Non-Malleable Codes

Stefan Dziembowski [*] Krzysztof Pietrzak [†] Daniel Wichs [‡]

Abstract

We introduce the notion of "non-malleable codes" which relaxes the notion of error-correction and error-detection. Informally, a code is non-malleable if the message contained in a modified codeword is either the original message, or a completely unrelated value. In contrast to error-correction and error-detection, non-malleability can be achieved for very rich classes of modifications.

We construct an efficient code that is non-malleable with respect to modifications that effect each bit of the codeword arbitrarily (i.e. leave it untouched, flip it or set it to either 0 or 1), but independently of the value of the other bits of the codeword. Using the probabilistic method, we also show a very strong and general statement: there exists a non-malleable code for *every* "small enough" family $\mathcal{F}$ of functions via which codewords can be modified. Although this probabilistic method argument does not directly yield efficient constructions, it gives us efficient non-malleable codes in the random-oracle model for very general classes of tampering functions — e.g. functions where every bit in the tampered codeword can depend arbitrarily on any 99% of the bits in the original codeword.

As an application of non-malleable codes, we show that they provide an elegant algorithmic solution to the task of protecting functionalities implemented in hardware (e.g. signature cards) against "tampering attacks". In such attacks, the secret state of a physical system is tampered, in the hopes that future interaction with the modified system will reveal some secret information. This problem, was previously studied in the work of Gennaro et al. in 2004 under the name "algorithmic tamper proof security" (ATP). We show that non-malleable codes can be used to achieve important improvements over the prior work. In particular, we show that any functionality can be made secure against a large class of tampering attacks, simply by encoding the secret-state with a non-malleable code while it is stored in memory.

1 Introduction

Consider the following three-step process, which we call the "tampering experiment":

1. A *source message* s is encoded via a (possibly randomized) procedure Enc, yielding a *codeword* $c = \mathsf{Enc}(s)$.
2. The codeword is modified under some *tampering-function* $f \in \mathcal{F}$ to an *erroneous-codeword* $\tilde{c} = f(c)$.
3. The erroneous-codeword $\tilde{c}$ is decoded using a procedure Dec, resulting in a *decoded-message* $\tilde{s} = \mathsf{Dec}(\tilde{c})$.

The tampering experiment can be used to model several interesting real-world settings, such as data transmitted over a noisy channel, or adversarial tampering of data stored in the memory of a physical device, which we will talk about more later. We would like to build special encoding/decoding procedures $(\mathsf{Enc}, \mathsf{Dec})$, which give us some meaningful guarantees about the results of the above tampering experiment, for large and interesting families $\mathcal{F}$ of tampering functions. Let us explore several possibilities for the type of guarantees that we may hope for.

[*] University of Rome, *La Sapienza* (stefan@dziembowski.net). The work leading to this invention has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n. 207908.

[†] CWI Amsterdam (K.Z.Pietrzak@cwi.nl)

[‡] New York University (wichs@cs.nyu.edu)

Error Correction. One very natural guarantee, called error correction, would be to require that for any tampering-function $f \in \mathcal{F}$ and any source-message s, the tampering experiment always produces the correct decoded-message $\tilde{s} = s$. This notion of correction was first studied in the seminal work of Shannon in 1948 [35] with respect to random noise (e.g each bit of the codeword c is flipped with some probability $p < 1/2$) and later by Hamming in 1950 [20], who introduced the notion of an *error-correcting code* with respect to worst-case errors. In particular, we can think of Hamming's notion of an error-correcting code (with distance d) as guaranteeing the error-correction property for the family $\mathcal{F}$ of functions f for which the *Hamming distance* between an erroneous-codeword $\tilde{c} = f(c)$ and c is small (at most $d/2 - 1$). Since their introduction, error-correcting codes have received much attention and found countless applications in both theory and practice. There has also been some study of error correction for other families $\mathcal{F}$, such as ones where tampering functions preserve edit distance instead of Hamming distance (see the survey of [27]). However, it is also clear that error correction cannot be achieved for many natural (and simple) function families – such as even the single "zero function" f that maps all values x to the all-zero string.

Error Detection. A weaker guarantee, called error-detection, requires that the tampering-experiment always results in either the correct value $\tilde{s} = s$ or a special symbol $\tilde{s} = \perp$ indicating that tampering has been detected (where the latter can only happen when the codeword has been modified to $\tilde{c} \neq c$). This notion of error-detection is a weaker guarantee than error-correction, and achievable for larger families $\mathcal{F}$ of tampering functions. For example, Hamming's notion of an error-correcting code with distance d can detect up to $d - 1$ errors (but only correct $d/2 - 1$ errors).

The recent work of Cramer et al. [12] introduced the notion of "Algebraic Manipulation Detection" (AMD) codes. Such codes have an inherently *randomized* encoding procedure and guarantee that for *any a-priori chosen error-vector* $\Delta \neq 0$, we get $\mathsf{Dec}(\mathsf{Enc}(s) + \Delta) = \perp$ with overwhelming probability over the randomness of the encoding.[1] In other words, these codes guarantee error-detection with respect to the family $\mathcal{F}_{\mathsf{err}}$ consisting of *all constant-error functions* f_Δ given by $f_\Delta(x) \overset{\text{def}}{=} x + \Delta$. The definition of AMD codes is an interesting departure from the traditional study of error correction/detection, where restrictions are usually placed on the relationship between the codeword c and the erroneous-codeword $\tilde{c}$ (e.g. they are assumed to be close in some metric space). Under the tampering-family $\mathcal{F}_{\mathsf{err}}$, the erroneous-codeword $\tilde{c} = f_\Delta(c) = c + \Delta$ can in principle take on *any* possible value, and hence the final relationship between c and $\tilde{c}$ can be arbitrary. However, the tampering functions $f_\Delta \in \mathcal{F}_{\mathsf{err}}$ are *restricted in how they derive the erroneous-codeword $\tilde{c}$ from c*, since the error-vector Δ is specified a-priori and independently of the value of c (which contains some randomness from the encoding procedure). We take a similar view in this work, and study restrictions on the family $\mathcal{F}$ of tampering functions rather than just the final relationship between a codeword and an erroneous codeword.

Unfortunately, even error detection cannot be achieved for many other natural (and simple) function families $\mathcal{F}$. For example, the family $\mathcal{F}_{\mathsf{const}}$ of all *constant-functions* $f_{c^*}(x) \overset{\text{def}}{=} c^*$, always contains some function that maps all values to a "valid" codeword c^* (for which $\mathsf{Dec}(c^*) \neq \perp$), and hence is neither correctable nor detectable.

Non-Malleability. As the main focus of this work, we present yet another meaningful and previously unexplored notion, which we call "non-malleability". A *non-malleable code* ensures that either the tampering experiment results in a correct decoded-message $\tilde{s} = s$, or the decoded-message $\tilde{s}$ is *completely independent of and unrelated to* the source-message s. In other words, our new notion of non-malleability for codes is similar, in spirit, to notions of non-malleability for cryptographic primitives (such as encryption[2], commitments and zero-knowledge proofs), introduced by the seminal work of Dolev, Dwork and Naor [14].

Compared to error correction or error detection, the "right" formalization of non-malleable codes is somewhat harder to define. Let Tamper^f_s be a random variable for the value of the decoded-message $\tilde{s}$, which results when we run the tampering experiment with source-message s and tampering-function f, over the randomness of the

[1] We will often just use the addition operation + over bit-strings, where an n-bit string is interpreted as a value in $\mathbb{F}_2^n$.

[2] For example, a non-malleable encryption scheme ensures that, by modifying the ciphertext in any way, the decrypted value is unrelated to that contained in the original ciphertext.

encoding procedure. Intuitively, we wish to say that the distribution of Tamper_s^f is independent of the encoded message s. Of course, we also want to allow for the case where the tampering experiment results in $\tilde{s} = s$ (for example, if the tampering function is identity), which clearly depends on s. Thus, we require that for every tampering-function $f \in \mathcal{F}$, there exists a distribution D_f which outputs either concrete values $\tilde{s}$ or a special $\mathsf{same}^\star$ symbol, and faithfully models the distribution of Tamper_s^f for all s in the following sense: for every source-message s, the distributions of Tamper_s^f and D_f are statistically close when the $\mathsf{same}^\star$ symbol is interpreted as s. That is, D_f correctly simulates the "outcome" of the tampering-experiment with a function $f \in \mathcal{F}$ without knowing the source-messages s, but it is allowed some ambiguity by outputting a $\mathsf{same}^\star$ symbol to indicate that the decoded-message should be the same as the source-message, without specifying what the exact value is. The fact that D_f depends on *only* f and *not* on s, shows that the outcome of Tamper_s^f is independent of s, exempting equality.

Error-correction w.r.t. to some class $\mathcal{F}$ of tampering functions trivially implies non-malleability w.r.t. to $\mathcal{F}$: simply define D_f to always output the $\mathsf{same}^\star$ symbol. Error-detection only implies non-malleability if for every $f \in \mathcal{F}$, the probability $\Pr[\mathsf{Dec}(f(\mathsf{Enc}(s))) = \perp]$ (i.e. that a tampered codeword is invalid) is the same for all source-message s. In that case, for each $f \in \mathcal{F}$ there is a distribution D_f over $\{\mathsf{same}^\star, \perp\}$ which satisfies the definition of non-malleability. For example, the above is the case for AMD codes where, for each $f_\Delta \in \mathcal{F}_{\mathsf{err}}$, $\Pr[\mathsf{Dec}(f_\Delta(\mathsf{Enc}(s))) = \perp] = \Pr[\mathsf{Dec}(\mathsf{Enc}(s) + \Delta) = \perp]$ is either 0 if $\Delta = 0$, or negligbily close to 1 if $\Delta \neq 0$, independently of s.

1.1 Main Results for Non-Malleability

We show that non-malleability can be achieved for many rich families $\mathcal{F}$ of tampering functions, for which error-correction and error-detection are impossible.

Bit-Wise Independent Tampering. As one very concrete example, we study non-malleability with respect to the family of functions f which specify, for each bit of the codeword c, whether to keep it as is, flip it, set it to 0, set it to 1. That is, each bit of the codeword is modified *arbitrarily* but independently of the value of the other bits of the codeword. We call this the "bit-wise independent tampering" family $\mathcal{F}_{\mathsf{BIT}}$. Note that this family contains constant functions $\mathcal{F}_{\mathsf{const}}$ and constant-error functions $\mathcal{F}_{\mathsf{err}}$ as subsets. Therefore, as we have mentioned, error-correction and error-detection cannot be achieved w.r.t. this family. Nevertheless, we show an efficient non-malleable code for this powerful family. Our construction, described in Section 4, is based on AMD codes (described above) and a type of linear secret-sharing scheme (over bits) with some additional properties.

All families of bounded size. Even non-malleability cannot be achieved with respect to the family of *all* tampering functions. No matter what encoding/decoding procedure one chooses, there is always some function f, which, on input c, computes $s = \mathsf{Dec}(c)$, sets s' to be the same as s but with e.g. the last bit flipped, and outputs $\tilde{c} = \mathsf{Enc}(s')$. However, we notice that this "bad" function f depends on the encoding/decoding procedure. Therefore, even though *for any* coding scheme, *there exists* some tampering function f that breaks non-malleability, we show that *for any* "small enough" tampering family $\mathcal{F}$, *there exists* some coding scheme which is non-malleable w.r.t. $\mathcal{F}$. Note that, when considering codewords of size n, the family $\mathcal{F}_{\mathsf{all}}$ of all tampering functions on n-bit codewords has size given by $\log(\log(|\mathcal{F}_{\mathsf{all}}|)) = n + \log(n)$. Our result shows that for any *slightly smaller* family $\mathcal{F}$ whose size is given by $\log(\log(|\mathcal{F}|)) < n$, there exist non-malleable codes w.r.t. $\mathcal{F}$ and, in particular, a random code is likely to be non-malleable with overwhelming probability. This is in contrast to error-correction and error-detection, for which there exist some small families $\mathcal{F}$ (of size $\log(\log(|\mathcal{F}|)) = \log(n)$) for which *no* code can be error-correcting or error-detecting (e.g. the family $\mathcal{F}_{\mathsf{const}}$).

