Constant-rate Non-malleable Codes in the Split-state Model

Divya Gupta* Hemanta K. Maji†, ‡, § Mingyuan Wang¶, §

November 3, 2017

Abstract

Dziembowski, Pietrzak, and Wichs (ICS–2010) introduced the notion of non-malleable codes as a useful message integrity assurance for scenarios where error-correction or, even, error-detection is impossible. Intuitively, a non-malleable code ensures that the tampered codeword encodes the original message or a message that is entirely independent of the original message. However, if the family of tampering functions is sophisticated enough to include the decoding algorithm itself, then such codes are impossible.

Motivated by several applications like non-malleable secret sharing schemes, one of the fundamental research directions in the field of non-malleable code construction considers encoding the message into k separate states, where $k \geqslant 2$, such that each state is tampered independently by an arbitrary function. The decoding procedure in this k-split-state model, on the other hand, relies on aggregating the information stored across all the k states. The general goal is to reduce the number of states k, thus, protecting from stronger tampering functions, and, simultaneously, achieve high encoding rate, i.e., the ratio of the message-length to the cumulative size of all the k encoded states.

The ideal result for this line of inquiry will be a 2-split-state non-malleable code with rate (close to) $1/2$, the upper-bound to maximum achievable rate. The current state-of-the-art construction, following a sequence of highly influential works guided by this goal, achieves rate $1/\log \ell$ (Li, STOC–2017), where ℓ is the length of the encoded message. Our work contributes to this research effort by constructing the first constant-rate ($\approx 1/3$) non-malleable code in the 3-split-state model, which is only half of the upper-bound on the maximal achievable rate in this model. The primary technical contribution of our work is a general bootstrapping technique to construct non-malleable codes that achieve high rate by leveraging a unique characteristic of the (rate-0) non-malleable code for 2-states provided by Aggarwal, Dodis, and Lovett (STOC–2014) in conjunction with an additional state.

We also study the construction of non-malleable codes in the streaming version of the k-split-state model, i.e., the tampering function of each state encounters the state as a stream, and it tampers each bit of the state based only on the part of the state seen thus far. We show that similar to the general k-split-state model, the maximum achievable rate of a non-malleable code is at most $1 - 1/k$ in the streaming version as well. We construct the first constant-rate ($\approx 1/3$) non-malleable code in the 2-split-state streaming model, which is only a factor-$(3/2)$ smaller than the upper-bound on the maximum rate.

*Microsoft Research, Bangalore, India. t-digu@microsoft.com

†Department of Computer Science, Purdue University. hmaji@purdue.edu.

‡The research effort is supported in part by an NSF CRII Award CNS–1566499, an NSF SMALL Award CNS–1618822, and an REU CNS–1724673.

§The research effort is supported in part by a Purdue Research Foundation grant.

¶Department of Computer Science, Purdue University. wang1929@purdue.edu.

Contents

1 Introduction

Dziembowski, Pietrzak, and Wichs [DPW10] introduced the notion of non-malleable codes as a useful message integrity assurance for scenarios where error-correction or, even, error-detection is impossible. For example, consider the simple tampering function that overwrites the encoding of the message with a fixed codeword. Intuitively, given a family of tampering function $\mathcal{F}$, our goal is to design a pair of encoding and decoding algorithms $(\mathsf{Enc}, \mathsf{Dec})$ for messages in $\{0,1\}^\ell$ such that the (possibly, randomized) encoding $c \in \{0,1\}^n$ of a message $m \in \{0,1\}^\ell$ has the following property. The decoding of the tampered codeword $\widetilde{c} = f(c)$, where $f \in \mathcal{F}$ is a tampering function, is either identical to the original message m or an entirely unrelated message. For instance, against the family of tampering functions that add an arbitrary error of low Hamming weight, we can design error-correcting codes to recover the original message. Moreover, against the family of tampering functions that add an arbitrary constant, we can design Algebraic Manipulation Detection codes to detect the tampering with high probability [CDF+08]. For even more complex families of tampering functions, non-malleable codes ensure that the decoding of the tampered codeword, i.e., the message $\mathsf{Dec}(f(\mathsf{Enc}(m))$, is either the original message m or a simulator Sim_f, which is entirely independent of the original message, can simulate it. Ensuring this weak message integrity turns out to be extremely useful for cryptography. For example, tampering the secret-key of a signature scheme either yields the original secret-key (in which case the signature's security already holds) or yields an unrelated secret-key (which, again, is useless for forging signatures using the original secret-key).

However, if $\mathcal{F}$ is the set of all functions from the domain $\{0,1\}^n$ to the range $\{0,1\}^n$, then no non-malleable code exists against this family of tampering functions. In particular, note that if the tampering family contains the function f that can decode the codeword c to retrieve the original message m and then write a particular encoding of the related message m_1^ℓ, where m_1 is the first bit of m, then it is impossible to achieve non-malleability against this tampering function. So, it is necessary to ensure that the decoding algorithm Dec (or its approximations) does not lie in the tampering function family itself. Therefore, given a fixed family of tampering functions $\mathcal{F}$, we design a non-malleable code that is resilient to any tampering function from that particular family of functions.

Split-State Tampering. A natural restriction on the tampering function family is that $k \geqslant 2$ separate states store the encoding, and the tampering function can only tamper each state independently, à la the k-split-state model. More formally, the message $m \in \{0,1\}^\ell$ is encoded as $c = (c_1, c_2, \ldots, c_k) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2} \times \cdots \times \{0,1\}^{n_k}$. A tampering function is a k-tuple of functions $f = (f_1, f_2, \ldots, f_k)$ such that the function f_i is an arbitrary function over $\{0,1\}^{n_i}$. Since the decoding algorithm can aggregate information across all the states while the tampering functions only have local influence, non-malleable codes in the k-split-state model, for $k \geqslant 2$, exist. Intuitively, the reduction in the number of states k increases the power of the tampering functions, thus, significantly escalating the complexity of designing non-malleable codes for smaller values of k.

However, if k is small then the k-split-state seems more naturally enforceable in cryptographic contexts, for example, the states can correspond to separated parts of memory or storage. The overhead of achieving non-malleability, measured by the rate $R = \ell/(n_1 + n_2 + \cdots n_k)$, is another crucial attribute determining the efficiency of using these codes in cryptographic contexts. Overall, this brings to fore the central guiding principle of non-malleable code design.

"Construct non-malleable code for the k-split-state model for small k that achieves high rate R."

The objective of reducing the number of states k is inimical to the objective of achieving high rate R.

The holy grail for this research endeavor is a rate 1/2 non-malleable code in the 2-split-state model,[1] which, by even the most optimistic estimates, seems distant. A sequence of highly influential works has established rate $1/\log \ell$ as the current state-of-the-art [Li17]. Our work constructs a rate 1/3 non-malleable code in the 3-split-state model, i.e., it achieves close-to-optimal rate by employing an additional state. In the context of tamper-resilient cryptographic applications of non-malleable codes, our results imply that if one can ensure three separate memory/storage units, then the overhead of non-malleability is not significant.

1.1 Our Contribution

Let $\mathcal{S}_n$ represent the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. Define $\mathcal{S}_{n_1} \times \mathcal{S}_{n_2} \times \cdots \times \mathcal{S}_{n_k}$ as the set of all k-tuples of functions $(f_1, f_2, \ldots, f_k)$ such that $f_i \in \mathcal{S}_{n_i}$ is an arbitrary function, for each $i \in \{1, 2, \ldots, k\}$. In the k-split-state, the codeword is distributed over k states of size $n_1, n_2, \ldots, n_k$, respectively, and the i-th state is tampered by f_i independently. The set of all tampering functions in this model is identical to $\mathcal{S}_{n_1} \times \cdots \times \mathcal{S}_{n_k}$, and the maximum achievable rate in this model is $(1 - 1/k)$ [CG14a]. Our first result proves the following theorem, and Figure 1 positions this result relative to relevant prior works.

Theorem 1 (Rate-1/3 NMC in 3-Split-State)**.** *There exists a non-malleable code, with negligible simulation error, in the 3-split-state model $\mathcal{S}_{n_1} \times \mathcal{S}_{n_2} \times \mathcal{S}_{n_3}$, where $n_1 = \ell$, $n_2 = (2 + o(1))$, $n_3 = o(\ell)$, and ℓ is the length of the message.*

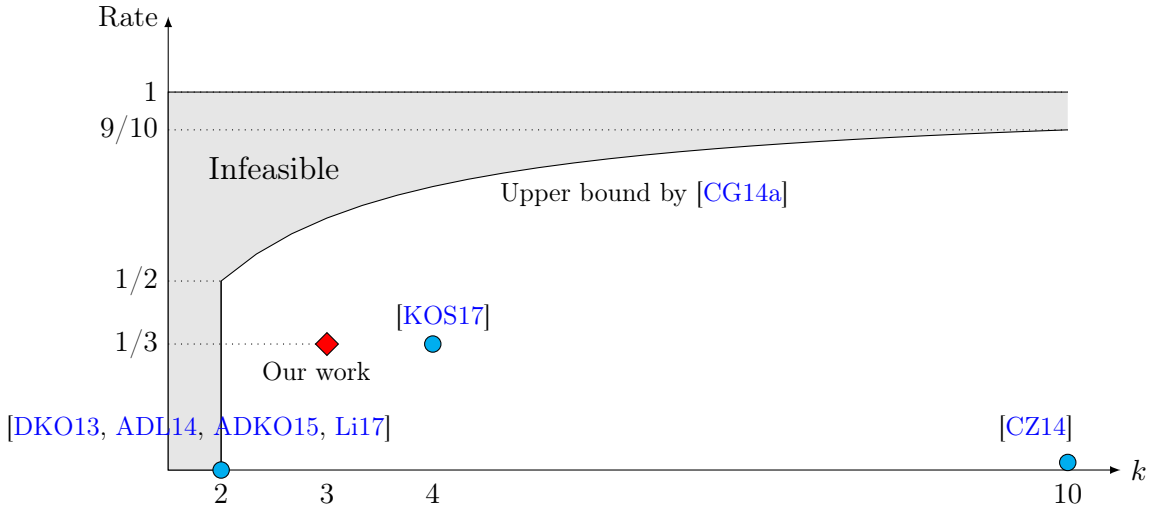

Figure 1: A pictorial summary of prior results for non-malleable codes in the k-split-state model and positioning our result among them.

