An extended abstract of this paper is published in the proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques — Eurocrypt 2014. This is the full version.

Efficient Non-Malleable Codes and Key-Derivation for Poly-Size Tampering Circuits

Sebastian Faust[*] Pratyay Mukherjee[†] Daniele Venturi[‡] Daniel Wichs[§]

May 15, 2014

Abstract

Non-malleable codes, defined by Dziembowski, Pietrzak and Wichs (ICS '10), provide roughly the following guarantee: if a codeword c encoding some message x is tampered to $c' = f(c)$ such that $c' \neq c$, then the tampered message x' contained in c' reveals no information about x. Non-malleable codes have applications to immunizing cryptosystems against tampering attacks and related-key attacks.

One *cannot* have an *efficient* non-malleable code that protects against *all efficient* tampering functions f. However, in this work we show "the next best thing": for any polynomial bound s given a-priori, there is an efficient non-malleable code that protects against all tampering functions f computable by a circuit of size s. More generally, for any family of tampering functions $\mathcal{F}$ of size $|\mathcal{F}| \leq 2^s$, there is an efficient non-malleable code that protects against all $f \in \mathcal{F}$. The *rate* of our codes, defined as the ratio of message to codeword size, approaches 1. Our results are information-theoretic and our main proof technique relies on a careful probabilistic method argument using limited independence. As a result, we get an efficiently samplable family of efficient codes, such that a random member of the family is non-malleable with overwhelming probability. Alternatively, we can view the result as providing an efficient non-malleable code in the "common reference string" (CRS) model.

We also introduce a new notion of non-malleable key derivation, which uses randomness x to derive a secret key $y = h(x)$ in such a way that, even if x is tampered to a different value $x' = f(x)$, the derived key $y' = h(x')$ does not reveal any information about y. Our results for non-malleable key derivation are analogous to those for non-malleable codes.

As a useful tool in our analysis, we rely on the notion of "leakage-resilient storage" of Davì, Dziembowski and Venturi (SCN '10) and, as a result of independent interest, we also significantly improve on the parameters of such schemes.

[*]EPFL. Lausanne, Switzerland. E-mail: `sebastian.faust@epfl.ch`.

[†]Aarhus University. Aarhus, Denmark. E-mail `pratyay@cs.au.dk`. Research supported by a European Research Commission Starting Grant (no. 279447), the CTIC and CFEM research center.

[‡]Sapienza University. Rome, Italy. E-mail: `daniele.venturi@uniroma1.it`.

[§]Northeastern University. Boston, MA, USA. E-mail: `wichs@ccs.neu.edu`. Research supported by NSF grant 1314722.

1 Introduction

Non-malleable codes were introduced by Dziembowski, Pietrzak and Wichs [20]. They provide meaningful guarantees on the integrity of an encoded message in the presence of tampering, even in settings where error-correction and error-detection may not be possible. Intuitively, a code $(\mathsf{Enc}, \mathsf{Dec})$ is non-malleable w.r.t. a family of tampering functions $\mathcal{F}$ if the message contained in a codeword modified via a function $f \in \mathcal{F}$ is either the original message, or a completely unrelated value. For example, it should not be possible to just flip 1 bit of the message by tampering the codeword via a function $f \in \mathcal{F}$. More formally, we consider an experiment Tamper_x^f in which a message x is (probabilistically) encoded to $c \leftarrow \mathsf{Enc}(x)$, the codeword is tampered to $c' = f(c)$ and, if $c' \neq c$, the experiment outputs the tampered message $x' = \mathsf{Dec}(c')$, else it outputs a special value $\mathsf{same}^\star$. We say that the code is non-malleable w.r.t. some family of tampering functions $\mathcal{F}$ if, for every function $f \in \mathcal{F}$ and every messages x, the experiment Tamper_x^f reveals almost no information about x. More precisely, we say that the code is ε-non-malleable if for every pair of messages x, x' and every $f \in \mathcal{F}$, the distributions Tamper_x^f and $\mathsf{Tamper}_{x'}^f$ are statistically ε-close. The encoding/decoding functions are public and do not contain any secret keys. This makes the notion of non-malleable codes different from (but conceptually related to) the well-studied notions of non-malleability in cryptography, introduced by the seminal work of Dolev, Dwork and Naor [18].

Relation to Error Correction/Detection. Notice that non-malleability is a weaker guarantee than error correction/detection; the latter ensure that any change in the codeword can be corrected or at least detected by the decoding procedure, whereas the former does allow the message to be modified, but only to an unrelated value. However, when studying error correction/detection we usually restrict ourselves to limited forms of tampering which preserve some notion of distance (e.g., usually hamming distance) between the original and tampered codeword. (One exception is [14], which studies error-detection for more complex tampering.) For example, it is already impossible to achieve error correction/detection for the simple family of functions $\mathcal{F}_{const}$ which, for every constant c^*, includes a "constant" function f_{c^*} that maps all inputs to c^*. There is always some function in $\mathcal{F}_{const}$ that maps everything to a *valid* codeword c^*. In contrast, it is trivial to construct codes that are non-malleable w.r.t $\mathcal{F}_{const}$, as the output of a constant function is clearly independent of its input. The prior works on non-malleable codes, together with the results from this work, show that one can construct non-malleable codes for highly complex tampering function families $\mathcal{F}$ for which error correction/detection are unachievable.

Applications to Tamper Resilience. The fact that non-malleable codes can be built for large and complex families of functions makes them particularly attractive as a mechanism for protecting memory against tampering attacks, known to be a serious threat for the security of cryptographic schemes [8, 2, 30, 13]. As shown in [20], to protect a scheme with some secret state against memory-tampering, we simply encode the state via a non-malleable code and store the encoding in the memory instead of the original secret. One can show that if the code is non-malleable with respect to function family $\mathcal{F}$, the transformed system is secure against tampering attacks carried out by any function in $\mathcal{F}$. See [20] for a discussion of the application of non-malleable codes to tamper resilience.

Limitations & Possibility. It is *impossible* to have codes that are non-malleable for *all* possible tampering functions. For any coding scheme $(\mathsf{Enc}, \mathsf{Dec})$, there exists a tampering function $f_{bad}(c)$ that recovers $x = \mathsf{Dec}(c)$, creates x' by (e.g.,) flipping the first bit of x, and outputs a valid encoding

c' of x'. Notice that if $\mathsf{Enc}, \mathsf{Dec}$ are *efficient*, then the function f_{bad} is efficient as well. Thus, it is also impossible to have an *efficient* code which is non-malleable w.r.t all *efficient* functions. Prior works [27, 11, 19, 1, 10, 21] (discussed shortly) constructed non-malleable codes for several rich and interesting function families. In all cases, the families are restricted through their *granularity* rather than their computational *complexity*. In particular, these works envision that the codeword is split into several (possibly just 2) components, each of which can only be tampered independently of the others. The tampering function therefore only operates on a "granular" rather than "global" view of the codeword.

1.1 Our Contribution

In this work, we are interested in designing non-malleable codes for large families of functions which are only restricted by their "computational complexity" rather than "granularity". As we saw, we cannot have a single efficient code that is non-malleable for all efficient tampering functions. However, we show the following positive result, which we view as the "next best thing":

Main Result: For any polynomial bound $s = s(n)$ in the codeword size n, and any tampering family $\mathcal{F}$ of size $|\mathcal{F}| \leq 2^s$, there is an efficient code of complexity $\mathsf{poly}(s, \log(1/\varepsilon))$ which is ε-non-malleable w.r.t. $\mathcal{F}$. In particular, $\mathcal{F}$ can be the family of all circuits of size at most s.

The code is secure in the information theoretic setting, and achieves *optimal rate* (message/codeword size) arbitrarily close to 1. It has a *simple* construction relying only on t-wise independent hashing.

The CRS Model. In more detail, if we fix some family $\mathcal{F}$ of tampering functions (e.g., circuits of bounded size), our result gives us a *family* of efficient codes, such that, with overwhelming probability, a random member of the family is non-malleable w.r.t $\mathcal{F}$. Each code in the family is indexed by some hash function h from a t-wise independent family of hash functions $\mathcal{H}$. This result already shows the *existence* of efficient non-malleable codes with some small *non-uniform advice* to indicate a "good" hash function h.

However, we can also efficiently sample a random member of the code family by sampling a random hash function h. Therefore, we find it most appealing to think of this result as providing a uniformly efficient *construction* of non-malleable codes in the "common reference string (CRS)" model, where a random public string consisting of the hash function h is selected once and fixes the non-malleable code. We emphasize that, although the family $\mathcal{F}$ (e.g., circuits of bounded size) is fixed prior to the choice of the CRS, the attacker can choose the tampering function $f \in \mathcal{F}$ (e.g., a particular small circuit) adaptively depending on the choice of h.

We argue that it is unlikely that we can completely de-randomize our construction and come up with a fixed uniformly-efficient code which is non-malleable for all circuits of size (say) $s = O(n^2)$. In particular, this would require a circuit lower bound, showing that the function f_{bad} (described above) *cannot* be computed by a circuit of size $O(n^2)$.

Non-Malleable Key-Derivation. As an additional contribution, we introduce a new primitive called *non-malleable key derivation*. Intuitively, a function $h : \{0,1\}^n \to \{0,1\}^k$ is a non-malleable key derivation for tampering family $\mathcal{F}$ if it guarantees that for any tampering function $f \in \mathcal{F}$, if we sample uniform randomness $x \leftarrow \{0,1\}^n$, the "derived key" $y = h(x)$ is statistically close to uniform even given $y' = h(f(x))$ derived from "tampered" randomness $f(x) \neq x$. Our positive results for non-malleable key derivation are analogous to those for non-malleable codes. One difference is that the rate k/n is now at most $1/2$ rather than 1, and we show that this is optimal.

While we believe that non-malleable key derivation is an interesting notion on its own (e.g., it can be viewed as a dual version of non-malleable extractors [17]), we also show it has useful applications for tamper resilience. For instance, consider some cryptographic scheme G using a uniform key in $y \leftarrow \{0,1\}^k$. To protect G against tampering attacks, we can store a bigger key $x \leftarrow \{0,1\}^n$ on the device and temporarily derive $y = h(x)$ each time we want to execute G. In Appendix A, we show that this approach protects any cryptographic scheme with a uniform key against one-time tampering attacks. The main advantage of using a non-malleable key-derivation rather than non-malleable codes is that the key x stored in memory is simply a uniformly random string with no particular structure (in contrast, the codeword in a non-malleable code requires structure).

In Section 6, we also show how to use non-malleable key derivation to build a tamper-resilient stream cipher. Our construction is based on a PRG $\mathsf{prg} : \{0,1\}^k \to \{0,1\}^{n+v}$ and a non-malleable key derivation function $h : \{0,1\}^n \to \{0,1\}^k$. For an initial key $s_0 \leftarrow \{0,1\}^n$, sampled uniformly at random, the output of the stream cipher at each round $i \in [q]$ is $(s_i, x_i) := \mathsf{prg}(h(s_{i-1}))$.

1.2 Our Techniques

Non-Malleable Codes. Our construction of non-malleable codes is incredibly simple and relies on t-wise independent hashing, where t is proportional to $s = \log |\mathcal{F}|$. In particular, if h_1, h_2 are two such hash functions, we encode a message x into a codeword $c = (r, z, \sigma)$ where r is randomness, $z = x \oplus h_1(r)$ and $\sigma = h_2(r, z)$. The security analysis, on the other hand, requires two independently interesting components. Firstly, we rely on the notion of *leakage-resilient encodings*, proposed by Davì, Dziembowski and Venturi [16]. These provide a method to encode a secret in such a way that a limited form of leakage on the encoding does not reveal anything about the secret. One of our contributions is to significantly improve the parameters of the construction from [16] by using a fresh and more careful analysis, which gives us such schemes with an essentially optimal rate. Secondly, we analyze a simpler/weaker notion of *bounded* non-malleability, which intuitively guarantees that an adversary seeing the decoding of a tampered codeword can learn only a bounded amount of information on the encoded value. This notion of bounded non-malleability is significantly simpler to analyze than full non-malleability. Finally, we show how to carefully combine leakage-resilient encodings with bounded non-malleability to get our full construction of non-malleable codes. On a very high (and not entirely precise) level, we can think of h_1 above as providing "leakage resilience" and h_2 as providing "bounded non-malleability".

We stress that the fact that t has to be proportional to s is not an artefact of our proof. In fact, one can see that whenever the hash function has seed size s, there is a family of 2^s functions that breaks the construction with probability 1: For each seed, just have a new function that decodes with that seed and encodes a related value. This shows that the t has to be proportional to $\log |\mathcal{F}|$.

Non-Malleable Key-Derivation. Our construction of non-malleable key-derivation functions is even simpler: a random t-wise independent hash function h already satisfies the definition with overwhelming probability, where t is proportional to $s = \log |\mathcal{F}|$. The analysis is again subtle and relies on a careful probabilistic method argument.

Similar to the case of non-malleable codes, the fact that t has to be proportional to s is necessary.