Unfortunately, our existential result is derived via a probabilistic method argument, and therefore does not directly lead to efficient constructions. However, as a corollary of the probabilistic method argument, we can show that efficient codes in the "random oracle model" are non-malleable w.r.t. large function families $\mathcal{F}$. For example, we get such non-malleable code w.r.t. all functions which tamper the left and right half of the codeword independently.[3] More generally, we get such codes for the family of functions f, where the ith bit in $f(c)$ is a function of an arbitrary fraction (say, 99%) of the bits of c.

1.2 Application to Tamper-Resilient Cryptography

Traditionally, cryptographic security notions assume that an adversary only has "black-box" access to an attacked system. That is, she can only observe the inputs/output behavior to the system, but gets no further information. Such models often fail to capture physical reality, where an adversary can attack an actual physical implementation of the system. Attacks which go beyond black-box access are called "implementation attacks", and we distinguish between two classes of such attacks, namely "leakage" and "tampering" attacks.[4] Leakage-attacks refer to all attacks where the adversary can learn some information about the secret internal state of a system by measuring some properties of its physical implementation (e.g. timing, radiation, heat, power consumption). Tampering attacks, on the other hand, refer to attacks where the adversary actively modifies the computation (e.g. by cutting wires in the circuit or heating it to introduce random errors in memory), and then learns some information from the input/output behavior of the modified computation. As an example of how such attacks are used, the work of Boneh, DeMillo and Lipton [5] shows that if random faults are introduced into the computation of a single RSA signature, then the resulting output can be used to factor the modulus. Of course, a physical attacker can be a combine both leakage and tampering attacks.

To get any non-trivial security for an implementation, some degree of security against both, leakage and tampering attacks, is necessary: no security can be achieved if the device just leaks the entire secret state, and no security can be achieved if the adversary can tamper the computation to the extent that she can simply replace the original computation with a computation that outputs the entire secret state. The cryptographic community has recently seen a flurry of work on cryptographic systems with leakage-resilience properties [26, 22, 17, 1, 13, 28, 23, 2, 16, 15, 31]. On the other hand, we are aware of only two works which formally study security against tampering attacks: the work of Genaro et al. [18] and Ishai et al. [21].[5] Our model of tamper-resilience is similar to that of [18] and we first describe this model and our results. We then we compare this to the results (and model) of [18, 21].

Model of Tamper-Resilient Security. Following Gennaro et al. [18], we consider the problem of constructing a secure device by combining secret "leakage-proof" but *not* "tamper-proof" memory with a public "tamper-proof" *and* "leakage-proof" circuit. Thus, only the parts of the device that are fixed and publicly known cannot be tampered with. The advantage of this approach is that one only has to manufacture some fixed tamper proof circuitry, but the memory – which must be read/written constantly and thus is harder to protect – need not be tamper-proof. On the other hand, as discussed in [26, 17], the requirement that all components of the device are completely "leakage-proof" is a very strong one and very hard, if not impossible, to satisfy in practice. Luckily, we can use the above-mentioned recent results on leakage-resilient cryptography to weaken this requirement, and allow some bounded amount of information to leak during computation. Therefore, we are left with the task of securing the tamper-*prone* memory against tampering attacks.

Our Results. We consider an arbitrary system $\langle G, s\rangle$ consisting of a *public functionality* G assumed to be implemented on a "tamper-proof" and "leakage-proof" circuit, and a *secret state* s stored in "leakage-proof" but "tamper-*prone*" memory. The system may be stateful and interactive: a user can invoke it periodically with inputs x by submitting an $\mathsf{Execute}(x)$ command, at which point the system computes $(y, s') \leftarrow G(x, s)$, updates its state to $s := s'$ and outputs y. We call this type of interaction with the system *black-box access*. We also assume that an adversary can, in addition, interact with the system by issuing $\mathsf{Tamper}(f)$ commands, for some tampering

[3] i.e. The family $\mathcal{F}$ of functions $f : \{0,1\}^n \to \{0,1\}^n$ which can be written as f_1, f_2 for $f_1, f_2 : \{0,1\}^{n/2} \to \{0,1\}^{n/2}$, such that, for any value $c = (c_1, c_2)$, we have $f(c_1, c_2) = (f_1(c_1), f_2(c_2))$.

[4] Leakage-attacks are often called "passive attacks", whereas tampering (or tampering combined with leakage) attacks are called "active attacks".

[5] Ad-hoc solutions for some particular systems were proposed by [19, 25]. Some of these were subsequently broken [8, 10], highlighting the need for formal study.

functions $f \in \mathcal{F}$, at which point the system state is modified to $s := f(s)$. We call this type of interaction (via both $\mathsf{Execute}$ and Tamper commands) *tampering access* to the system. Our goal is to take any system $\langle G, s\rangle$ and compile it into a new system $\langle G', s'\rangle$, which acts exactly the same as the original in any black-box interaction (i.e. preserves functionality), but is also *tamper resilient*. We formalize the latter, by requiring that, for any adversary $\mathcal{A}$ with tampering access to the compiled system $\langle G', s'\rangle$, there is a simulator $\mathcal{S}$ which only has black-box access to the original system $\langle G, s\rangle$ and "learns" the same information as $\mathcal{A}$. In other words, tampering-access in the compiled system does not give $\mathcal{A}$ any advantage over black-box access to the original.

As our main result for tamper-proof security, we show that any coding scheme $(\mathsf{Enc}, \mathsf{Dec})$, which is non-malleable w.r.t. some family $\mathcal{F}$, can be used to compile *any* system $\langle G, s\rangle$ into a system $\langle G^{\mathsf{Enc},\mathsf{Dec}}, \mathsf{Enc}(s)\rangle$ which is tamper-resilient w.r.t. $\mathcal{F}$. The compiled system $\langle G^{\mathsf{Enc},\mathsf{Dec}}, \mathsf{Enc}(s)\rangle$ stores its state in encoded form. As the first step in any computation, the compiled system decodes its state and then runs the original computation with the decoded state and the input. Lastly, at the end of the computation, the compiled system re-encodes its updated state and updates the memory contents with the new state. On a high-level, the compiled system is tamper-resilient since tampering with the encoded state c via a $f \in \mathcal{F}$ produces a predictable result, which is independent of the underlying state s. More precisely, the tampering interaction can be simulated by sampling from the distribution D_f used to define non-malleability: if the outcome is the $\mathsf{same}^\star$ symbol, tampering-access to the compiled system can be simulated by black-box access to the original system, and, if the outcome is some specific value $\tilde{s}$, tampering-access can be simulated by running $\langle G, \tilde{s}\rangle$ independently of the original system.

Comparison to "Algorithmic Tamper-Proof Security" of [18]. As we mentioned, our model of tamper-resilient security, where we assume a "tamper-proof"/"leakage-proof" public circuit and a "leakage-proof" but *not* "tamper-proof" memory was proposed and studied by Gennaro et al. [18]. The main solution of that work consists of using a (strong) *public-key signature scheme* and storing the state s along with a signature $\sigma = \mathsf{Sign}_{sk}(s)$ on memory. In our context, we can think of this solution as using a *keyed* encoding scheme $(\mathsf{Enc}_{sk}, \mathsf{Dec}_{pk})$ where $\mathsf{Enc}_{sk}(s)$ is now a *secret encoding algorithm* which outputs s along with a signature $\sigma = \mathsf{Sign}_{sk}(s)$ under the secret-key sk. The decoding function $\mathsf{Dec}_{pk}(s, \sigma)$ verifies the signature σ using pk, outputs $\perp$ if the signature is invalid, and s otherwise. We compare the above solution with our solution using non-malleable codes.

Limited Functionality: The security notion achieved by the signature-based coding scheme does not satisfy our notion of non-malleability, even for simple/natural function families like bit-wise independent tampering $\mathcal{F}_{\mathsf{BIT}}$. The problem is that a tampering-attacker can learn some limited information about the state s by (for example) setting the first bit of the encoded state (s, σ) to 0. Then the decoded-message is s *if and only if* the first bit of s already was 0, and $\perp$ otherwise (as w.h.p. σ will not be a valid signature). Therefore, the adversary can learn some bits of s via tampering attacks.[6] This issue was already recognized in the work of [18], but it was noted that tamper-resilient security is still achieved for some functionalities such as *signatures and encryption schemes* where leaking logarithmically many bits about the secret key can be tolerated without breaking the security. Unfortunately, even in these special cases, some problems come up: for example, security breaks down if an adversary has tampering-access to *many* signature/encryption devices that use the same secret key, and encryption security breaks down if the adversary can adaptively perform a tampering attack based on the challenge-ciphertext. More generally, the signature-based solution does not provide a general method for making any functionality G tamper-resilient. In contrast, our solution using non-malleable codes gives us strong guarantees *for all functionalities* G, ensuring that a tampering attack can always be simulated.

Secret-keyed Encoding: Another big difference between our solution and that of [18] is that our encoding/decoding procedures are public and un-keyed, while that of [18] requires a secret-keyed encoding procedure (and a public-keyed decoding). This has several implications. Firstly, the use of a secret-keyed encoding means that state of G has to be decided by an outside party (that knows the secret signing-key sk) and cannot be

[6] In fact, using this type of an attack, one can learn up to logarithmically many bits of s with polynomial probability. A "self-destruct" feature, which we will discuss below, will prevent the adversary from launching this attack many times on different bits of the state, ultimately learning the entire secret.

done by the system itself. In particular, this solution will *not* work for stateful functionalities G which need to update their own state. Secondly, the use of keys brings up some key-management problems where the "software" designer (who decides on the state s) needs to coordinate with the hardware designer (who knows the signing key sk). In contrast, our solution has public encoding/decoding and therefore is applicable to stateful functionalities G without requiring key-management.

Assumptions: The signature-based solution requires computational assumptions (one-way functions), while our main constructions of non-malleable codes are information theoretic.

Tampering family $\mathcal{F}$: The main benefit of the signature-based solution is that it allows for the most general family $\mathcal{F}$ of *all efficient tampering functions*. In contrast, all of our solutions (necessarily) only consider smaller classes of tampering functions, namely those for which non-malleable codes exist. In this aspect, our result is weaker but still well motivated since tampering-attacks are unlikely to be "too complicated". To the best of our knowledge, bit-wise independent tampering (considered in Section 4) already covers the majority of real-world tampering attacks that have been demonstrated in practice.

Self-Destruct/Self-Update: The ATP model of [18] assumes a *"self-destruct"* feature, which means that whenever the memory content decodes to $\perp$ (i.e. is not of the form (s, σ) where σ is a valid signature of s), the device is "destroyed". This can be done by overwriting the memory content with some dummy value, say the all-zero string, or setting some (tamper-proof) flag that prevents future use of the device.

Our construction of a tamper-proof systems via non-malleable codes has a similar self-destruct feature implicitly built into it: the system freshly re-encodes its state after each invocation, where, if the decoding gives $\perp$, the re-encoded value is set to a dummy all-zero string. For functionalities $G(.,.)$ which do not update their state, this requirement imposes some overhead as memory must be written to.

It would be desirable to have a transformation achieving tamper-proof security w.r.t. to some class $\mathcal{F}$ from codes which are non-malleable w.r.t. $\mathcal{F}$ (or some similar notion), where also the transformed system is stateless. We leave this problem for future work. Here, we would only like to mention that our construction of non-malleable codes for bit-wise independent tampering ($\mathcal{F}_{\mathsf{BIT}}$) has some particular properties[7] which allow us to prove that our transformation (applied to stateless functionalities) remains tamper-proof even if we simply omitt updating the sate. (We then need an explicit self-destruct as described above.)

