Non-malleable Codes against Lookahead Tampering*

Divya Gupta
Microsoft Research, Bangalore, India
divyagupta.iitd@gmail.com

Hemanta K. Maji
Department of Computer Science, Purdue University, USA
hmaji@purdue.edu

Mingyuan Wang
Department of Computer Science, Purdue University, USA
wang1929@purdue.edu

Abstract

There are natural cryptographic applications where an adversary only gets to tamper a high-speed data stream on the fly based on her view so far, namely, the lookahead tampering model. Since the adversary can easily substitute transmitted messages with her messages, it is farfetched to insist on strong guarantees like error-correction or, even, manipulation detection. Dziembowski, Pietrzak, and Wichs (ICS–2010) introduced the notion of non-malleable codes that provide a useful message integrity for such scenarios. 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.

Our work studies the following tampering model. We encode a message into $k \geqslant 1$ secret shares, and we transmit each share as a separate stream of data. Adversaries can perform lookahead tampering on each share, albeit, independently. We call this *k-lookahead model*.

First, we show a hardness result for the k-lookahead model. To transmit an ℓ-bit message, the cumulative length of the secret shares must be at least $\frac{k}{k-1}\ell$. This result immediately rules out the possibility of a solution with $k = 1$. Next, we construct a solution for 2-lookahead model such that the total length of the shares is 3ℓ, which is only 1.5x of the optimal encoding as indicated by our hardness result.

Prior work considers stronger model of split-state encoding that creates $k \geqslant 2$ secret shares, but protects against adversaries who perform arbitrary (but independent) tampering on each secret share. The size of the secret shares of the most efficient 2-split-state encoding is $\ell \log \ell / \log\log \ell$ (Li, ECCC–2018). Even though k-lookahead is a weaker tampering class, our hardness result matches that of k-split-state tampering by Cheraghchi and Guruswami (TCC–2014). However, our explicit constructions above achieve much higher efficiency in encoding.

Keywords and phrases Non-malleable Codes, Lookahead Tampering, Split-state, Constant-rate

Funding The research effort is supported in part by an NSF CRII Award CNS-1566499, an NSF SMALL Award CNS-1618822, an REU CNS-1724673 and Purdue Research Foundation grant.

Contents

1 Introduction

Dziembowski, Pietrzak, and Wichs [DPW10] introduced the powerful notion of *non-malleable codes* for message integrity for scenarios where error-correction or, even, error-detection is impossible. Some of the main applications of non-malleable codes are tamper resilient storage and computation [DPW10], and *non-malleable message transmission* between two parties [GK18]. In this work, we focus on the application of non-malleable message transmission. Intuitively, non-malleable coding scheme guarantees that the decoding of the *tampered* codeword is either the original message or an unrelated message and the probability of either of these events happening is independent of the original message. To build such a scheme against some of the simpler tampering functions such as adding an arbitrary low Hamming weight error, the sender can encode the message using appropriate error-correcting codes, and the receiver would always recover the original message (by error correcting). Moreover, against the family of tampering functions that add an arbitrary constant, the sender can use Algebraic Manipulation Detection codes to help the receiver detect the tampering with high probability [CDF+08]. However, against more complex tampering functions, where error correction or detection are impossible, non-malleable codes can still give the following meaningful guarantee: Let $(\mathsf{Enc}, \mathsf{Dec})$ be the encoding and decoding algorithms for messages in $\{0,1\}^\ell$ against the tampering family $\mathcal{F}$. Then, for any message $m \in \{0,1\}^\ell$, $f \in \mathcal{F}$, 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 its distribution. 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, it is impossible to construct non-malleable codes that are secure against class of all tampering functions. For instance, the adversary can intercept the entire encoding, decode the transmitted 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. So, it is necessary to ensure that the decoding algorithm Dec (or any of its approximations) does not lie in the tampering function family itself. Therefore, non-malleable codes are typically constructed against a restricted class of tampering functions. Next, we discuss some tampering families considered in this work.

1.0.0.1 Lookahead Tampering & Non-malleable Messaging.

Consider the motivating application of non-malleable message transmission, where the high-speed network switches routing the communication between parties shall forward their data packets at several gigabits per second. An adversary, who is monitoring the communication at a network switch, cannot block or slow the information stream, which would outrightly signal her intrusion. So, the adversary is naturally left to innocuously substituting data packets based on all the information that she has seen so far, namely, the *lookahead tampering* model [ADKO15, CGM+16]. This restricts the tampering power of the adversary as she cannot tamper the encoding arbitrarily.

1.0.0.2 Split-State Tampering.

A widely studied setting is k-split-state tampering [DKO13, ADL14, CZ14, ADKO15, Li17, KOS17]. Here, message is encoded into k states and the adversary can only tamper each

of the states independently (and arbitrarily). 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 $f = (f_1, f_2, \ldots, f_k)$ s.t. the function $f_i : \{0,1\}^{n_i} \to \{0,1\}^{n_i}$ is an arbitrary function. Note that the tampering function only sees single states locally, and decoding requires aggregating information across all states.