This result relies on leveraging a unique characteristic of the (rate-0) non-malleable code in 2-split-state provided by Aggarwal, Dodis, and Lovett [ADL14], namely *augmented non-malleability*, which was identified by [AAG+16]. At a high-level, our construction preserves a small digest of the message and the randomness used in our encoding procedure using the augmented non-malleable code of [ADL14] in the 2-split-state model. Using one additional state, we can prove that our entire encoding is non-malleable in the 3-split-state. This design principle allows us to achieve

[1] In the information-theoretic setting, the maximum achievable rate of any non-malleable code in the 2-split-state model is 1/2 [CG14a].

close-to-optimum rate. Additionally, we can merge two appropriate states in our encoding if the tampering functions are suitably restricted, which yields the 2-split-state result in the streaming model summarized below.

Streaming Model. Instead of considering an arbitrary function, we consider tampering functions that encounter the information of each state as a stream. Let $\mathcal{LA}_{n_1,n_2,\ldots,n_B}$ be the set of all functions $f\colon \{0,1\}^{n_1+n_2+\cdots+n_B} \to \{0,1\}^{n_1+n_2+\cdots+n_B}$ such that there exists functions $f^{(1)}, f^{(2)}, \ldots, f^{(B)}$ with the following properties.

1. For inputs $x_1 \in \{0,1\}^{n_1}, x_2 \in \{0,1\}^{n_2}, \ldots, x_B \in \{0,1\}^{n_B}$,
2. For each $1 \leqslant i \leqslant B$, we have $f^{(i)}\colon \{0,1\}^{n_1+n_2+\cdots+n_i} \to \{0,1\}^{n_i}$, and
3. The function $f(x_1, x_2, \ldots, x_B)$ is the concatenation of $f^{(i)}(x_1, x_2, \ldots, x_i)$, for $1 \leqslant i \leqslant B$.

Intuitively, the state arrives as B blocks of information, and the i-th block is tampered based on all the previous blocks $\{1, 2, \ldots, i\}$. Note that this is similar to the notion of lookahead functions introduced in [ADKO15] (who introduced this tampering family to assist the construction of non malleable codes in the 2-split-state model) and block-wise tampering functions introduced by [CGM+16]. In the streaming version of the k-split-state model, the tampering function for each state is a streaming function.

Note that $\mathcal{LA}_{\underbrace{1,1,\ldots,1}_{n\text{-times}}}$ (for brevity, alternatively represented by $\mathcal{LA}_{1^{\otimes n}}$) is the set of all streaming functions where each bit is a block, and every $\mathcal{LA}_{n_1,n_2,\ldots,n_B}$ tampering family, such that $n_1 + n_2 + \cdots + n_B = n$, contains this tampering family. Analogous to the result of Cheraghchi and Guruswami [CG14a] for the k-split-state model, we prove that the rate of any non-malleable code against the tampering family $\mathcal{LA}_{1^{\otimes n_1}} \times \cdots \times \mathcal{LA}_{1^{\otimes n_k}}$ in the k-split-state model is at most $1-1/k$ (see Theorem 5 in Appendix B). The current state-of-the-art in non-malleable code construction in the streaming version of the k-split-state model coincides with the general k-split-state model. In particular, prior to our work, no constant-rate non-malleable codes for $k = 2$ and $k = 3$ even in the restricted streaming version of the k-split-state model was known. We resolve this problem in the positive for the most complex model, i.e., for $k = 2$.

Theorem 2 (Rate-1/3 NMC in Streaming 2-Split-State)**.** *There exists a non-malleable code, with negligible simulation error, in the streaming version of the 2-split-state model $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $n_1 = (2 + o(1))\ell$, $n_2 = o(\ell)$, $n_3 = o(\ell)$, $n_4 = \ell$, and ℓ is the length of the message.*

[ADKO15] motivated achieving non-malleability against streaming functions along with another particular family of functions (namely, forgetful functions) as an intermediate step to constructing constant-rate non-malleable codes in the 2-split-state model. We achieve partial progress towards this goal, and Theorem 3 summarizes this result.

1.2 Prior Relevant Works

It is not possible to do justice to the vast literature on the related topics of non-malleability, error-correcting codes, and algebraic manipulation detection codes, and summarize them in one section. Even the field of non-malleable codes and extractors is sufficiently immense that an exhaustive survey is beyond the scope of this paper.

As explained earlier, it is impossible to construct non-malleable codes against the set of all tampering functions. If the size of the tampering family $\mathcal{F}$ is bounded then Monte-Carlo constructions of non-malleable codes exist [FMVW14, CG14a]. However, explicit constructions are

known only for a few tampering families. For example, (1) bit-level perturbation and permutations [DPW10, CG14b, AGM+15], and (2) local or AC^0 tampering functions [BDKM16, CL16] are a few representative families of tampering functions.

Another famous tampering function family is the k-split-state model, for $k \geqslant 2$, where the tampering function tampers each state independently. Cheraghchi and Guruswami [CG14a] proved an upper bound of $1-1/k$ on the rate of any non-malleable code in the k-split-state model. Decreasing the number of states k escalates the complexity of constructing non-malleable codes significantly. For $k = 2$, technically the most challenging problem and most reliable for cryptographic applications, [DKO13] constructed the first explicit non-malleable code for one-bit messages. In a breakthrough result, Aggarwal, Dodis, and Lovett [ADL14] presented the first multi-bit non-malleable code with rate $O(\ell^{-\rho})$, for a suitable constant $\rho > 1$. The subsequent work of [ADKO15] introduced the general notions of non-malleable reductions and transformations and exhibited their utility for modular constructions of non-malleable codes. Currently, the best rate of $1/\log \ell$ is achieved by [Li17].

For higher values of k, Chattopadhyay, and Zuckerman [CZ14] constructed the first constant-rate non-malleable code when $k = 10$. Note that a non-malleable code in the k-split-state model implies non-malleable codes for a k'-split-state model, for all $k' \geqslant k$. Recently, [KOS17] constructed a rate-$1/3$ non-malleable code in the 4-split-state model. The construction of constant rate non-malleable codes in the 2-split-state and 3-split-state models was open.

Computational 2-Split-State. The computational version of this problem restricts to only computationally efficient tampering functions, and [AAG+16] provided the qualitatively and quantitatively optimal solution. In the 2-split-state model, they showed that one-way functions are necessary to surpass the upper bound of $1/2$ on the rate in the information-theoretic setting [CG14a], and one-way functions suffice to achieve rate-1.

Streaming Model. In the streaming model, tampering functions encounter the state as a stream, and the tampering functions tampers a block of the state based solely on the blocks of the state it has seen thus far. This family of tampering has also been considered by [ADKO15] (lookahead tampering) as an interesting pit stop on the route to constructing non-malleable codes in the 2-split-state model. [CGM+16] also considers this family of tampering functions (referred to as block-wise tampering) but their results are in the computational setting.

Observe that a non-malleable code in the k-split-state model is also a non-malleable code in its streaming version. Currently, the state-of-the-art in the streaming k-split-state model coincides with the general k-split-state-model. In particular, there are no known constant-rate non-malleable codes in the information-theoretic setting for $k = 2$ and $k = 3$.

1.3 Technical Overview

We provide a summary of the intuition underlying our constructions. We use the rate-$1/3$ non-malleable code construction in the 3-split-state model as a representative example to illustrate our primary technical contribution. Figure 3 presents the encoding and decoding procedures for this non-malleable code. Figure 2 provides a block-diagram of our construction, which employs a hybrid of two forms of message integrity guarantees.

First, we XOR the message m with a random private-key r and obtain the cipher-text c. In the final construction, we derive the key r from a source w using a seed s. However, to build intuition, it suffices to assume that c and r form the payload. The cipher-text c and the key r must be allocated to separate states; otherwise, a tampering function can recover m and then perform a message-dependent tampering.

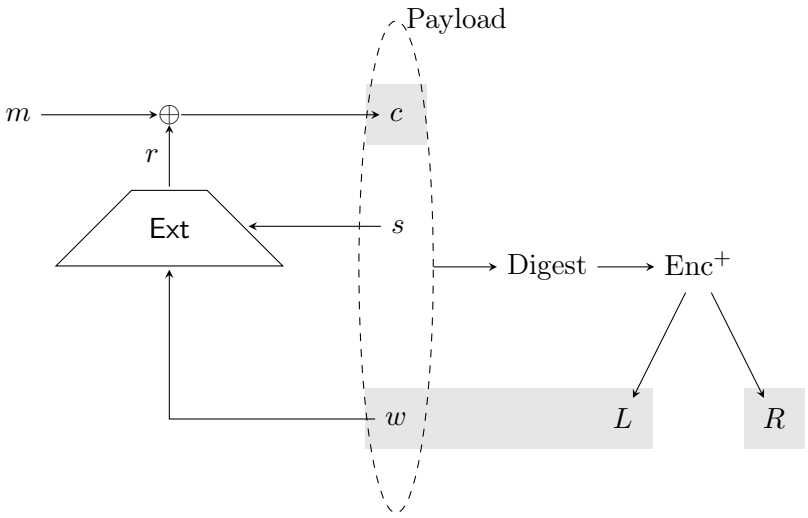

Figure 2: An intuitive summary of our non-malleable code construction in the 3-split-state model, where the message is m. Each gray rectangle represents a state.

1. The first message integrity primitive ensures that the payload (c, r) cannot change if the associated secret-key remains untouched. This component generates a short digest of the payload such that the total size of the digest and the secret-key is at most ℓ^{σ}, where σ is an arbitrarily small positive constant.

2. The second message integrity primitive protects the critical, although small-sized, digest and key of the first primitive against tampering using an elaborate mechanism, which results in encodings of size $\leqslant \ell^{\rho\sigma}$, where $\rho > 1$ is an appropriate constant. We can choose the σ in the first step small enough so that $\rho\sigma \leqslant 1$. A rate-$o(1)$ non-malleable code in the 2-split-state model, for instance, suffices. Let its two states be L and R.

Since we are working with $k = 3$, either the cipher-text c or the key r must be in the same state as L or R. Suppose, the key r and L are in the same state. Now, the tampering on the key r can depend on L, which creates two main issues.

1. Ordinary non-malleable codes in the 2-split-state only ensure that the "the digest and the secret-key" remain intact or is replaced by an unrelated pair of digest and key. However, they *cannot* ensure that the tampering on r using L is not dependent on the digest, which in turn depends on m. To resolve this, we need the *augmented non-malleability* of the 2-split-state non-malleable code of [ADL14].

2. Tampering r based on L creates circularity issues as well, which is evident from Figure 2. The direction of the arrows indicates the direction of information flow in our scheme. The addition of an arrow from L to r, representing a tampering of r based on L, introduces a cycle. To resolve this, we employ techniques from leakage-resilient cryptography. Instead of using r, we use a (sufficiently large) source w and seed s to derive the key r. In the hybrid arguments, we can emulate the effect of the tampering functions by performing suitable bounded-size leakage on the source w such that the source retains sufficient min-entropy. Using this min-entropy, the extracted r (roughly) becomes independent of the source, thus, breaking the circularity.