Comparison to [21]. Ishai et al. [21] build a general circuit compiler where an adversary, who can tamper with the wires of a compiled circuit, does not learn any more information than an adversary that only has black-box access to the original circuit. The transformation uses techniques from multi-party computation and is quite delicate. The model considered in that work is stronger as ours as it allows tampering of the "circuit" and "memory" (essentially, memory corresponds to a subset of the wires associated with the state), whereas we only consider tampering of "memory" but assume the circuit is tamper-proof. However, in this very general setting, the work of [21] only considers a very limited tampering-function family that modifies a bounded subset of the wires between each invocation (to tolerate tampering of t wires per invocaton, the compiler will blow up the cirucit size by a factor of at least t). When restricted to memory, this would be analogous to standard error-correcting codes which provide security against a bounded number of individual bit modification. In contrast, we achieve security for much richer families $\mathcal{F}$ of tampering functions. It is an interesting open problem to combine the two approaches and achieve tamper-proof security for circuits as [21], but for larger families $\mathcal{F}$, and, ideally for all families for which there exist non-malleable codes.

[7] Informally, given some history $f_1, \ldots, f_{i-1}$ of tampering functios that were applied to some unknown codeword, we can compute (assuming the codeword is still valid) the probabilities that applying a tampering function f_i will (1) produce an invalid codeword (2) leave the encoded message unchanged or (3) produce an encoding of a different message. In the lats case we can also output a codeword having the right distribution.

2 Preliminaries & Notation

For a randomized function g, we write $g(x;r)$ to denote the value of g on input x using randomness r. Sometimes we don't want to make the randomness explicit and only write $g(x)$ to denote the random variable $g(x;R)$ for a uniformly random R. For a random variable X we use the notation $x \leftarrow X$ to denote that x is sampled randomly according to X. For a set S we use the notation $s \leftarrow S$ to denote that s is sampled uniformly at random from S.

The *Hamming weight* of a binary string $x \in \{0,1\}^n$, denoted $w_H(x)$ is the number of 1's in x. The *Hamming distance* of two strings x, x', denoted $d_H(x,x') \overset{\text{def}}{=} w_H(x - x')$ is the number of positions in which they differ.

The *Statistical Distance* of two random variables X_0, X_1 with support $\mathcal{X}$ is

$$\mathsf{SD}(X_0, X_1) \overset{\text{def}}{=} 1/2 \sum_{x \in \mathcal{X}} |P_{X_0}(x) - P_{X_1}(x)|.$$

If the statistical distance between X_0 and X_1 is negligible (as a function of some well-defined security parameter), we say that X_0 and X_1 are statistically indistinguishable and write $X_0 \approx X_1$. We write $X_0 \equiv X_1$ if they are identically distributed.

3 Non-Malleable Codes

Definition 3.1 (Coding Scheme). *A coding scheme consists of two functions: a* randomized *encoding function* $\mathsf{Enc} : \{0,1\}^k \to \{0,1\}^n$*, and* deterministic *decoding function* $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k \cup \{\perp\}$ *such that, for each* $s \in \{0,1\}^k$*,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(s)) = s] = 1$ *(over the randomness of the encoding algorithm).*

We can also talk about coding-scheme ensembles, parametrized by a message length $k \in \mathbb{N}$ and security-parameter $\lambda \in \mathbb{N}$, so that, for each choice of k, λ, the ensemble contains an efficient code with $n = n(k, \lambda)$. We will often assume such ensembles implicitly, but will try not to complicate the exposition with additional notation.

We now define non-malleability w.r.t. some family $\mathcal{F}$ of tampering functions. We want to say that a code is non-malleable, if the result of tampering with a codeword is independent of the encoded message. To do so, we require that for each $f \in \mathcal{F}$ there is a universal distribution D_f which, *for all* s, indicates what the likely outcomes are when applying a tampering function f to a (random) encoding of s. The distribution D_f only gets (black-box) access to $f(.)$, but does not get s. We allow D_f to output a special $\mathsf{same}^\star$ symbol, to indicate that tampering via f does not change the initial encoded message (without needing to commit to what that message is).

Definition 3.2 (Non-Malleability). *Let* $\mathcal{F}$ *be some family of tampering functions. For each* $f \in \mathcal{F}$ *and* $s \in \{0,1\}^k$*, define the tampering-experiment*

$$\mathsf{Tamper}_s^f \overset{\text{def}}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \textit{Output: } \tilde{s}. \end{array} \right\}$$

which is a random-variable over the randomness of the encoding function Enc*. We say that a coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is* non-malleable w.r.t. $\mathcal{F}$ *if for each* $f \in \mathcal{F}$*, there exists a distribution* D_f *over* $\{0,1\}^k \cup \{\perp, \mathsf{same}^\star\}$*, such that, for all* $s \in \{0,1\}^k$*, we have:*

$$\mathsf{Tamper}_s^f \approx \left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \textit{Output } s \textit{ if } \tilde{s} = \mathsf{same}^\star, \textit{ and } \tilde{s} \textit{ otherwise.} \end{array} \right\} \tag{1}$$

and D_f *is efficiently samplable given oracle access to* $f(\cdot)$*. Here "*$\approx$*" can refer to statistical or computational indistinguishability. In case of statistical indistinguishability, we say the scheme has exact-security* ε*, if the statistical distance above is at most* ε*.*

We also define a slightly stronger notion, called *strong non-malleability*, which essentially says that, whenever the codeword is actually modified to $\tilde{c} \neq c$, the decoded-message $\tilde{s}$ is independent of s. This is in contrast to the plain notion of non-malleability where some modifications of the codeword c could preserve the value of the contained message $\tilde{s} = s$. The stronger notion of non-malleability is somewhat simple to define.

Definition 3.3 (Strong Non-Malleability). *Let $\mathcal{F}$ be a family of functions. We say that an coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is* strongly non-malleable w.r.t. $\mathcal{F}$ *if for any $s_0, s_1 \in \{0,1\}^k$ and any $f \in \mathcal{F}$, we have:*

$$\mathsf{StrongNM}^f_{s_0} \approx \mathsf{StrongNM}^f_{s_1} \quad \textit{where} \quad \mathsf{StrongNM}^f_s \overset{def}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \textit{Output } \mathsf{same}^\star \textit{ if } \tilde{c} = c, \textit{ and } \tilde{s} \textit{ otherwise.} \end{array} \right\} \tag{2}$$

Here "$\approx$" can refer to statistical, or computational indistinguishability. In case of statistical indistinguishability, we say the scheme has exact-security ε, if the statistical distance is at most ε.

As implied by our naming, strong non-malleability is a strictly stronger property than (regular) non-malleability.

Theorem 3.1. *If a coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is* strongly *non-malleable w.r.t. to some family $\mathcal{F}$, then it is non-malleable w.r.t. $\mathcal{F}$ with the same exact security.*

Proof. Set $D_f = \mathsf{StrongNM}^f_{s_0}$ for some constant $s_0 \in \{0,1\}^k$. Then, for each $s \in \{0,1\}^k$, $D_f \approx \mathsf{StrongNM}^f_s$ and so

$$\left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^\star, \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\} \approx \left\{ \begin{array}{c} \tilde{s} \leftarrow \mathsf{StrongNM}^f_s \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^\star, \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\}$$
$$\equiv \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \text{Output: } \tilde{s}. \end{array} \right\}.$$

□

We also note that our notion of *strong* non-malleability bears a very close resemblance to the standard notion of non-malleable encryption as defined in [14]. On the other hand, our default notion of non-malleability (where tampering may leave the message unchanged) is closer, in analogy, to a relaxed notion of cryptographic non-malleability called "replayable" non-malleable security, defined in [7]. Indeed, in that work it was already noted that for many cryptographic applications this relaxed notion of non-malleable encryption suffices. When it comes to non-malleable codes, we chose to make the relaxed notion the default one since:

- Having a code which ensures that tampering *preserves* the message entirely seems like a desirable feature rather than a bug when designing coding schemes. In particular, we view non-malleability as a weakening of error-correction. This view is only valid with the relaxed notion of non-malleability, since error-correction does *not* imply *strong non-malleability*.
- Our default relaxed notion of non-malleability is sufficient for our application to tamper-resilience.

The main disadvantage of our default notion of non-malleability (compared to strong non-malleability) seems to be a more complicated definition, which is harder to state and understand. Inspired by [7], we also give an alternative definition of our default notion of non-malleability in Appendix A, and prove the equivalence of the two definitions.

4 Bit-Wise Independent Tampering

With $\mathcal{F}_{\mathsf{BIT}}$ we denote the family which contains all tampering functions that tamper every bit independently. Formally, this family contains all functions $f : \{0,1\}^n \to \{0,1\}^n$ that are defined by n functions $f_i : \{0,1\} \to \{0,1\}$ (for $i = 1, \ldots, n$) as $f(c_1, \ldots, c_n) = (f_1(c_1), \ldots, f_n(c_n))$. Note that there are only 4 possible choices for each f_i (i.e. how to modify a particular bit) and we name these "set to 0", "set to 1", "flip", "keep" where the meanings should be intuitive. We call the above family the *bit-wise independent tampering family*.

As discussed in the introduction, standard notions of "error correction" and "error detection" trivially cannot be achieved against this class. Note that the family $\mathcal{F}_{\mathsf{err}}$ of constant-error functions which was studied in the context of AMD codes is a subset of $\mathcal{F}_{\mathsf{BIT}}$.[8] Bitwise independent tampering is interesting in the context of tamper-proof security, as most practical attacks tamper with each bit in the computing circuit and/or the memory individually. In Section Section 6 we'll show how non-malleable codes can be used to protect any functionality against tampering of the memory.

[8] They are defined as the functions in $\mathcal{F}_{\mathsf{BIT}}$, but where the f_i can only be "keep" or "flip", but not "set to 0" or "set to 1".

4.1 Construction

Before proceeding with the construction, we first define two primitives which are interesting in their own right: a type of secret sharing scheme over bits that also has a large distance, and AMD codes which we mentioned previously in the introduction.

Definition 4.1 (LECSS Schemes). *Let* (E, D) *be a coding scheme with source-messages* $s \in \{0,1\}^k$ *and codewords* $c \in \{0,1\}^n$. *We say that the scheme is a* (d,t)-linear error-correcting secret-sharing (LECSS) *scheme if the following properties hold:*

- Linearity: *For all* $c \in \{0,1\}^n$, *such that* $\mathsf{D}(c) \neq \perp$, *all* $\Delta \in \{0,1\}^n$, *we have*

$$\mathsf{D}(c+\Delta) = \begin{cases} \perp & \textit{if } \mathsf{D}(\Delta) = \perp \\ \mathsf{D}(c) + \mathsf{D}(\Delta) & \textit{otherwise} \end{cases}$$

- Distance d*: For all non-zero* $\tilde{c} \in \{0,1\}^n$ *with hamming-weight* $w_H(\tilde{c}) < d$, *we have* $\mathsf{D}(\tilde{c}) = \perp$.
- Secrecy t*: For any fixed* $s \in \{0,1\}^k$, *we define the random variables* $C = (C_1, \ldots, C_n) = \mathsf{E}(s)$, *where* C_i *denotes the bit of* C *in position* i *(and where the randomness comes from the encoding procedure). Then the random variables* $\{C_i\}_{1 \leq i \leq n}$, *are individually uniform over* $\{0,1\}$ *and* t*-wise independent.*

Definition 4.2 (AMD Codes [12]). *Let* (A, V) *be a coding scheme with* $\mathsf{A} : \{0,1\}^k \to \{0,1\}^n$. *We say that* (A, V) *is a* ρ-secure algebraic manipulation detection (AMD) code *if for all* $s \in \{0,1\}^k$ *and all non-zero* $\Delta \in \{0,1\}^n$ *we have* $\Pr[\mathsf{V}(\mathsf{A}(m) + \Delta) \neq \perp] \leq \rho$, *where the probability is over the randomness of the encoding.*

Our main theorem of this section shows that, by composing an AMD code with a LECSS scheme, we get a non-malleable code for the bit-wise independent tampering family $\mathcal{F}_{\mathsf{BIT}}$.

