Y) = z$.

[7] A function $C : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^t$ is called an *ε-almost universal hash function* if for any $x, y \in \{0,1\}^n$ such that $x \neq y$, $\Pr_{R \leftarrow \{0,1\}^s}(C(R, x) = C(R, y)) \leq \varepsilon$ The following is a standard construction of a polynomial evaluation ε-universal hash function. The parameters are from [40]. For any $n, t > 2 \log n$, there exists an efficiently computable $2^{-t/2}$-almost univeral hash function $C : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^t$ with $s = 2t$.

[8] We require 4 almost universal hash functions instead of a single, joint check in order to ensure that X, Y are independent given $C(\sigma_1, X \| Y \| A \| B)$, and A, B are independent given $C(\sigma_1, X \| Y \| A \| B)$.

L, R, thereby fixing $L' = \ell', R' = r'$.

Thus, X', Y' are deterministic functions of X, Y, respectively. Now we further partition the space $\{0,1\}^{25k} \times \{0,1\}^{25k}$ based on the functions f_2, g_2. First we consider the case where $\mathbf{H}_\infty(X') + \mathbf{H}_\infty(Y') \gg 26k$. In this case, by using the fact that inner product is a strong 2-source extractor, and noting that X, Y, and hence X', Y' is independent of the message m, we have that z' (and hence z'_1 is close to uniform and independent of the message m, and ℓ', r'. Thus, the probability that $\sigma'_2 = c'_2 \oplus z'_1$ is negligible, and hence the decoding algorithm outputs $\perp$ with high probability.

The only remaining case is when one of $\mathbf{H}_\infty(X|X')$ or $\mathbf{H}_\infty(Y|Y')$ (say $\mathbf{H}_\infty(X|X')$) is at least $10k$, in which case $\mathbf{H}_\infty(X|X', A, A') \geq 5k$, and hence by the strong extractor property of the inner product, we have that $z = \mathsf{Ext}_2(X, Y)$ is independent of X', Y', A', B', A, B and hence is independent of the tampered codeword (since we already fixed L', R'). The tampered codeword is thus independent of the message[9].

B. A non-malleable code secure against 2-lookahead and forgetful tampering

Theorem 5.3: There is an $2^{-k^{\Omega(1)}}$- non-malleable code for k-bit messages against the tampering family $\mathcal{LA}^{\leftarrow 3}_{O(k),O(k),O(k)} \times \mathcal{LA}^{\leftarrow 3}_{O(k),O(k),O(k)} \cup \mathcal{FOR}^{6}_{O(k),O(k),O(k),O(k),O(k),O(k)}$.

Proof: We modify the construction in Section V-A to get non-malleability against the forgetful family.

Construction.: Our construction (E^*, D^*) depicted in Figure 2 that achieves the above result is as follows.

Encode: Our encoding algorithm is as follows.

- Given a message $m^* \in \{0,1\}^{k-2t}$, let $m = 0^{2t} \| m^*$.
- Let $X, A, Y, B, L, R, s, \sigma_1, \sigma_2, z, z_1, z_2, c_1, c_2$ be as in the encoding of $E(m)$, where E is the encoding algorithm from Section V-A.
- Choose X_1, Y_1 uniformly at random from $\{0,1\}^{25k}$, and let $X_2 = X \oplus X_1, Y_2 = Y \oplus Y_1$.
- Output the three parts of the first lookahead as $((L, X_1), X_2, A)$, and the three parts of the second lookahead as $((R, Y_1), Y_2, B)$.

Decode: The decoding algorithm is as follows.

- Given $((L, X_1), X_2, A), ((R, Y_1), Y_2, B)$, compute $X = X_1 \oplus X_2$, and $Y = Y_1 \oplus Y_2$.
- Then $D^*\left(((L, X_1), X_2, A), ((R, Y_1), Y_2, B)\right) := D\left((L, X, A), (R, Y, B)\right)$, where D is as defined in Section V-A.

We now give a simple argument that shows that this construction is secure against the tampering family $\mathcal{LA}^{\leftarrow 3}_{O(k),O(k),O(k)} \times \mathcal{LA}^{\leftarrow 3}_{O(k),O(k),O(k)} \cup \mathcal{FOR}^{6}_{O(k),O(k),O(k),O(k),O(k),O(k)}$ assuming the construction given in Theorem 5.2 is secure against the tampering family $\mathcal{LA}^{\leftarrow 3}_{O(k),O(k),O(k)} \times \mathcal{LA}^{\leftarrow 3}_{O(k),O(k),O(k)}$.