1.0.0.3 Our Objective.

Motivated by applications like non-malleable message transmission over high-speed networks, our work studies the limits of the efficiency of constructing non-malleable codes in the k-split-state model where a lookahead adversary tampers each state independently, i.e., the *k-lookahead model*. We know that constructing non-malleable codes against single state, i.e., $k = 1$, lookahead adversary is impossible [CGM+16]. So, we consider the next best setting of 2-split-state lookahead tampering, where the message is encoded into 2 states and transmitted using 2 independent paths. Each of these states is tampered independently using lookahead tampering. Since split-state lookahead tampering is a sub-class of split-state tampering, a conservative approach is to use generic non-malleable codes in the k-split-state, which protect against arbitrary split-state tamperings. Prior to our work, the most efficient non-malleable codes achieved rate $R = \log\log\ell/\log\ell$ for $k = 2$ [Li18], and rate $R = 1/3$ for $k = 4$ [KOS17]. In a concurrent and independent work, [KOS18] achieves rate $R = 1/3$ for $k = 3$.

As illustrated above, there are natural cryptographic applications where lookahead attacks appropriately model the adversarial threat. We ask the following question: Can we leverage the structure of the lookahead tampering to construct a constant rate non-malleable code that requires establishing least number of, i.e., only 2, independent communication routes between the sender and the receiver?

1.0.0.4 Our Results.

We first prove an upper-bound that the rate of any non-malleable code in the 2-split-state lookahead model is at most 1/2. Next, we construct a non-malleable code for the 2-lookahead model with rate $R = 1/3$, which is 2/3-close to the above mentioned optimal upper-bound. En route, we also independently construct a 3-split-state non-malleable code that achieves rate $R = 1/3$. The starting point of all our non-malleable code constructions is the recent construction of [KOS17] in the 4-split-state model.

Finally, we interpret our results in the context of the original motivating example of non-malleable message transmission. It is necessary to establish at least two independent routes of communication to facilitate non-malleable message transmission between two parties. We show that the cumulative size of the encoding of the message sent by the sender must be at least twice the message length when the sender transmits the shares of the encoded message over two independent routes. For this setting, we provide a construction where the encoding of the message is (roughly) three-times the size of the message (1.5x the optimal solution).

1.1 Our Contribution

Let $\mathcal{S}_n$ represent the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. We call any subset $\mathcal{F} \subseteq \mathcal{S}_n$ a *tampering family* on $\{0,1\}^n$. We denote k-split-state tampering families on $\{0,1\}^{n_1+n_2+\cdots+n_k}$ by $\mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_k$, where $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$ are tampering families

on $\{0,1\}^{n_1}, \{0,1\}^{n_2}, \ldots, \{0,1\}^{n_k}$. Here, the codeword is distributed over k states of size $n_1, n_2, \ldots, n_k$.

1.1.0.1 (Split-State) Lookahead Tampering.

Motivated by the example in the introduction, instead of considering an arbitrary tampering function for each state, we consider tampering functions that encounter the information 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 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
2. The function $f(x_1, x_2, \ldots, x_B)$ is the concatenation of $f^{(i)}(x_1, x_2, \ldots, x_i)$, i.e., $f(x_1, x_2, \ldots, x_B) = f^{(1)}(x_1)||f^{(2)}(x_1, x_2)||\cdots||f^{(B)}(x_1, x_2, \ldots, x_B)$

Intuitively, the codeword arrives as B blocks of information, and the i-th block is tampered based on all the blocks so far $\{1, 2, \ldots, i\}$. In the k-split-state lookahead tampering, denoted by *k-lookahead*, the tampering function for each state is a lookahead function. The k-lookahead tampering family was introduced in [ADKO15] for the purpose of constructing non-malleable codes in the 2-split-state model. A similar notion called block-wise tampering function was introduced by [CGM+16]. Our first result is the hardness result. We give a more precise statement of this result in Theorem 5.

▶ **Theorem 1.** *For k-lookahead tampering family, the best achievable rate is $1 - 1/k$.*

In fact, we prove the above upper bound for the weakest tampering family in this class where each block in lookahead tampering is a single bit, i.e., $\mathcal{LA}_{n_1,n_2,\ldots,n_B}$ s.t. $B = n$ and $n_i = 1$. For brevity, we represent this function by $\mathcal{LA}_{1^{\otimes n}}$. Surprisingly, analogous to the result of Cheraghchi and Guruswami [CG14a] for the k-split-state model, we prove that even against significantly more restricted k-lookahead tampering $\mathcal{LA}_{1^{\otimes n_1}} \times \cdots \times \mathcal{LA}_{1^{\otimes n_k}}$, the rate of any non-malleable code is at most $1 - 1/k$ (see Subsection 3.1).

We use Fig. 1 to summarize our positive results in k-lookahead and k-split-state model and position our results relative to relevant prior works. Intuitively, lower the k, the more powerful is the tampering family, and the harder it is to construct the non-malleable codes. The state-of-the-art in non-malleable code construction against k-lookahead coincides with the general k-split-state model. In particular, no constant-rate non-malleable codes are known even against the restricted 2-lookahead model. We resolve this open question in the positive (with 2/3 the optimal rate).