Theorem 4.1. *Let* (E, D) *be a* (d,t)*-LECSS with* $\mathsf{E} : \{0,1\}^k \to \{0,1\}^{n'}$ *and distance* $d > n/4$. *Let* (A, V) *be a* ρ*-secure AMD code where* $\mathsf{A} : \{0,1\}^{k'} \to \{0,1\}^n$ *and* $k' = n'$. *Define the composed code* $(\mathsf{Enc}, \mathsf{Dec})$ *by*

$$\mathsf{Enc}(s) \overset{def}{=} \mathsf{E}(\mathsf{A}(s)) \ \ , \ \ \mathsf{Dec}(c) \overset{def}{=} \begin{cases} \perp & \textit{if } \mathsf{D}(c) = \perp \\ \mathsf{V}(\mathsf{D}(c)) & \textit{otherwise} \end{cases}$$

Then $(\mathsf{Enc}, \mathsf{Dec})$ *is non-malleable with respect to the family* $\mathcal{F}_{\mathsf{BIT}}$ *with exact security* $\varepsilon \leq \max\left(\ \rho\ ,\ 2^{-\Omega(t)}\ \right)$.

The proof of the theorem appears in Appendix B, and is fairly delicate. Below, we sketch only the high-level idea. To prove the theorem, we must show that for each $f \in \mathcal{F}_{\mathsf{BIT}}$, there is a distribution D_f which satisfies Definition 3.2. To do so, we look at how many of the component function f_i that make up f, are of the form "set to 0" or "set to 1". If this number is too small (less than t) then tampering via f is really equivalent to adding some offset Δ to $\mathsf{A}(s)$, from a distribution which is independent of $\mathsf{A}(s)$. Therefore, there is some universal probability with which $\Delta = 0$ (in which case D_f should output $\mathsf{same}^\star$) and $\Delta \neq 0$ (in which case D_f should output $\perp$ since the security of the AMD codes ensure that such modifications are likely to be detected). On the other hand, if the number of f_i that are "set to 0"/"set to 1" is too high (more than $n-t$) then the values in the positions of $\tilde{c} = f(\mathsf{Enc}(s))$ that are not "set" are uniformly random. Therefore D_f can sample the value of $\tilde{c}$ independently of s. Lastly, we show that when the number of f_i that are "set to 0"/"set to 1" is in between t and $n-t$, then the erroneous codeword decodes to $\perp$ with overwhelming probability. This is where we need the distance of the LECSS to be $d > n/4$.[9] Essentially, we show that in this case there exists some codeword c^* such that the tampered value $\tilde{c}$ is neither likely to math c^* exactly nor is it likely to be be far-away enough from it to be a valid codeword under the LECSS scheme.

We note that the idea of composing AMD codes together with linear secret-sharing schemes also appeared explicitly in [12] (and implicitly in [29, 6, 30]) in the context of "robust secret sharing", but the application in this work and the main ideas behind our proof are very different.

[9] Interestingly, the requirement that $d > n/4$ is not an artifact of the proof-technique but an optimal bound at this level of generality. In particular, there are examples of LECSS constructions with $d < n/4$ for which our construction would not be secure.

4.2 Instantiating the Construction

To actually instantiate a construction of the non-malleable code defined above, we need constructions of AMD codes and of LECSS-schemes. For the former, very efficient constructions of AMD codes were given by [12] where, to achieve security $\rho \leq 2^{-\lambda}$, the encoding has an additive overhead of roughly 2λ bits. On the other hand, LECSS-schemes have not been explicitly defined or constructed in literature. We note that there is a clear connection between LECSS-schemes, linear ramp secret-sharing schemes, and error-correcting codes. Ramp secret-sharing schemes have mostly been considered for larger alphabet sizes (e.g. Shamir Secret Sharing [34]) and not over bits, an exception is the work of Pueyo et al. [32]. However, no known constructions of ramp secret-sharing schemes over bits achieve distance $d > n/4$ over bits, as required for our application. In Appendix C, we show how to instantiate LECSS schemes with large d, t *efficiently*, based on random linear error-correcting codes (recall, that our construction does not require efficient error-decoding). The main ideas for this construction are based on the work of Chen et al. [9], which shows how to relate the secrecy t of a LECSS to the dual-distance of an underlying error correcting code. In addition, it shows that a random linear error-correcting code will (with overwhelming probability) achieve a large distance d and a large dual distance yielding a large t. Combining the analysis in Appendix C with Theorem 4.1 we get the following parameters.

Theorem 4.2. *For any constant* $\delta > 0$*, any* $n \in \mathbb{N}$*, there exist* efficient *non-malleable codes w.r.t. the family* $\mathcal{F}_{\mathsf{BIT}}$*, with codeword size* n*, message size* $k \geq (1 - H(1/4) - \delta)n$ *and security* $\varepsilon = 2^{-\Omega(n)}$*, where* $H(x) \overset{def}{=} -x\log_2(x) - (1-x)\log_2(1-x)$ *is the Shannon entropy function. In particular, the rate of the code can be made to approach* $k/n \approx (1 - H(1/4)) \approx .1887$*. Moreover, there is an efficient procedure which, given* k *and* n*, outputs a description of such a code with probability* $1 - 2^{-\Omega(n)}$.

4.3 Improving Efficiency

One issue with our construction is that the rate k/n of the scheme is less than $1/5$ and therefore relatively poor. We now address this issue by showing that *any* non-malleable code w.r.t $\mathcal{F}_{\mathsf{BIT}}$, with any *constant* rate k/n, can be converted into *computationally secure* non-malleable code w.r.t $\mathcal{F}_{\mathsf{BIT}}$, where the new rate k'/n' asymptotically approaches 1. The main idea is to combine an encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ from the previous sections with an authenticated encryption scheme $(\mathsf{AuthEncrypt}, \mathsf{AuthDecrypt})$ in the following way: to encode a message s we take a random key key and store $(\mathsf{Enc}(\mathsf{key}), \mathsf{AuthEncrypt}_{\mathsf{key}}(s))$. The decoding of (c_0, c_1) is straightforward: first compute $\mathsf{key}' = \mathsf{Dec}(c_0)$, and then output $\mathsf{AuthDecrypt}_{\mathsf{key}'}(c_1)$. Security of this construction is shown in the proof of the following theorem (which appears in Appendix D).

Theorem 4.3. *Assuming the existence of one-way functions, there are efficient* computational *non-malleable codes w.r.t the family* $\mathcal{F}_{\mathsf{BIT}}$*, where the rate* k/n *asymptotically approaches* 1.

5 A General Result for Tampering-Function Families of Bounded Size

5.1 A Probabilistic Method Approach

We now show that, for *any* "small enough" function family $\mathcal{F}$, there *exists* a (possibly inefficient) coding scheme which is non-malleable w.r.t. $\mathcal{F}$. Moreover, we show that for a fixed "small enough" function family $\mathcal{F}$, a *random* coding scheme is likely to be non-malleable w.r.t. $\mathcal{F}$ with overwhelming probability. Unfortunately, random coding schemes cannot be efficiently represented, nor is the encoding/decoding function likely to be efficient. Therefore, this result should merely be thought of as showing "possibility" and providing a target that we should then strive to match constructively. Moreover, this result also highlights the difference between "error-correction//detection" and "non-malleability" since a result of this form could not be true for the former notions.

Let us now clarify what we mean by "small enough". The family $\mathcal{F}_{\mathsf{all}}$ of all functions $f : \{0,1\}^n \to \{0,1\}^n$ has size $\log(\log(|\mathcal{F}_{\mathsf{all}}|)) = n + \log(n)$. Clearly, no code is non-malleable w.r.t. the family $\mathcal{F}_{\mathsf{all}}$, since this family includes e.g. the function that decodes the codeword, flips the last bit of the source message, and re-encodes it.

However, we show that for any function family $\mathcal{F}$ of slightly smaller size $\log(\log(|\mathcal{F}|)) < n$, there exist codes that are non-malleable w.r.t. $\mathcal{F}$.

Lastly, let us clarify what we mean by "random" coding scheme. We can define a coding scheme completely by only specifying the decoding function $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k \cup \perp$, which then implicitly defines an encoding function Enc that maps a source message s uniformly at random into the set $\{c \in \{0,1\}^n | \mathsf{Dec}(c) = s\}$. For us, a random coding scheme is a scheme of this type, where the decoding function is chosen *uniformly* at random from all function $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k$; that is, we specify the decoding function by taking each value $c \in \{0,1\}^n$ and letting it decode to a uniformly random source message $s \in \{0,1\}^k$. Note that there are *no* invalid values $c \in \{0,1\}^n$ that decode to $\perp$. See Appendix E for a proof of the following theorem.

Theorem 5.1. *Let $\mathcal{F}$ be any function family consisting of functions $f : \{0,1\}^n \to \{0,1\}^n$. Let $\varepsilon, \rho > 0$ be arbitrary values and $k, n > 0$ be integers such that*

$$n > \log(\log(|\mathcal{F}|)) + 3k + \log(k) + 2\log(1/\varepsilon) + \log(\log(1/\rho)) + 9.$$

Then there exists a strongly non-malleable code w.r.t. $\mathcal{F}$, with k-bit source-messages and n-bit codewords, and exact security ε. Moreover, a uniformly random decoding function $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k$ gives rise to such a code with probability $\geq 1 - \rho$.

5.2 Constructions in the Random Oracle Model

It is not clear what the bound from Theorem 5.1 actually implies. For example it does tell us that non-malleable codes exist with respect to all efficient functions, but this is misleading as we know that *efficient* non-malleable codes (and ultimately we are only interested in such) cannot be non-malleable w.r.t. this class.

Nonetheless, as we'll explain below, the result by the probabilistic method method does give us codes which are non-malleable w.r.t. very general classes of functions in the random oracle model. By Theorem 5.1, a random decoding $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k$ will be non-malleable, thus we could instantiate Dec with a random oracle, but then it is ont obvious how to implement the corresponding encoding Enc efficiently.

Coron et al. [11] show how to construct a permutation (which can be queried from both sides) form a random oracle which is *indifferentiable* from a uniformly random permutation. Thus, instead of a random oracle, we can assume (cf. Proposition 2 in [24]) that we have a uniformly random permutation $\mathcal{P}$ over $\{0,1\}^n$. We then define the encoding $\mathsf{Enc}(x; r) \overset{\text{def}}{=} \mathcal{P}(x, r)$ for messages $x \in \{0,1\}^k$ and randomness $r \in \{0,1\}^{n-k}$. The decoding of $c \in \{0,1\}^n$ are simply the first k bits of $\mathcal{P}^{-1}(c)$.[10]

Thus, we can get a non-malleable code w.r.t. any $\mathcal{F}$ as in Theorem 5.1 relative to a random oracle. It is important to note that here the tampering functions $f \in \mathcal{F}$ cannot have access to the random permutation $\mathcal{P}$.[11] For this reason, this should *not* be considered a result in the random oracle model, where one usually assumes that the oracle is accessible by all parties.[12] To get a result in the random oracle model, we must give all parties, *including the tampering functions*, access to the random oracle.