Non-malleability against 2-lookahead tampering.: We first argue security against 2-lookahead tampering. Let the tampering functions be $f_1, g_1 : \mathbb{F}_p^n \times \{0,1\}^{25k} \to \mathbb{F}_p^n$, $f_2, g_2 : \mathbb{F}_p^n \times \{0,1\}^{25k} \to \{0,1\}^{25k}$, $f_3, g_3 : \mathbb{F}_p^n \times \{0,1\}^{50k} \to \{0,1\}^{25k}$, $f_4, g_4 : \mathbb{F}_p^n \times \{0,1\}^{55k} \to \{0,1\}^{5k}$, such that $L'_1 = f_1(L, X_1)$, $X'_1 = f_2(L, X_1)$, $X'_2 = f_3(L, X_1, X_2)$, $A' = f_4(L, X_1, X_2, A)$, and $R'_1 = g_1(R, Y_1)$, $Y'_1 = g_2(R, Y_1)$, $Y'_2 = g_3(R, Y_1, Y_2)$, $B' = g_4(R, Y_1, Y_2, B)$, We show the result for every possible fixing of $X_1 = x$ and $Y_1 = y$. We define the functions f^*_1, f^*_2, f^*_3 as $f^*_1(L) := f_1(L, x)$, $f^*_2(L, X) := f_2(L, x) \oplus f_3(L, x, X \oplus x)$, $f^*_3(L, X, A) := f_4(L, x, X \oplus x, A)$, and similarly define g^*_1, g^*_2, g^*_3, which is an attack in $\mathcal{LA}^{\leftarrow 3} \times \mathcal{LA}^{\leftarrow 3}$ against the construction from Theorem 5.2. With this change, the proof is identical to that of Theorem 5.2.

Non-malleability against forgetful tampering.: In order to argue security against forgetful tampering, consider the case where the adversary loses information about one of A or B (say A), but knows $L, R, X_1, X_2, Y_1, Y_2, B$. We assume that A, B, X_1, X_2, Y_1, Y_2 are uniformly distributed and L, R is computed as in the E^* given A, B, X_1, X_2, Y_1, Y_2. In this case, since $\mathbf{H}_\infty(A|C(\sigma_1, X\|Y\|A\|B)) \geq 5k - t$, and A, B are independent given $C(\sigma_1, X\|Y\|A\|B)$, we have that[10]

$$\Delta(\mathsf{Ext}_3(A, B)\ ;\ U_k \mid B, X_1, X_2, Y_1, Y_2, L, R) \leq 2^{-1.5k}\ .$$

For any message m^*, we have that $\mathsf{Ext}_3(A, B) \oplus \mathsf{Ext}_2(X_1, X_2) = m^*$ (respectively $U_k \oplus \mathsf{Ext}_2(X_1, X_2) = m$), and[11] we have that upto statistical distance $2 \cdot 2^{-0.5k}$, $B, X_1, X_2, Y_1, Y_2, L, R$ are independent of the message m.

Similarly, if the adversary loses information about one of X_2 or Y_2 (say X_2), then a similar argument shows that z_2 is uniform and independent of $A, B, X_1, Y_1, Y_2, L, R$, and hence conditioning on $(z_1\|z_2) \oplus \mathsf{Ext}_3(A, B) = 0^{2t}\|m^*$, which implies that upto statistical distance $2^{-\Omega(k)}$, m^* is independent of $A, B, X_1, Y_1, Y_2, L, R$.

Losing one of (L, X_1) or (R, Y_1) (say (L, X_1)) is clearly worse for the adversary, and so the adversary cannot distinguish between the tampered codeword of any two messages. The result follows. ∎

C. Final result via a non-malleable reduction from [19]

Setting $q = 125k$ in Theorem 5.4, and padding the required number of 0's as a prefix to each part of the codeword, we obtain the following

[9] Since $m = \mathsf{Ext}_2(X, Y) + \mathsf{Ext}_3(A, B)$

[10] Recall that $C(\sigma_1, X\|Y\|A\|B)$ is shorthand for $C_1(\sigma_1, X)\|C_2(\sigma_1, Y)\|C_3(\sigma_1, A)\|C_4(\sigma_1, B)$.