1.3 Related Works

Granular Tampering. Most of the earlier works on non-malleable codes focus on granular tampering models, where the tampering functions are restricted to act on individual components of the codeword independently. The original work of [20] (see also [12]) gives an efficient construction for bit-tampering (i.e., the adversary can tamper with each bit of the codeword independently of every other bit). Very recently, Cheraghchi and Guruswami [10] gave a construction with improved rate and better efficiency for the same family. Choi *et al.* [11] considered an extended tampering family, where the tampering function can be applied to a small (logarithmic in the security parameter) number of blocks independently.

Perhaps the least granular and most general such model is the so-called *split-state* model, where the encoding consists of two parts L (left) and R (right), and the adversary can tamper L and R *arbitrarily but independently.* Starting with the random oracle construction of [20], a few other constructions of non-malleable split-state codes have been proposed, both in the computational setting [27, 21] and in the information theoretic setting [19, 1, 10]. Notice that the family $\mathcal{F}_{split}$ of all split-state tampering functions (without restricting efficiency), has doubly exponential size $2^{O(2^{n/2})}$ in the codeword size n, and therefore it is not covered by our results, which can efficiently handle at most singly-exponential-size families $2^{\mathsf{poly}(n)}$. On the other hand, the split-state model doesn't cover "computationally simple" functions, such as the function computing the XOR or the bit-wise inner-product of L, R. Therefore, although the works are technically orthogonal, we believe that looking at computational complexity may be more natural.

Global Tampering. The work of [20] gives an *existential* (inefficient) construction of non-malleable codes for doubly-exponential sized function families. More precisely, for any constant $0 < \alpha < 1$ and any family $\mathcal{F}$ of functions of size $|\mathcal{F}| \leq 2^{2^{\alpha n}}$ in the codeword size n, there exists an inefficient non-malleable code w.r.t. $\mathcal{F}$; indeed a completely random function gives such a code with high probability. The code is clearly not efficient, and this should be expected for such a broad result: the families $\mathcal{F}$ can include all circuits of size (e.g.,) $s(n) = 2^{n/2}$, which means that the efficiency of the code must exceed $O(2^{n/2})$. Unfortunately, there is no direct way to "scale down" the result in [20] so as to get an efficient construction for singly-exponential-size families. (One can view our work as providing such "scaled down" result.) Moreover, the analysis only yielded a rate of at most $(1-\alpha)/3 < 1/3$, and it was previously not known if such codes can achieve a rate close to 1, even for "small" function families. We note that [20] also showed that the probabilistic method construction can yield efficient non-malleable codes for large function families in the *random oracle model.* However, this only considers function families that don't have access to the random oracle. For example, one cannot interpret this as giving any meaningful result for tampering functions with bounded complexity.

Concurrent and Independent Work. In a concurrent and independent work, Cheraghchi and Guruswami [9] give two related results. Firstly, they improve the probabilistic method construction of [20] and show that, for families $\mathcal{F}$ of size $|\mathcal{F}| \leq 2^{2^{\alpha n}}$, there exist (inherently inefficient) non-malleable codes with rate $1 - \alpha$, which they also show to be optimal. This gives the first characterization of the rate of non-malleable codes. Secondly, similar to our results, they use limited independence to construct efficient non-malleable codes when restricted to tampering families $\mathcal{F}$ of size $|\mathcal{F}| \leq 2^{s(n)}$ for a polynomial $s(n)$. However, the construction of [9] is not "efficient" in the usual cryptographic sense: to get error-probability ε, the encoding and decoding procedures require complexity $\mathsf{poly}(1/\varepsilon)$. If we set ε to be negligible, as usually desired in cryptography, then the encoding/decoding procedures would require super-polynomial time. In contrast, the encoding/decoding procedures in our construction have efficiency $\mathsf{poly}(\log(1/\varepsilon))$, and therefore we can set ε to be negligible while maintaining polynomial-time encoding/decoding.

Other Approaches to Achieve Tamper Resilience. There is a vast body of literature that considers tampering attacks using other approaches besides non-malleable codes. See, e.g., [5, 23, 25, 4, 22, 26, 3, 24, 28, 31, 6, 15]. The reader is referred to (e.g.,) [20] for a more detailed comparison between these approaches and non-malleable codes.

2 Preliminaries

Notation. We denote the set of first n natural numbers, i.e. $\{1, \ldots, n\}$, by $[n]$. Let X, Y be random variables with supports $S(X), S(Y)$, respectively. We define

$$\mathbf{SD}(X,Y) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{s \in S(X) \cup S(Y)} |\Pr[X = s] - \Pr[Y = s]|$$

to be their *statistical distance*. We write $X \approx_\varepsilon Y$ and say that X and Y are ε-statistically close to denote that $\mathbf{SD}(X,Y) \leq \varepsilon$. We let U_n denote the uniform distribution over $\{0,1\}^n$. We use the notation $x \leftarrow X$ to denote the process of sampling a value x according to the distribution X. If f is a randomized algorithm, we write $f(x; r)$ to denote the execution of f on input x with random coins r. We let $f(x)$ denote a random variable over the random coins.

We recall two lemmas from [7].

Lemma 2.1 (Lemma 2.3 of [7])**.** *Let $t \geq 4$ be an even integer. Suppose $X_1, \ldots, X_n$ are t-wise independent random variables taking values in $[0,1]$. Let $X := \sum_{i=1}^n X_i$ and define $\mu := \mathbf{E}[X]$ be the expectation of the sum. Then, for any $A > 0$, $\Pr[|X - \mu| \geq A] \leq K_t \left(\frac{t\mu + t^2}{A^2}\right)^{t/2}$ where $K_t \leq 8$.*

Lemma 2.2 (Lemma A.1 of [7])**.** *Let $t \geq 2$ be an even integer. Suppose $X_1, X_2, \ldots, X_n$ are t-wise independent random variables taking values in $[0,1]$. Let $X := \sum_{i=1}^n X_i$ and $\mu := \mathbf{E}[X]$ be the expectation of the sum. Then, $\mathbf{E}[(X - \mu)^t] \leq K_t \left(t\mu + t^2\right)^{t/2}$ where $K_t \leq 8$.*

Notice that Lemma 2.2 is slightly modified from Lemma A.1 of [7], where the random variables are *fully* independent. However it is easy to extend the statement in [7] to the one above by a simple observation (also stated in [7]) that, by linearity of expectation, the value of $\mathbf{E}[(X - \mu)^t]$ can be computed under the assumption that $X_1, \ldots, X_n$ are fully independent, whenever they are at least t-wise independent.

2.1 Definitions of Non-Malleable Codes

Definition 2.3 (Coding Scheme)**.** *A (k,n)-coding scheme consists of two functions: a* randomized *encoding function* $\mathsf{Enc} : \{0,1\}^k \to \{0,1\}^n$*, and a* deterministic *decoding function* $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k \cup \{\bot\}$ *such that, for each $x \in \{0,1\}^k$,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(x)) = x] = 1$.

We now define non-malleability w.r.t. some family $\mathcal{F}$ of tampering functions. The work of [20] defines a default and a strong version of non-malleability. The main difference is that, in the default version, the tampered codeword $c' \neq c$ may still encode the original message x whereas the strong version ensures that any change to the codeword completely destroys the original message. We only define the strong version below. We then add an additional strengthening which we call *super* non-malleability.

Definition 2.4 (Strong Non-Malleability [20]). *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a* (k, n)*-coding scheme and* $\mathcal{F}$ *be a family of functions* $f : \{0,1\}^n \to \{0,1\}^n$*. We say that the scheme is* $(\mathcal{F}, \varepsilon)$-non-malleable *if for any* $x_0, x_1 \in \{0,1\}^k$ *and any* $f \in \mathcal{F}$*, we have* $\mathsf{Tamper}^f_{x_0} \approx_\varepsilon \mathsf{Tamper}^f_{x_1}$ *where*

$$\mathsf{Tamper}^f_x \stackrel{def}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(x), c' := f(c), x' = \mathsf{Dec}(c') \\ \textit{Output } \mathsf{same}^\star \textit{ if } c' = c, \textit{ and } x' \textit{ otherwise.} \end{array} \right\}.$$

For *super* non-malleable security (defined below), if the tampering manages to modify c to c' such that $c' \neq c$ and $\mathsf{Dec}(c') \neq \bot$, then we will even give the attacker the tampered codeword c' in *full* rather than just giving $x' = \mathsf{Dec}(c')$. We do not immediately see a concrete application of this strengthening, but it seems sufficiently interesting to define explicitly.

Definition 2.5 (Super Non-Malleability). *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a* (k, n)*-coding scheme and* $\mathcal{F}$ *be a family of functions* $f : \{0,1\}^n \to \{0,1\}^n$*. We say that the scheme is* $(\mathcal{F}, \varepsilon)$-super non-malleable *if for any* $x_0, x_1 \in \{0,1\}^k$ *and any* $f \in \mathcal{F}$*, we have* $\mathsf{Tamper}^f_{x_0} \approx_\varepsilon \mathsf{Tamper}^f_{x_1}$ *where:*

$$\mathsf{Tamper}^f_x \stackrel{def}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(x), c' := f(c) \\ \textit{Output } \mathsf{same}^\star \textit{ if } c' = c, \textit{ output } \bot \textit{ if } \mathsf{Dec}(c') = \bot, \\ \textit{and else output } c'. \end{array} \right\}.$$

3 Improved Leakage-Resilient Codes

We will rely on leakage resilience as an important tool in our analysis. The following notion of leakage-resilient codes was defined by [16]. Informally, a code is leakage-resilient w.r.t. some leakage family $\mathcal{F}$ if, for any $f \in \mathcal{F}$, "leaking" $f(c)$ for a codeword c does not reveal anything about the encoded value.

Definition 3.1 (Leakage-Resilient Codes [16]). *Let* $(\mathsf{LREnc}, \mathsf{LRDec})$ *be a* (k, n)*-coding scheme. For a function family* $\mathcal{F}$*, we say that* $(\mathsf{LREnc}, \mathsf{LRDec})$ *is* $(\mathcal{F}, \varepsilon)$*-leakage-resilient, if for any* $f \in \mathcal{F}$ *and any* $x \in \{0,1\}^k$ *we have* $\mathbf{SD}(f(\mathsf{LREnc}(x)), f(U_n)) \leq \varepsilon$*.*

The work of [16] gave a probabilistic method construction showing that such codes exist and can be efficient when the size of the leakage family $|\mathcal{F}|$ is singly-exponential. However, the rate k/n was at most some small constant ($< \frac{1}{4}$), even when the family size $|\mathcal{F}|$ and the leakage size ℓ are small. Here, we take the construction of [16] and give an improved analysis with improved parameters, showing that the rate can approach 1. In particular, the additive overhead of the code is very close to the leakage-amount ℓ, which is optimal. Our result and analysis are also related to the "high-moment crooked leftover hash lemma" of [29], although our construction is somewhat different, relying only on high-independence hash functions rather than permutations.

Construction. Let $\mathcal{H}$ be a t-wise independent function family consisting of functions $h : \{0,1\}^v \to \{0,1\}^k$. For any $h \in \mathcal{H}$ we define the $(k, n = k + v)$-coding scheme $(\mathsf{LREnc}_h, \mathsf{LRDec}_h)$ where: (i) $\mathsf{LREnc}_h(x) := (r, h(r) \oplus x)$ for $r \leftarrow \{0,1\}^v$; (ii) $\mathsf{LRDec}_h((r, z)) := z \oplus h(r)$.

Theorem 3.2. *Fix any function family* $\mathcal{F}$ *consisting of functions* $f : \{0,1\}^n \to \{0,1\}^\ell$*. With probability* $1 - \rho$ *over the choice of a random* $h \leftarrow \mathcal{H}$*, the coding scheme* $(\mathsf{LREnc}_h, \mathsf{LRDec}_h)$ *is* $(\mathcal{F}, \varepsilon)$*-leakage-resilient as long as:*

$$t \geq \log|\mathcal{F}| + \ell + k + \log(1/\rho) + 3 \quad \textit{and} \quad v \geq \ell + 2\log(1/\varepsilon) + \log(t) + 3.$$

Proof. Fix a function family $\mathcal{F}$. Now, taking probabilities (only) over the choice of h, let $\textsc{Bad}$ be the event that $(\mathsf{LREnc}, \mathsf{LRDec})$ is not an $(\mathcal{F}, \varepsilon)$-leakage-resilient code. Then,

$$\begin{aligned}
\Pr[\textsc{Bad}] &= \Pr_{h \leftarrow \mathcal{H}}[\exists f \in \mathcal{F}, x \in \{0,1\}^k \quad : \quad \mathbf{SD}(f(\mathsf{LREnc}_h(x)), f(U_n)) > \varepsilon] \\
&\leq \sum_{f \in \mathcal{F}} \sum_{x \in \{0,1\}^k} \Pr_{h \leftarrow \mathcal{H}} \left[\sum_{\alpha \in \{0,1\}^\ell} \left| \Pr_{r \leftarrow \{0,1\}^v}[f(r, h(r) \oplus x) = \alpha] - \Pr[f(U_n) = \alpha] \right| > 2\varepsilon \right] \quad (1)
\end{aligned}$$

where Eq. (1) follows by taking a union bound over all $f \in \mathcal{F}$ and $x \in \{0,1\}^k$.