Definition 5.1 (Closed Class of Tampering Functions.). *Consider a class $\mathcal{F}$ of oracle tampering functions, that is, a function $f \in \mathcal{F}$ can have special oracle gates. With $f^{\mathcal{O}}$ we denote f where the oracle queries are answered by an oracle $\mathcal{O}$. For some (compatible) oracle $\mathcal{O}$, $\mathcal{F}^{\mathcal{O}}$ denotes the class of functions $f^{\mathcal{O}}$ where $f \in \mathcal{F}$. We say $\mathcal{F}$ is* closed, *if for every oracle $\mathcal{O}$ and every $f^{\mathcal{O}} \in \mathcal{F}^{\mathcal{O}}$, there exists an $f' \in \mathcal{F}^{\mathcal{O}}$ (where f' has no oracle gates) such that f' and $f^{\mathcal{O}}$ are the same function.*

[10] The distribution on Dec is not exactly the distribution of a random decoding function as analyzed in Theorem 5.1, as now we are using a random permutation instead of a random function. But when considering efficient adversaries that only make a polynomial (in n) number of queries, this is irrelevant as the distance in the distributions will be exponentially small.

[11] Otherwise e.g. the class $\mathcal{F}$ containing the single function f, where $f^{\mathcal{P}}(c)$ decodes c and then re-encodes the message with the last bit flipped, is a counterexample.

[12] Security proofs in the random oracle model only exploit the fact that an exponential amount of uniform randomness is efficiently accessible by all parties, not that some entities (like, in our case, tampering functions) cannot access this randomness.

Clearly, if $\mathcal{F}$ is closed, then the subset $\hat{\mathcal{F}} \subset \mathcal{F}$ of functions with no oracle gates makes up the entire class, i.e. $\hat{\mathcal{F}} = \mathcal{F}^O$ for any $\mathcal{O}$. So when considering a code that is non-malleable w.r.t. to a closed class $\mathcal{F}$ relative to a random oracle model, it doesn't make a difference whether we give the tampering functions access to the random oracle or not. Summarizing the above discussion, we get the following theorem.

Theorem 5.2. *Let $\mathcal{F}, \varepsilon, \delta, n, k$ be as in Theorem 5.1, but where $f \in \mathcal{F}$ can have oracle gates. If $\mathcal{F}$ is closed, then there exists an efficient non-malleable code w.r.t. to $\mathcal{F}$ in the random oracle model with exact security $\varepsilon + \delta + O(q^{16}/2^n)$. (where q is the number of oracle queries made by the adevrsary.)*

The extra $O(q^{16}/2^n)$ error term comes from the bound proven in [11], it can be avoided if we directly assume access to a uniformly random permutation (this is called the ideal-cipher model). The distance in the distributions as mentioned in footnote 10 is also captured by this term.

Examples of Closed Classes. An interesting and natural closed family consists of functions that tamper each half of the codeword independently: that is $f \in \mathcal{F}_{\mathsf{half}}$ if we can write $f : \{0,1\}^n \to \{0,1\}^n$ as $f(c_1, c_2) \overset{\text{def}}{=} (f_1(c_1), f_2(c_2))$ for $f_1, f_2 : \{0,1\}^{n/2} \to \{0,1\}^{n/2}$. This family has size $\log(\log(|\mathcal{F}_{\mathsf{half}}|)) = n/2 + \log(n)$ as required by Theorem 5.1, and thus by Theorem 5.2, there exist non-malleable codes for $\mathcal{F}_{\mathsf{half}}$ with rate $k/n \approx 1/6$.

More generally, for any $\delta < 1$ we can consider the class $\mathcal{F}_\delta$ where $f \in \mathcal{F}_\delta$ if every bit of $f(c)$ depends only on a subset of at most $\lfloor \delta n \rfloor$ bits of c. ($\mathcal{F}_{\mathsf{half}}$ just discussed is thus a subset of $\mathcal{F}_{0.5}$.) The size of this class is $\log(\log(|\mathcal{F}_\delta|) \leq \delta \cdot n + 2\log(n)$ as required by Theorem 5.1.

6 Tamper-Resilient Security

In this section, we formally define the notion of tamper-resilient security as outlined in the introduction. We consider some *interactive stateful system* $\langle G, s \rangle$ consisting of a *pubic* (possibly randomized) functionality $G : \{0,1\}^u \times \{0,1\}^n \to \{0,1\}^v \times \{0,1\}^n$ and *secret* initial state $s \in \{0,1\}^n$. We consider two ways of interacting with the system:

$\mathsf{Execute}(x)$**:** A user can provide the system with $\mathsf{Execute}(x)$ queries, for $x \in \{0,1\}^u$, in which case the system computes $(y, s') \leftarrow G(x, s)$, updates the state of the system to $s := s'$ and outputs y.

$\mathsf{Tamper}(f)$**:** We also consider tampering attacks against the system, modeled by $\mathsf{Tamper}(f)$ commands, for functions $f : \{0,1\}^n \to \{0,1\}^n$. Upon receiving such command, the system state is set to $s := f(s)$.

Honest users only interact with a system via Execute queries. As we discussed in the introduction, an attacker that can also interact with the system via Tamper queries can potentially learn significantly more about the secret state, even recover it entirely. Therefore, we would like to have a general method for securing systems against tampering attacks, so that the ability to issue Tamper queries (at least for functions f in some large family $\mathcal{F}$) cannot provide the attacker with additional information. We show how to use non-malleable codes for this purpose. First, we define what it means to *harden* a functionality G via a code $(\mathsf{Enc}, \mathsf{Dec})$.

Definition 6.1. *Let $(\mathsf{Enc}, \mathsf{Dec})$ be any coding scheme with k-bit messages and n-bit codewords. Let $G : \{0,1\}^u \times \{0,1\}^k \to \{0,1\}^v \times \{0,1\}^k$ be an arbitrary functionality with k-bit state. We define the* hardened functionality *$G^{\mathsf{Enc},\mathsf{Dec}} : \{0,1\}^u \times \{0,1\}^n \to \{0,1\}^v \times \{0,1\}^n$ to be the functionality (with n-bit state) which, on input $(x, c) \in \{0,1\}^u \times \{0,1\}^n$, computes $s \leftarrow \mathsf{Dec}(c)$ and checks if $s \overset{?}{=} \bot$. If so, it outputs a dummy-value $(0^u, 0^n)$. Otherwise it computes $(y, s') \leftarrow G(x, s)$ and outputs $(y, \mathsf{Enc}(s'))$.*

It is easy to see that the original system $\langle G, s \rangle$ and the hardened system $\langle G^{\mathsf{Enc},\mathsf{Dec}}, \mathsf{Enc}(s) \rangle$ always behave exactly the same with respect to any series of Execute commands. Of course, they do *not* act the same with respect to Tamper commands, and that's the point; we wish to show that the hardened system $\langle G^{\mathsf{Enc},\mathsf{Dec}}, \mathsf{Enc}(s) \rangle$ is tamper-resilient and renders tampering attacks useless. We formally define this property next, as essentially saying that for any adversary $\mathcal{A}$ that interacts with the hardened system $\langle G^{\mathsf{Enc},\mathsf{Dec}}, \mathsf{Enc}(s) \rangle$ via Execute and Tamper queries, there is a simulator $\mathcal{S}$ that interacts with the original system $\langle G, s\rangle$ *only via* Execute queries and learns the same information as $\mathcal{A}$. In particular we say that a coding scheme $(\mathsf{Enc}, \mathsf{Dec})$ is *tamper simulatable* if it can be used to securely harden *any* system $\langle G, s\rangle$ into a tamper-resilient system $\langle G^{\mathsf{Enc},\mathsf{Dec}}, \mathsf{Enc}(s)\rangle$.

Definition 6.2 (Tamper-Simulatability). *A coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *with* k*-bit messages and* n*-bit codewords is* tamper-simulatable w.r.t. a function family $\mathcal{F}$*, if there exists a simulator* $\mathcal{S}$ *such that, for any functionality* G *with* k *bit state, any adversary* $\mathcal{A}$*, and any initial state* $s \in \{0,1\}^k$ *we have*

$$\mathsf{TamperInteract}(\mathcal{A}, \mathcal{F}, \langle G^{\mathsf{Enc},\mathsf{Dec}}, \mathsf{Enc}(s)\rangle) \approx \mathsf{BBInteract}(\mathcal{S}, \langle G, s\rangle)$$

where TamperInteract *and* BBInteract *are defined as follows.*

$\mathsf{TamperInteract}(\mathcal{A}, \mathcal{F}, \langle G, s\rangle)$**:** *The adversary* $\mathcal{A}$ *interacts with the system* $\langle G, s\rangle$ *for arbitrarily many rounds of interactions where, in each round:*

1. *The adversary can "tamper" by executing a* $\mathsf{Tamper}(f)$ *command against the system, for some* $f \in \mathcal{F}$*.*
2. *The adversary runs an* $\mathsf{Execute}(x)$ *query for some* $x \in \{0,1\}^u$*, and receives the output* y*.*

The output of the game consists of the output of the adversary $\mathcal{A}$ *at the end of the interaction, along with all of the execute queries* $x_1, x_2, \ldots$

$\mathsf{BBInteract}(\mathcal{S}, \langle G, s\rangle)$**:** *The simulator interacts with the system* $\langle G, s\rangle$ *for arbitrarily many rounds of interaction where, in each round, it runs an* $\mathsf{Execute}(x)$ *query for some* $x \in \{0,1\}^u$ *and receives the output* y*. The output of the game consists of the output of the simulator* $\mathcal{S}$ *at the end of the interaction, along with all of the execute-query inputs* x*.*

The simulator $\mathcal{S}$ *gets black-box access to the adversary* $\mathcal{A}$*, the functionality* $G(\cdot,\cdot)$*, and any tampering functions* $f(\cdot)$ *produced by* $\mathcal{A}$ *(but does not get the state* s*). Here* $\approx$ *can refer to statistical, or computational indistinguishability.*

Note that the TamperInteract and BBInteract games include all of the "execute" queries submitted by $\mathcal{A}$ and $\mathcal{S}$ in their outputs. This prevents the simulator from submitting queries which $\mathcal{A}$ would not have asked: e.g. if $\mathcal{A}$ performs tampering attacks but only queries the system on some three inputs x_1, x_2, x_3 then $\mathcal{S}$ must simulate this without tampering queries *by only querying its system on the same inputs* x_1, x_2, x_3. Also, note that in the definition of the TamperInteract game, we only allow the adversary to submit *one* tampering query $\mathsf{Tamper}(f)$ in each round. For families $\mathcal{F}$ which are closed under composition (i.e. for any $f_1, f_2 \in \mathcal{F}$, we have $f_1 \circ f_2 \in \mathcal{F}$) this is without loss of generality, as submitting several tamper queries in a row can always be subsumed via a single query. We now show that a code $(\mathsf{Enc}, \mathsf{Dec})$ is tamper-simulatable w.r.t. $\mathcal{F}$ if it is non-malleable w.r.t. $\mathcal{F}$. See Appendix E for a proof of the following theorem.

Theorem 6.1. *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be any coding scheme which is non-malleable w.r.t.* $\mathcal{F}$*. Then* $(\mathsf{Enc}, \mathsf{Dec})$ *is also tamper-simulatable w.r.t.* $\mathcal{F}$*.*

7 Conclusions

In this work, we introduce and explore the notion of non-malleable codes and present several constructions. We show the application of such codes to the problem of tamper-resilience. There are several interesting open problems left by this work. Firstly, we would like to design efficient non-malleable codes for more function families $\mathcal{F}$. For example, we would like to build such codes for the families as discussed in Section 5.2 without relying on random oracles. Another interesting open problem would be to figure out the optimal rates of such codes for various families. For example, even our existential result only achieved a rate of at most $1/3$ (for even small function families $\mathcal{F}$) and our efficient construction for bit-wise independent tampering only achieved a rate of at most .1887. It would be interesting to see if these can be improved or if there are some inherent lower bounds.

8 Acknowledgements

We would like to thank Ronald Cramer, Yevgeniy Dodis, Eike Kiltz, Swastik Kopparty, Ignacio Cascudo Pueyo and Madhu Sudan for the many helpful discussions and comments they provided for this project.