[11] Using following lemma: let $X_1, Y_1 \in \mathcal{A}_1$, and $Y_1, Y_2 \in \mathcal{A}_2$ be random variables such that $\Delta((X_1, X_2)\ ;\ (Y_1, Y_2)) \leq \varepsilon$. Then, for any non-empty set $\mathcal{A}' \subseteq \mathcal{A}_1$, we have

$$\Delta(X_2 \mid X_1 \in \mathcal{A}'\ ;\ Y_2 \mid Y_1 \in \mathcal{A}') \leq \frac{2\varepsilon}{\Pr(X_1 \in \mathcal{A}')}\ .$$

Theorem 5.4: There is an $2^{-q^{\Omega(1)}}$- non-malleable code for $k - O(k^{1/5})$-bit messages against the tampering family $\mathcal{LA}^{\leftarrow 3}_{q,q,q} \times \mathcal{LA}^{\leftarrow 3}_{q,q,q} \cup \mathcal{FOR}^{6}_{q,q,q,q,q,q}$.

Theorem 1.1 then follows from Theorem 3.1 and Theorem 5.1.

ACKNOWLEDGMENTS

This work was partially supported by the Singapore National Research Foundation under NRF RF Award No. NRF-NRFF2013-13, the Ministry of Education, Singapore under grants MOE2012-T3-1-009, and MOE2019-T2-1-145.

REFERENCES

[1] S. Dziembowski, K. Pietrzak, and D. Wichs, "Non-malleable codes," in *ICS*. Tsinghua University Press, 2010, pp. 434–452.

[2] F.-H. Liu and A. Lysyanskaya, "Tamper and leakage resilience in the split-state model," in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 517–532.

[3] S. Dziembowski, T. Kazana, and M. Obremski, "Non-malleable codes from two-source extractors," in *Advances in Cryptology-CRYPTO 2013*. Springer, 2013.

[4] D. Aggarwal, Y. Dodis, and S. Lovett, "Non-malleable codes from additive combinatorics," in *STOC*. ACM, 2014.

[5] M. Cheraghchi and V. Guruswami, "Capacity of non-malleable codes," in *ITCS*, 2014.

[6] ——, "Non-malleable coding against bit-wise and split-state tampering," in *TCC*, 2014.

[7] D. Aggarwal, "Affine-evasive sets modulo a prime," *Information Processing Letters*, vol. 115, no. 2, pp. 382–385, 2015.

[8] E. Chattopadhyay, V. Goyal, and X. Li, "Non-malleable extractors and codes, with their many tampered extensions," in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. ACM, 2016, pp. 285–298.

[9] X. Li, "Improved non-malleable extractors, non-malleable codes and independent source extractors," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2017, pp. 1144–1156.

[10] ——, "Non-malleable extractors and non-malleable codes: Partially optimal constructions," *CCC*, 2019.

[11] V. Goyal, O. Pandey, and S. Richelson, "Textbook non-malleable commitments," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, D. Wichs and Y. Mansour, Eds. ACM, 2016, pp. 1128–1141.

[12] V. Goyal and A. Kumar, "Non-malleable secret sharing," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, I. Diakonikolas, D. Kempe, and M. Henzinger, Eds. ACM, 2018, pp. 685–698.

[13] ——, "Non-malleable secret sharing for general access structures," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, ser. Lecture Notes in Computer Science, H. Shacham and A. Boldyreva, Eds., vol. 10991, 2018, pp. 501–530.

[14] D. Aggarwal, I. Damgård, J. B. Nielsen, M. Obremski, E. Purwanto, J. Ribeiro, and M. Simkin, "Stronger leakage-resilient and non-malleable secret-sharing schemes for general access structures," *IACR Cryptology ePrint Archive*, vol. 2018, p. 1147, 2018.

[15] S. Badrinarayanan and A. Srinivasan, "Revisiting non-malleable secret sharing," *IACR Cryptology ePrint Archive*, vol. 2018, p. 1144, 2018.

[16] A. Srinivasan and P. N. Vasudevan, "Leakage resilient secret sharing and applications," *IACR Cryptology ePrint Archive*, vol. 2018, p. 1154, 2018.

[17] D. Aggarwal and J. Briët, "Revisiting the sanders-bogolyubov-ruzsa theorem in f p n and its application to non-malleable codes," in *Information Theory (ISIT), 2016 IEEE International Symposium on*. Ieee, 2016, pp. 1322–1326.

[18] S. Dziembowski and K. Pietrzak, "Intrusion-resilient secret sharing," in *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. IEEE, 2007, pp. 227–237.

[19] D. Aggarwal, Y. Dodis, T. Kazana, and M. Obremski, "Non-malleable reductions and applications," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, 2015, pp. 459–468.

[20] E. Chattopadhyay and D. Zuckerman, "Non-malleable codes in the constant split-state model," *To appear in FOCS*, 2014.

[21] B. Kanukurthi, S. L. B. Obbattu, and S. Sekar, "Four-state non-malleable codes with explicit constant rate," in *Theory of Cryptography Conference*. Springer, 2017, pp. 344–375.