▶ **Theorem 2** (Rate-1/3 NMC against 2-Lookahead)**.** *There exists a computationally efficient non-malleable code, with negligible simulation error, against the 2-lookahead tampering $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $n_1 = (2 + o(1))\ell$, $n_2 = o(\ell)$, $n_3 = o(\ell)$, $n_4 = \ell$, where ℓ is the length of the message.*

We start from the construction of 4-split-state non-malleable codes by Kanukruthi et al. [KOS17] and leverage a unique characteristic of the (rate-0) 2-split-state code of Aggarwal, Dodis, and Lovett [ADL14], namely *augmented non-malleability* that was identified by [AAG+16].

By manipulating the way we store information in the construction of Theorem 2, we also obtain the first constant-rate non-malleable codes in 3-split-state.[1]

[1] Concurrent and independent work of [KOS18] obtained similar result.

▶ **Theorem 3** (Rate-1/3 NMC in 3-Split-State)**.** *There exists a computationally efficient 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* $n_1 = \ell$*,* $n_2 = (2 + o(1))\ell$*,* $n_3 = o(\ell)$*, where* ℓ *is the length of the message.*

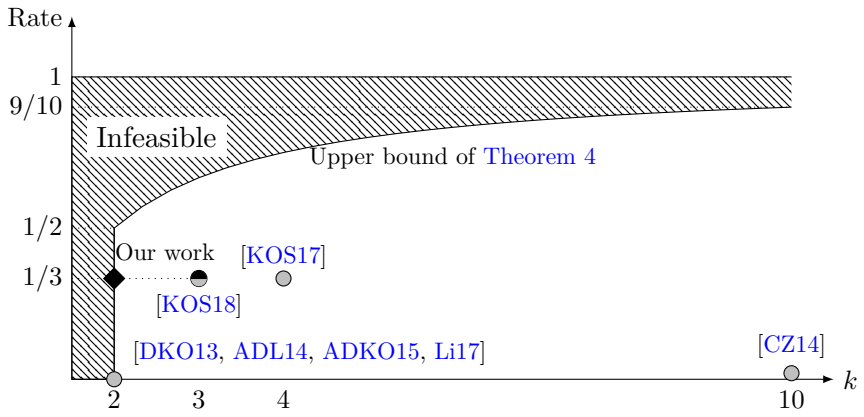

Figure 1 A comparison of the efficiency of our 2-lookahead non-malleable code with the efficiency of generic k-split-state non-malleable codes in the information-theoretic setting. The diamond represents a k-lookahead result, and the circles represent k-split-state results. Black color represents our results, and gray color represents other known results (includes both prior and concurrent works).

Lastly, [ADKO15] motivated constant-rate construction achieving non-malleability against 2-lookahead tampering 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 6 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, for a single state, 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 well-studied restriction on the tampering class is the k-split-state, 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 $\log\log\ell/\log\ell$ is achieved by [Li18]. For higher values of k, Chattopadhyay, and Zuckerman [CZ14] constructed the first constant-rate non-malleable code when $k = 10$. 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.

The computational version of this problem restricts to only computationally efficient tampering, 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 rate-1/2 in the information-theoretic setting [CG14a], and one-way functions suffice to achieve rate-1.

1.2.0.1 Lookahead Model.

In the lookahead 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. [CGM+16] first considered this family of tampering functions (referred to as block-wise tampering). In fact, they focused on the 1-lookahead family and showed the impossibility even in the computational setting. Thus, they relaxed the non-malleability guarantee and gave a construction using computational assumptions.

This family of tampering has also been considered by [ADKO15] as an interesting pit stop on the route to constructing non-malleable codes in the 2-split-state model. Specifically, they showed that given any non-malleable codes against k-lookahead tampering family together with another so-called forgetful tampering, they can transform it to get a 2-split-state non-malleable codes with only constant overhead on the rate.

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

1.2.0.2 Concurrent and Independent Work.

In a recent, concurrent and independent work, Kanukurthi et al. [KOS18] obtain a similar construction for non-malleable codes in the 3-split-state setting that achieves an identical rate as our construction. In their work, they study the problem of storing random secrets. While, the primary focus of our work is to study the family of tampering function in the lookahead model and explores the hardness of achieving non-malleability against this family of tampering functions.

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

[2] In light of the objection raised by [Li17] in the argument of [ADKO15], their constructions against lookahead tampering are flawed.