Figure 4 presents our simulator and Appendix A presents the proof of security, thus proving Theorem 1.

A similar construction, where the first state is (w, R) and the second state is (L, c), is a non-malleable code in the streaming version of the 2-split-state model (refer to Figure 5 and the proof of security in Appendix C), which yields Theorem 2.

2 Preliminaries

For any natural number n, the symbol $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For a probability distribution A over a finite sample space Ω, $A(x)$ denotes the probability of sampling $x \in \Omega$ according to the distribution A and $x \sim A$ denotes that x is sampled from Ω according to A. For any $n \in \mathbb{N}$, U_n denotes the uniform distribution over $\{0,1\}^n$. Similarly, for a set S, U_S denotes the uniform distribution over S. For two probability distributions A and B over the same sample space Ω, the *statistical distance* between A and B, represented by $\mathrm{SD}(A, B)$, is defined to be $\frac{1}{2}\sum_{x\in\Omega} |A(x) - B(x)|$.

Let $f : \{0,1\}^p \times \{0,1\}^q \longrightarrow \{0,1\}^p \times \{0,1\}^q$. For any $x \in \{0,1\}^p, y \in \{0,1\}^q$, let $(\widetilde{x}, \widetilde{y}) = f(x, y)$. Then, we define $f_x(y) = \widetilde{y}$ and $f_y(x) = \widetilde{x}$. Note that $f_x : \{0,1\}^q \to \{0,1\}^q$ and $f_y : \{0,1\}^p \to \{0,1\}^p$.
.

2.1 Non-malleable codes.

We follow the presentation in previous works and define non-malleable codes below.

Definition 1 (Coding Schemes)**.** *Let* $\mathrm{Enc}\colon \{0,1\}^\ell \to \{0,1\}^n$ *and* $\mathrm{Dec}\colon \{0,1\}^n \to \{0,1\}^k \cup \{\perp\}$ *be functions such that* Enc *is a randomized function (that is, it has access to private randomness) and* Dec *is a deterministic function. The pair* $(\mathrm{Enc}, \mathrm{Dec})$ *is called a coding scheme with block length* n *and message length* ℓ *if it satisfies perfect correctness, i.e., for all* $m \in \{0,1\}^\ell$*, over the randomness of* Enc*,* $\Pr[\mathrm{Dec}(\mathrm{Enc}(m)) = m] = 1$.

A non-malleable code is defined w.r.t. a family of tampering functions. For an encoding scheme with block length n, let $\mathcal{F}_n$ denote the set of all functions $f\colon \{0,1\}^n \to \{0,1\}^n$. Any subset $\mathcal{F} \subseteq \mathcal{F}_n$ is considered to a family of tampering functions. Please refer to Section 1.1 for definition of k-split-state tampering function family $\mathcal{S}_{n_1} \times \mathcal{S}_{n_2} \times \cdots \times \mathcal{S}_{n_k}$ and the streaming version of the k-split-state tampering function family $\mathcal{LA}_{1^{\otimes n_1}} \times \cdots \times \mathcal{LA}_{1^{\otimes n_k}}$.

Next, we define the non-malleable codes against a family $\mathcal{F}$ of tampering functions. We need the following $\mathrm{copy}(x, y)$ function defined as follows:

$$\mathrm{copy}(x, y) = \begin{cases} y, & \text{if } x = \text{same*}; \\ x, & \text{otherwise.} \end{cases}$$

Definition 2 ((n, ℓ, ε)-Non-malleable Codes)**.** *A coding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *with block length* n *and message length* ℓ *is said to be non-malleable against tampering family* $\mathcal{F} \subseteq \mathcal{F}_n$ *with error* ε *if for all function* $f \in \mathcal{F}$*, there exists a distribution* Sim_f *over* $\{0,1\}^\ell \cup \{\perp\} \cup \{\text{same*}\}$ *such that for all* $m \in \{0,1\}^\ell$*,*

$$\mathrm{Tamper}_f^m \approx_\varepsilon \mathrm{copy}\left(\mathrm{Sim}_f, m\right)$$

where Tamper_f^m *stands for the following tampering distribution*

$$\mathrm{Tamper}_f^m := \left\{ \begin{array}{c} c \sim \mathrm{Enc}(m),\ \widetilde{c} = f(c),\ \widetilde{m} = \mathrm{Dec}(\widetilde{c}) \\ \textit{Output: } \widetilde{m}. \end{array} \right\}$$

The rate of a non-malleable code is defined as ℓ/n.

Our constructions rely on leveraging a unique characteristic of the non-malleable code in 2-split-state ($\mathcal{S}_{n_1} \times \mathcal{S}_{n_2}$ s.t. $n_1 + n_2 = n$) provided by Aggarwal, Dodis, and Lovett [ADL14], namely *augmented non-malleability*, which was identified by [AAG+16]. We formally define this notion next. Below, we denote the two states of the codeword as $(L, R) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2}$.

Definition 3 (($n_1, n_2, \ell, \varepsilon$)-Augmented Non-malleable Codes against 2-split-state tampering family)**.** *A coding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *with message length* ℓ *is said to be a augmented non-malleable coding scheme against tampering family* $\mathcal{S}_{n_1} \times \mathcal{S}_{n_2}$ *with.* $n_1 + n_2 = n$ *with error* ε *if for all function* $f, g \in \mathcal{S}_{n_1} \times \mathcal{S}_{n_2}$*, there exists a distribution* $\mathrm{SimPlus}_{f,g}$ *over* $\{0,1\}^{n_1} \times (\{0,1\}^{\ell} \cup \{\perp\} \cup \{\text{same}^*\})$ *such that for all* $m \in \{0,1\}^{\ell}$*,*

$$\mathrm{TamperPlus}^{m}_{f,g} \approx_{\varepsilon} \mathrm{copy}\left(\mathrm{SimPlus}_{f,g}, m\right)$$

where $\mathrm{TamperPlus}^{m}_{f,g}$ *stands for the following augmented tampering distribution*

$$\mathrm{TamperPlus}^{m}_{f,g} := \left\{ \begin{array}{c} (L,R) \sim \mathrm{Enc}(m),\ \widetilde{L} = f(L),\ \widetilde{R} = g(R) \\ \textit{Output}\ \left(L, \mathrm{Dec}(\widetilde{L}, \widetilde{R})\right) \end{array} \right\}$$

Note that above we abuse notation for $\mathrm{copy}\left(\mathrm{SimPlus}_{f,g}, m\right)$. Formally, it is defined as follows: $\mathrm{copy}\left(\mathrm{SimPlus}_{f,g}, m\right) = (L, m)$ when $\mathrm{SimPlus}_{f,g} = (L, \text{same}^*)$ and $\mathrm{SimPlus}_{f,g}$ otherwise.

It was shown in [AAG+16] that the construction of Aggarwal et al. [ADL14] satisfies this stronger definition of augmented non-malleability with rate $1/\operatorname{poly}(\ell)$ and negligible error ε. More formally, following holds.

Imported Theorem 1 ([AAG+16])**.** *For any message length* ℓ*, there is a coding scheme* $(\mathrm{Enc}^+, \mathrm{Dec}^+)$ *of block length* $n = p(\ell)$ *(where* p *is a polynomial) that satisfies augmented non-malleability against 2-split-state tampering functions with error that is negligible in* ℓ*.*

2.2 Building Blocks.

Next, we describe building blocks average min-entropy seeded extractors with small seed and one-time message authentication codes that we use in our construction.

Definition 4 (Average conditional min-entropy)**.** *The average conditional min-entropy of a distribution* A *conditioned on distribution* L *is defined to be*

$$\widetilde{H}_{\infty}(A|L) = -\log\left(\mathbb{E}_{\ell \sim L}\left[2^{-H_{\infty}(A|L=\ell)}\right]\right)$$

Following lemma holds for average conditional min-entropy in the presence of leakage.

Lemma 1 ([DORS08])**.** *Let* L *be an arbitrary* κ*-bit leakage on* A*, then* $\widetilde{H}_{\infty}(A|L) \geqslant H_{\infty}(A) - \kappa$*.*

Definition 5 (Seeded Average Min-entropy Extractor)**.** *We say* $\mathrm{Ext} : \{0,1\}^{n} \times \{0,1\}^{d} \longrightarrow \{0,1\}^{\ell}$ *is a* (k, ε)*-average min-entropy strong extractor if for every joint distribution* (A, L) *such that* $\widetilde{H}_{\infty}(A|L) \geqslant k$*, we have that* $(\mathrm{Ext}(A, U_d), U_d, L) \approx_{\varepsilon} (U_{\ell}, U_d, L)$*.*

It is proved in [Vad12] that any extractor is also a average min-entropy extractor with only a loss of constant factor on error. Also, [GUV07] gave strong extractors with small seed length that extract arbitrarily close to k uniform bits. We summarize these in the following lemma.

Combining this results with the following known construction for extractors, we have that there exists average min-entropy extractor that require seed length $O(\log n + \log(1/\varepsilon))$ and extracts randomness with length arbitrary close to conditional min-entropy.

Lemma 2 ([GUV07, Vad12]). *For all constant $\alpha > 0$ and all integers $n \geqslant k$, there exists an efficient (k, ε)-average min-entropy strong extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \longrightarrow \{0,1\}^\ell$ *with seed length* $d = O(\log n + \log(1/\varepsilon))$ *and* $\ell = (1-\alpha)k - O(\log(n) + \log(1/\varepsilon))$.

Next, we define one-time message authentication codes.

Definition 6 (Message authentication code). *A μ-secure one-time message authentication code (MAC) is a family of pairs of function*

$$\left\{ \mathrm{Tag}_k : \{0,1\}^\alpha \longrightarrow \{0,1\}^\beta,\ \mathrm{Verify}_k : \{0,1\}^\alpha \times \{0,1\}^\beta \longrightarrow \{0,1\} \right\}_{k \in K}$$

such that
(1) For all m, k, $\mathrm{Verify}_k(m, \mathrm{Tag}_k(m)) = 1$.
(2) For all $m \neq m'$ and t, t', $\Pr_{k \sim U_K}[\mathrm{Tag}_k(m) = t \mid \mathrm{Tag}_k(m') = t'] \leqslant \mu$.

Message authentication code can be constructed from μ-almost pairwise hash function family with the key length $2\log(1/\mu)$. For the completeness of our proof, we give a construction in the Appendix E.