Fix any $f \in \mathcal{F}, x \in \{0,1\}^k$. For any $\alpha \in \{0,1\}^\ell, r \in \{0,1\}^v$, define $p_\alpha := \Pr[f(U_n) = \alpha]$ and $p_{r,\alpha} := \Pr[f(r, U_k) = \alpha]$. Let $\widetilde{p}_\alpha := \max\{\ p_\alpha\ ,\ 2^{-\ell}\ \}$. Note that

$$\sum_{\alpha \in \{0,1\}^\ell} \widetilde{p}_\alpha \leq \sum_\alpha p_\alpha + \sum_\alpha 2^{-\ell} \leq 2. \quad (2)$$

Define the random variable $Y_{r,\alpha}$ such that $Y_{r,\alpha} = 1$ if $f(r, h(r) \oplus x) = \alpha$, where the randomness is over the choice of $h \leftarrow \mathcal{H}$. Then $\Pr[Y_{r,\alpha} = 1] = p_{r,\alpha}$ and, for a fixed α, the random variables $\{Y_{r,\alpha}\}_{r \in \{0,1\}^v}$ are t-wise independent. Moreover, $\mathbf{E}[\sum_{r \in \{0,1\}^v} Y_{r,\alpha}] = 2^v p_\alpha$. Therefore, we have:

$$\begin{aligned}
& \Pr_{h \leftarrow \mathcal{H}} \left[\sum_{\alpha \in \{0,1\}^\ell} \left| \Pr_{r \leftarrow \{0,1\}^v}[f(r, h(r) \oplus x) = \alpha] - \Pr[f(U_n) = \alpha] \right| > 2\varepsilon \right] \\
\leq\ & \Pr_{h \leftarrow \mathcal{H}} \left[\exists \alpha \in \{0,1\}^\ell \quad : \quad \left| \Pr_{r \leftarrow \{0,1\}^v}[f(r, h(r) \oplus x) = \alpha] - p_\alpha \right| > \varepsilon \cdot \widetilde{p}_\alpha \right] \quad (3) \\
\leq\ & \sum_{\alpha \in \{0,1\}^\ell} \Pr_{h \leftarrow \mathcal{H}} \left[\left| \sum_{r \in \{0,1\}^v} Y_{r,\alpha} - 2^v p_\alpha \right| > 2^v \varepsilon \cdot \widetilde{p}_\alpha \right] \\
\leq\ & \sum_{\alpha \in \{0,1\}^\ell} K_t \left(\frac{t 2^v p_\alpha + t^2}{(2^v \varepsilon \widetilde{p}_\alpha)^2} \right)^{t/2} \quad (4) \\
\leq\ & \sum_{\alpha \in \{0,1\}^\ell} K_t \left(\frac{2t 2^v \cdot \widetilde{p}_\alpha}{(2^v \varepsilon \widetilde{p}_\alpha)^2} \right)^{t/2} \leq 2^\ell K_t \left(\frac{2t}{2^{v-\ell} \varepsilon^2} \right)^{t/2}, \quad (5)
\end{aligned}$$

where (3) follows from (2), and (4) follows from Lemma 2.1, with $K_t \leq 8$. Finally, (5) follow from the fact that the theorem's parameters ensure: $2^v \cdot \widetilde{p}_\alpha \geq 2^{v-\ell} \geq t$.

Combining the above with (1), we get: $\Pr[\textsc{Bad}] \leq |\mathcal{F}| \cdot 2^{\ell+k} \cdot 8 \left(\frac{2t}{2^{v-\ell}\varepsilon^2}\right)^{t/2}$. Therefore to satisfy $\Pr[\textsc{Bad}] \leq \rho$ we can set:

$$v \geq \ell + 2\log(1/\varepsilon) + \log(t) + 3 \quad \text{and} \quad t \geq \log|\mathcal{F}| + \ell + k + \log(1/\rho) + 3.$$

□

4 Non-Malleable Codes

We now construct a non-malleable code for any family $\mathcal{F}$ of sufficiently small size. We will rely on leakage resilience as an integral part of the analysis.

Construction. Let $\mathcal{H}_1$ be a family of hash functions $h_1 : \{0,1\}^{v_1} \to \{0,1\}^k$, and $\mathcal{H}_2$ be a family of hash functions $h_2 : \{0,1\}^{k+v_1} \to \{0,1\}^{v_2}$ such that $\mathcal{H}_1$ and $\mathcal{H}_2$ are both t-wise independent. For any $(h_1, h_2) \in \mathcal{H}_1 \times \mathcal{H}_2$, define $\mathsf{Enc}_{h_1,h_2}(x) = (r, z, \sigma)$ where $r \leftarrow \{0,1\}^{v_1}$ is random, $z := x \oplus h_1(r)$ and $\sigma := h_2(r, z)$. The codewords are of size $n := |(r, z, \sigma)| = k + v_1 + v_2$. Correspondingly define $\mathsf{Dec}((r, z, \sigma))$ which first checks $\sigma \overset{?}{=} h_2(r, z)$ and if this fails, outputs $\perp$, else outputs $z \oplus h_1(r)$. Notice that, we can think of (r, z) as being a leakage-resilient encoding of x; i.e., $(r, z) = \mathsf{LREnc}_{h_1}(x; r)$.

Theorem 4.1. *For any function family $\mathcal{F}$, the above construction $(\mathsf{Enc}_{h_1,h_2}, \mathsf{Dec}_{h_1,h_2})$ is an $(\mathcal{F}, \varepsilon)$-super non-malleable code with probability $1 - \rho$ over the choice of h_1, h_2 as long as:*

$$\begin{aligned} t &\geq t^* \quad \textit{for some} \quad t^* = O(\log|\mathcal{F}| + n + \log(1/\rho)) \\ v_1 &> v_1^* \quad \textit{for some} \quad v_1^* = 3\log(1/\varepsilon) + 3\log(t^*) + O(1) \\ v_2 &> v_1 + 3. \end{aligned}$$

For example, in the above theorem, if we set $\rho = \varepsilon = 2^{-\lambda}$ for "security parameter" λ, and $|\mathcal{F}| = 2^{s(n)}$ for some polynomial $s(n) = n^{O(1)} \geq n \geq \lambda$, then we can set $t = O(s(n))$ and the message length $k := n - (v_1 + v_2) = n - O(\lambda + \log n)$. Therefore the rate of the code k/n is $1 - O(\lambda + \log n)/n$ which approaches 1 as n grows relative to λ.

4.1 Proof of Theorem 4.1

Useful Notions. For a coding scheme $(\mathsf{Enc}, \mathsf{Dec})$, we say that $c \in \{0,1\}^n$ is *valid* if $\mathsf{Dec}(c) \neq \perp$. For any function $f : \{0,1\}^n \to \{0,1\}^n$, we say that $c' \in \{0,1\}^n$ is δ-heavy for f if $\Pr[f(\mathsf{Enc}(U_k)) = c'] \geq \delta$. Define

$$H_f(\delta) = \{c' \in \{0,1\}^n : c' \text{ is } \delta\text{-heavy for } f\}.$$

Notice that $|H_f(\delta)| \leq 1/\delta$.

Definition 4.2 (Bounded-malleable)**.** *We say that a coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is* $(\mathcal{F}, \delta, \tau)$-bounded-malleable *if for all $f \in \mathcal{F}, x \in \{0,1\}^k$ we have*

$$\Pr[c' \neq c \wedge c' \textit{ is valid } \wedge c' \notin H_f(\delta) \mid c \leftarrow \mathsf{Enc}(x), c' = f(c)] \leq \tau,$$

where the probability is over the randomness of the encoding.

Intuition. The above definition says the following. Take any message $x \in \{0,1\}^k$, tampering function $f \in \mathcal{F}$ and do the following: choose $c \leftarrow \mathsf{Enc}(x)$, set $c' = f(c)$, and output: (i) $\mathsf{same}^\star$ if $c' = c$, (ii) $\perp$ if c' is not valid, (iii) c' otherwise. Then, with probability $1 - \tau$ the output of the above experiment takes on one of the values: $\{\mathsf{same}^\star, \perp\} \cup H_f(\delta)$. Therefore, the output of the above tampering experiment only leaks a bounded amount of information about c; in particular it leaks at most $\ell = \lceil \log(1/\delta + 2) \rceil$ bits. Furthermore the "leakage" on c is independent of the choice of the code, up to knowing which codewords are valid and which are δ-heavy. In particular, in our construction, the "leakage" only depends on the choice of h_2 but *not* on the choice of h_1. This will allow us to then rely on the fact that $\mathsf{LREnc}_{h_1}(x; r) = (r, h_1(r) \oplus x)$ is a leakage-resilient encoding of x to argue that the output of the above experiment is the same for x as for a uniformly random value. We formalize this intuition below.

From Bounded-Malleable to Non-Malleable. For any "tampering function" family $\mathcal{F}$ consisting of functions $f : \{0,1\}^n \to \{0,1\}^n$, any $\delta > 0$, and any $h_2 \in \mathcal{H}_2$ we define the "leakage function" family $\mathcal{G} = \mathcal{G}(\mathcal{F}, h_2, \delta)$ which consists of the functions $g_f : \{0,1\}^{k+v_1} \to H_f(\delta) \cup \{\mathsf{same}^\star, \perp\}$ for each $f \in \mathcal{F}$. The functions are defined as follows:

- $g_f(c_1)$: Compute $\sigma = h_2(c_1)$. Let $c := (c_1, \sigma), c' = f(c)$. If c' is not valid output $\perp$. Else if $c' = c$ output $\mathsf{same}^\star$. Else if $c' \in H_f(\delta)$ output c'. Lastly, if none of the above cases holds, output $\perp$.

Notice that the notion of "δ-heavy" and the set $H_f(\delta)$ are completely specified by h_2 and do not depend on h_1. This is because the distribution $\mathsf{Enc}_{h_1,h_2}(U_k)$ is equivalent to $(U_{k+v_1}, h_2(U_{k+v_1}))$ and therefore c' is δ-heavy if and only if $\Pr[f(U_{k+v_1}, h_2(U_{k+v_1})) = c'] \geq \delta$. Therefore the family $\mathcal{G} = \mathcal{G}(\mathcal{F}, h_2, \delta)$ is fully specified by $\mathcal{F}, h_2, \delta$. Also notice that $|\mathcal{G}| = |\mathcal{F}|$ and that the output length of the functions g_f is given by $\ell = \lceil \log(|H_f(\delta)| + 2) \rceil \leq \lceil \log(1/\delta + 2) \rceil$.

Lemma 4.3. *Let $\mathcal{F}$ be any function family and let $\delta > 0$. Fix any h_1, h_2 such that $(\mathsf{Enc}_{h_1,h_2}, \mathsf{Dec}_{h_1,h_2})$ is $(\mathcal{F}, \delta, \varepsilon/4)$-bounded-malleable and $(\mathsf{LREnc}_{h_1}, \mathsf{LRDec}_{h_1})$ is $(\mathcal{G}(\mathcal{F}, h_2, \delta), \varepsilon/4)$-leakage-resilient, where the family $\mathcal{G} = \mathcal{G}(\mathcal{F}, h_2, \delta)$, with size $|\mathcal{G}| = |\mathcal{F}|$, is defined above, and the leakage amount is $\ell = \lceil \log(1/\delta + 2) \rceil$. Then $(\mathsf{Enc}_{h_1,h_2}, \mathsf{Dec}_{h_1,h_2})$ is $(\mathcal{F}, \varepsilon)$-non-malleable.*

Proof. For any $x_0, x_1 \in \{0,1\}^k$ and any $f \in \mathcal{F}$:

$$\mathsf{Tamper}^f_{x_0} \quad = \quad \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}_{h_1,h_2}(x_0), c' := f(c) \\ \text{Output: } \mathsf{same}^\star \text{ if } c' = c, \perp \text{ if } \mathsf{Dec}_{h_1,h_2}(c') = \perp, \\ c' \text{ otherwise.} \end{array} \right\}$$

$$\overset{\text{stat}}{\approx}_{\varepsilon/4} \left\{ \begin{array}{c} c_1 \leftarrow \mathsf{LREnc}_{h_1}(x_0) \\ \text{Output: } g_f(c_1) \end{array} \right\} \tag{6}$$

$$\overset{\text{stat}}{\approx}_{\varepsilon/4} \left\{ \begin{array}{c} c_1 \leftarrow \mathsf{LREnc}_{h_1}(U_k) \\ \text{Output: } g_f(c_1) \end{array} \right\} \tag{7}$$

$$\overset{\text{stat}}{\approx}_{\varepsilon/4} \left\{ \begin{array}{c} c_1 \leftarrow \mathsf{LREnc}_{h_1}(x_1) \\ \text{Output: } g_f(c_1) \end{array} \right\} \tag{8}$$

$$\overset{\text{stat}}{\approx}_{\varepsilon/4} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}_{h_1,h_2}(x_1), c' := f(c) \\ \text{Output: } \mathsf{same}^\star \text{ if } c' = c, \perp \text{ if } \mathsf{Dec}_{h_1,h_2}(c') = \perp, \\ c' \text{ otherwise.} \end{array} \right\} \tag{9}$$

$$= \mathsf{Tamper}^f_{x_1}.$$

Eq. (6) and Eq. (9) follow as $(\mathsf{Enc}_{h_1,h_2}, \mathsf{Dec}_{h_1,h_2})$ is an $(\mathcal{F}, \delta, \varepsilon/4)$-bounded-malleable code, and Eq. (7) and Eq. (8) follow as the code $(\mathsf{LREnc}_{h_1}, \mathsf{LRDec}h_1)$ is $(\mathcal{G}(\mathcal{F}, \delta), \varepsilon/4)$-leakage-resilient. □

We can use Theorem 3.2 to show that $(\mathsf{LREnc}_{h_1}, \mathsf{LRDec}_{h_1})$ is $(\mathcal{G}(\mathcal{F}, h_2, \delta), \varepsilon/4)$-leakage-resilient with overwhelming probability. Therefore, it remains to show that our construction is $(\mathcal{F}, \delta, \tau)$-bounded-malleable, which we do below.