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 $\mathrm{Dec}\colon \{0,1\}^n \to \{0,1\}^\ell \cup \{\perp\}$ 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 lookahead 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

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

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 an augmented non-malleable coding scheme against tampering family $\mathcal{S}_{n_1} \times \mathcal{S}_{n_2}$ with $n_1 + n_2 = n$ and

error ε if for all functions $(f, g) \in \mathcal{S}_{n_1} \times \mathcal{S}_{n_2}$, there exists a distribution $\text{SimPlus}_{f,g}$ over $\{0,1\}^{n_1} \times (\{0,1\}^{\ell} \cup \{\perp\} \cup \{\text{same}^*\})$ such that for all $m \in \{0,1\}^{\ell}$,

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

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

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

Note that above we abuse notation for $\text{copy}\left(\text{SimPlus}_{f,g}, m\right)$. Formally, it is defined as follows: $\text{copy}\left(\text{SimPlus}_{f,g}, m\right) = (L, m)$ when $\text{SimPlus}_{f,g} = (L, \text{same}^*)$ and $\text{SimPlus}_{f,g}$ 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 $1/\text{poly}(\ell)$ and negligible error ε. More formally, the following holds.

▶ **Imported Theorem 1** ([AAG+16]). *For any message length ℓ, there is a coding scheme $(\text{Enc}^+, \text{Dec}^+)$ 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 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 $\text{Ext} : \{0,1\}^n \times \{0,1\}^d \longrightarrow \{0,1\}^{\ell}$ 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 $(\text{Ext}(A, U_d), U_d, L) \approx_{\varepsilon} (U_{\ell}, U_d, L)$.

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 these 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 uniform random strings of length arbitrarily close to the conditional min-entropy of the source.

▶ **Lemma 2** ([GUV07, Vad12]). *For all constants $\alpha > 0$ and all integers $n \geqslant k$, there exists an efficient (k, ε)-average min-entropy strong extractor $\text{Ext} : \{0,1\}^n \times \{0,1\}^d \longrightarrow \{0,1\}^{\ell}$ 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 completeness, we give a construction in Supporting Material A.

3 Non-malleable Codes against k-Lookahead

In this section, we study the k-lookahead tampering family. We first prove an upper-bound on the maximum rate that can be achieved for any non-malleable code against k-lookahead tampering family. For this, Theorem 5 states that the maximum rate that can be achieved is roughly $1 - 1/k$. Surprisingly, this matches the upper bound on the rate non-malleable codes against much stronger tampering family of k-split-state by [CG14a]. Our upper bound as well as the impossibility result by [CGM+16] rules the information theoretic construction against single state lookahead tampering. On the constructive side, for 2-lookahead model, the technically most challenging setting among k-lookahead tampering families, we construct a non-malleable code that achieves rate 1/3.

3.0.0.1 Notation.

Recall that $\mathcal{LA}_{n_1,n_2,\ldots,n_B} \subseteq (\{0,1\}^n)^{\{0,1\}^n}$, where $n = \sum_{i\in[B]} n_i$, denotes the family of lookahead tampering functions $f = (f^{(1)}, f^{(2)}, \ldots, f^{(B)})$ for $f^{(i)} : \{0,1\}^{\sum_{j\in[i]} n_j} \to \{0,1\}^{n_i}$ such that

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

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

3.1 Impossibility Results for the Split-State Lookahead Model

In this section, we first prove an upper-bound on the rate of any non-malleable encoding against 2-lookahead tampering function, where each bit is treated as a block, 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 imported lemma is used in their proof of theorem 5.3 (see [CG13]).[3]

[3] Specifically, in their proof of Theorem 5.3, they picked two messages s_0, s_1 along with X_η that satisfy the property we require for m_0, m_1 in the imported lemma. Also, we stress that their proof not only showed s_0 and s_1 exist, but there are *multiple choices* for the pair. This gives us the freedom when we pick our m_0 and m_1. We make use of this in our proof.

▶ **Imported Lemma 1.** *For any constant $0 < \delta < \alpha$ and any encoding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *with block length n and rate $1 - \alpha + \delta$, the following holds. Let the 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, $\varepsilon > \delta/8$.*

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[4] $\widehat{m}$. 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 exist 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 $\widehat{c}_{1,n/2}^{(1)}$ 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 lookahead 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 k-lookahead tampering as follows:

▶ **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 $1 - 1/k + \delta$, for any constant $\delta > 0$ and simulation error ε. Then, $\varepsilon > k\delta/16$.*

Proof Outline. The proof follows by doing a similar analysis as above for the largest state. Without loss of generality, let the first state be the largest state, i.e., $n_1 \geqslant n_i$ for all

[4] 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.

$i \in \{2, \ldots, k\}$. By averaging argument it holds that $n_1 \geqslant n/k$, where n is the block length. Now, the theorem follows along the same lines as the proof of 2-lookahead tampering above when we consider the code for the first state as c_1 and rest of the code as c_2. We note that the above proof does not require c_1 and c_2 to have the same size.

3.2 Rate-1/3 Non-malleable Code in 2-Lookahead Model

In this section, we present our construction for non-malleable codes against 2-lookahead tampering functions. Our construction relies on the following tools. Let $(\mathsf{Tag}, \mathsf{Verify})$ (resp., $(\mathsf{Tag}', \mathsf{Verify}')$) 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 $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^\ell$ be a (k, ε_1) average min-entropy strong extractor. We define k later during parameter setting. Finally, let $(\mathsf{Enc}^+, \mathsf{Dec}^+)$ 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).