3 Construction for 3-Split State Non-malleable Code

In this section, we provide our construction for 3-state non-malleable code against split-state tampering functions. Our construction relies on the following tools. Let $(\mathrm{Tag}, \mathrm{Verify})$ (resp., $(\mathrm{Tag}', \mathrm{Verify}')$) be a μ (resp., μ') secure message authentication code with message length ℓ (resp., n), tag length β (resp., β') and key length γ (resp., γ'). Let $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ be a (k, ε_1) average min-entropy strong extractor. We define k later during parameter setting. Finally, let $(\mathrm{Enc}^+, \mathrm{Dec}^+)$ be $(n_1^+, n_2^+, \ell^+, \varepsilon^+)$-augmented 2-split-state non-malleable code (see Definition 3), where $\ell^+ = \gamma + \gamma' + \beta + \beta' + d$. We denote the codewords of this scheme as (L, R) and given a tampering function, we denote the output of the simulator SimPlus as (L, Ans).

Construction Overview. We define our encoding and decoding functions formally in Figure 3. In our encoding procedure, we first sample a uniform source w of n bits and a uniform seed s of d bits. Next, we extract a random r from (w, s) using the extractor Ext. We hide the message m using the key r as one-time pad to obtain a ciphertext c. Next, we authenticate the ciphertext c using Tag_{k_1} and authenticate the source w using Tag'_{k_2} to tags t_1 and t_2, respectively. Now, we think of (k_1, k_2, t_1, t_2, s) as the digest and protect it using an augmented 2-state non-malleable encoding Enc^+ to obtain (L, R). Finally, our codeword is (c_1, c_2, c_3) where $c_1 = c$, $c_2 = (w, L)$ and $c_3 = R$.

We also note that $n_1 := |c_1| = |m| = \ell$, $n_2 := |c_2| = |w| + |L| = n + n_1^+$ and $n_3 := |c_3| = n_2^+$. From the Figure 3, it is evident that our construction satisfies perfect correctness.

Proof of Non-malleability against split-state tampering. Given a tampering function $(f, g, h) \in \mathcal{S}_{n_1} \times \mathcal{S}_{n_2} \times \mathcal{S}_{n_3}$, we define our simulator $\mathrm{Sim}_{f,g,h}$. We formally describe our simulator in Figure 4.

Note that the function g acts on both (w, L) to produce $(\widetilde{w}, \widetilde{L})$. Our simulator describes a leakage function $\mathcal{L}(w)$ that captures the leakage required on the source w in order to simulate the tampering experiment. This leakage has four parts $(\mathrm{Ans}, \mathrm{flag}_1, \mathrm{flag}_2, \mathrm{mask})$. Ans denotes the second part of output of SimPlus on tampering function (g_w, h), where g_w represents the tampering function on L given w. Next, for the case when $\mathrm{Ans} = \mathrm{same}^*$, flag_1 denotes the bit $\widetilde{w} = w$. When

$\mathrm{Enc}(m)$:	$\mathrm{Dec}(c_1, c_2, c_3)$:
1. Sample $w \sim U_n$, $s \sim U_d$, $k_1 \sim U_\gamma$, $k_2 \sim U_{\gamma'}$ 2. Compute $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$ 3. Compute the tags $t_1 = \mathrm{Tag}_{k_1}(c)$ and $t_2 = \mathrm{Tag}'_{k_2}(w)$ 4. Compute the 2-state non-malleable encoding $(L, R) \sim \mathrm{Enc}^+(k_1, k_2, t_1, t_2, s)$ 5. Output the three states $\Big(c, (w, L), R\Big)$	1. Let the tampered states be $\widetilde{c} := c_1$, $(\widetilde{w}, \widetilde{L}) := c_2$, $\widetilde{R} := c_3$ 2. Decrypt $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{Dec}^+(\widetilde{L}, \widetilde{R})$ 3. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$ 4. (Else) If $\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0$, then output $\perp$ 5. (Else) Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Figure 3: Compiling an augmented 2-state non-malleable code $(\mathrm{Enc}^+, \mathrm{Dec}^+)$ into a 3-state non-malleable code.

1. $w \sim U_n$
2. For the tampering function (g, h) we define the following leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$
   (a) $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g_w, h}$, $\widetilde{w} = g_L(w)$
   (b) If Ans =
      - Case same*: $\mathrm{flag}_1 = 1$ iff $(\widetilde{w} = w)$.
      - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\mathrm{flag}_2 = 1$ iff $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$. And $\mathrm{mask} = \mathrm{Ext}(\widetilde{w}, \widetilde{s})$.
   (c) $\mathcal{L}(w) := (\mathrm{Ans}, \mathrm{flag}_1, \mathrm{flag}_2, \mathrm{mask})$
3. $r \sim U_\ell$, $c = 0^\ell \oplus r$, $\widetilde{c} = f(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ AND $\mathrm{flag}_1\Big)$, output same*, else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=1 AND $\mathrm{flag}_2\Big)$, output $\widetilde{c} \oplus \mathrm{mask}$, else $\perp$.

Figure 4: The simulator $\mathrm{Sim}_{f,g,h}$ for the non-malleable code in the 3-split-state model.

$\mathrm{Ans} = (\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$, flag_2 captures the bit $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})$, i.e., whether the new key $\widetilde{k_2}$ and tag $\widetilde{t_2}$ are valid authentication on new source $\widetilde{w}$. In this case, the last part of leakage mask is set of new extracted value.

We give the formal proof on indistinguishability between simulated and tampering distributions in Appendix A using a series of statistically close hybrids.

Rate analysis. We will use λ as our security parameter. By Corollary 7, we will let k_1, k_2 be of length 2λ, i.e. $\gamma = \gamma' = 2\lambda$ and t_1, t_2 will have length λ, i.e. $\beta = \beta' = \lambda$ and both $(\mathrm{Tag}, \mathrm{Verify})$ and $(\mathrm{Tag}', \mathrm{Verify}')$ will have error $2^{-\lambda}$.

Since we will need to extract ℓ bits as a one-time pad to mask the message, by Lemma 2, we will set min-entropy k to be $(1+\alpha')\ell$ for some constant α' and let Ext be a $((1+\alpha')\ell, 2^{-\lambda})$-strong average min-entropy extractor that extract ℓ-bit randomness with seed length $O(\log n + \lambda)$. By our analysis in Appendix A, it is suffice to have $n - 6\lambda - \ell - O(\log n + \lambda) - 3 \geqslant (1+\alpha')\ell$. Hence, we will set $n = (2+\alpha)\ell$ for some constant $\alpha > \alpha'$.

Now the message length for our augmented 2-state non-malleable code will be $2\lambda + 2\lambda + \lambda +$

$\lambda + O(\log n + \lambda) = O(\log n + \lambda)$. Now by Theorem 1, we will let ζ be the constant such that $p(n^\zeta) = o(n)$ and set $\lambda = O(n^\zeta)$. Hence, the length of (L, R) will be $o(n)$. Therefore, the total length of our coding scheme will be $\ell + (2+\alpha)\ell + o(n)$ and the rate is $\frac{1}{3+\alpha}$ with error $O(2^{-n^\zeta})$. This completes the proof for Theorem 1.

4 Streaming version of the Split State Model

We prove an upper-bound on the maximum achievable rate for any non-malleable code in the streaming version of the k-split-state model. Theorem 5 (proven in Appendix B), roughly, states that the maximum achievable rate is $1 - 1/k$, similar to the result of [CG14a]. On the other hand, for $k = 2$, the technically most challenging model, we construct a non-malleable code that achieves rate $1/3$. Figure 5 presents the non-malleable code in the streaming version of the 2-split-state model. The construction is similar to the construction in Figure 3, except the manner of merging the payload with the 2-split-state encoding (L and R) of the digest. Appendix C provides the security proof.

$\mathrm{Enc}(m)$:

1. Sample $w \sim U_n$, $s \sim U_d$, $k_1 \sim U_\gamma$, $k_2 \sim U_{\gamma'}$
2. Compute $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$
3. Compute the tags $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
4. Compute the 2-state non-malleable encoding $(L, R) \sim \mathrm{Enc}^+(k_1, k_2, t_1, t_2, s)$
5. Output the states $\Big((w, R), (L, c)\Big)$

$\mathrm{Dec}\Big((c_1, c_2), (c_3, c_4)\Big)$:

1. Let the tampered states be $\widetilde{w} := c_1$, $\widetilde{R} := c_2$, $\widetilde{L} := c_3$, $\widetilde{c} := c_4$
2. Decrypt $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{Dec}^+(\widetilde{L}, \widetilde{R})$
3. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
4. (Else) If $\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0$, output $\perp$
5. (Else) Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Figure 5: Non-malleable coding scheme against $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $n_1 = |w|$, $n_2 = |R|$, $n_3 = |L|$, and $n_4 = |c|$.

5 Forgetful tampering in the Streaming 2-Split-State Model

In this section we restrict ourselves to the streaming version of the 2-split-state model. Let us define an additional family of tampering functions. Consider a tampering function $f\colon \{0,1\}^{n_1+n_2+n_3+n_4} \to \{0,1\}^{n_1+n_2+n_3+n_4}$. The function f is *1-forgetful*, if there exists a function $g\colon \{0,1\}^{n_2+n_3+n_4} \to \{0,1\}^{n_1+n_2+n_3+n_4}$ such that $f(x_1, x_2, x_3, x_4) = g(x_2, x_3, x_4)$ for all $x_1 \in \{0,1\}^{n_1}$, $x_2 \in \{0,1\}^{n_2}$, $x_3 \in \{0,1\}^{n_3}$, and $x_4 \in \{0,1\}^{n_4}$. Intuitively, the tampering function f forgets its first n_1-bits of the codeword. The set of all such functions that are 1-forgetful are represented by $\mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}}$. Analogously, we define $\mathcal{FOR}_{n_1,n_2,n_3,n_4-\{i\}}$, for each $i \in \{2,3,4\}$.

[ADKO15] proved that we can construct constant-rate non-malleable code in the 2-split-state from a constant-rate non-malleable code that protects against the tampering family

$$\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4} \quad \bigcup_{i=1}^{4} \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{i\}}$$

We make partial progress towards this goal and prove the following theorem in Appendix D.

Theorem 3. *For all constant α, there exists a constant ζ and a non-malleable coding scheme against $\big(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\big) \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$ with rate $\frac{1}{4+\alpha}$ and error 2^{-n^ζ}.*

References

[AAG+16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 393–417, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49099-0_15.

[ADKO15] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 459–468, Portland, OR, USA, June 14–17, 2015. ACM Press.