Analysis of Bounded-Malleable Codes. We now show that the code $(\mathsf{Enc}_{h_1,h_2}, \mathsf{Dec}_{h_1,h_2})$ is bounded-malleable with overwhelming probability. As a very high-level intuition, if a tampering function f can often map valid codewords to other valid codewords (and many different ones), then it must guess the output of h_2 on many different inputs. If the family $\mathcal{F}$ is small enough, it is highly improbable that it would contain some such f. For more detailed intuition, we show that the following two properties hold for any message x and any function f with overwhelming probability: (i) there is at most some "small" set of q valid codewords c' that we can hit by tampering some encoding of x via f; (ii) for each such codeword c' which is not in δ-heavy, the probability of landing in c' after tampering an encoding of x cannot be higher than 2δ. This shows that the total probability of tampering an encoding of x and landing in a valid codeword which not δ-heavy is at most $2q\delta$, which is small. Property (i) roughly follows by showing that f would need to "predict" the output of h_2 on q different inputs, and property (ii) follows by using "leakage resilience" of h_1 to argue that we cannot distinguish an encoding of x from an encoding of a random message, for which the probability of landing in c' is at most δ.

Lemma 4.4. *For any function family $\mathcal{F}$, any $\delta > 0$, the code $(\mathsf{Enc}_{h_1,h_2}, \mathsf{Dec}_{h_1,h_2})$ is $(\mathcal{F}, \delta, \tau)$-bounded-malleable with probability $1 - \psi$ over the choice of h_1, h_2 as long as:*

$$\begin{aligned} \tau &\geq 2(\log|\mathcal{F}| + k + \log(1/\psi) + 2)\delta \\ t &\geq \log|\mathcal{F}| + n + k + \log(1/\psi) + 5 \\ v_1 &\geq 2\log(1/\delta) + \log(t) + 4 \quad \textit{and} \quad v_2 \geq v_1 + 3. \end{aligned}$$

Proof. Set $q := \lceil \log|\mathcal{F}| + k + \log(1/\psi) + 1 \rceil$. For any $f \in \mathcal{F}, x \in \{0,1\}^k$ define the events $E_1^{f,x}$ and $E_2^{f,x}$ over the random choice of h_1, h_2 as follows:

1. $E_1^{f,x}$ occurs if there exist at least q distinct values $c'_1, \ldots, c'_q \in \{0,1\}^n$ such that each c'_i is valid and $c'_i = f(c_i)$ for some $c_i \neq c'_i$ which encodes the message x (i.e., $c_i = \mathsf{Enc}_{h_1,h_2}(x; r_i)$ for some r_i).

2. $E_2^{f,x}$ occurs if there exists some $c' \in \{0,1\}^n \setminus H_f(\delta)$ such that $\Pr_{r \leftarrow \{0,1\}^{v_1}}[f(\mathsf{Enc}_{h_1,h_2}(x; r)) = c'] \geq 2\delta$.

Let $E_1 = \bigvee_{f,x} E_1^{f,x}$, $E_2 = \bigvee_{f,x} E_2^{f,x}$ and $\textsc{Bad} = E_1 \vee E_2$. Assume (h_1, h_2) are any hash functions for which the event $\textsc{Bad}$ does *not* occur. Then, for every $f \in \mathcal{F}, x \in \{0,1\}^k$:

$$\begin{aligned} &\Pr[f(C) \neq C \wedge f(C) \text{ is valid } \wedge f(C) \notin H_f(\delta)] \\ = &\sum_{c' \colon c' \text{valid and } c' \notin H_f(\delta)} \Pr[f(C) = c' \wedge C \neq c'] < 2q\delta \leq \tau, \end{aligned} \tag{10}$$

where $C = \mathsf{Enc}_{h_1,h_2}(x; U_{v_1})$ is a random variable. Eq. (10) holds since (i) given that E_1 does not occur, there are fewer than q values c' that are valid and for which $\Pr[f(C) = c' \wedge C \neq c'] > 0$, and (ii) given that E_2 does not occur, for any $c' \notin H_f(\delta)$, we also have $\Pr[f(C) = c' \wedge C \neq c'] \leq \Pr[f(C) = c'] < 2\delta$.

Therefore, if the event $\textsc{Bad}$ does not occur, then the code is $(\mathcal{F}, \delta, \tau)$-bounded-malleable. This means:

$$\Pr_{h_1,h_2}[(\mathsf{Enc}_{h_1,h_2}, \mathsf{Dec}_{h_1,h_2}) \quad \text{is not} \quad (\mathcal{F}, \delta, \tau)\text{-bounded-malleable}] \leq \Pr[\textsc{Bad}] \leq \Pr[E_1] + \Pr[E_2].$$

So it suffices to show that $\Pr[E_1]$ and $\Pr[E_2]$ are both bounded by $\psi/2$, which we do next.

Claim 4.5. $\Pr[E_1] \leq \psi/2$.

Proof. Fix some message $x \in \{0,1\}^k$ and some function $f \in \mathcal{F}$. Assume that the event $E_1^{f,x}$ occurs for some choice of hash functions (h_1, h_2). Then there must exist some values $\{r_1, \ldots, r_q\}$ such that: if we define $c_i := \mathsf{Enc}(x; r_i), c'_i := f(c_i)$ then $c'_i \neq c_i$, c'_i is valid, and $|\{c'_1, \ldots, c'_q\}| = q$. The last condition also implies $|\{c_1,\ldots,c_q\}| = q$. However, it is possible that $c_i = c'_j$ for some $i \neq j$. We claim that we can find a subset of at least $s := \lceil q/3 \rceil$ of the indices such that the $2s$ values $\{c_{a_1},\ldots,c_{a_s},c'_{a_1},\ldots,c'_{a_s}\}$ are all distinct. To do so, notice that if we want to keep some index i corresponding to values c_i, c'_i, we need to take out at most two indices j, k in case $c'_j = c_i$ or $c_k = c'_i$.[1] To summarize, if $E_1^{f,x}$ occurs, then (by re-indexing) there is some set $R = \{r_1,\ldots,r_s\} \subseteq \{0,1\}^{v_1}$ of size $|R| = s$ satisfying the following two conditions:

(i) If we define $c_i := \mathsf{Enc}(x; r_i)$, $c'_i \neq c_i$ and c'_i is valid meaning that $c'_i = (r'_i, z'_i, \sigma'_i)$ where $\sigma' = h_2(r', z')$.

(ii) $|\{c_1,\ldots,c_s,c'_1,\ldots,c'_s\}| = 2s$.

Therefore we have:

$$\begin{aligned}
\Pr[E_1^{f,x}] &\leq \Pr_{h_1,h_2}[\exists R \subseteq \{0,1\}^{v_1}, |R| = s, R \text{ satisfies } (i) \text{ and } (ii)] \leq \sum_R \Pr_{h_1,h_2}[R \text{ satisfies } (i) \text{ and } (ii)] \\
&\leq \sum_{R=\{r_1,\ldots,r_s\}} \max_{h_1,\sigma_1,\ldots,\sigma_s} \Pr_{h_2}\left[\forall i\ ,\ c'_i \text{ valid} \;\middle|\; \begin{array}{c} c_i := (r_i, z_i = h_1(r_i) \oplus x, \sigma_i), c'_i := f(c_i), c'_i \neq c_i \\ |\{c_1,\ldots,c_s,c'_1,\ldots,c'_s\}| = 2s \end{array}\right] \\
&\leq \binom{2^{v_1}}{s} 2^{-sv_2} \leq \left(\frac{e2^{v_1}}{s}\right)^s 2^{-sv_2} \leq 2^{s(v_1-v_2)} \leq 2^{q(v_1-v_2)/3} \leq 2^{-q}, \qquad (11)
\end{aligned}$$

where Eq. (11) follows from the fact that, even if we condition on any choice of the hash function h_1 which fixes $z_i = h_1(r_i) \oplus x$, and any choice of the s values $\sigma_i = h_2(r_i, z_i)$, which fixes $c_i := (r_i, z_i = h_1(r_i) \oplus x, \sigma_i), c'_i := f(c_i)$ such that $c'_i \neq c_i$ and $|\{c_1,\ldots,c_s,c'_1,\ldots,c'_s\}| = 2s$, then the probability that $h_2(r'_i, z'_i) = \sigma'_i$ for all $i \in [s]$ is at most 2^{-sv_2}. Here we use the fact that $\mathcal{H}_2$ is t-wise independent where $t \geq q \geq 2s$. Now, we calculate

$$\Pr[E_1] \leq \sum_{f\in\mathcal{F}} \sum_{x\in\{0,1\}^k} \Pr[E_1^{f,x}] \leq |\mathcal{F}| 2^{k-q} \leq \psi/2,$$

where the last inequality follows from the assumption, $q = \lceil \log|\mathcal{F}| + k + \log(1/\psi) + 1 \rceil$. □

Claim 4.6. $\Pr[E_2] \leq \psi/2$.

Proof. For this proof, we will rely on the leakage resilience property of the code $(\mathsf{LREnc}_{h_1}, \mathsf{LRDec}_{h_1})$ as shown in Theorem 3.2. First, let us write:

$$\begin{aligned}
\Pr[E_2] &= \Pr_{h_1,h_2}\left[\exists (f,x,c') \in \mathcal{F} \times \{0,1\}^k \times \{0,1\}^n \setminus H_f(\delta) \ :\ \Pr[f(\mathsf{Enc}_{h_1,h_2}(x;U_{v_1})) = c'] \geq 2\delta\right] \\
&\leq \Pr_{h_1,h_2}\Big[\exists (f,x,c') \in \mathcal{F} \times \{0,1\}^k \times \{0,1\}^n \setminus H_f(\delta) \ : \qquad (12) \\
&\qquad \left|\Pr[f(\mathsf{Enc}_{h_1,h_2}(x;U_{v_1})) = c'] - \Pr[f(\mathsf{Enc}_{h_1,h_2}(U_k;U_{v_1})) = c']\right| \geq \delta\Big]
\end{aligned}$$

since, for any $c' \notin H_f(\delta)$, we have $\Pr[f(\mathsf{Enc}_{h_1,h_2}(U_k;U_{v_1})) = c'] < \delta$ by definition. Notice that we can write $\mathsf{Enc}_{h_1,h_2}(x;r) = (c_1,c_2)$ where $c_1 = \mathsf{LREnc}_{h_1}(x;r)$, $c_2 = h_2(c_1)$. We will now rely on the leakage resilience of the code $(\mathsf{LREnc}_{h_1}, \mathsf{LRDec}_{h_1})$ to bound the above probability by $\psi/2$. In fact,

[1] In other words, if we take any set of tuples $\{(c_i, c'_i)\}$ such that all the left components are distinct $c_i \neq c_j$ and all the right components are distinct $c'_i \neq c'_j$, but there may be common values $c_i = c'_j$, then there is a subset of at least 1/3 of the tuples such that all left and right components in this subset are mutually distinct.

we show that the above holds even if we take the probability over h_1 only, for a worst-case choice of h_2.

Let us fix some choice of h_2 and define the family $\mathcal{G} = \mathcal{G}(h_2)$ of leakage functions $\mathcal{G} = \{g_{f,c'} : \{0,1\}^{k+v_1} \to \{0,1\} \mid f \in \mathcal{F}, c' \in \{0,1\}^n\}$ with output size $\ell = 1$ bits as follows:

- $g_{f,c'}(c_1)$: Set $c = (c_1, c_2 = h_2(c_1))$. If $f(c) = c'$ output 1, else output 0.