3.2.0.1 Construction Overview.

We define our encoding and decoding functions formally in Fig. 2. 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 randomness r from (w, s) using the strong extractor Ext. We hide the message m using r as the one-time pad to obtain a ciphertext c. Next, we sample random keys k_1, k_2 and authenticate the ciphertext c using Tag_{k_1} and the source w using Tag'_{k_2} to obtain 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 $\big((c_1, c_2), (c_3, c_4)\big)$ where $c_1 = w$, $c_2 = R$, $c_3 = L$ and $c_4 = c$.

We also note that $n_1 := |c_1| = |w| = n$, $n_2 := |c_2| = |R| = n_2^+$, $n_3 := |c_3| = |L| = n_1^+$ and $n_4 := |c_4| = |c| = \ell$. From Fig. 2, it is evident that our construction satisfies perfect correctness.

$\mathsf{Enc}(m)$:	$\mathsf{Dec}\Big((c_1, c_2), (c_3, c_4)\Big)$:
1. Sample $w \sim U_n$, $s \sim U_d$, $k_1 \sim U_\gamma$, $k_2 \sim U_{\gamma'}$	**1.** Let the tampered states be $\widetilde{w} := c_1$, $\widetilde{R} := c_2$, $\widetilde{L} := c_3$, $\widetilde{c} := c_4$
2. Compute $r = \mathsf{Ext}(w, s)$, $c = m \oplus r$	**2.** Decrypt $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathsf{Dec}^+(\widetilde{L}, \widetilde{R})$
3. Compute the tags $t_1 = \mathsf{Tag}_{k_1}(c)$, $t_2 = \mathsf{Tag}'_{k_2}(w)$	**3.** If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
4. Compute the 2-state non-malleable encoding $(L, R) \sim \mathsf{Enc}^+(k_1, k_2, t_1, t_2, s)$	**4.** (Else) If $\mathsf{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}) = 0$ or $\mathsf{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0$, output $\perp$
5. Output the states $\Big((w, R), (L, c)\Big)$	**5.** (Else) Output $\widetilde{c} \oplus \mathsf{Ext}(\widetilde{w}, \widetilde{s})$

Figure 2 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 $n_4 = |c|$.

3.2.0.2 Proof of Non-malleability against 2-lookahead tampering.

Given a tampering function $(f, g) \in \mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$, where $f = (f^{(1)}, f^{(2)})$ and $g = (g^{(1)}, g^{(2)})$, we formally describe our simulator in Fig. 3.

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 five parts

1. $w \sim U_n$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ 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 $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$, Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
        If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
   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 $\widetilde{c} \oplus \text{mask}$

Figure 3 The simulator $\text{Sim}_{f,g}$ for the non-malleable code against 2-lookahead tampering family.

$(L, \text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$. The values L and Ans are the outputs of simulator SimPlus on tampering function $(g^{(1)}, f_w^{(2)})$, where $f_w^{(2)}$ represents the tampering function on R given w. Next, for the case when Ans = same*, flag_1 denotes the bit $\widetilde{w} = w$. When $\text{Ans} = (\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$, flag_2 captures the bit $\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})$, 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 value mask is the extracted output of tampered source $\widetilde{w}$ using tampered seed $\widetilde{s}$.

We give the formal proof on indistinguishability between simulated and tampering distributions in Subsection 3.3 using a series of statistically close hybrids.

3.2.0.3 Rate analysis.

We will use λ as our security parameter. By Corollary 1, 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 $(\text{Tag}, \text{Verify})$ and $(\text{Tag}', \text{Verify}')$ 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 Subsection 3.3, it suffices to have $n - (\ell + n_1^+ + \ell^+ + 3) = n - \ell - p(\log n + \lambda) \geqslant (1+\alpha')\ell$. 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 2.

3.3 Proof of Non-Malleability against 2-lookahead (Theorem 2)

In this section, we prove that our code scheme Fig. 2 is secure against the tampering family $\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}$. 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 $g = (g^{(1)}, g^{(2)})$, our simulator as defined in Fig. 3 satisfies that, for all m, we have

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

The following sequence of hybrids will lead us from tampering experiment to the simulator. 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.

The initial hybrid represents the tampering experiment $\mathrm{Tamper}^{m}_{f,g}$ 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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w},\widetilde{s})$

Next, we rewrite $\widetilde{R} = f^{(2)}(w,R)$ and $\widetilde{c} = g^{(2)}(L,c)$ as $\widetilde{R} = f^{(2)}_{w}(R)$ and $\widetilde{c} = g^{(2)}_{L}(c)$. Now, rearrange the steps leads us 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. $\widetilde{L} = g^{(1)}(L)$, $\widetilde{R} = f^{(2)}_{w}(R)$
6. $(\widetilde{k_1},\widetilde{k_2},\widetilde{t_1},\widetilde{t_2},\widetilde{s}) = \mathrm{Dec}^{+}(\widetilde{L},\widetilde{R})$
7. $\widetilde{c} = g^{(2)}_{L}(c)$
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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w},\widetilde{s})$