References

[1] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC 2009: 6th Theory of Cryptography Conference*, Lecture Notes in Computer Science, pages 474–495. Springer-Verlag, Berlin, Germany, 2009.

[2] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, Lecture Notes in Computer Science, pages 36–54. Springer-Verlag, Berlin, Germany, August 2009.

[3] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, Berlin, Germany, December 2000.

[4] Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science*, pages 276–287. IEEE Computer Society Press, November 1994.

[5] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, Berlin, Germany, May 1997.

[6] Sergio Cabello, Carles Padró, and Germán Sáez. Secret sharing schemes with detection of cheaters for a general access structure. *Des. Codes Cryptography*, 25(2):175–188, 2002.

[7] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO*, pages 565–582, 2003.

[8] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag, Berlin, Germany, August 1999.

[9] Hao Chen, Ronald Cramer, Shafi Goldwasser, Robbert de Haan, and Vinod Vaikuntanathan. Secure computation from random error correcting codes. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 291–310. Springer-Verlag, Berlin, Germany, May 2007.

[10] Jean-Sébastien Coron and Louis Goubin. On Boolean and arithmetic masking against differential power analysis. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 231–237. Springer-Verlag, Berlin, Germany, August 2000.

[11] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, Lecture Notes in Computer Science, pages 1–20. Springer-Verlag, Berlin, Germany, August 2008.

[12] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, Lecture Notes in Computer Science, pages 471–488. Springer-Verlag, Berlin, Germany, April 2008.

[13] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *41st Annual ACM Symposium on Theory of Computing*, pages 621–630. ACM Press, 2009.

[14] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552, 1991.

[15] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science*, pages 293–302. IEEE Computer Society Press, 2008.

[16] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy Rothblum. Leakage-resilient signatures. In *TCC 2010: 7th Theory of Cryptography Conference*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2010.

[17] Sebastian Faust, Leonid Reyzin, and Eran Tromer. Protecting circuits from computationally-bounded leakage. Cryptology ePrint Archive, Report 2009/379, 2009. `http://eprint.iacr.org/`.

[18] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer-Verlag, Berlin, Germany, February 2004.

[19] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer-Verlag, Berlin, Germany, August 1999.

[20] Richard W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950.

[21] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer-Verlag, Berlin, Germany, May / June 2006.

[22] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer-Verlag, Berlin, Germany, August 2003.

[23] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *Advances in Cryptology – ASIACRYPT 2009*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, December 2009.

[24] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer-Verlag, Berlin, Germany, February 2004.

[25] Thomas S. Messerges. Securing the AES finalists against power analysis attacks. In Bruce Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer-Verlag, Berlin, Germany, April 2000.

[26] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer-Verlag, Berlin, Germany, February 2004.

[27] Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. In *SWAT*, pages 1–3, 2008.

[28] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, August 2009.

[29] Wakaha Ogata and Kaoru Kurosawa. Optimum secret sharing scheme secure against cheating. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 200–211. Springer-Verlag, Berlin, Germany, May 1996.

[30] Wakaha Ogata, Kaoru Kurosawa, Douglas R. Stinson, and Hajime Saido. New combinatorial designs and their applications to authentication codes and secret sharing schemes. *Discrete Mathematics*, 279(1-3):383–405, 2004.

[31] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, Lecture Notes in Computer Science, pages 462–482. Springer-Verlag, Berlin, Germany, April 2009.

[32] Ignacio Cascudo Pueyo, Hao Chen, Ronald Cramer, and Chaoping Xing. Asymptotically good ideal linear secret sharing with strong multiplication over *ny* fixed finite field. In *CRYPTO*, pages 466–486, 2009.

[33] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. In *SODA*, pages 331–340, 1993.

[34] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[35] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

A Alternative Definition of Non-Malleable Codes

In this section we propose an alternative definition of non-malleability, inspired by an analogous definition for replayable CCA secure encryption in [7], which we show to be equivalent to our default notion. Essentially, the alternative definition says that we cannot distinguish the outcome of the tampering experiment that starts with source-message s_0 from that which starts with source-message s_1, *unless* the outcome is one of $\{s_0, s_1\}$.

Definition A.1 (Alternative-Non-Malleability). *Let $\mathcal{F}$ be a family of functions. We say that an coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is* alternative-non-malleable w.r.t. $\mathcal{F}$ *if for any $s_0, s_1 \in \{0,1\}^k$ and any $f \in \mathcal{F}$, we have:*

$$\mathsf{AltNM}^f_{s_0,s_1}(0) \approx \mathsf{AltNM}^f_{s_0,s_1}(1)$$

where we define the two experiments by

$$\mathsf{AltNM}^f_{s_0,s_1}(b) \stackrel{def}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s_b), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \textit{Output } \mathsf{same}^\star \textit{ if } \tilde{s} \in \{s_0, s_1\}, \textit{ and } \tilde{s} \textit{ otherwise.} \end{array} \right\}$$

Here "$\approx$" can refer to statistical, or computational indistinguishability. In case of statistical indistinguishability, we say the scheme has exact-security ε, if the statistical distance is at most ε.

We now show the equivalence of the two definitions of non-malleability. This equivalence only holds when the domain size of source-messages is super-polynomial, as otherwise alternative-non-malleability may be a weaker notion. The following theorem makes this precise.

Theorem A.1. *For any function family $\mathcal{F}$:*

1. *If a coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is non-malleable w.r.t $\mathcal{F}$ then it is alternatively-non-malleable w.r.t $\mathcal{F}$. In case of statistical indistinguishability, if the scheme is non-malleable with security ε, then it is alternatively-non-malleable with security 2ε.*
2. *If a coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is alternatively-non-malleable w.r.t $\mathcal{F}$,* and *its domain size is super-polynomial (i.e. $k = \omega(\log(\lambda))$ where λ is a security parameter), then it is also non-malleable w.r.t. $\mathcal{F}$. In case of statistical indistinguishability, if the code is alternatively non-malleable with security ε, then it is non-malleable with security $\varepsilon + 2^{-k}$.*
3. *When the domain-size of the coding scheme is super-polynomial, a coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is non-malleable* if and only if *it is alternatively-non-malleable.*

Proof. Let's start with the first part of the theorem. Assume $(\mathsf{Enc}, \mathsf{Dec})$ is non-malleable w.r.t. $\mathcal{F}$. Then, for each $f \in \mathcal{F}$ there exists some distribution D_f satisfying the definition of non-malleability. Therefore, for any $s_0, s_1 \in \{0,1\}^k$,

$$\begin{aligned} \mathsf{AltNM}^f_{s_0,s_1}(0) &\equiv \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s_0), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \text{Output } \mathsf{same}^\star \text{ if } \tilde{s} \in \{s_0, s_1\}, \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\} \\ &\approx \left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \text{Output } \mathsf{same}^\star \text{ if } \tilde{s} \in \{s_0, s_1, \mathsf{same}^\star\} \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\} \\ &\approx \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s_1), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \text{Output } \mathsf{same}^\star \text{ if } \tilde{s} \in \{s_0, s_1\}, \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\} \\ &\equiv \mathsf{AltNM}^f_{s_0,s_1}(0) \end{aligned}$$

In case of statistical indistinguishability, each "$\approx$" above hides a factor of ε statistical distance, giving us a total of 2ε.

For the second part of the theorem, assume that $(\mathsf{Enc}, \mathsf{Dec})$ is alternatively-non-malleable w.r.t. $\mathcal{F}$. We construct the distribution D_f as follows:

$$D_f \overset{\text{def}}{=} \left\{ \begin{array}{c} \text{Sample } s \leftarrow \{0,1\}^k, c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \text{Output } \mathsf{same}^\star \text{ if } \tilde{s} = s \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\}$$

In other words, D_f tries running the tampering experiment on a random s and outputs $\mathsf{same}^\star$ if the value remains unchanged. We show that D_f satisfies the definition of non-malleability. For any $f \in \mathcal{F}$, $s_0 \in \{0,1\}^k$:

$$\begin{array}{cl} & \left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \text{Output } s_0 \text{ if } \tilde{s} = \mathsf{same}^\star \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\} \\ \equiv & \left\{ \begin{array}{c} s \leftarrow \{0,1\}^k, \tilde{s} \leftarrow \mathsf{AltNM}^f_{s_0,s}(1) \\ \text{Output } s_0 \text{ if } \tilde{s} = \mathsf{same}^\star \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\} \\ \approx & \left\{ \begin{array}{c} s \leftarrow \{0,1\}^k, \tilde{s} \leftarrow \mathsf{AltNM}^f_{s_0,s}(0) \\ \text{Output } s_0 \text{ if } \tilde{s} = \mathsf{same}^\star \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\} \\ \approx & \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s_0), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \text{Output } \tilde{s}. \end{array} \right\} \end{array}$$

For statistical indistinguishability, the first "$\approx$" hides a factor of ε while the second "$\approx$" (which is always statistical) hides a factor of 2^{-k} since the third and fourth experiments only differ when $\mathsf{Dec}(\tilde{c}) = s$, where s is chosen randomly and independently of how $\tilde{c}$ is generated, which only happens with probability 2^{-k}.

The third part of the theorem simply follows from the previous two. □

B Proof of Theorem 4.1

We prove the following exact security. For every even $t > 6$,

$$\varepsilon \leq \max\left(\rho \ , \ \frac{1}{2^t} + \left(\frac{t}{n(d/n - 1/4)^2} \right)^{t/2} \right).$$

Proof of Theorem 4.1. Fix any tampering function $f = (f_1, \ldots, f_n) \in \mathcal{F}$. We show that there exists a distribution D_f which satisfies the definition of non-malleability. To help with notation, for every $s \in \{0,1\}^k$, we define the random variables:

$$C^s \overset{\text{def}}{=} \mathsf{Enc}(s) \quad , \quad \tilde{C}^s \overset{\text{def}}{=} f(C^s) \quad , \quad \Delta^s \overset{\text{def}}{=} \tilde{C}^s - C^s \quad , \quad \tilde{S}^s \overset{\text{def}}{=} \mathsf{Dec}(\tilde{C}^s).$$

We use the notation C^s_i (resp. $\tilde{C}^s_i$, Δ^s_i) to denote the bit of C^s (resp. $\tilde{C}^s$, Δ^s) in position $i \in \{1, \ldots, n\}$. We also define the distribution $\mathsf{Patch}(D_f, s)$ which samples $\tilde{s} \leftarrow D_f$ and outputs s if $\tilde{s} = \mathsf{same}^\star$ and $\tilde{s}$ otherwise. To prove the theorem, we must show that there exists a distribution D_f such that $\mathsf{SD}(\tilde{S}^s \ , \ \mathsf{Patch}(D_f, s)) \leq \varepsilon$ for all $s \in \{0,1\}^k$. Let q be the number of positions $i \in \{1, \ldots, n\}$ for which f_i is one of the functions "set to 0"/"set to 1". We split the rest of the proof into 4 cases based on the value of q.

Case 1, $q \leq t$**:** In this case, the distribution of Δ^s is the same for all s. In particular:

- For i such that $f_i =$"keep", $\Delta^s_i = 0$. For i such that f_i="flip", $\Delta^s_i = 1$.
- Let A be the set of i for which $f_i =$"set to 0" or $f_i =$"set to 1". For $i \in A$, Δ^s_i are uniformly random over $\{0,1\}$ and independent.
  Since $|A| = q \leq t$, we rely on the t-secrecy of the LECSS, which tells us that the random variables $\{C^s_i\}_{i \in A}$ are independent and uniform over $\{0,1\}$. Since $\{f_i(C^s_i)\}_{i \in A}$ are constants, we see that $\{\Delta^s_i = C^s_i - f_i(C^s_i)\}_{i \in A}$ are uniformly random and independent variables for all s.