Notice that the size of the family $\mathcal{G}$ is $2^n|\mathcal{F}|$ and the family does not depend on the choice of h_1. Therefore, continuing from inequality (12), we get:

$$\begin{aligned}
\Pr[E_2] &\leq \Pr_{h_1,h_2}\Big[\exists (f,x,c') \in \mathcal{F} \times \{0,1\}^k \times \{0,1\}^n \setminus H_f(\delta) \ : \\
&\qquad\qquad \big|\Pr[f(\mathsf{Enc}_{h_1,h_2}(x;U_{v_1})) = c'] - \Pr[f(\mathsf{Enc}_{h_1,h_2}(U_k;U_{v_1})) = c']\big| \geq \delta\Big] \\
&\leq \max_{h_2}\Pr_{h_1}\left[\exists (g_{f,c'},x) \in \mathcal{G}(h_2) \times \{0,1\}^k \ : \ \left|\begin{array}{c}\Pr[g_{f,c'}(\mathsf{LREnc}_{h_1}(x;U_{v_1})) = 1] \\ -\Pr[g_{f,c'}(\mathsf{LREnc}_{h_1}(U_k;U_{v_1})) = 1]\end{array}\right| \geq \delta\right] \\
&= \max_{h_2}\Pr_{h_1}\left[\exists (g_{f,c'},x) \in \mathcal{G}(h_2) \times \{0,1\}^k \ : \ \left|\begin{array}{c}\Pr[g_{f,c'}(\mathsf{LREnc}_{h_1}(x;U_{v_1})) = 1] \\ -\Pr[g_{f,c'}(U_{k+v_1}) = 1]\end{array}\right| \geq \delta\right] \\
&\leq \max_{h_2}\Pr_{h_1}\big[\ (\mathsf{LREnc}_{h_1},\mathsf{LRDec}_{h_1}) \text{ is not } (\mathcal{G}(h_2),\delta)\text{-Leakage-Resilient }\ \big] \leq \psi/2,
\end{aligned}$$

where the last inequality follows from Theorem 3.2 by the choice of parameters. □

□

Putting it All Together. Lemma 4.3 tells us that for any $\delta > 0$ and any function family $\mathcal{F}$:

$$\begin{aligned}
&\Pr[(\mathsf{Enc}_{h_1,h_2},\mathsf{Dec}_{h_1,h_2}) \text{ is not } (\mathcal{F},\varepsilon)\text{-super-non-malleable}] \\
&\leq \Pr[(\mathsf{Enc}_{h_1,h_2},\mathsf{Dec}_{h_1,h_2}) \text{ is not } (\mathcal{F},\delta,\varepsilon/4)\text{-bounded-malleable}] \qquad (13)\\
&\qquad + \Pr[(\mathsf{LREnc}_{h_1},\mathsf{LRDec}_{h_1}) \text{ is not } (\mathcal{G}(\mathcal{F},h_2,\delta),\varepsilon/4)\text{-leakage-resilient}], \qquad (14)
\end{aligned}$$

where $\mathcal{G} = \mathcal{G}(\mathcal{F},h_2,\delta)$ is of size $|\mathcal{G}| = |\mathcal{F}|$ and consists of function with output size $\ell = \lceil \log(1/\delta{+}2)\rceil$.

Let us set $\delta := (\varepsilon/8)(\log|\mathcal{F}| + k + \log(1/\rho) + 3)^{-1}$. This ensures that the first requirement of Lemma 4.4 is satisfied with $\tau = \varepsilon/4$. We choose $t^* = O(\log|\mathcal{F}| + n + \log(1/\rho))$ such that $\log(1/\delta) \leq \log(1/\varepsilon) + \log(t^*) + O(1)$. Notice that the leakage amount of $\mathcal{G}$ is $\ell = \lceil\log(1/\delta+2)\rceil \leq \log(1/\varepsilon) + \log(t^*) + O(1)$. With v_1, v_2 as in Theorem 4.1, we satisfy the remaining requirements of Lemma 4.4 (bounded-malleable codes) and Theorem 3.2 (leakage-resilient codes) to ensure that the probabilities (13), (14) are both bounded by $\rho/2$, which proves our theorem.

5 Non-Malleable Key-Derivation

In this section we introduce a new primitive, which we name non-malleable key derivation. Intuitively a function h is a non-malleable key derivation function if $h(x)$ is close to uniform even given the output of h applied to a related input $f(x)$, as long as $f(x) \neq x$.

Definition 5.1 (Non-Malleable Key-Derivation). *Let $\mathcal{F}$ be any family of functions $f : \{0,1\}^n \to \{0,1\}^n$. We say that a function $h : \{0,1\}^n \to \{0,1\}^k$ is an $(\mathcal{F},\varepsilon)$*-non-malleable key derivation function *if for every $f \in \mathcal{F}$ we have $\mathbf{SD}\big(\mathsf{Real}_h(f);\ \mathsf{Sim}_h(f)\big) \leq \varepsilon$ where $\mathsf{Real}_h(f)$ and $\mathsf{Sim}_h(f)$ denote the output distributions of the corresponding experiments described in Fig. 1.*

Experiment $\mathsf{Real}_h(f)$ **vs.** $\mathsf{Sim}_h(f)$

```
Experiment $\mathsf{Real}_h(f)$:
    Sample $x \leftarrow U_n$.
    If $f(x) = x$:
        Output $\big(h(x), \mathsf{same}^\star)\big)$.
    Else
        Output $\big(h(x), h(f(x))\big)$.
```

```
Experiment $\mathsf{Sim}_h(f)$:
    Sample $x \leftarrow U_n$; $y \leftarrow U_k$
    If $f(x) = x$:
        Output $\big(y, \mathsf{same}^\star\big)$.
    Else
        Output $\big(y, h(f(x))\big)$.
```

Figure 1: Experiments defining a non-malleable key derivation function h

Note that the above definition can be interpreted as a dual version of the definition of non-malleable extractors [17].[2] The theorem below states that by sampling a function h from a set $\mathcal{H}$ of t-wise independent hash functions, we obtain a non-malleable key derivation function with overwhelming probability.

Theorem 5.2. *Let $\mathcal{H}$ be a $2t$-wise independent function family consisting of functions $h : \{0,1\}^n \to \{0,1\}^k$ and let $\mathcal{F}$ be some function family as above. Then with probability $1-\rho$ over the choice of a random $h \leftarrow \mathcal{H}$, the function h is an $(\mathcal{F}, \varepsilon)$-non-malleable key-derivation function as long as:*

$$n \geq 2k + \log(1/\varepsilon) + \log(t) + 3 \quad \textit{and} \quad t > \log(|\mathcal{F}|) + 2k + \log(1/\rho) + 5.$$

Proof. For any $h \in \mathcal{H}$ and $f \in \mathcal{F}$, define a function $h_f : \{0,1\}^n \to \{0,1\}^k \cup \mathsf{same}^\star$ such that if $f(x) = x$ then $h_f(x) = \mathsf{same}^\star$ otherwise $h_f(x) = h(f(x))$. Fix a function family $\mathcal{F}$. Now, taking probabilities (only) over the choice of h, let $\textsc{Bad}$ be the event that h is not an $(\mathcal{F}, \varepsilon)$-non-malleable-key-derivation function. Then:

$$\begin{aligned}
\Pr[\textsc{Bad}] &= \Pr_{h \leftarrow \mathcal{H}}\Big[\exists f \in \mathcal{F} \;:\; \mathbf{SD}(\ \mathsf{Real}_h(f)\ ,\ \mathsf{Sim}_h(f)\ ) > \varepsilon\Big] \\
&= \Pr_{h \leftarrow \mathcal{H}}\Big[\exists f \in \mathcal{F} \;:\; \mathbf{SD}(\ (h(X), h_f(X))\ ,\ (U_k, h_f(X))\ ) > \varepsilon\Big] \\
&\leq \sum_{f \in \mathcal{F}} \Pr_{h \leftarrow \mathcal{H}}\left[\sum_{y \in \{0,1\}^k} \sum_{y' \in \{0,1\}^k \cup \mathsf{same}^\star} \left| \begin{array}{c} \Pr[h(X) = y \wedge h_f(X) = y'] \\ -\Pr[U_k = y \wedge h_f(X) = y'] \end{array} \right| > 2\varepsilon \right] \\
&\leq \sum_{f \in \mathcal{F}} \Pr_{h \leftarrow \mathcal{H}}\left[\exists\ y \in \{0,1\}^k, y' \in \{0,1\}^k \cup \mathsf{same}^\star \;:\; \left| \begin{array}{c} \Pr[h(X) = y \wedge h_f(X) = y'] \\ -\Pr[U_k = y \wedge h_f(X) = y'] \end{array} \right| > 2^{-2k}\varepsilon \right] \\
&\leq \sum_{f \in \mathcal{F}} \sum_{y \in \{0,1\}^k} \sum_{y' \in \{0,1\}^k \cup \mathsf{same}^\star} \Pr_{h \leftarrow \mathcal{H}}\left[\left| \begin{array}{c} \Pr[h(X) = y \wedge h_f(X) = y'] \\ -2^{-k}\Pr[h_f(X) = y'] \end{array} \right| > 2^{-2k}\varepsilon \right] \qquad (15)
\end{aligned}$$

Fix f, y, y'. For every $x \in \{0,1\}^n$, define a random variable C_x over the choice of $h \leftarrow \mathcal{H}$, such that

$$C_x = \begin{cases} 1 - 2^{-k} & \text{if } h(x) = y \wedge h_f(x) = y' \\ -2^{-k} & \text{if } h(x) \neq y \wedge h_f(x) = y' \\ 0 & \text{otherwise.} \end{cases}$$

[2]The duality comes from the fact that the output of a non-malleable extractor is close to uniform even given a certain number of outputs computed with related seeds (whereas for non-malleable key derivation the seed is unchanged but the input can be altered).

Notice that each C_x is 0 on expectation. However, the random variables C_x are not even pairwise independent.[3] In Section 5.1, we prove the following lemma about the variables C_x.

Lemma 5.3. *There exists a partitioning of $\{0,1\}^n$ into four disjoint subsets $\{A_j\}_{j=1}^4$, such that for any $A > 0$ and for all $j = 1, \ldots, 4$:*

$$\Pr\left[\left|\sum_{x \in A_j} C_x\right| > A\right] < K_t \left(\frac{t}{A}\right)^t,$$

where $K_t \leq 8$.

Continuing from Eq. (15), we get:

$$\begin{aligned}
&\Pr_{h \leftarrow \mathcal{H}}\left[\left|\Pr[h(X) = y \wedge h_f(X) = y'] - 2^{-k}\Pr[h_f(X) = y']\right| > 2^{-2k}\varepsilon\right] \\
&= \Pr_{h \leftarrow \mathcal{H}}\left[\left|\sum_{x \in \{0,1\}^n} C_x\right| > 2^{n-2k}\varepsilon\right] \qquad (16) \\
&\leq \sum_{j=1}^{4} \Pr_{h \leftarrow \mathcal{H}}\left[\left|\sum_{x \in A_j} C_x\right| > 2^{n-2k-2}\varepsilon\right] < 4K_t \left(\frac{t}{2^{n-2k-2}\varepsilon}\right)^t. \qquad (17)
\end{aligned}$$

Eq. (16) follows from the definitions of the variables C_x and Eq. (17) follows by applying Lemma 5.3 to the sum. Combining Eq. (15) and Eq. (17), we get $\Pr[\textsc{Bad}] < |\mathcal{F}| 2^{2k} \left[4K_t \left(\frac{t}{2^{n-2k-2}\varepsilon}\right)^t\right]$. In particular, it holds that $\Pr[\textsc{Bad}] \leq \rho$ as long as:

$$n \geq 2k + \log(1/\varepsilon) + \log(t) + 3 \quad \text{and} \quad t > \log(|\mathcal{F}|) + 2k + \log(1/\rho) + 5.$$

□

5.1 Proof of Lemma 5.3

It will be convenient to represent the tampering function f as a graph. In particular, we define a directed graph $G = (V, E)$ with vertices $V = \{0,1\}^n$ and edges $E = \{(x, x') : x' = f(x)\}$. Each vertex has out-degree 1, and the graph may contain self-loops. We choose a random $h \leftarrow \mathcal{H}$, and label each vertex x with a value $h(x)$. Each random variable C_x depends only on the labels of the vertices x and $f(x)$. As we mentioned, the random variables C_x are not independent. However, we show how to partition the variables into 4 sets such that, within each set, the variables are either independent or "as good as independent".

First we partition the values $x \in \{0,1\}^n$ into two sets S_f (self-loop) and $\overline{S}_f$ (no self-loop) such that $x \in S_f$ if and only if $f(x) = x$. We prove the following lemma.

Lemma 5.4. *Let $t > 2$ be an even integer. Consider the set of random variables $\{C_x\}_{x \in S_f}$. Then for any $A > 0$,*

$$\Pr\left[\left|\sum_{x \in S_f} C_x\right| > A\right] < K_t \left(\frac{t}{A}\right)^t.$$

where $K_t \leq 8$.

[3] For example if $f(x) = f(x')$ and $C_x = 0$ then $C_{x'} = 0$ as well.

Proof. First consider the case when $y' \neq \mathsf{same}^\star$. In this case, for all $x \in S_f$, we have that $C_x = 0$ for any choice of h and thus the statement is verified.

On the other hand when $y' = \mathsf{same}^\star$ the variables $\{C_x\}_{x \in S_f}$ are t-wise independent. Hence, the statement follows directly from Lemma 2.1. $\square$

Now, consider the subgraph $G(\overline{S}_f)$ induced by $\overline{S}_f \subseteq V$. In $\overline{S}_f$, there is no self-loop and each vertex has at most one outgoing edge.[4] We prove the following lemma about $G(\overline{S}_f)$.