Note that shaded steps in the previous hybrid formulate a 2-state tampering experiment onto (L,R). Therefore, we could 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. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}, f^{(2)}_w}$
5. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{copy}\left(\mathrm{Ans}, (k_1, k_2, t_1, t_2, s)\right)$.
6. $\widetilde{c} = g^{(2)}_L(c)$
7. If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
8. 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 $\perp$
9. Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

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

$H_3(f, g, m)$:

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. $(L, \mathrm{Ans}) \sim \mathrm{SimPlus}_{g^{(1)}, f^{(2)}_w}$
5. $\widetilde{c} = g^{(2)}_L(c)$
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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Next, in hybrid $H_3(f, g, m)$ 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 $H_4(f, g, m)$, we check if the ciphertext c and source w are the same.

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

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

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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$\Big(\ , m\Big)$

We note that the variables k_1, k_2, t_1, t_2 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 $H_4(f, g, m)$ and $H_5(f, g, m)$ are identical.

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

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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$\Big(\ , m\Big)$

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 $H_5(f, g, m)$ to get $H_6(f, g, m)$. 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 decoding the tampered message. We note that the hybrids $H_5(f, g, m)$ and $H_6(f, g, m)$ are identical.

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

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 $\widetilde{c} \oplus \text{mask}$

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

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

1. $w \sim U_n$
2. Leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ be 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 $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: $\text{flag}_1 = 0$, Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
        If $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
   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 $\widetilde{c} \oplus \text{mask}$

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

Since $\mathcal{L}(w)$ outputs a $\ell + n_1^+ + \ell^+ + 3$ bits of leakage, by Lemma 1, $H_\infty(W|\mathcal{L}(W)) \geqslant n - (\ell + n_1^+ \ell^+ + 3)$. Here, W denotes the random variable corresponding to w. We will pick n such that $n - (\ell + n_1^+ + \ell^+ + 3) > \ell$ for the min-entropy extraction to give a uniform string (see Lemma 2).

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

1. $w \sim U_n$
2. Leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ be 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 $(\widetilde{k}_1, \widetilde{k}_2, \widetilde{t}_1, \widetilde{t}_2, \widetilde{s})$: $\text{flag}_1 = 0$, Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
        If $\left(\text{Verify}'_{\widetilde{k}_2}(\widetilde{w}, \widetilde{t}_2)\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
   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 $\widetilde{c} \oplus \text{mask}$

$\Big($ $, m\Big)$

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 $H_8(f, g, m) = H_9(f, g, m)$. Notice that $H_9(f, g, m) = \text{copy}\left(\text{Sim}_{f,g}, m\right)$.

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

1. $w \sim U_n$
2. Leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ be 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 $(\widetilde{k}_1, \widetilde{k}_2, \widetilde{t}_1, \widetilde{t}_2, \widetilde{s})$: $\text{flag}_1 = 0$, Let $\text{mask} = \text{Ext}(\widetilde{w}, \widetilde{s})$
        If $\left(\text{Verify}'_{\widetilde{k}_2}(\widetilde{w}, \widetilde{t}_2)\right) = 1$, $\text{flag}_2 = 1$; Else $\text{flag}_2 = 0$
   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 $\widetilde{c} \oplus \text{mask}$

$\Big($ $, m\Big)$

4 Construction for 3-Split-State Non-malleable Code

By re-organizing the information between states, we also obtain a rate-1/3 3-split-state non-malleable codes. Our coding scheme is defined in Fig. 4. Specifically, instead of storing w

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

Figure 4 Non-malleable coding scheme against 3-split-state tampering.

with R and L with c, we merge w and L into one state and store c, (w, L) and R independently. We present the proof of non-malleability for this coding scheme next. By similar analysis as in 2-lookahead case, it is easy to see our non-malleable codes in 3-split-state scheme also has rate-1/3.

4.1 Proof of 3-Split-State Non-malleability (Theorem 3)

Here we will prove that the encoding scheme shown in Fig. 4 is secure against 3-split-state tampering. More formally, we will show that there exists a simulator $\mathrm{Sim}_{f,g,h}$ such that

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

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 $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1}){=}0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
8. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

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 $\Big(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0$ or $\mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\Big)$, output $\perp$
10. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

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

Now, we open up the different cases of Ans. This hybrid is identical to the previous one.

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

1. $w \sim U_n,\ s \sim U_d,\ k_1 \sim U_\gamma,\ k_2 \sim U_{\gamma'}$
2. $r = \mathrm{Ext}(w, s),\ c = m \oplus r,\ t_1 = \mathrm{Tag}_{k_1}(c),\ t_2 = \mathrm{Tag}'_{k_2}(w)$
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 $\Big(\mathrm{Verify}_{k_1}(\widetilde{c}, t_1)=0$ 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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Now we use the properties of message authentication codes.

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

copy

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 $\left(\widetilde{c} = c \text{ and } \widetilde{w} = w\right) = 1$, 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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m$

Clean up and remove the redundant steps.