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 774–783, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

[AGM+15] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 375–397, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_16.

[BDKM16] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 881–908, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_31.

[CDF+08] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[CG14a] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In Moni Naor, editor, *ITCS 2014: 5th Innovations in Theoretical Computer Science*, pages 155–168, Princeton, NJ, USA, January 12–14, 2014. Association for Computing Machinery.

[CG14b] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 440–464, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_19.

[CGM+16] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. In Ioannis Chatzigiannakis, Michael

Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016: 43rd International Colloquium on Automata, Languages and Programming*, volume 55 of *LIPIcs*, pages 31:1–31:14, Rome, Italy, July 11–15, 2016. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2016.31.

[CL16] Eshan Chattopadhyay and Xin Li. Explicit non-malleable extractors, multi-source extractors, and almost optimal privacy amplification protocols. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 158–167, New Brunswick, NJ, USA, October 9–11, 2016. IEEE Computer Society Press. doi:10.1109/FOCS.2016.25.

[CZ14] Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *55th Annual Symposium on Foundations of Computer Science*, pages 306–315, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press. doi:10.1109/FOCS.2014.40.

[DKO13] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 239–257, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_14.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 434–452, Tsinghua University, Beijing, China, January 5–7, 2010. Tsinghua University Press.

[FMVW14] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 111–128, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_7.

[GUV07] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 96–108, 2007. URL: https://doi.org/10.1109/CCC.2007.38.

[KOS17] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. Cryptology ePrint Archive, Report 2017/930, 2017. http://eprint.iacr.org/2017/930.

[Li17] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 1144–1156, Montreal, QC, Canada, June 19–23, 2017. ACM Press.

[Vad12] S.P. Vadhan. *Pseudorandomness*. Foundations and trends in theoretical computer science. Now Publishers, 2012. URL: https://books.google.com/books?id=iam4lAEACAAJ.

A Proof of 3-Split-State Non-malleability (Theorem 1)

Here we will prove that the encoding scheme shown in Figure 3 is secure against 3-split-state tampering. The formal description of the simulator $\mathsf{Sim}_{f,g,h}$ was provided in Figure 4. More formally, we will use a series of statistically close hybrids to show that

$$\left\{\begin{array}{c}(c,(w,L),R) \sim \mathrm{Enc}(m) \\ \widetilde{c}=f(c),\ (\widetilde{w},\widetilde{L})=g(w,L),\ \widetilde{R}=h(R) \\ \text{Output: } \tilde{m}=\mathrm{Dec}\left(\widetilde{c},(\widetilde{w},\widetilde{L}),\widetilde{R}\right)\end{array}\right\} = \mathrm{Tamper}^{m}_{f,g,h} \approx \mathrm{copy}\left(\mathrm{Sim}_{f,g,h},m\right)$$

Throughout this section, we use the following color/highlight notation. In a current hybrid, the text in red denotes the changes from the previous hybrid. The text in shaded part represents the steps that will be replaced by red part of the next hybrid.

Our first hybrid is exactly the same as $\mathrm{Tamper}^{m}_{f,g,h}$. We just open up the definition of Enc and Dec.

$H_0(f,g,h,m)$:

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r=\mathrm{Ext}(w,s),\ c=m\oplus r,\ t_1=\mathrm{Tag}_{k_1}(c),\ t_2=\mathrm{Tag}'_{k_2}(w)$
3. $(L,R) \sim \mathrm{Enc}^+(k_1,k_2,t_1,t_2,s)$
4. $\widetilde{c}=f(c),\ (\widetilde{w},\widetilde{L})=g(w,L),\ \widetilde{R}=h(R)$
5. $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s})=\mathrm{Dec}^+(\widetilde{L},\widetilde{R})$
6. If $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s})=\perp$, output $\perp$
7. Else if $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c},\widetilde{t_1})=0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w},\widetilde{t_2})=0\right)$, output $\perp$
8. Else output $\widetilde{c}\oplus\mathrm{Ext}(\widetilde{w},\widetilde{s})$

In the next hybrid, we re-write $(\widetilde{w},\widetilde{L})=g(w,L)$ as $\widetilde{w}=g_L(w)$ and $\widetilde{L}=g_w(L)$. The hybrids $H_0(f,g,h,m)$ and $H_1(f,g,h,m)$ are identical.

$H_1(f,g,h,m)$:

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r=\mathrm{Ext}(w,s),\ c=m\oplus r,\ t_1=\mathrm{Tag}_{k_1}(c),\ t_2=\mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{c}=f(c)$
4. $(L,R) \sim \mathrm{Enc}^+(k_1,k_2,t_1,t_2,s)$
5. $\widetilde{L}=g_w(L),\ \widetilde{R}=h(R)$
6. $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s})=\mathrm{Dec}^+(\widetilde{L},\widetilde{R})$
7. $\widetilde{w}=g_L(w)$
8. If $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s})=\perp$, output $\perp$
9. Else if $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c},\widetilde{t_1})=0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w},\widetilde{t_2})=0\right)$, output $\perp$
10. Else output $\widetilde{c}\oplus\mathrm{Ext}(\widetilde{w},\widetilde{s})$

Notice that step 4,5,6 in $H_1(f,g,h,m)$ is exactly $\mathrm{TamperPlus}^{(k_1,k_2,t_1,t_2,s)}_{g_w,h}$, replace this with simulator $\mathrm{SimPlus}_{g_w,h}$ gives us $H_2(f,g,h,m)$. We note that hybrids $H_1(f,g,h,m)$ and $H_2(f,g,h,m)$ are ε^+-close. If not, we can use the tampering function (g_w,h) and message (k_1,k_2,t_1,t_2,s) to break the ε^+ augmented non-malleability of $(\mathrm{Enc}^+,\mathrm{Dec}^+)$.

$H_2(f, g, h, m)$:

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \text{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \text{Tag}_{k_1}(c)$, $t_2 = \text{Tag}'_{k_2}(w)$
3. $\widetilde{c} = f(c)$
4. $(L, \text{Ans}) \sim \text{SimPlus}_{g_w, h}$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \text{copy}\left(\text{Ans}, (k_1, k_2, t_1, t_2, s)\right)$
6. $\widetilde{w} = g_L(w)$
7. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
8. Else if $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\right)$, output $\perp$
9. Else output $\widetilde{c} \oplus \text{Ext}(\widetilde{w}, \widetilde{s})$

Now in hybrid $H_3(f, g, h, m)$, instead of doing copy(), we do a case analysis on Ans. We note that the hybrids $H_2(f, g, h, m)$ and $H_3(f, g, h, m)$ are identical.

$H_3(f, g, h, m)$:

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \text{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \text{Tag}_{k_1}(c)$, $t_2 = \text{Tag}'_{k_2}(w)$
3. $\widetilde{c} = f(c)$
4. $(L, \text{Ans}) \sim \text{SimPlus}_{g_w, h}$
5. $\widetilde{w} = g_L(w)$
6. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\text{Verify}_{k_1}(\widetilde{c}, t_1)=0 \text{ or } \text{Verify}'_{k_2}(\widetilde{w}, t_2) = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{Ext}(\widetilde{w}, s)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{Ext}(\widetilde{w}, \widetilde{s})$

Next, in hybrid $H_4(f, g, h, m)$ we change the case when Ans = same*. Note that Ans = same* says that the both the authentication keys k_1, k_2 as well as the tags are unchanged. Hence, with probability at least $(1 - \mu - \mu')$, both authentications would verify only if w and c are unchanged. Hence, in $H_5(f, g, h, m)$, we check if the ciphertext c and source w are the same.

Given that $(\text{Tag}, \text{Verify})$ and $(\text{Tag}', \text{Verify}')$ are μ and μ'-secure message authentication codes, $H_3(f, g, h, m) \approx_{\mu+\mu'} H_4(f, g, h, m)$.

$H_4(f, g, h, m)$:

copy $\Big($

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s),\ c = m \oplus r,\ t_1 = \mathrm{Tag}_{k_1}(c),\ t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{c} = f(c)$
4. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g_w, h}$
5. $\widetilde{w} = g_L(w)$
6. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

We note that the variables k_1, k_2, t_1, t_2, s are no longer used in the hybrid. Hence, we remove the sampling of these in the next hybrid. It is clear that the two hybrids $H_4(f, g, h, m)$ and $H_5(f, g, h, m)$ are identical.

$H_5(f, g, h, m)$:

copy $\Big($

1. $w \sim U_n,\ s \sim U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r$
2. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g_w, h}$
3. $\widetilde{c} = f(c),\ \widetilde{w} = g_L(w)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})$=0 or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Now, we wish to use the property of average min-entropy extractor to remove the dependence between c and w. Before we do the trick, we shall first rearrange the steps in $H_5(f, g, h, m)$ to get $H_6(f, g, h, m)$. We process all the leakage we need at the first part of our hybrid and use only the leakage of w in the remaining. Intuitively, when Ans = same*, flag_1 records whether $\widetilde{w} = w$ and when Ans = $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$, flag_2 records whether $\widetilde{w}$ can pass the MAC verification under new key and tag and mask is the new one-time pad we need for the decoding the tampered message. We note that the hybrids $H_5(f, g, h, m)$ and $H_6(f, g, h, m)$ are identical.

$H_6(f, g, h, m)$:

copy $\Big($

1. $w \sim U_n$
2. $(L, \text{Ans}) \sim \text{SimPlus}_{g_w,h}$, $\widetilde{w} = g_L(w)$
3. If Ans $=$
   - Case same*: $\text{flag}_1 = 1$ iff $(\widetilde{w} = w)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_2 = 1$ iff $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$. Set mask $= \text{Ext}(\widetilde{w}, \widetilde{s})$.
4. $s \sim U_d$, $r = \text{Ext}(w, s)$, $c = m \oplus r$, $\widetilde{c} = f(c)$
5. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\text{flag}_1\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\text{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus$ mask

$, m\Big)$

In the next hybrid, we formalize $(\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$ as the leakage on source w. Note that the hybrids $H_6(f, g, h, m)$ and $H_7(f, g, h, m)$ are identical.

$H_7(f, g, h, m)$:

copy $\Big($

1. $w \sim U_n$
2. For the tampering function (g, h) we define the following leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$
   (a) $(L, \text{Ans}) \sim \text{SimPlus}_{g_w,h}$, $\widetilde{w} = g_L(w)$
   (b) If Ans $=$
   - Case same*: $\text{flag}_1 = 1$ iff $(\widetilde{w} = w)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_2 = 1$ iff $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$. Set mask $= \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $s \sim U_d$, $r = \text{Ext}(w, s)$, $c = m \oplus r$, $\widetilde{c} = f(c)$
4. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\text{flag}_1\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\text{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus$ mask

$, m\Big)$

In the next hybrid, we replace the extracted output r with a uniform random ℓ bit string. We argue that the hybrids $H_7(f, g, h, m)$ and $H_8(f, g, h, m)$ are ε_1 close for appropriate length n of source w.