Lemma 5.5. *The graph $G(\overline{S}_f)$ is 3-colorable.*

Proof. We prove the lemma by constructing an algorithm to color $G(\overline{S}_f)$ with 3 colors (say R, B and W). We recall that, by definition, in graph $G(\overline{S}_f)$ each vertex has at most one outgoing edge. Without loss of generality we assume that each vertex in the graph has exactly one outgoing edge. It is easy to see that this is the worst-case for graph coloring, as more edges might enforce to use more colors. This fact we exploit heavily in this proof. The algorithm is described as follows:

1. Pick up a vertex randomly and color it by some arbitrary color, say R.

2. Follow the unique outgoing edge to color the next vertex by a different color, say B.

3. Continue to color vertices alternately following unique outgoing edges successively, with the following check at each step. To color a vertex check if its successor vertex is uncolored. If it is, then color the vertex with R or B, whichever appropriate. Otherwise, there may be a situation such that both the predecessor and the successor of a vertex are colored with different colors, which enforces to color that vertex with the third color, namely W. After that, pick up another uncolored vertex randomly and repeat from the beginning of step 3. If no such vertex is left then stop.

Note that the algorithm always terminates; this is because whenever we encounter a loop we choose a new vertex from the set of uncolored vertices, and there are only finitely many vertices.

To prove the correctness, we observe the following invariant is maintained throughout. According to the algorithm, once we are done with coloring some vertex v we move to color its unique uncolored successor v'. Now we claim that all the other predecessors v'' of v' which are different from v must be uncolored. To see this, we assume by contradiction that there is some v'' different from v which is colored. Now, since v' is the unique successor of v'', following the algorithm we must have colored v' immediately after we color v''. Therefore, assuming v'' is colored leads to the fact that v' is already colored which is a contradiction. So, the only neighbor of v' which might be colored is its unique successor $\hat{v}$. Note that this may enforce us to color v' differently from both its predecessor v and its unique successor $\hat{v}$ with the third color, which we are allowed to do. We conclude that the algorithm imposes a proper 3-coloring on $G(\overline{S}_f)$. $\square$

By the above lemma we can partition the set $\overline{S}_f \subseteq \{0,1\}^n$ into three disjoint subsets A_1, A_2, A_3 (i.e., the three colors) such that for $j \in \{1,2,3\}$, and all $x, x' \in A_j$, $f(x) \neq x'$. We consider the set of variables $\{C_x\}_{x \in A_j}$. Intuitively, the above partitioning removed some "bad" dependence between variables C_x and $C_{f(x)}$ by seeparating them into different subsets. However, even within any set A_j, the variables are not t-wise independent. For example if $f(x) = f(x')$ then C_x and $C_{x'}$ may be in the same set A_j but are correlated. Nevertheless, we prove that the correlation within each set goes in the "right" direction and allows us to bound the sum. In particular, we prove the following

[4]Note that it might happen that some outgoing edge lands in S_f. In that case the induced subgraph $G(\overline{S}_f)$ would exclude that edge.

lemma about the set of variables $\{C_x\}_{x\in A_j}$; note that Lemma 5.4 and Lemma 5.6 imply Lemma 5.3 by letting $S_f = A_4$.

Lemma 5.6. *Let $t > 2$ be an even integer. Consider the set of random variables $\{C_x\}_{x\in A_j}$ for some $j \in \{1,2,3\}$. Denote their sum by $\Sigma_j = \sum_{x\in A_j} C_x$. Then for any $A > 0$ and for all $j \in \{1,2,3\}$,*

$$\Pr[\Sigma_j > A] < K_t \left(\frac{t}{A}\right)^t ,$$

where $K_t \le 8$.

Proof. Fix some $j \in \{1,2,3\}$. First consider the case when $y' = \mathsf{same}^\star$. In this case, for all $x \in A_j$, we have that $C_x = 0$ for any choice of h. Thus, $\Pr[\Sigma_j > A] = 0$ and the statement of the lemma is verified. For the remaining of this proof we will assume that $y' \neq \mathsf{same}^\star$.

Let $|A_j| = m$ for some $m \in [2^n]$, and denote the variables $\{C_x\}_{x\in A_j}$ by $\{C_{x_1},\ldots,C_{x_m}\}$. For each variable C_{x_i} ($i \in [m]$) we can define the corresponding conditional random variable $C_{x_i}|(h(f(x_i)) = y')$ as follows:

$$\widetilde{C}_i := C_{x_i}|(h(f(x_i)) = y') = \begin{cases} 1-p & \text{with probability } \Pr_h[h(x_i) = y] = p \\ -p & \text{with probability } \Pr_h[h(x_i) \neq y] = 1-p \end{cases}$$

where $p = 2^{-k}$. Note that the variables $\{\widetilde{C}_i\}_{i=1}^m$ satisfy the following properties: (i) They are t-wise independent; (ii) Each $\widetilde{C}_i$ is 0 on expectation and hence $\mathbf{E}[\widetilde{C}] = 0$, where $\widetilde{C} = \sum_{i=1}^m \widetilde{C}_i$. We can thus apply Lemma 2.2 to the variables $\widetilde{C}_1, \widetilde{C}_2, \cdots, \widetilde{C}_m$ with $\mu = \mathbf{E}[\widetilde{C}] = 0$ to get the following:

$$\mathbf{E}[\widetilde{C}^t] \le K_t \cdot t^t, \tag{18}$$

where $K_t \le 8$.

The next claim shows that $\mathbf{E}[\Sigma_j^t] < \mathbf{E}[\widetilde{C}^t]$. Note that from Eq. (18) and Claim 5.7 we get that for all $j \in \{1,2,3\}$, $\mathbf{E}[\Sigma_j^t] < K_t \cdot t^t$; applying Markov's inequality we obtain that for any $A > 0$, $\Pr[\Sigma_j > A] < K_t \cdot \left(\frac{t}{A}\right)^t$. This concludes the proof of Lemma 5.6.

Claim 5.7. $\mathbf{E}[\Sigma_j^t] < \mathbf{E}[\widetilde{C}^t]$.

Proof. For any m variables $Y_1, Y_2, \ldots, Y_m$, we have that $(Y_1 + Y_2 + \ldots + Y_m)^t$ is a polynomial of degree t. Therefore, to prove the above claim, using linearity of expectation, it is sufficient to show that the expectation of each monomial in the right hand side of the inequality is individually greater than the expectation of each monomial in the left hand side. From both sides, we take a term of the form[5] $\mathbf{E}[\prod_{i=1}^l Y_{a_i}^{e_i}]$ where $e_i, l \in [t]$, $a_i \in [m]$ and $\sum_{i=1}^l e_i = t$.

Note that the variables $C_{x_1}, C_{x_2}, \ldots, C_{x_m}$ are not independent. However, since within a set A_j there is no pair x, x' such that $f(x) = x'$, the only possibility of dependence among the variables arises from the event: $f(x) = f(x')$. We can further partition the variables in the product $\prod_{i=1}^l C_{x_{a_i}}^{e_i}$ into sub-products $\Pi_b = \prod_{i\in S_b} C_{x_{a_i}}^{e_i}$ for disjoint subsets $S_b \subseteq [l]$ where $b \in \{1,2,\ldots,l'\}$ and $1 \le l' \le l$, such that, in each sub-product Π_b, for all $i \in S_b$, $f(x_{a_i}) = x'_b$ (for some $x'_b \in A_{j'} \cup S_f$, with $j' \neq j$). Now, since (i) by definition, dependent variables are within the same sub-product and (ii) the hash function h is $2t$-wise independent, the sub-products $\{\Pi_b\}_{b=1}^{l'}$ are mutually independent.

Next, we compute the expectation $\mathbf{E}[\Pi_b]$:

[5] We ignore the multiplicative constant as it is the same on both the sides.

$$\begin{aligned}\mathbf{E}[\Pi_b] &= \mathbf{E}[\Pi_b | h(x'_b) = y'] \Pr[h(x'_b) = y'] + \mathbf{E}[\Pi_b | h(x'_b) \neq y'] \Pr[h(x'_b) \neq y'] \\ &= \mathbf{E}\left[\prod_{i \in S_b} C^{e_i}_{x_{a_i}} | h(x'_b) = y'\right] \Pr[h(x'_b) = y'] = p \cdot \mathbf{E}\left[\prod_{i \in S_b} \widetilde{C}^{e_i}_{a_i}\right]. \end{aligned} \tag{19}$$

The second equality of Eq. (19) follows from the fact that $C_x = 0$ whenever $h(f(x)) \neq y'$; the third equality follows from the definition of $\widetilde{C}_{a_i}$. We now compute the expectation of the product $\prod_{i=1}^{l} C^{e_i}_{x_{a_i}}$, as follows:

$$\mathbf{E}\left[\prod_{i=1}^{l} C^{e_i}_{x_{a_i}}\right] = \prod_{b=1}^{l'} \mathbf{E}[\Pi_b] = \prod_{b=1}^{l'} \left(p \cdot \mathbf{E}\left[\prod_{i \in S_b} \widetilde{C}^{e_i}_{a_i}\right]\right) = p^{l'}\, \mathbf{E}\left[\prod_{i=1}^{l} (\widetilde{C}^{e_i}_{a_i})\right] < \mathbf{E}\left[\prod_{i=1}^{l} (\widetilde{C}^{e_i}_{a_i})\right]. \tag{20}$$

The first equality of Eq. (20) follows from the fact that the sub-products Π_b are mutually independent, the second equality follows from Eq. (19) and the third equality from the t-wise independence of the variables $\widetilde{C}_{a_i}$. This concludes the proof of the claim. □

□

Optimal Rate of Non-Malleable Key-Derivation. We can define the rate of a key derivation function $h : \{0,1\}^n \to \{0,1\}^k$ as the ratio k/n. Notice that our construction achieves rate arbitrary close to 1/2. We claim that this is *optimal* for non-malleable key derivation. To see this, consider a tampering function $f : \{0,1\}^n \to \{0,1\}^n$ which is a permutation and never identity: $f(x) \neq x$. In this case the joint distribution $(h(X), h(f(X)))$ is ε-close to $(U_k, h(f(X)))$ which is ε-close to the distribution (U_k, U'_k) consisting of $2k$ random bits. Since all of the randomness in $(h(X), h(f(X)))$ comes from X, this means that X must contain at least $2k$ bits of randomness, meaning that $n > 2k$.

6 A Tamper-resilient Stream Cipher

Throughout this section, we write $\mathbf{CD}^{\mathsf{A}}(X_1; X_2) = |\Pr[\mathsf{A}(X_1) = 1] - \Pr[\mathsf{A}(X_2) = 1]|$ for the advantage of a PPT adversary A in distinguishing two random variables X_1 and X_2 (defined over some space $\mathcal{X}$).

A stream cipher SC takes as input an initial key $s_0 \in \{0,1\}^n$ and is executed in rounds. For $i \in [q]$, it computes in the i-th round $(s_i, x_i) = \mathsf{SC}(s_{i-1})$ where $x_i \in \{0,1\}^v$ is given to the adversary. We write $\mathsf{SC}^i(s_0) = (s_i, x_i)$ to denote the i-th output of SC, when run for i rounds with initial key s_0. We also write (s_i, x_i) for the corresponding random variables, generated by sampling $s_0 \leftarrow \{0,1\}^n$ and running the stream cipher on this key.

The standard security requirement says that even given $x_1, \ldots, x_{i-1}$, the adversary cannot distinguish between the next block x_i and a uniform random sample $u \leftarrow U_v$. We strengthen this security requirement and allow the adversary to tamper additionally with the secret state of SC. Of course, if an adversary can apply an arbitrary function f_i to the state, he may just overwrite it with a known key, which clearly contradicts the pseudorandomness of x_{i+1}. Notice that such an "overwriting" makes the cipher useless and does not help the adversary to, e.g., decrypt ciphertexts that were encrypted with $\mathsf{SC}(s_0)$. To model tamper resilience of SC we consider an adversary A that obtains both the correctly evaluated outputs $\mathbf{x} = (x_1, \ldots, x_i)$ of $\mathsf{SC}(S_0)$ and the faulty outputs $\mathbf{x}' = (x'_1, \ldots, x'_i)$ where $(s'_{i+1}, x'_{i+1}) = \mathsf{SC}(f_{i+1}(s'_i))$. We say that SC is secure against tampering

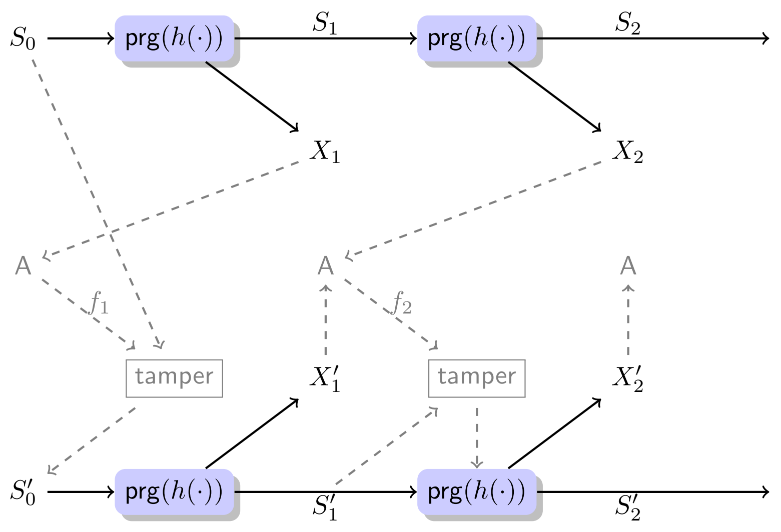

Figure 2: Construction of a tamper-resilient stream cipher. The regular evaluation is shown in black (at the top line), the attack related part is shown in gray with dashed lines, and the corresponding tampered evaluation is shown in black (at the bottom line). Recall that the choice of the tampering function is adaptive from the a-priori fixed set $\mathcal{F}$ that is tolerated by the underlying non-malleable key-derivation.

attacks if even given the outputs $(\mathbf{x}, \mathbf{x}')$ the random variable x_{i+1} is computationally close to uniform.