[22] ——, "Non-malleable randomness encoders and their applications," in *EUROCRYPT*. Springer, 2018, pp. 589–617.

[23] D. Gupta, H. K. Maji, and M. Wang, "Constant-rate non-malleable codes in the split-state model," Technical Report Report 2017/1048, Cryptology ePrint Archive, Tech. Rep., 2018.

[24] D. Aggarwal, S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran, "Optimal computational split-state non-malleable codes," in *Theory of Cryptography Conference*. Springer, 2016, pp. 393–417.

[25] S. Faust, P. Mukherjee, J. Nielsen, and D. Venturi, "Continuous non-malleable codes," in *Theory of Cryptography Conference - TCC*. Springer, 2014.

[26] D. Aggarwal, S. Dziembowski, T. Kazana, and M. Obremski, "Leakage-resilient non-malleable codes," in *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, 2015, pp. 398–426.

[27] D. Aggarwal, T. Kazana, and M. Obremski, "Inception makes non-malleable codes stronger," in *Theory of Cryptography Conference*. Springer, 2017, pp. 319–343.

[28] D. Aggarwal, N. Dottling, J. B. Nielsen, M. Obremski, and E. Purwanto, "Continuous non-malleable codes in the 8-split-state model," Cryptology ePrint Archive, Report 2017/357, Tech. Rep., 2017.

[29] H. Chabanne, G. Cohen, J. Flori, and A. Patey, "Non-malleable codes from the wire-tap channel," in *Information Theory Workshop (ITW), 2011 IEEE*. IEEE, 2011, pp. 55–59.

[30] H. Chabanne, G. Cohen, and A. Patey, "Secure network coding and non-malleable codes: Protection against linear tampering," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 2546–2550.

[31] S. G. Choi, A. Kiayias, and T. Malkin, "Bitr: built-in tamper resilience," in *Advances in Cryptology–ASIACRYPT 2011*. Springer, 2011, pp. 740–758.

[32] S. Faust, P. Mukherjee, D. Venturi, and D. Wichs, "Efficient non-malleable codes and key-derivation for poly-size tampering circuits," in *Eurocrypt*. Springer, 2014, to appear.

[33] S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran, "Explicit non-malleable codes resistant to permutations and perturbations," *IACR Cryptology ePrint Archive*, vol. 2014, p. 841, 2014.

[34] ——, "A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations," in *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, 2015, pp. 375–397.

[35] M. Ball, D. Dachman-Soled, M. Kulkarni, and T. Malkin, "Non-malleable codes for bounded depth, bounded fan-in circuits," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 881–908.

[36] S. Faust, K. Hostáková, P. Mukherjee, and D. Venturi, "Non-malleable codes for space-bounded tampering," in *Annual International Cryptology Conference*. Springer, 2017, pp. 95–126.

[37] M. Ball, D. Dachman-Soled, M. Kulkarni, and T. Malkin, "Non-malleable codes from average-case hardness: Decision trees, and streaming space-bounded tampering," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 618–650.

[38] M. Ball, D. Dachman-Soled, S. Guo, T. Malkin, and L.-Y. Tan, "Non-malleable codes for small-depth circuits," in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018, pp. 826–837.

[39] D. Dolev, C. Dwork, and M. Naor, "Nonmalleable cryptography," *SIAM*, vol. 30, pp. 391–437, 2000.

[40] Y. Dodis and D. Wichs, "Non-malleable extractors and symmetric key cryptography from weak secrets," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, M. Mitzenmacher, Ed. Bethesda, MD, USA: ACM, 2009, pp. 601–610.

[41] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology—CRYPTO 2003*, ser. LNCS, D. Boneh, Ed., vol. 2729. Springer-Verlag, 2003.

[42] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin, "Algorithmic Tamper-Proof (ATP) security: Theoretical foundations for security against hardware tampering," in *First Theory of Cryptography Conference — TCC 2004*, ser. LNCS, M. Naor, Ed., vol. 2951. Springer-Verlag, Feb. 19–21 2003, pp. 258–277.

[43] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner, "Private circuits II: Keeping secrets in tamperable circuits," in *Advances in Cryptology—EUROCRYPT 2006*, ser. LNCS, S. Vaudenay, Ed., vol. 4004. Springer-Verlag, 2006, pp. 308–327.

[44] Y. T. Kalai, B. Kanukurthi, and A. Sahai, "Cryptography with tamperable and leaky memory," in *Advances in Cryptology–CRYPTO 2011*. Springer, 2011, pp. 373–390.