Since $\mathcal{L}(w)$ outputs a $\ell+\ell^+ +3$ bits of leakage, by Lemma 1, $H_\infty(W|\mathcal{L}(W)) = k \geqslant n-(\ell+\ell^+ +3)$. Here, W denotes the random variable corresponding to w. We will pick n such that $k > \ell$ for the min-entropy extraction to give a uniform string (see Lemma 2).

$H_8(f,g,h,m)$:

$\text{copy}\Big($

1. $w \sim U_n$
2. For the tampering function (g,h) we define the following leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$
   (a) $(L, \text{Ans}) \sim \text{SimPlus}_{g_w,h}$, $\widetilde{w} = g_L(w)$
   (b) If Ans =
   - Case same*: $\text{flag}_1 = 1$ iff $(\widetilde{w} = w)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_2 = 1$ iff $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$. Set mask $= \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \sim U_\ell$, $c = m \oplus r$, $\widetilde{c} = f(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c \text{ and } \text{flag}_1\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$, m\Big)$

Finally, notice that the distribution of c is independent of m and we can use the message 0^ℓ. This gives us our simulator. Clearly $H_8(f,g,h,m) = H_9(f,g,h,m)$. Notice that $H_9(f,g,h,m) = \text{copy}\Big(\text{Sim}_{f,g,h}, m\Big)$.

$H_9(f,g,h,m)$:

$\text{copy}\Big($

1. $w \sim U_n$
2. For the tampering function (g,h) we define the following leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$
   (a) $(L, \text{Ans}) \sim \text{SimPlus}_{g_w,h}$, $\widetilde{w} = g_L(w)$
   (b) If Ans =
   - Case same*: $\text{flag}_1 = 1$ iff $(\widetilde{w} = w)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_2 = 1$ iff $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 1$. Set mask $= \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \sim U_\ell$, $c = 0^\ell \oplus r$, $\widetilde{c} = f(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c \text{ and } \text{flag}_1\Big) = 1$, output same*, else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \text{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$, m\Big)$

B Impossibility results for Streaming version of the Split State Model

Let $\mathcal{LA}_{n_1,n_2,\ldots,n_k} \subset (\{0,1\}^n)^{\{0,1\}^n}$, where $n = \sum_{i\in[k]} n_i$, denote the family of look-ahead tampering functions $f = (f^{(1)}, f^{(2)}, \ldots, f^{(k)})$ for $f^{(i)} : \{0,1\}^{\sum_{j\in[i]} n_j} \to \{0,1\}^{n_i}$ such that

$$\tilde{c} := f(c) = f^{(1)}(c_1)||f^{(2)}(c_1, c_2)||\ldots f^{(i)}(c_1, \ldots, c_i)||\ldots||f^{(k)}(c_1, \ldots, c_k)$$

for $c = c_1||c_2||\ldots||c_k$ and for all $i \in [k]$, $c_i \in \{0,1\}^{n_i}$. That is, if c consists of k parts such that i^{th} part has length n_i, then i^{th} tampered part depends on first i parts of c. We use $\mathcal{LA}_{m^{\otimes k}}$ to denote the family of look-ahead tampering functions $\mathcal{LA}_{\underbrace{m, m, \ldots, m}_{k\text{-times}}}$.

In this section, we prove an upper-bound on rate of any non-malleable encoding against 2-split-state lookahead tampering function, i.e., $\mathcal{LA}_{1^{\otimes n/2}} \times \mathcal{LA}_{1^{\otimes n/2}}$.

In our proof, we use ideas similar to [CG14a] and the following result of [CG14a] that was used in proof of Theorem 5.3.

Imported Lemma 1. *For any encoding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *with block length* n *and rate* $1 - \alpha + \delta$*, the following holds. Any codeword* c *be written as* $(c_1, c_2) \in \{0,1\}^{\alpha n} \times \{0,1\}^{(1-\alpha)n}$*. Let* $\eta = \frac{\delta}{4\alpha}$*. Then, there exists a set* $X_\eta \subseteq \{0,1\}^{\alpha n}$ *and two messages* m_0, m_1 *such that*

$$\Pr[c_1 \in X_\eta | \,\mathrm{Dec}(c) = m_0] \geqslant \eta$$
$$\Pr[c_1 \in X_\eta | \,\mathrm{Dec}(c) = m_1] \leqslant \eta/2$$

Theorem 4. *Let* $(\mathrm{Enc}, \mathrm{Dec})$ *be any encoding scheme that is non-malleable against the family of tampering functions* $\mathcal{LA}_{1^{\otimes n/2}} \times \mathcal{LA}_{1^{\otimes n/2}}$ *and achieves rate* $1/2 + \delta$*, for any constant* $\delta > 0$ *and simulation error* ε*. Then,* $\varepsilon > \delta/16\alpha$*.*

Proof. Note that any codeword c in support of Enc consists of two states c_1 and c_2, each of length $n/2$. We use $c_{i,j}$ for $i \in \{1,2\}$ and $j \in \{1, \ldots, n/2\}$ to denote the j^{th} bit in state i. Any tampering function $f = (f_1, f_2)$ generates a tampered codeword $\widetilde{c} = (\widetilde{c_1}, \widetilde{c_2}) = (f_1(c_1), f_2(c_2))$. Below, we will construct a tampering function f^* such that any simulated distribution Sim_{f^*} will be ε far from tampering distribution Tamper_{f^*}.

Next, we fix a message $\widehat{m}$ and its codeword $\widehat{c}^{(0)} = (\widehat{c}_1^{(0)}, \widehat{c}_2^{(0)}) \in \mathrm{Enc}(\widehat{m})$ such that the following holds. Let $\widehat{c}^{(1)} \in \{0,1\}^n$ be such that for all $j \in \{1, \ldots, n/2 - 1\}$, $\widehat{c}_{1,j}^{(0)} = \widehat{c}_{1,j}^{(1)}$, $\widehat{c}_{1,n/2}^{(0)} \neq \widehat{c}_{1,n/2}^{(1)}$ and $\widehat{c}_2^{(0)} = \widehat{c}_2^{(1)}$. Moreover, we require that $\mathrm{Dec}(\widehat{c}^{(1)}) \neq \widehat{m}$. That is, the two codewords are identical except the last bit of first block and the second codeword does not encode the same message[2] $\widehat{m}$. Above condition is still satisfied if $\mathrm{Dec}(\widehat{c}^{(1)}) = \perp$.

Since the rate of the given scheme $(\mathrm{Enc}, \mathrm{Dec})$ is $1 - 1/2 + \delta$ (with a constant δ), by Imported Lemma 1, we have that there exists special messages m_0, m_1 and set X_η with the above guarantees where c_1 corresponds to the first state. In fact, Imported Lemma 1 gives many such pair of messages and we will pick such that $\widehat{m}, m_0, m_1$ are all unique.

Now, our tampering function $f^* = (f^*_1, f^*_2)$ is as follows: f^* tampers a codeword $c = (c_1, c_2)$ to $\widetilde{c} = (\widetilde{c_1}, \widetilde{c_2})$ such that for all $j \in \{1, \ldots, n/2 - 1\}$, $\widetilde{c}_{1,j} = \widehat{c}_{1,j}^{(0)}$, $\widetilde{c}_{1,n/2} = \widehat{c}_{1,n/2}^{(0)}$ if $c_1 \in X_\eta$, else $\widehat{c}_{1,n/2}^{(1)}$

[2]We note that such codewords would exist otherwise we can show that the last bit of the first state is redundant for decoding. This way we can obtain a smaller encoding. Then, w.l.o.g., we can apply our argument on this new encoding.

and $\widetilde{c}_2 = \widehat{c}_2^{(0)}$. That is, if $c_1 \in X_\eta$, the resulting codeword is $\widehat{c}^{(0)}$, else it is $\widehat{c}^{(1)}$. Note that the above tampering attack can be done using a split-state look-ahead tampering function.

Finally, it is evident that for message m_0, the tampering experiment results in $\widehat{m}$ with probability at least η. On the other hand, for message m_1, the tampering experiment results in $\widehat{m}$ with probability at most $\eta/2$. Hence, probability assigned by $\mathrm{Tamper}_{f^*}^{m_0}$ and $\mathrm{Tamper}_{f^*}^{m_1}$ to message $\widehat{m}$ differs by at least $\eta/2$. Since $\widehat{m}$ is different from m_0, m_1, it holds that ε, the simulation error of non-malleable code, is at least $\eta/4$ by triangle inequality. ■

The above result can be extended to look-ahead tampering in the k-split-state model to give us the following result:

Theorem 5. *Let* $(\mathrm{Enc}, \mathrm{Dec})$ *be any encoding scheme that is non-malleable against the family of tampering functions* $\mathcal{LA}_{1^{\otimes n_1}} \ldots \times \ldots \mathcal{LA}_{1^{\otimes n_k}}$ *and achieves rate* $1/k + \delta$*, for any constant* $\delta > 0$ *and simulation error* ε*. Then,* $\varepsilon > \delta/16\alpha$*.*

Proof Outline. The proof follows by doing a similar analysis as above for the largest state. We note that the above proof does not require all states to have the same size. It suffices to have a state whose size is larger than n/k that holds by averaging argument.

C Proof of Non-Malleability in the Streaming Model (Theorem 2)

In this section, we prove that our code scheme Figure 5 is secure against the tampering family $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$.

Figure 5 presents the formal description of our coding scheme and we have the following theorem.

Theorem 6. *For all constant* α*, there exists a constant* ζ *and a non-malleable coding scheme which is secure against* $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$ *with rate* $\frac{1}{3+\alpha}$ *and error* 2^{-n^ζ}*.*

Again the perfect correctness is trivial. In order to prove the non-malleability, we need to show that for all tampering functions $(f, g) \in \mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $f = (f^{(1)}, f^{(2)})$ and $g = (g^{(1)}, g^{(2)})$, there exists a simulator $\mathrm{Sim}_{f,g}$ such that for all m, we have the following guarantee.

$$\left\{ \begin{array}{c} \big((w,R),(L,c)\big) \sim \mathrm{Enc}(m) \\ \widetilde{w} = f^{(1)}(w),\ \widetilde{R} = f^{(2)}(w,R) \\ \widetilde{L} = g^{(1)}(L),\ \widetilde{c} = g^{(2)}(L,c) \\ \text{Output: } \tilde{m} = \mathrm{Dec}\big((\widetilde{w},\widetilde{R}),(\widetilde{L},\widetilde{c})\big) \end{array} \right\} = \mathrm{Tamper}_{f,g}^{m} \approx \mathrm{copy}\left(\mathrm{Sim}_{f,g}\,,\, m\right)$$

The following sequence of hybrids will lead us from tampering experiment to the simulator. The initial hybrid represents the tampering experiment $\mathrm{Tamper}_{f,g}^{m}$ and the last hybrid represents $\mathrm{copy}(\mathrm{Sim}_{f,g}, m)$.