More formally, consider the following experiment (running with a PPT adversary A and a family of functions $\mathcal{F}$) and denote its output by $\mathsf{view}_{\mathsf{A},\mathcal{F}}(q)$:

1. Sample $s_0 \leftarrow U_n$ and let $s_0' := s_0$.
2. The adversary repeats the following for each $i \in [q]$:
   (a) *Compute untampered output:* Compute $(x_i, s_i) = \mathsf{SC}(s_{i-1})$ and give x_i to A.
   (b) *Compute tampered output:* Receive $f_i \in \mathcal{F}$ from A. Compute $(x_i', s_i') = \mathsf{SC}(f_i(s_{i-1}'))$ and give x_i' to A.
3. The output of the experiment is defined as $\mathbf{x} = (x_1, \ldots, x_q)$ and $\mathbf{x}' = (x_1', \ldots, x_q')$.

Definition 6.1. *Let $\mathsf{SC} : \{0,1\}^n \to \{0,1\}^v \times \{0,1\}^n$. We say that SC is continuous $(\mathcal{F}, \varepsilon)$-tamper-resilient if for all PPT adversary A*

$$\mathbf{CD}^{\mathsf{A}}\big((x_{q+1}, \mathsf{view}_{\mathsf{A},\mathcal{F}}(q)); (U_v, \mathsf{view}_{\mathsf{A},\mathcal{F}}(q))\big) \le \varepsilon,$$

where $\mathsf{view}_{\mathsf{A},\mathcal{F}}(q)$ is defined as above, $x_{q+1} := \mathsf{SC}^q(s_0)$, and SC^q denotes q executions of the stream cipher.

The construction. Recall that a pseudorandom generator (PRG) is a function $\mathsf{prg} : \{0,1\}^{k_1} \to \{0,1\}^{k_2}$; we say that prg is ε-secure if for all PPT adversaries A we have $\mathbf{CD}^{\mathsf{A}}(\mathsf{prg}(U_{k_1}); U_{k_2}) \le \varepsilon$.

Consider the following construction of a stream cipher $\mathsf{SC}^{h,\mathsf{prg}}$, based on a PRG $\mathsf{prg} : \{0,1\}^k \to \{0,1\}^{n+v}$ and a non-malleable key-derivation function $h : \{0,1\}^n \to \{0,1\}^k$ (see also Figure 2). At the beginning a description of the function h and of the pseudorandom generator prg are output as public parameters. Then a key $s_0 \leftarrow \{0,1\}^n$ is sampled uniformly at random and for each $i \in [q]$ we define the output of the stream cipher at round i as $(s_i, x_i) := \mathsf{prg}(h(s_{i-1}))$.

Theorem 6.2. *Assume that* prg *is an* $\varepsilon_{\mathsf{prg}}$*-secure pseudorandom generator and that* h *is an* $(\mathcal{F}, \varepsilon)$*-non-malleable key-derivation function. Then the stream cipher* $\mathsf{SC}^{h,\mathsf{prg}}$ *defined above is continuous* $(\mathcal{F}, \varepsilon')$*-tamper-resilient, where* $\varepsilon' \leq (2q+1)\varepsilon + 2q\varepsilon_{\mathsf{prg}}$.

Proof. Consider the distribution $D_{f,h}$ over $\{0,1\}^k \cup \mathsf{same}^\star$, which samples $s \leftarrow U_n$ and outputs $\mathsf{same}^\star$ if $f(s) = s$ and $h(f(s))$ otherwise.

Before giving the formal proof, let us discuss some intuition. We show the desired computational indistinguishability through several intermediate hybrid games. Notice that the interaction of the challenger and the adversary in the definition, can be viewed as if there were two chains of values: (i) an "untampered" chain, similar to the standard stream cipher game (c.f. Step 2a in the definition); and (ii) a "tampered chain", where the adversary can tamper with each input of the function h (c.f. Step 2b in the definition). In the i-th hybrid we replace the i-th output of h in the untampered chain, by a uniform random value. In the tampered chain, if the adversary is yet to tamper, then we replace the output of h with a random sample from the distribution $D_{f_i,h}$. In case $D_{f_i,h}$ returns some value which is not $\mathsf{same}^\star$, i.e. the adversary tampered, then we stop sampling further and continue to simulate the output from that point on using the last sampled value. Non-malleability of the key-derivation function h guarantees that, once the adversary tampers, the modified value reveals almost no information about the "extracted" key. Notice that, if before entering the i-th round the adversary had already tampered, then there is no difference between the i-th and $(i+1)$-th hybrids in the tampered chain. We now proceed with the formal proof.

Define the following hybrid games for all $i \in \{0, \ldots, q\}$ and all $b \in \{0,1\}$.

$\mathsf{Game}_i(b)$:

1. Sample $s_0 \leftarrow U_n$ and let $s'_0 := s_0$. Set initial mode to *normal mode.*
2. For all $j = 1, \ldots, i$ do the following.
   (a) *Compute untampered output:* Sample $\kappa_j \leftarrow U_k$. Compute $(x_j, s_j) \leftarrow \mathsf{prg}(\kappa_j)$ and give x_j to A.
   (b) *Compute tampered output:* Receive $f_j \in \mathcal{F}$ from A. Depending on the current mode behave in one of the following ways:
      - *Normal Mode*: Sample $\widetilde{\kappa}_j \leftarrow D_{f_j,h}$. If $\widetilde{\kappa}_j = \mathsf{same}^\star$, then set $(x'_j, s'_j) := (x_j, s_j)$ and give x'_j to A. Else, if $\widetilde{\kappa}_j \neq \mathsf{same}^\star$, then set the current mode to *overwritten mode*, set $(x'_j, s'_j) := \mathsf{prg}(\widetilde{\kappa}_j)$ and give x'_j to A.
      - *Overwritten Mode*: Compute $(x'_j, s'_j) \leftarrow \mathsf{prg}(h(f_j(s'_{j-1})))$, give x'_j to A.
3. For all $j = i+1, \ldots, q$ do the following.
   (a) *Compute untampered output:* Compute $(x_j, s_j) \leftarrow \mathsf{prg}(h(s_{j-1}))$ and give x_j to A.
   (b) *Compute tampered output:* Receive $f_j \in \mathcal{F}$ from A. Compute $(x'_j, s'_j) \leftarrow \mathsf{prg}(h(f_j(s'_{j-1})))$ and give x'_j to A.
4. *Challenge Phase.* If $b = 0$, then set $(x_{q+1}, s_{q+1}) \leftarrow \mathsf{prg}(h(s_q))$; otherwise, sample $x_{q+1} \leftarrow U_v$. The output of the experiment is defined as $\mathsf{Game}_i(b) = (x_{q+1}, \mathbf{x}, \mathbf{x}')$ where $\mathbf{x} = (x_1, \ldots, x_q)$ and $\mathbf{x}' = (x'_1, \ldots, x'_q)$.

Notice that the output distribution of $\mathsf{Game}_0(b)$ is either equal to $(x_{q+1}, \mathsf{view}_{\mathsf{A},\mathcal{F}}(q))$ (in case $b = 0$) or to $(u \leftarrow U_v, \mathsf{view}_{\mathsf{A},\mathcal{F}}(q))$ (in case $b = 1$); thus our goal is to bound $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_0(0); \mathsf{Game}_0(1)\big)$. By using the triangle inequality, we write:

$$\begin{aligned}\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_0(0); \mathsf{Game}_0(1)\big) &\le \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_0(0); \mathsf{Game}_1(0)\big) + \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_1(0); \mathsf{Game}_q(0)\big) \\ &\quad + \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_q(0); \mathsf{Game}_q(1)\big) + \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_q(1); \mathsf{Game}_1(1)\big) \\ &\quad + \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_1(1); \mathsf{Game}_0(1)\big) \\ &\le 3\varepsilon + 2\varepsilon_{\mathsf{prg}} + \sum_{b\in\{0,1\}} \sum_{i=1}^{q-1} \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_i(b); \mathsf{Game}_{i+1}(b)\big) \qquad (21) \\ &\le (2q+1)\varepsilon + 2q\varepsilon_{\mathsf{prg}}. \qquad (22)\end{aligned}$$

Eq. (21) follows by applying Claim 6.3 and Claim 6.5 whereas Eq. (22) follows from Claim 6.4. This concludes the proof of the theorem except the proofs of the claims which are given below.

Claim 6.3. $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_q(0); (\mathsf{Game}_q(1)\big) \le \varepsilon + 2\varepsilon_{\mathsf{prg}}$.

Proof. Consider the following modified games.

Game'_q: First run $\mathsf{Game}_q(0)$ until Step 3. In Step 4 sample $s_q \leftarrow U_n$ and compute $(x_{q+1}, s_{q+1}) \leftarrow \mathsf{prg}(h(s_q))$. Output $\mathsf{Game}'_q = (x_{q+1}, \mathbf{x}, \mathbf{x}')$.

Game''_q: First run $\mathsf{Game}_q(1)$ until Step 3. In Step 4 sample $\kappa_{q+1} \leftarrow U_k$ and compute $(x_{q+1}, s_{q+1}) \leftarrow \mathsf{prg}(\kappa_{q+1})$. Output $\mathsf{view}''_q = (x_{q+1}, \mathbf{x}, \mathbf{x}')$.

We notice that the only difference between $\mathsf{Game}_q(0)$ and Game'_q is in the challenge phase: in the former s_q is computed as the output of $\mathsf{prg}(\kappa_{q-1})$ for some uniformly random $\kappa_{q-1} \leftarrow U_k$, whereas in the latter s_q is uniformly random. Clearly, $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_q(0); \mathsf{Game}'_q\big) \le \varepsilon_{\mathsf{prg}}$. By a similar argument, $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}''_q; \mathsf{Game}_q(1)\big) \le \varepsilon_{\mathsf{prg}}$.

Let us now compare the output distribution of Game'_q and Game''_q. Again, the only difference is in the challenge phase: in the former the input of prg is computed as $h(s_q)$ for some uniformly random $s_q \leftarrow U_n$, whereas in the latter the input of prg is sampled uniformly as $\kappa_{q+1} \leftarrow U_k$. Now the fact that h is non-malleable clearly implies that the output of h on a random input is close to uniform.[6] This shows that $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}'_q; \mathsf{Game}''_q\big) \le \varepsilon$.

Combining the above arguments, we conclude

$$\begin{aligned}\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_q(0); \mathsf{Game}_q(1)\big) &\le \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_q(0); \mathsf{Game}'_q\big) \\ &\qquad + \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}'_q; \mathsf{Game}''_q\big) + \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}''_q; \mathsf{Game}_q(1)\big) \\ &\le \varepsilon + 2\varepsilon_{\mathsf{prg}},\end{aligned}$$

as desired. □

Claim 6.4. *For all* $b \in \{0,1\}$ *and all* $i \in [q-1]$, $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_i(b); \mathsf{Game}_{i+1}(b)\big) \le \varepsilon + \varepsilon_{\mathsf{prg}}$.

Proof. Fix some $b \in \{0,1\}$ and $i \in [q-1]$. Consider the following modified game.

[6] In this step we only require that the output of h is close to uniform, which is obviously a weaker property than non-malleability.

$\mathsf{Game}'_i(b)$: First run $\mathsf{Game}_i(b)$ until Step 2. In Step 3 first sample $s_i \leftarrow U_n$ and then do the following for all $j = i+1, \ldots, q$:

(a) *Compute untampered output:* Compute $(x_j, s_j) \leftarrow \mathsf{prg}(h(s_{j-1}))$ and give x_j to A.

(b) *Compute tampered output:* Receive $f_j \in \mathcal{F}$ from A. In case the execution is still in normal mode, set $s'_i := s_i$. In the next step, irrespective of the mode, compute $(x'_j, s'_j) \leftarrow \mathsf{prg}(h(f_j(s'_{j-1})))$ and give x'_j to A.

Step 4 is defined exactly as in $\mathsf{Game}_i(b)$.

We notice that the only difference between $\mathsf{Game}_i(b)$ and $\mathsf{Game}'_i(b)$ is in Step 3: in the i-th round the last output of prg (i.e., s_i) is replaced by a uniform random value in the latter. Note that in case the execution has already entered the overwritten mode, this change does not affect the tampered output; on the other hand, in case the execution is still in normal mode, then in the tampered output the i-th output of prg is replaced by the above uniform value s_i. Thus, clearly, $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}'_i(b); \mathsf{Game}_i(b)\big) \le \varepsilon_{\mathsf{prg}}$.