Therefore, there is a universal distribution Δ such that $\Delta \equiv \Delta^s$ for all s and so:

$$\begin{aligned}\tilde{S}^s &\equiv \mathsf{Dec}(\tilde{C}^s) \equiv \mathsf{V}(\mathsf{D}(C^s + \Delta^s)) \\ &\equiv \mathsf{V}(\mathsf{D}(C^s) + \mathsf{D}(\Delta^s)) \equiv \mathsf{V}(\mathsf{A}(s) + \mathsf{D}(\Delta^s)) \qquad (3) \\ &\equiv \mathsf{V}(\mathsf{A}(s) + \mathsf{D}(\Delta))\end{aligned}$$

where (3) follows by the linearity property of the LECSS. Therefore:

1. Conditioned on $\mathsf{D}(\Delta) \neq 0$, $\Pr[\tilde{S}^s = \perp] \geq (1-\rho)$.
   This follows by the security of the AMD code, and the fact that $\mathsf{D}(\Delta)$ is independents of the randomness of the AMD encoding.

2. Conditioned on $\mathsf{D}(\Delta) = 0$, $\Pr[\tilde{S}^s = s] = 1$.

We define the distribution D_f which samples $\delta \leftarrow \Delta$ and outputs $\mathsf{same}^\star$ if $\mathsf{D}(\delta) = 0$ and $\perp$ otherwise. By conditions 1,2 it is easy to see that, for all $s \in \{0,1\}^k$,

$$\mathsf{SD}(\tilde{S}^s, \mathsf{Patch}(D_f, s)) \leq \rho$$

which concludes the proof of Case 1.

Case 2, $q \geq n - t$**:** In this case, the distribution of $\tilde{C}^s$ is the same for all s. In particular:

- For $f_i =$"set to 0", $\tilde{C}^s_i = 0$. For f_i="set to 1", $\tilde{C}^s_i = 1$.
- Let B be the set of i for which $f_i =$"keep" or $f_i =$"flip". For $i \in B$, $\tilde{C}^s_i$ are uniformly random over $\{0,1\}$ and independent.
  This is because $|B| = n - q \leq t$. By the t-secrecy of the LECSS, the random variables $\{C^s_i\}_{i \in B}$ are uniformly random and independent and hence so are the variables $\{\tilde{C}^s_i = f_i(C^s_i)\}_{i \in B}$.

Therefore, there is a universal distribution $\tilde{C}$ such that $\tilde{C} \equiv \tilde{C}^s$ for all s and so $\tilde{S}^s \equiv \mathsf{Dec}(\tilde{C}^s) \equiv \mathsf{Dec}(\tilde{C})$. We define the distribution D_f which samples $\tilde{C}$ as above, and computes $\mathsf{Dec}(\tilde{C})$. It is clear that, for all $s \in \{0,1\}^k$,

$$\mathsf{SD}(\tilde{S}^s, \mathsf{Patch}(D_f, s)) \leq \mathsf{SD}(\tilde{S}^s, D_f) = 0.$$

which concludes the proof of Case 2.

Case 3, $t < q \leq n/2$**:** In this case, the distribution D_f just outputs $\perp$. Fix any $s \in \{0,1\}^k$. It is easy to see that

$$\Pr[\tilde{S}^s \neq \perp] \leq \Pr[\mathsf{Dec}(\tilde{C}^s) \neq \perp] \leq \Pr[\mathsf{D}(\Delta^s) \neq \perp].$$

We are now only left to show that $\Pr[\mathsf{D}(\Delta^s) \neq \perp]$ is small. Define A be the set of indices i for which f_i is of the form "set to 0"/"set to 1", so that $|A| = q$. Note that $\{\Delta^s_i\}_{i \notin A}$ are completely fixed by f to some constants (Δ^s_i is fixed to 0 if f_i is "keep" and fixed to 1 if f_i is "flip"). Let $\delta^* \in \{0,1\}^n$ be any value which is consistent with the fixed bits of Δ so that $\{\Delta^s_i = \delta^*_i\}_{i \notin A}$, and for which $\mathsf{D}(\delta^*) \neq \perp$. If no such value exists then we are done since $\mathsf{D}(\Delta^s) = \perp$ with probability 1, so let's assume that some such value exists.

It is easy to see that $\Pr[\Delta^s = \delta^*] \leq \frac{1}{2^t}$ since the random variables $\{\Delta^s_i\}_{i \in A}$ are uniform over $\{0,1\}$ and t-wise independent. So Δ^s is unlikely to match δ^* exactly. On the other hand, we show that the Hamming distance $d_H(\Delta^s, \delta^*)$ is unlikely to be too large. In particular the expected value of this distance is:

$$\mathsf{E}[d_H(\Delta^s, \delta^*)] = \mathsf{E}\left[\sum_{i=1}^{n} d_H(\Delta^s_i, \delta^*_i)\right] = \mathsf{E}\left[\sum_{i \in A} d_H(\Delta^s_i, \delta^*_i)\right] = \sum_{i \in A} \mathsf{E}\left[d_H(\Delta^s_i, \delta^*_i)\right] = \frac{q}{2}.$$

where the second equality follows by the fact that $\Delta_i^s = \delta_i^*$ for $i \notin A$, the third equality follows by the linearity of expectation and and the fourth equality follows since, each Δ_i^s for $i \in A$ is individually uniform. We now wish to bound the probability that $d_H(\Delta^s, \delta^*) = \sum_{i \in A} d_H(\Delta_i^s, \delta_i^*)$ is greater than the distance d of the LECSS scheme. Since $\{d_H(\Delta_i^s, \delta_i^*)\}_{i \in A}$ are t-wise independent random variables, we can use (Chernoff-Hoeffding type) tail inequalities for bounded independence (see [33, 4]) to bound the above probability. In particular:

$$\begin{aligned}
\Pr\left[d_H(\Delta^s, \delta^*) \geq d\right] &\leq \Pr\left[\ |d_H(\Delta^s, \delta^*) - q/2| \ \geq d - q/2\ \right] \\
&\leq \left(\frac{nt}{(d-q/2)^2}\right)^{t/2} \leq \left(\frac{nt}{(d-n/4)^2}\right)^{t/2} \qquad (4)\\
&\leq \left(\frac{t}{n(d/n-1/4)^2}\right)^{t/2}
\end{aligned}$$

where (4) follows by the tail inequality of Bellare and Rompel [4], Lemma 2.2. Finally, we see that

$$\Pr[\mathsf{D}(\Delta^s) \neq \perp] \quad \leq \quad \Pr[\Delta^s = \delta^* \ \vee \ d_H(\Delta^s, \delta^*) \geq d] \quad \leq \quad \frac{1}{2^t} + \left(\frac{t}{n(d/n-1/4)^2}\right)^{t/2}$$

which concludes the analysis of Case 3.

Case 4, $n/2 < q < n - t$**:** In this case, the distribution D_f just outputs $\perp$. Fix any $s \in \{0,1\}^k$. It is easy to see that $\Pr[\tilde{S}^s \neq \perp] \leq \Pr[\mathsf{D}(\tilde{C}^s) \neq \perp]$ and so we are left to show that this last probability is small. To show this, we essentially re-use the analysis of Case 3 and note the symmetry between the distribution of $\tilde{C}^s$ in this case and Δ^s in Case 3. In particular, define B be the set of indices i for which f_i =“keep”/“flip”, so $|B| = n - q$ and hence $t \leq |B| \leq n/2$. Note that $\{\tilde{C}_i^s\}_{i \notin B}$ are completely fixed by f. Let $\tilde{c}^* \in \{0,1\}^n$ be any value which is consistent with the fixed portion of $\tilde{C}^s$ so that $\{\tilde{C}_i^s = \tilde{c}_i^*\}_{i \notin B}$. Again, if no such value exists then we are done. Otherwise, we re-use the exact same analysis as in Case 3 to argue that

$$\Pr[\mathsf{D}(\tilde{C}^s) \neq \perp] \quad \leq \quad \Pr[\tilde{C}^s = \tilde{c}^* \ \vee \ d_H(\tilde{C}^s, \tilde{c}^*) \geq d] \quad \leq \quad \frac{1}{2^t} + \left(\frac{t}{n(d/n-1/4)^2}\right)^{t/2}$$

which concludes the analysis of Case 4. Since these are the only possible cases for q, this concludes the proof of the theorem.

□

C Construction of Linear Error-Correcting Secret Sharing

Recently, Chen et al. [9] formalized the connection between secret-sharing schemes and error-correcting codes with large *dual distance*. Since our terminology and exact construction differ slightly, we state and prove the following lemma, which is essentially equivalent to the result of [9].

Lemma C.1 (Based on [9]). *Let G be a $(k+\ell) \times n$ generator matrix over some field $\mathbb{F}$, generating a code $\mathcal{C} \subseteq \mathbb{F}^n$ with distance d. Let $\hat{G}$ be the sub-matrix consisting of the last ℓ rows of G, let $\hat{\mathcal{C}} \subseteq \mathcal{C}$ be the sub-code generated by $\hat{G}$ and let $\hat{\mathcal{C}}^\perp$ be the dual-code to $\hat{\mathcal{C}}$. Let $\hat{d}^\perp$ be the distance of $\hat{\mathcal{C}}^\perp$.*

We define the coding scheme (E, D), with source messages $s \in \mathbb{F}^k$. The encoding algorithm computes $\mathsf{E}(s) \stackrel{def}{=} (s, r)G$ for a uniformly random $r \in \mathbb{F}^\ell$. The decoding algorithm $\mathsf{Dec}(c)$ checks if c is a codeword in $\mathcal{C}$: if so, it decodes c to $(s', r') \in \mathbb{F}^k \times \mathbb{F}^\ell$ and outputs s'; otherwise, it outputs $\perp$.Then (E, D) is a (d, t)-LECSS scheme where $t = \hat{d}^\perp - 1$.

Proof. The linearity and distance d of the LECSS follows directly from the linearity and distance of $\mathcal{C}$. To analyze privacy, note that for any $s \in \mathbb{F}^k$, we have $\mathsf{E}(s) = (s, r)G = (s, 0)G + r\hat{G}$ and so we just need show that the components of $\hat{c} = r\hat{G}$ are individually uniform and t-wise independent for a random r. To show this, we use a well-known fact from coding theory that, if the distance of $\hat{\mathcal{C}}^{\perp}$ is $\hat{d}^{\perp}$ then *any* $t = \hat{d}^{\perp} - 1$ columns of $\hat{G}$ must be linearly independent. Therefore, for any fixed set of indices $A \subseteq \{1, \dots, n\}$ of size $|A| = t$, multiplication by $\hat{G}$ is a surjective linear function when restricted to the columns $i \in A$ and so the components of $\hat{c} = r\hat{G}$ in positions $i \in A$ are uniformly random and independent. □

Unfortunately, it is unclear how to construct an error-correcting code $\mathcal{C}$ which has a good distance $d > n/4$ and having a sub-code $\hat{\mathcal{C}}$ with large dual-distance $\hat{d}^{\perp}$, over a binary alphabet. For example, we can explicitly get codes with large distance (approaching $\frac{1}{2}$) using concatenated codes (such as Forney or Justesen codes), but the dual-distance of a concatenated code may only be equal to the dual-distance of the inner code, which is usually small. Therefore, we do not know of any explicit constructions of such codes, and it remains as an interesting open problem. However, it is relatively easy to show that a *random linear error-correcting code* is likely (with overwhelming probability) to simultaneously achieve very high distance d for $\mathcal{C}$ and high dual-distance $\hat{d}^{\perp}$ for $\hat{\mathcal{C}}^{\perp}$ as defined in Lemma C.1. We note that, in our context, using random linear error-correcting codes is *very efficient*, since the representation size of such code, the encoding procedure, checking if a value c is a codeword, and decoding a codeword *without errors* are all very efficient. In particular, the construction in Lemma C.1 does *not* require the ability to efficiently *correct* errors, which is believed to be hard for random linear codes. The following result, explicitly shown by Chen et. al. [9] (see Theorem 4 and Corollary 1), follows from a Gilbert-Varshamov type of argument showing that random linear error-correcting codes achieve good distance and also good dual-distance.