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

copy

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 $\left(\widetilde{c} = c \text{ and } \widetilde{w} = w\right) = 1$, 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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m$

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

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

copy

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

$, m$

Formally define the information as a leakage function of w.

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

copy

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 $\left(\widetilde{w} = w\right)$
      - 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. $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$

$\Big($, $m\Big)$

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 $\left(\widetilde{c} = c \text{ and } \text{flag}_1\right) = 1$, 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 $\widetilde{c} \oplus \text{mask}$

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

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

copy

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 $\left(\widetilde{w} = w\right)$
      - 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. $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$

$\Big($, $m\Big)$

3. $r \sim U_\ell$, $c = m \oplus r$, $\widetilde{c} = f(c)$
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 $\widetilde{c} \oplus \text{mask}$

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

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

copy

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 $\left(\widetilde{w} = w\right)$
     - 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. $\mathcal{L}(w) := (\text{Ans}, \text{flag}_1, \text{flag}_2, \text{mask})$

   $, m\Big)$
3. $r \sim U_\ell$, $c = 0^\ell \oplus r$, $\widetilde{c} = f(c)$
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 $\widetilde{c} \oplus \text{mask}$

5 Forgetful tampering in the 2-lookahead Model

In this section we restrict ourselves to the 2-lookhead 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 and do the entire tampering using only x_2, x_3, x_4. The set of all 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 $i \in \{2,3,4\}$.

Aggarwal et al. [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 following tampering family[5]

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

We make partial progress towards the goal of constructing non-malleable codes secure against above tampering family (and hence, constant rate codes against 2-split-state family), and prove the following theorem.

▶ **Theorem 6.** *For all constants α, there exists a constant ζ and a computationally efficient non-malleable coding scheme against $\left(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\right) \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$ with rate $\frac{1}{4+\alpha}$ and error 2^{-n^ζ}.*

We provide a formal proof next.

[5] Specifically, Theorem 30 in [ADKO15] states that there exists a constant-rate non-malleable reduction from 2-split-state tampering family to the following tampering function family consisting of union of split-state lookahead and forgetful tampering functions.

5.1 Proof of Non-malleability against Forgetful Functions (Theorem 6)

In this section, we shall prove Theorem 6. We now give a construction Fig. 5 of constant-rate non-malleable code against $\left(\mathcal{LA}_{n_1,n_2} \times \mathcal{LA}_{n_3,n_4}\right) \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{1\}} \cup \mathcal{FOR}_{n_1,n_2,n_3,n_4-\{3\}}$.

$\mathrm{Enc}(m)$:	$\mathrm{Dec}\left(c_1, c_2, c_3, c_4\right)$:
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 $w := (w_1, w_2)$	**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. Compute $r = \mathrm{Ext}(w, s)$, $c = m \oplus r$	**2.** Decrypt $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{Dec}^+(\widetilde{L}, \widetilde{R})$
3. Compute the tags $t_1 = \mathrm{Tag}_{k_1}(c)$, $t_2 = \mathrm{Tag}'_{k_2}(w)$	**3.** If $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \perp$, output $\perp$
4. Compute the 2-state non-malleable encoding $(L, R) \sim \mathrm{Enc}^+(k_1, k_2, t_1, t_2, s)$	**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. Output the four states $w_1, R, (w_2, L), c$	**5.** Else Output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

Figure 5 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 5.1.1, 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 5.1.2, 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.

5.1.1 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 $m \neq m'$,

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

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

$$c_{-1}^{m} \approx_{\varepsilon_1} c_{-1}^{m'}$$

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

$$c_{-3}^{m} \approx c_{-3}^{m'}$$

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

5.1.2 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 $\mathsf{Sim}_{f,g}$ such that for all m,

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

The following hybrids 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 $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\right)$, output $\perp$
8. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

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 $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\right)$, output $\perp$
10. Else output $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

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. $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s}) = \mathrm{copy}\left(\mathrm{Ans}, (k_1, k_2, t_1, t_2, s)\right)$
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 $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) = 0\right)$, output $\perp$
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 $\left(\mathrm{Verify}_{k_1}(\widetilde{c}, t_1)=0 \text{ or } \mathrm{Verify}'_{k_2}(\widetilde{w}, t_2) = 0\right)$, 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 $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \mathrm{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2}) =, \text{ output } \perp\right.$
     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 $\left(\widetilde{c} = c \text{ and } \widetilde{w} = w\right) = 1$, 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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m\Big)$

Remove the redundant steps.

$H_5(f, g, m)$:

copy

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 $\left(\widetilde{c} = c \text{ and } \widetilde{w} = w\right) = 1$, 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 $\widetilde{c} \oplus \mathrm{Ext}(\widetilde{w}, \widetilde{s})$

$, m)$

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

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 $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{w}, \widetilde{t_2})\right) = 1, \mathrm{flag}_2 = 1$, Else $\mathrm{flag}_2 = 0$
     Let $\mathrm{mask} = \mathrm{Ext}(\widetilde{w}, \widetilde{s})$
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 $\left(\widetilde{c} = c \text{ and } \mathrm{flag}_1\right) = 1$, output same*; Else output $\perp$
   - Case $(\widetilde{k_1}, \widetilde{k_2}, \widetilde{t_1}, \widetilde{t_2}, \widetilde{s})$: If $\left(\mathrm{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})=0 \text{ or } \mathrm{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \mathrm{mask}$

$, m)$

Formally define the leakage function.