$H_0(f, g, m)$:

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $(L, R) \sim \mathrm{Enc}^+(k_1, k_2, t_1, t_2, s)$
4. $\widetilde{w} = f^{(1)}(w)$, $\widetilde{R} = f^{(2)}(w, R)$, $\widetilde{L} = g^{(1)}(L)$, $\widetilde{c} = g^{(2)}(L, c)$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{Dec}^+(\widetilde{L}, \widetilde{R})$
6. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
7. Else If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
8. Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Rearranging steps leads to the next hybrid.

$H_1(f, g, m)$:

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w} = f^{(1)}(w)$
4. $(L, R) \sim \mathrm{Enc}^+(k_1, k_2, t_1, t_2, s)$
5. $\widetilde{L} = g^{(1)}(L)$, $\widetilde{R} = f_w^{(2)}(R)$
6. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{Dec}^+(\widetilde{L}, \widetilde{R})$
7. $\widetilde{c} = g_L^{(2)}(c)$
8. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
9. Else If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
10. Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Now we use the augmented simulator to replace the tampering experiment of augmented two-state non-malleable codes.

$H_2(f, g, m)$:

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w} = f^{(1)}(w)$
4. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}, f_w^{(2)}}$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{copy}\Big(\mathrm{Ans}, (k_1, k_2, t_1, t_2, s)\Big)$.
6. $\widetilde{c} = g_L^{(2)}(c)$
7. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
8. Else If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
9. Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Rearrange steps gives us $H_3(f, g, m)$.

$H_3(f, g, m)$:

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s),\ c = m \oplus r,\ t_1 = \mathrm{Tag}_{k_1}(c),\ t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w} = f^{(1)}(w)$
4. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}, f_w^{(2)}}$
5. $\widetilde{c} = g_L^{(2)}(c)$
6. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\mathrm{Verify}_{k_1}(\widetilde{c}, t_1){=}0 \text{ or } \mathrm{Verify}'_{k_2}(\widetilde{w}, t_2) = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, s)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Now we use the properties of message authentication codes.

$H_4(f, g, m)$:

$\mathrm{copy}\Big($

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s),\ c = m \oplus r,\ t_1 = \mathrm{Tag}_{k_1}(c),\ t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w} = f^{(1)}(w)$
4. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}, f_w^{(2)}}$
5. $\widetilde{c} = g_L^{(2)}(c)$
6. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \widetilde{w} = w\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Clean up and remove the redundant steps.

$H_5(f, g, m)$:

$\mathrm{copy}\Big($

1. $w \sim U_n,\ s \sim U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r$
2. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}, f_w^{(2)}}$
3. $\widetilde{w} = f^{(1)}(w),\ \widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \widetilde{w} = w\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Now, compute information about w we need in the first part of the hybrid.

$H_6(f, g, m)$:

copy $\Big($

1. $w \sim U_n$
2. $(L, \text{Ans}) \sim \text{SimPlus}_{g^{(1)}, f_w^{(2)}},\ \widetilde{w} = f^{(1)}(w)$
3. If Ans =
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$.
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
4. $s \sim U_d,\ r = \text{Ext}(w, s),\ c = m \oplus r,\ \widetilde{c} = g_L^{(2)}(c)$
5. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$, m\Big)$

Formally define the information as a leakage function of w.

$H_7(f, g, m)$:

copy $\Big($

1. $w \sim U_n$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{p_1} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ as the following function:
   (a) $(L, \text{Ans}) \sim \text{SimPlus}_{g^{(1)}, f_w^{(2)}},\ \widetilde{w} = f^{(1)}(w)$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0,\ \text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $s \sim U_d,\ r = \text{Ext}(w, s)$, $c = m \oplus r,\ \widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$, m\Big)$

Using the property of average min-entropy extractor to replace the extraction step with uniform random bits.

$H_8(f, g, m)$:

copy $\Big($

1. $w \sim U_n$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{p_1} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ as the following function:
   (a) $(L, \text{Ans}) \sim \text{SimPlus}_{g^{(1)}, f_w^{(2)}}$, $\widetilde{w} = f^{(1)}(w)$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0$, $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \sim U_\ell$, $c = m \oplus r$, $\widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$, m\Big)$

Finally, fixing the message to 0^ℓ would not affect the distribution of the output of our hybrid. This last hybrid is our simulator.

$H_9(f, g, m)$:

copy $\Big($

1. $w \sim U_n$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{p_1} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ as the following function:
   (a) $(L, \text{Ans}) \sim \text{SimPlus}_{g^{(1)}, f_w^{(2)}}$, $\widetilde{w} = f^{(1)}(w)$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0$, $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0$, $\text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \sim U_\ell$, $c = 0^\ell \oplus r$, $\widetilde{c} = g_L^{(2)}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$, m\Big)$

D Proof of Non-malleability against Forgetful Functions (Theorem 3)

In this section, we shall prove Theorem 3. We now give a construction Figure 6 of constant-rate non-malleable code against $\left(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\right) \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$.

$\text{Enc}(m)$:

1. Sample $w_1 \sim U_n$, $w_2 \sim U_{n'}$, $s \sim U_d$, $k_1 \sim U_\gamma$, $k_2 \sim U_{\gamma'}$, Let $w := (w_1, w_2)$
2. Compute $r = \text{Ext}(w, s)$, $c = m \oplus r$
3. Compute the tags $t_1 = \text{Tag}_{k_1}(c)$, $t_2 = \text{Tag}'_{k_2}(w)$
4. Compute the 2-state non-malleable encoding $(L, R) \sim \text{Enc}^+(k_1, k_2, t_1, t_2, s)$
5. Output the four states $w_1, R, (w_2, L), c$

$\text{Dec}\left(c_1, c_2, c_3, c_4\right)$:

1. Let the tampered states be $\widetilde{w_1} := c_1$, $\widetilde{R} := c_2$, $(\widetilde{w_2}, \widetilde{L}) := c_3$, $\widetilde{c} := c_4$, Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
2. Decrypt $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \text{Dec}^+(\widetilde{L}, \widetilde{R})$
3. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
4. Else If $\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0$ or $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0$, output $\perp$
5. Else Output $\widetilde{c} \oplus \text{Ext}(\widetilde{w}, \widetilde{s})$

Figure 6: Non-malleable coding scheme against $\left(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\right) \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$

Now we divide the proof of non-malleability into two parts. In Subsection D.1, we show our coding scheme is non-malleable against tampering from $\mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$. In Subsection D.2, we show non-malleability against $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$. Together they prove the

non-malleability of our coding scheme.

D.1 Non-malleability against $\mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$

In this section, for codeword $c = (c_1, c_2, \ldots, c_k)$, we write c_{-i} to denote $(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_k)$. Intuitively, our scheme is secure when the tampering function forget about the first or third state because forgetting any one of those two states essentially means forgetting about the message. Specifically, if we use c^m to denote the random variable $\mathrm{Enc}(m)$, we are going to show that for all $m \neq m'$,

$$c^m_{-i} \approx_{\varepsilon_1} c^{m'}_{-i} \quad i = 1 \text{ or } 3 \tag{1}$$

Recall ε_1 is the error of our extractor Ext. This would immediately imply non-malleability because for all $f \in \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{i\}}$, we could write (see Section 5 for definition of forgetful family)

$$\mathrm{Dec}(f(\mathrm{Enc}(m))) = \mathrm{Dec}(g(c^m_{-i})) \approx_{\varepsilon_1} \mathrm{Dec}(g(c^{m'}_{-i})) = \mathrm{Dec}(f(\mathrm{Enc}(m')))$$

We shall prove Equation 1 for $i = 1$ next. Fix keys k_1, k_2, if given leakage t_2 and w_2, we still have $\widetilde{H}_\infty(w|t_2, w_2) \geqslant k$, by the property of our strong average min-entropy extractor, we have

$$k_1, k_2, t_2, w_2, s, \mathrm{Ext}(w, s) \approx_{\varepsilon_1} k_1, k_2, t_2, w_2, s, U_\ell$$

Therefore, we have (recall we use r to denote $\mathrm{Ext}(w, s)$)

$$k_1, k_2, t_2, w_2, s, r \oplus m \approx_{\varepsilon_1} k_1, k_2, t_2, w_2, s, r \oplus m'$$

which leads to (since t_1 is a deterministic function of k_1 and $c = r \oplus m$)

$$(k_1, k_2, t_1, t_2, s), w_2, r \oplus m \approx_{\varepsilon_1} (k_1, k_2, t_1, t_2, s), w_2, r \oplus m'$$

which implies

$$R, (w_2, L), r \oplus m \approx_{\varepsilon_1} R, (w_2, L), r \oplus m'$$

which is equivalent to

$$c^m_{-1} \approx_{\varepsilon_1} c^{m'}_{-1}$$

Using similar arguments, as long as $\widetilde{H}_\infty(w|t_2, w_1) \geqslant k$, we have

$$c^m_{-3} \approx c^{m'}_{-3}$$

Note that this also requires w_2 to have length $l + o(l)$.

D.2 Non-malleability against $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$

In order to prove non-malleability, we need to show that for all tampering $(f, g) \in \mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $f = (f^{(1)}, f^{(2)})$ and $g = (g^{(1)}, g^{(2)})$, there exists a simulator $\mathrm{Sim}_{f,g}$ such that for all m,

$$\left\{\begin{array}{c} \big((w_1, R, (w_2, L), c)\big) \sim \mathrm{Enc}(m) \\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{R} = f^{(2)}(w_1, R) \\ (\widetilde{w_2}, \widetilde{L}) = g^{(1)}(w_2, L),\ \widetilde{c} = g^{(2)}(w_2, L, c) \\ \text{Output: } \tilde{m} = \mathrm{Dec}\left(\widetilde{w_1}, \widetilde{R}, (\widetilde{w_2}, \widetilde{L}), \widetilde{c}\right) \end{array}\right\} = \mathrm{Tamper}^m_{f,g} \approx \mathrm{copy}\left(\mathrm{Sim}_{f,g}\,,\, m\right)$$

The following hybrid will lead us from tampering experiment to the simulator.