Let us now compare the output distribution of $\mathsf{Game}'_i(b)$ and $\mathsf{Game}_{i+1}(b)$. Again the only differences are as follows:

- Regarding the untampered output, the $(i+1)$-th output of h is replaced by a uniform random value κ_{i+1} in $\mathsf{Game}_{i+1}(b)$ whereas in $\mathsf{Game}'_i(b)$ it is computed as $h(s_i)$ for some uniform s_i.
- Regarding the tampered output, there is no difference in case the execution has already entered the overwritten mode. In case the execution is still normal mode, then $\mathsf{Game}_{i+1}(b)$ samples $\widetilde{\kappa}_{i+1} \leftarrow D_{f_{i+1},h}$. If $\widetilde{\kappa}_{i+1}$ is not $\mathsf{same}^{\star}$, then the output will be computed by running prg on $\widetilde{\kappa}_{i+1}$; else, the output is just copied from the untampered output.

We observe that $\mathsf{Game}'_i(b)$ can be computed as a deterministic function of $\mathsf{Real}_h(f_{i+1})$ and $\mathsf{Game}_{i+1}(b)$ as a deterministic function of $\mathsf{Sim}_h(f_{i+1})$ (c.f. Figure 1). This shows that $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}'_i(b); \mathsf{Game}_{i+1}(b)\big) \le \varepsilon$.

Combining the above arguments, we conclude that for any $b \in \{0,1\}$ and all $i \in [q-1]$

$$\begin{aligned}\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_i(b); \mathsf{Game}_{i+1}(b)\big) &\le \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_i(b); \mathsf{Game}'_i(b)\big) + \mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}'_i(b); \mathsf{Game}_{i+1}(b)\big)\\ &\le \varepsilon + \varepsilon_{\mathsf{prg}},\end{aligned}$$

as desired. □

Claim 6.5. *For all $b \in \{0,1\}$, $\mathbf{CD}^{\mathsf{A}}\big(\mathsf{Game}_0(b); \mathsf{Game}_1(b)\big) \le \varepsilon$.*

Proof. The statement can be shown in a similar way as in the proof of Claim 6.4, with the only difference that in this case the first input to h is a uniformly random value s_0 (rather that some "pseudorandom" value), and thus the modified game $\mathsf{Game}'_0(b)$ collapses to $\mathsf{Game}_0(b)$. □

□

Acknowledgements

We thank Ivan Damgård for useful discussions at the early stages of this research.

References

[1] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:81, 2013.

[2] Ross Anderson, Markus Kuhn, and England U. S. A. Tamper resistance — a cautionary note. In *In Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, 1996.

[3] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *ICS*, pages 45–60, 2011.

[4] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *CRYPTO*, pages 666–684, 2010.

[5] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In *EUROCRYPT*, pages 491–506, 2003.

[6] Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. RKA security beyond the linear barrier: IBE, encryption and signatures. In *ASIACRYPT*, pages 331–348, 2012.

[7] Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *FOCS*, pages 276–287. IEEE Computer Society, 1994.

[8] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *J. Cryptology*, 14(2):101–119, 2001.

[9] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:118, 2013.

[10] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. *IACR Cryptology ePrint Archive*, 2013:565, 2013.

[11] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. BiTR: Built-in tamper resilience. In *ASIACRYPT*, pages 740–758, 2011.

[12] Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. *IACR Cryptology ePrint Archive*, 2014:324, 2014.

[13] Jean-Sébastien Coron, Antoine Joux, Ilya Kizhvatov, David Naccache, and Pascal Paillier. Fault attacks on rsa signatures with partially unknown messages. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 444–456. Springer, 2009.

[14] 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, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 2008.

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

[16] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In Juan A. Garay and Roberto De Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2010.

[17] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In *STOC*, pages 601–610, 2009.

[18] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

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

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

[21] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *TCC*, pages 465–488, 2014.

[22] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In *ICALP (1)*, pages 391–402, 2011.

[23] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.

[24] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In *TCC*, pages 182–200, 2011.

[25] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.

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

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

[28] Krzysztof Pietrzak. Subspace LWE. In *TCC*, pages 548–563, 2012.

[29] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2013.

[30] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. pages 2–12. Springer-Verlag, 2002.

[31] Hoeteck Wee. Public key encryption against related key attacks. In *Public Key Cryptography*, pages 262–279, 2012.

A One-Time Tamper Simulatability

Similarly to [20], we show that a non-malleable key-derivation function can be use to protect any stateless functionality against tampering attacks. We consider two main differences with respect to the setting considered in [20]: (i) the original functionality is stateless and works with a uniformly chosen state; (ii) the attacker can only tamper once.

A stateless functionality $\langle \mathsf{G}, \kappa \rangle$ consists of a public (possibly randomized) function $\mathsf{G} : \{0,1\}^k \times \{0,1\}^u \to \{0,1\}^v$ and a secret initial state $\kappa \in \{0,1\}^k$. Whenever the state κ is chosen uniformly at random from $\{0,1\}^k$, we say that the functionality is *regular*. The main idea is to transform $\langle \mathsf{G}, \kappa \rangle$ into some "hardened" functionality $\langle \mathsf{G}^h, s \rangle$ via a non-malleable key-derivation function h.

Definition A.1 (Hardened functionality)**.** *Let $h : \{0,1\}^n \to \{0,1\}^k$ be a function. Let $\mathsf{G} : \{0,1\}^k \times \{0,1\}^u \to \{0,1\}^v$ be any* stateless, regular *functionality with k-bit state. We define the hardened functionality $\mathsf{G}^h : \{0,1\}^n \times \{0,1\}^u \to \{0,1\}^v$ to be the functionality that takes as input $(s, x) \in \{0,1\}^n \times \{0,1\}^u$ and outputs $y \leftarrow \mathsf{G}(h(s), x)$.*

Security of G^h is defined via the comparison of a real and an ideal experiment. The real experiment features an adversary A interacting with G^h; the adversary is allowed to honestly run the functionality on any chosen input, but also to tamper with the original state and interact with the modified functionality. The ideal experiment features a simulator S; the simulator is given black-box access to the original functionality G and to the adversary A, but is *not* allowed any tampering query. The two experiments are formally described below.

Experiment $\mathsf{Real}_{\mathsf{A},\mathcal{F}}^{\mathsf{G}^h(s,\cdot)}$. A value $s \leftarrow \{0,1\}^n$ is chosen uniformly at random. Then A can issue the following commands (in any order):

- $\langle \texttt{Eval}, x \rangle$: In response to an evaluation query, return $y \leftarrow \mathsf{G}(h(s), x)$. This command can be run a polynomial number of times.
- $\langle \texttt{Tamper}, f \rangle$: Upon input $f : \{0,1\}^n \to \{0,1\}^n$, with $f \in \mathcal{F}$, replace s by $f(s)$. This command can be run a single time.

The output of the experiment is defined as

$$\mathsf{Real}_{\mathsf{A},\mathcal{F}}^{\mathsf{G}^h(s,\cdot)} = ((x_1, y_1), (x_2, y_2), \dots)).$$

Experiment $\mathsf{Ideal}_{\mathsf{S}}^{\mathsf{G}(\kappa,\cdot)}$. A value $\kappa \leftarrow \{0,1\}^k$ is chosen uniformly at random. The simulator is given black-box access to the functionality $\mathsf{G}(\kappa, \cdot)$ and the adversary A. The output of the experiment is defined as

$$\mathsf{Ideal}_{\mathsf{S}}^{\mathsf{G}(\kappa,\cdot)} = ((x_1, y_1), (x_2, y_2), \dots),$$

where (x_j, y_j) are the input/output tuples simulated by S.

Definition A.2 (One-time tamper simulatability)**.** *Let $h : \{0,1\}^n \to \{0,1\}^k$ be a function and consider a* stateless, regular *functionality G. Denote with G^h the hardened functionality corresponding to G. We say that h is one-time $(\mathcal{F}, \varepsilon)$-tamper simulatable for G, if for all PPT adversaries A there exists a PPT simulator S such that for any initial state κ,*

$$\mathsf{Real}_{\mathsf{A},\mathcal{F}}^{\mathsf{G}^h(s,\cdot)} \approx \mathsf{Ideal}_{\mathsf{S}}^{\mathsf{G}(\kappa,\cdot)},$$

where $\approx$ refers either to statistical or computational indistinguishability.

The theorem below states that whenever h is a non-malleable key-derivation function, then it is also one-time tamper simulatable.

Theorem A.3. *Consider a stateless, regular functionality* G*. Let* h *be an* $(\mathcal{F}, \varepsilon)$*-non-malleable key-derivation function, then* h *is one-time* $(\mathcal{F}, \varepsilon)$*-tamper simulatable for* G*.*

Proof. Define the following distribution $D_{f,h}$ over $\{0,1\}^k \cup \mathsf{same}^\star$ (which is efficiently samplable given black-box access to functions f and h): Sample $s \leftarrow \{0,1\}^n$ and define $\widetilde{\kappa} = \mathsf{same}^\star$ if $f(s) = s$ and $\widetilde{\kappa} = h(f(s))$ otherwise. The simulator S, having black-box access to the adversary A and the original functionality $\langle \mathsf{G}, \kappa \rangle$, will use $D_{f,h}$ to answer A's tampering query.

At the beginning, S runs A and then it works in one of two modes defined below (starting with the "normal mode"):

1. *Normal mode.* While A continues issuing queries, answer as follows:
   - Upon input $\langle \mathtt{Eval}, x \rangle$, forward x to $\mathsf{G}(\kappa, \cdot)$ and return the output y back to A.
   - Upon input $\langle \mathtt{Tamper}, f \rangle$, sample $\widetilde{\kappa} \leftarrow D_{f,h}$. Hence,
     - In case $\widetilde{\kappa} = \mathsf{same}^\star$, stay in the current mode.
     - In case $\widetilde{\kappa} \neq \mathsf{same}^\star$, go to the "overwritten mode" defined below with state $\widetilde{\kappa}$.
2. *Overwritten mode.* Given state $\widetilde{\kappa}$ simulate all further evaluation queries by running $\mathsf{G}(\widetilde{\kappa}, \cdot)$.
3. Output whatever A does.

We argue that $\mathsf{Real}_{\mathsf{A},\mathcal{F}}^{\mathsf{G}^h(s,\cdot)} \approx \mathsf{Ideal}_{\mathsf{S}}^{\mathsf{G}(\kappa,\cdot)}$. By non-malleability of the key-derivation function h, we know that for all functions $f \in \mathcal{F}$ it holds that:

$$\mathsf{Real}_h(f) := \left\{ \begin{array}{c} s \leftarrow \{0,1\}^n \\ \text{If } f(s) = s, \quad \text{Output: } (h(s), \mathsf{same}^\star) \\ \text{Else } \text{ Output: } (h(s), h(f(s))) \end{array} \right\} \approx_\varepsilon \left\{ \begin{array}{c} \widetilde{\kappa} \leftarrow D_{f,h}, \kappa \leftarrow \{0,1\}^k \\ \text{Output: } (\kappa, \widetilde{\kappa}) \end{array} \right\} := \mathsf{Sim}_h(f).$$

To argue indistinguishability of the simulation, we observe that the real experiment and the ideal experiment are identical after the simulator enters in overwritten mode. To conclude the proof, we show that they are also indistinguishable for each round up to and including the round in which the simulator enters in overwritten mode. Without loss of generality, assume that A applies the tampering function f at round q, for some $q \in \mathbb{N}$. Denote with $\overline{\mathsf{Real}}_{\mathsf{A},\mathcal{F}}^{\mathsf{G}^h(s,\cdot)}$ the random variable corresponding to the output of the real experiment until round q. The random variable $\overline{\mathsf{Ideal}}_{\mathsf{S}}^{\mathsf{G}(\kappa,\cdot)}$ is defined analogously.

Consider now the following function g, taking as input a pair of values $\kappa^* \in \{0,1\}^k$ and $\widetilde{\kappa} \in (\{0,1\}^k \cup \mathsf{same}^\star)$, together with any sequence of inputs $x_1, \ldots, x_q \in \{0,1\}^u$: Output $y_i = \mathsf{G}(\kappa^*, x_i)$ for all $i \leq q-1$ and $y_q = \mathsf{G}(\kappa^*, x_q)$ if $\widetilde{\kappa} = \mathsf{same}^\star$ and $y_q = \mathsf{G}(\widetilde{\kappa}, x_q)$ otherwise. It is not hard to see that

$$\overline{\mathsf{Real}}_{\mathsf{A},\mathcal{F}}^{\mathsf{G}^h(s,\cdot)} = g(\mathsf{Real}_h(f)) \quad \text{and} \quad \overline{\mathsf{Ideal}}_{\mathsf{S}}^{\mathsf{G}(\kappa,\cdot)} = g(\mathsf{Sim}_h(f)).$$

Thus,

$$\mathbf{SD}(\overline{\mathsf{Real}}_{\mathsf{A},\mathcal{F}}^{\mathsf{G}^h(s,\cdot)}; \overline{\mathsf{Ideal}}_{\mathsf{S}}^{\mathsf{G}(\kappa,\cdot)}) = \mathbf{SD}(g(\mathsf{Real}_h(f)); g(\mathsf{Sim}_h(f))) \leq \mathbf{SD}(\mathsf{Real}_h(f); \mathsf{Sim}_h(f)) \leq \varepsilon,$$

concluding the proof. □