$H_7(f, g, m)$:

copy

1. $w_1 \sim U_n,\ w_2 \sim U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ 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. $s \sim U_d,\ r = \text{Ext}(w, s)$, $c = m \oplus r,\ \widetilde{c} = g^{(2)}_{w_2, L}(c)$
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 $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})\text{=0 or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$\Big($ $\Big)$, m

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

$H_8(f, g, m)$:

copy

1. $w_1 \sim U_n,\ w_2 \sim U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ 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 $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, 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 $\left(\text{Verify}_{\widetilde{k_1}}(\widetilde{c}, \widetilde{t_1})\text{=0 or } \text{flag}_2 = 0\right)$, output $\perp$
     Else output $\widetilde{c} \oplus \text{mask}$

$\Big($ $\Big)$, m

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

$H_9(f, g, m)$:

copy

1. $w_1 \sim U_n,\ w_2 \sim U_{n'}$ Let $w := (w_1, w_2)$
2. Define leakage function $\mathcal{L}(w) : \{0,1\}^n \longrightarrow \{0,1\}^{n'} \times \{0,1\}^{n_1^+} \times \{0,1\}^{\beta+\beta'+\gamma+\gamma'+d+1} \times \{0,1\} \times \{0,1\} \times \{0,1\}^\ell$ 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 $\left(\text{Verify}'_{\widetilde{k_2}}(\widetilde{w}, \widetilde{t_2})\right) = 1$, 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 $\widetilde{c} \oplus \text{mask}$

Notice that in our hybrid argument, we have some additional leakage w_2 of w, which is of length $\ell + o(\ell)$ by our analysis in Subsection 5.1.1. Therefore, the total leakage of w is $2\ell + o(\ell)$ and that makes w of length $3\ell + o(\ell)$ in our construction.

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. 5, 7, 8, 9

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. 3, 5, 6, 7, 23

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. 3, 5, 6, 8, 9

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. 6

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. 6

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. 3

CG13 Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. *CoRR*, abs/1309.0458, 2013. URL: http://arxiv.org/abs/1309.0458, arXiv:1309.0458. 10

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. 5, 6, 7, 10

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. 6

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. 3, 4, 5, 7, 10

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. 6

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. 3, 6, 7

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. 3, 6

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. 9

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. 3, 6

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. 6

GK18 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, 2018. 3

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. 9

KOS17 Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. In *TCC 2017: 15th Theory of Cryptography Conference, Part II*, Lecture Notes in Computer Science, pages 344–375. Springer, Heidelberg, Germany, March 2017. 3, 4, 5, 6, 7

KOS18 Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable randomness encoders and their applications. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, pages 589–617, 2018. URL: https://doi.org/10.1007/978-3-319-78372-7_19, doi:10.1007/978-3-319-78372-7_19. 4, 5, 6, 7

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. 3, 6, 7

Li18 Xin Li. Pseudorandom correlation breakers, independence preserving mergers and their applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:28, 2018. URL: https://eccc.weizmann.ac.il/report/2018/028. 4, 7

Vad12 S.P. Vadhan. *Pseudorandomness*. Foundations and trends in theoretical computer science. Now Publishers, 2012. URL: https://books.google.com/books?id=iam4lAEACAAJ. 9

A 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.*

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

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

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

◀

▶ **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* $h_{k_1,k_2}(m) = k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell$*, 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,

$$\Pr_{k\sim U_{2\beta}}[h_k(m) = t] = \Pr_{k_2\sim U_\beta}\left[\Pr_{k_1\sim U_\beta}\left[k_1+m_1k_2+m_2k_2^2+\cdots+m_\ell k_2^\ell = t\right]\right] = \Pr_{k_2\sim U_\beta}\left[2^{-\beta}\right] = 2^{-\beta}$$

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

$$\begin{aligned}
&\Pr_{k\sim U_{2\beta}}[h_k(m) = t \wedge h_k(m') = t'] \\
&= \Pr_{k_1\sim U_\beta, k_2\sim U_\beta}\left[k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell = t \wedge k_1 + m_1'k_2 + m_2'k_2^2 + \cdots + m_\ell' k_2^\ell = t'\right] \\
&= \Pr_{k_2\sim U_\beta}\left[\sum_{i=1}^{\ell}(m_i - m_i')k_2^i = t - t'\right] \cdot \Pr_{k_1\sim U_\beta}\left[k_1 + m_1k_2 + m_2k_2^2 + \cdots + m_\ell k_2^\ell = t\right] \\
&\leqslant \frac{\ell}{2^\beta}\cdot 2^{-\beta}
\end{aligned}$$

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 1.** *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*.*