$H_0(f, g, m)$:

1. $w_1 \sim U_n,\ w_2 \sim U_{n'}\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$. Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $(L, R) \sim \mathrm{Enc}^+(k_1, k_2, t_1, t_2, s)$
4. $\widetilde{w_1} = f^{(1)}(w_1)$, $\widetilde{R} = f^{(2)}(w_1, R)$, $(\widetilde{w_2}, \widetilde{L}) = g^{(1)}(w_2, L)$, $\widetilde{c} = g^{(2)}(w_2, L, c)$. Let $\widetilde{w} = (\widetilde{w_1}, \widetilde{w_2})$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{Dec}^+(\widetilde{L}, \widetilde{R})$
6. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
7. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
8. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Decompose the shaded equation into individual tampering equations.

$H_1(f, g, m)$:

1. $w_1 \sim U_n,\ w_2 \sim U_{n'},\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$. Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w_1} = f^{(1)}(w_1)$
4. $(L, R) \sim \mathrm{Enc}^+(k_1, k_2, t_1, t_2, s)$
5. $\widetilde{L} = g^{(1)}_{w_2}(L)$, $\widetilde{R} = f^{(2)}_{w_1}(R)$
6. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{Dec}^+(\widetilde{L}, \widetilde{R})$
7. $\widetilde{c} = g^{(2)}_{w_2, L}(c)$, $\widetilde{w_2} = g^{(1)}_{L}(w_2)$. Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
8. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
9. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
10. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Use SimPlus to replace the tampering experiment of augmented 2-state non-malleable code.

$H_2(f, g, m)$:

1. $w_1 \sim U_n,\ w_2 \sim U_{n'},\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$. Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w_1} = f^{(1)}(w_1)$
4. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}}$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{copy}\Big(\mathrm{Ans}, (k_1, k_2, t_1, t_2, s)\Big)$
6. $\widetilde{c} = g^{(2)}_{w_2, L}(c)$, $\widetilde{w_2} = g^{(1)}_{L}(w_2)$. Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
7. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
8. Else if $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
9. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Rearrange steps.

$H_3(f, g, m)$:

1. $w_1 \sim U_n,\ w_2 \sim U_{n'},\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$ Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w_1} = f^{(1)}(w_1)$
4. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}}$
5. $\widetilde{c} = g^{(2)}_{w_2,L}(c)$, $\widetilde{w_2} = g^{(1)}_{L}(w_2)$. Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
6. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\mathrm{Verify}_{k_1}(\widetilde{c}, t_1){=}0$ or $\mathrm{Verify}'_{k_2}(\widetilde{w}, t_2) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, s)$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Use the property of message authentication codes.

$H_4(f, g, m)$:

copy $\Big($

1. $w_1 \sim U_n,\ w_2 \sim U_{n'},\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$ Let $w := (w_1, w_2)$
2. $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$, $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$
3. $\widetilde{w_1} = f^{(1)}(w_1)$
4. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}}$
5. $\widetilde{c} = g^{(2)}_{w_2,L}(c)$, $\widetilde{w_2} = g^{(1)}_{L}(w_2)$. Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
6. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Remove the redundant steps.

$H_5(f, g, m)$:

copy $\Big($

1. $w_1 \sim U_n,\ w_2 \sim U_{n'},\ s \sim U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r$ Let $w := (w_1, w_2)$
2. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}}$
3. $\widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2}),\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
4. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\widetilde{w} = w\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Process the leakage on w in the first part of our hybrid and only use the leakage in the remainder of our hybrid.

$H_6(f, g, m)$:

copy $\Big($

1. $w_1 \sim U_n,\ w_2 \sim U_{n'}$. Let $w := (w_1, w_2)$
2. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}},\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
3. If Ans $=$
   - Case same*: If $(\widetilde{w} = w)$, $\mathrm{flag}_1 = 1$; Else $\mathrm{flag}_1 = 0$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{w}, \widetilde{t_2})\Big) = 1, \mathrm{flag}_2 = 1$, Else $\mathrm{flag}_2 = 0$
     Let mask $= \mathrm{Ext}(\widetilde{w}, \widetilde{s})$
4. $s \sim U_d,\ r = \mathrm{Ext}(w, s),\ c = m \oplus r,\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
5. If Ans $=$
   - Case $\perp$: Output $\perp$
   - Case same*: If $\Big(\widetilde{c} = c$ and $\mathrm{flag}_1\Big) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{flag}_2 = 0\Big)$, output $\perp$
     Else output $\widetilde{c} \oplus$ mask

$, m\Big)$

Formally define the leakage function.

$H_7(f, g, m)$:

copy $\Big($

1. $w_1 \sim U_n,\ w_2 \sim U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{p_1} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^{\ell}$ as the following function:
   (a) $(L, \text{Ans}) \sim \text{SimPlus}^{(1)}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}},\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0,\ \text{flag}_2 = 0,\ \text{mask} = 0^{\ell}$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0,\ \text{mask} = 0^{\ell}$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
   (c) $\mathcal{L}(w) := (w_2, L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $s \sim U_d,\ r = \text{Ext}(w, s)$, $c = m \oplus r,\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$, m\Big)$

Use the property of min-entropy extractor to replace extraction step with true uniform bits.

$H_8(f, g, m)$:

copy $\Big($

1. $w_1 \sim U_n,\ w_2 \sim U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{p_1} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ as the following function:
   (a) $(L, \text{Ans}) \sim \text{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}},\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0,\ \text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$
     $\text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$
     If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
     Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (w_2, L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \sim U_\ell,\ c = m \oplus r$, $\widetilde{c} = g^{(2)}_{w_2, L}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same*
     Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$, m\Big)$

Now, we are finally ready to replace m with 0^ℓ. And this give us the hybrid.

$H_9(f, g, m)$:

copy $\Big($

1. $w_1 \sim U_n,\ w_2 \sim U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{p_1} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^{\ell}$ as the following function:
   (a) $(L, \text{Ans}) \sim \text{SimPlus}_{g^{(1)}_{w_2}, f^{(2)}_{w_1}},\ \widetilde{w_1} = f^{(1)}(w_1),\ \widetilde{w_2} = g^{(1)}_L(w_2)$ Let $\widetilde{w} := (\widetilde{w_1}, \widetilde{w_2})$
   (b) If Ans =
   - Case $\perp$: $\text{flag}_1 = 0,\ \text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case same*: If $(\widetilde{w} = w)$, $\text{flag}_1 = 1$; Else $\text{flag}_1 = 0$ $\text{flag}_2 = 0,\ \text{mask} = 0^\ell$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$ If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$ Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$

   (c) $\mathcal{L}(w) := (w_2, L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$
3. $r \sim U_\ell,\ c = 0^\ell \oplus r,\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
4. If Ans =
   - Case $\perp$: Output $\perp$
   - Case same*: If $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, output same* Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0 \text{ or } \text{flag}_2 = 0\right)$, output $\perp$ Else output $\widetilde{c} \oplus \text{mask}$

$\Big)$, m

Notice that in our hybrid argument, we have some additional leakage w_2 of w, which is of length $l + o(l)$ by our analysis in Subsection D.1. Therefore, the total leakage of w is $2l + o(l)$ and that makes w of length $3l + o(l)$ in our construction.

E Message Authentication Code: Choice of Parameters

Lemma 3. *Suppose $\{h_k : \{0,1\}^\alpha \longrightarrow \{0,1\}^\beta\}$ is a μ-almost pairwise independent hash family. Then the following family of pair of functions is a μ-secure message authentication code.*

$$\left\{\begin{matrix}\text{Tag}_k(x) = h_k(x) \\ \text{Verify}_k(x, y) = 1 \text{ if and only if } y = h_k(x)\end{matrix}\right\}_{k \in K}$$

Proof. Obviously, for all m, k, $\text{Verify}(m, h_k(m)) = 1$. Also, for all $m \neq m'$ and t, t',

$$\Pr_{k \sim U_K}\Big[\text{Tag}_k(m') = t' \Big| \text{Tag}_k(m) = t\Big] = \frac{\Pr_{k \sim U_K}[\text{Tag}_k(m') = t' \wedge \text{Tag}_k(m) = t]}{\Pr_{k \sim U_K}[\text{Tag}_k(m) = t]} \leqslant \frac{\mu \cdot 2^{-\beta}}{2^{-\beta}} = \mu \quad \blacksquare$$

Lemma 4. *Suppose $\alpha = \ell \cdot \beta$ and write m as $(m_1, m_2, \ldots, m_\ell)$ where $m_i \in \{0,1\}^\beta$. Let $K = \{0,1\}^\beta \times \{0,1\}^\beta$ and write k as (k_1, k_2). Define $h_{k_1,k_2}(m) = k_1 + m_1 k_2 + m_2 k_2^2 + \cdots + m_l k_2^\ell$, which is seen as a polynomial in $\mathbb{GF}[2^\beta]$. Then $\{h_k\}$ is a $\frac{\alpha}{\beta \cdot 2^\beta}$-almost pairwise independent hash family.*

Proof. For all m, t,

$$\Pr_{k \sim U_{2\beta}}[h_k(m) = t] = \Pr_{k_2 \sim U_\beta}\left[\Pr_{k_1 \sim U_\beta}\left[k_1 + m_1 k_2 + m_2 k_2^2 + \cdots + m_l k_2^\ell = t\right]\right] = \Pr_{k_2 \sim U_\beta}\left[2^{-\beta}\right] = 2^{-\beta}$$

For all $m \neq m'$ and t, t',

$$\begin{aligned}
&\Pr_{k\sim U_{2\beta}}[h_k(m) = t \wedge h_k(m') = t'] \\
&= \Pr_{k_1\sim U_\beta, k_2\sim U_\beta}\left[k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell = t \wedge k_1 + m'_1k_2 + m'_2k_2^2 + \cdots + m'_\ell k_2^\ell = t'\right] \\
&= \Pr_{k_2\sim U_\beta}\left[\sum_{i=1}^{\ell}(m_i - m'_i)k_2^i = t - t'\right] \cdot \Pr_{k_1\sim U_\beta}\left[k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell = t\right] \\
&\leqslant \frac{\ell}{2^\beta} \cdot 2^{-\beta}
\end{aligned}$$

where the last inequality is because a degree ℓ polynomial in a field can have at most ℓ many zeros. Since $\ell = \alpha/\beta$, this completes the proof. ■

Corollary 7. *For all message length α and Tag length β, there exists a $\frac{\alpha}{2^\beta}$-secure message authentication code scheme with key length 2β.*