Lemma C.2 ([9]). *Let G be a uniformly random generator matrix from the space of $(k + \ell) \times n$ matrices over $\mathbb{F}_2$ with linearly-independent rows. Let $\mathcal{C}$, $\hat{\mathcal{C}}$, $\hat{\mathcal{C}}^{\perp}$, d and $\hat{d}^{\perp}$ be defined as in Lemma C.1. Then for any $0 < \alpha, \hat{\alpha} < \frac{1}{2}$,*

$$\Pr[\, (d > \alpha n) \text{ and } (\hat{d}^{\perp} > \hat{\alpha} n)] > 1 - 2^{k+\ell-n(1-H(\alpha))} - 2^{\ell - nH(\hat{\alpha})}$$

where the probability is taken (only) over the choice of G, and $H(x) \overset{def}{=} -x\log_2(x) - (1-x)\log_2(1-x)$.

C.1 Proof of Theorem 4.2

Using the above two lemmas, we are now ready to prove Theorem 4.2

Proof of Theorem 4.2. First, given $\delta > 0$ there are constants $\delta_0 > 0, \delta_1 > 0$ such that

$$(1 - H(1/4) - \delta) = (1 - H(1/4 + \delta_0) - \delta_1).$$

Let $k' = (1 - H(1/4 + \delta_0) - \delta_1/4)n$ and $\ell = (\delta_1/4)n$. Let $\hat{\alpha}$ be a constant such that $H(\hat{\alpha}) \le \delta_1/8$. If we choose a random $(k' + \ell) \times n$ matrix G over $\mathbb{F}_2$, and define $\mathcal{C}, \hat{\mathcal{C}}, \hat{\mathcal{C}}^{\perp}, d, \hat{d}^{\perp}$ as in Lemma C.1, then using the bounds of Lemma C.2, we will get $d \ge (1/4 + \delta_0)n$ and $\hat{d}^{\perp} \ge \hat{\alpha}n$ with probability

$$1 - 2^{k'+\ell-n(1-H(1/4+\delta_0))} - 2^{\ell-nH(\hat{\alpha})} = 1 - 2^{-(\delta_1/2)n} - 2^{-(\delta_1/8)n} = 1 - 2^{-\Omega(n)}.$$

Assuming this is the case, we now show how to instantiate our non-malleable code using G, so that we achieve the claimed parameters.

1. Using Lemma C.1, we see that G gives rise to an efficient (d, t)-LECSS scheme with message size k' and codeword size n, and with $d \ge (1/4 + \delta_0)n$ and $t = \hat{d}^{\perp} - 1 \ge \delta_2 n$ for some constant $\delta_2 < \delta_0^2/2$.

2. Using the result of [12], there exist AMD codes with message size k, codeword size k' and security

$$\rho \le \frac{k}{2^{(k'-k)/2}} \le \frac{n}{2^{(\delta_1/4)n}} = 2^{-\Omega(n)}.$$

Now we use Theorem 4.1 to combine (1) and (2) above, yielding a non-malleable code with message size k, codeword size n and security

$$\varepsilon \le \max\left(\rho\ ,\ \frac{1}{2^{\delta_2 n}} + \left(\frac{\delta_2 n}{n(\delta_0)^2}\right)^{(\delta_2 n)/2}\right) \le 2^{-\Omega(n)}$$

□

D Proof of Theorem 4.3

As a tool we use a *symmetric-key authenticated-encryption scheme* that we construct using an encryption scheme $(\mathsf{Encrypt}, \mathsf{Decrypt})$ with key-size k_0, and a message authentication code (MAC) $(\mathsf{Tag}, \mathsf{Verify})$ with key-size k_1. The encryption scheme $(\mathsf{Encrypt}, \mathsf{Decrypt})$ has to always satisfy the following: $\mathsf{Decrypt}(\mathsf{key}_0, \mathsf{Encrypt}(\mathsf{key}_0, m)) = m$. Security of $(\mathsf{Encrypt}, \mathsf{Decrypt})$ is defined as follows: for every two messages m and m' of equal length, and a random key_0 we have that $\mathsf{Encrypt}(\mathsf{key}_0, m)$ and $\mathsf{Encrypt}(\mathsf{key}_0, m')$ are computationally indistinguishable.

The authentication scheme $(\mathsf{Tag}, \mathsf{Verify})$ has always to satisfy: $\mathsf{Verify}(\mathsf{key}_1, m, \mathsf{Tag}(\mathsf{key}_1, m)) = \mathsf{yes}$. Security of $(\mathsf{Tag}, \mathsf{Verify})$ is defined as follows: for every message m and for a random key_1 no poly-time adversary $\mathcal{A}$, after seeing $(m, \mathsf{Tag}(\mathsf{key}, m))$, can output (t, m') such that $\mathsf{Verify}(\mathsf{key}_1, m', t) = \mathsf{yes}$ and $m \neq m'$, except with a negligible probability.

An authenticated encryption scheme that we use has a key (k_0, k_1). On a message m it outputs $\mathsf{Tag}(k_1, \mathsf{Encrypt}(k_0, m))$ (cf. [3] for a general discussion on how to combine authentication with encryption).

Theorem D.1. *Assume that* $(\mathsf{Enc}, \mathsf{Dec})$ *is a non-malleable code w.r.t. the family* $\mathcal{F}$ *of bit-wise independent tampering functions with message-size* k *and codeword-size* n. *Let* $(\mathsf{Encrypt}, \mathsf{Decrypt})$ *and* $(\mathsf{Tag}, \mathsf{Verify})$ *be as above. Define:*

$$\mathsf{Enc}'(s) \overset{\mathit{def}}{=} \left\{ \begin{array}{c} (\mathsf{key}_0, \mathsf{key}_1) \leftarrow U_{k_0+k_1}, x \leftarrow \mathsf{Encrypt}(\mathsf{key}_0, s), t \leftarrow \mathsf{Tag}(\mathsf{key}_1, x), \\ c \leftarrow \mathsf{Enc}(\mathsf{key}_0 || \mathsf{key}_1) \\ \textit{Output: } c = (c, x, t) \end{array} \right\}$$

$$\mathsf{Dec}'(c, x, t) \overset{\mathit{def}}{=} \left\{ \begin{array}{c} \widetilde{\mathsf{key}}_0 || \widetilde{\mathsf{key}}_1 = \mathsf{Dec}(c), \\ \textit{If } \widetilde{\mathsf{key}}_0 || \widetilde{\mathsf{key}}_1 = \bot \textit{ or } \mathsf{Verify}(\mathsf{key}_1, x, t) \neq \mathsf{yes} \textit{ output } \bot, \\ \textit{else output } \mathsf{Decrypt}(\mathsf{key}_0, x) \end{array} \right\}$$

Then $(\mathsf{Enc}', \mathsf{Dec}')$ *is a computational non-malleable code w.r.t* $\mathcal{F}$.

Proof of D.1. Intuitively, the reason why the theorem holds is that the adversary has a choice to either (1) make the key encoding $(\mathsf{key}_0, \mathsf{key}_1)$ decode to $\bot$ — in this case the whole encoding decodes to $\bot$ not matter what the encoded message was; or (2) do not change the encoded key $(\mathsf{key}_0, \mathsf{key}_1)$ — in this case the adversary cannot substitute the encode message s with any other s' without breaking the MAC, moreover the security of the encryption scheme guarantees the probability that the decoding yields $\bot$ has to be (almost) the same for every message s; finally he can (3) change the encoded key to some random key $(\widetilde{\mathsf{key}}_0, \widetilde{\mathsf{key}}_1)$ (that has a distribution that is independent on $(\mathsf{key}_0, \mathsf{key}_1)$)— in this case it follows from the security of the encryption scheme that the decoded message cannot depend on the encoded message s, as otherwise one could construct a distinguisher that breaks the encryption scheme by sampling $(\widetilde{\mathsf{key}}_0, \widetilde{\mathsf{key}}_1)$ himself.

Let f' be a function that is tampering the output of $(\mathsf{Enc}', \mathsf{Dec}')$. We will construct a distribution $D'_{f'}$ such that for every s (1) is satisfied. Since the output of Enc' has a form (c, x, t), and since f' only modifies the individual bits, we can think of f' as of three functions f'_0, f'_1, and f'_2, where $f'(c, x, t) = (f'_0(c), f'_1(x), f'_2(t))$. The distribution $D'_{f'}$ is constructed as follows. First, we take an arbitrary message s and a random pair $(\mathsf{key}_0, \mathsf{key}_1)$. Then, we let $x^s \leftarrow \mathsf{Encrypt}(\mathsf{key}_0, s)$ and $t^s \leftarrow \mathsf{Tag}(\mathsf{key}_1, x),$. Form the assumption that $(\mathsf{Enc}, \mathsf{Dec})$ is a non-malleable code there exists a distribution $D_{f'_0}$ such that (1) holds. We sample $\tilde{c}$ from this distribution. If $\tilde{c} = \mathsf{same}^\star$ we let $(\widetilde{\mathsf{key}}_0, \widetilde{\mathsf{key}}_1) := (\mathsf{key}_0, \mathsf{key}_1)$, and otherwise we let $(\widetilde{\mathsf{key}}_0, \widetilde{\mathsf{key}}_1) := \tilde{c}$. We then output

$$\mathsf{out}^s := \begin{cases} \bot & \text{if } \mathsf{Verify}(\widetilde{\mathsf{key}}_1, f'_1(x^s), f'_2(t^s)) \neq \mathsf{yes} \text{ or } \tilde{c} = \bot, \\ \mathsf{same}^\star & \text{if } f'_1(x^s) = x^s \text{ and } \mathsf{Verify}(\widetilde{\mathsf{key}}_1, f'_1(x^s), f'_2(t^s)) = \mathsf{yes} \text{ and } \tilde{c} = \mathsf{same}^\star, \\ \mathsf{Dec}(\widetilde{\mathsf{key}}_0, f'_1(x^s)) & \text{otherwise.} \end{cases}$$

To prove that this construction is correct we need to show that for every s^0 and s^1 the distributions out^{s^0} and out^{s^0} are computationally-indistinguishable. After showing this we will be done, since this will mean that the result of our simulation (with an arbitrary value s) is computationally indistinguishable from a result of the simulation with any other s, and out^s with $\mathsf{same}^\star$ substituted with s is simply distributed identically to the output of the experiment of the left hand side of (1). Obviously, the distribution of $\tilde{c}$ does not depend on s. Therefore we can consider separately the following cases.

- $\tilde{c} = \perp$ — in this case obviously out^s is always equal to $\perp$, no matter what s is,
- $\tilde{c} = \mathsf{same}^\star$ — in this case first observe that

  $$|P(\mathsf{out}^{s^0} = \perp) - P(\mathsf{out}^{s^1} = \perp)| \text{ is negligible.}$$

  This is because otherwise one could construct an adversary that breaks the security of the encryption scheme $(\mathsf{Encrypt}, \mathsf{Decrypt})$ by distinguishing if x is an encryption of s^0 and s^1 in the following way: generate a random key_1 and output 1 if and only if $\mathsf{Verify}(\mathsf{key}_1, f_1'(x), f_2'(\mathsf{Tag}(\mathsf{key}_1, x)))$.

  On the other hand if $\mathsf{out}^s \neq \perp$ and $f_1'(x^s) \neq x^s$ then we can break the security of the MAC in the following way: after seeing a tag t^s on x^s an adversary can output a message $f_1'(x^s)$ with a tag $f_1'(x^s)$.
- $\tilde{c} = (\tilde{\mathsf{key}}_0, \tilde{\mathsf{key}}_1)$ — if in this case out^{s^0} can be distinguished from out^{s^1} by some poly-time machine $\mathcal{A}$ then one easily distinguish if x is an encryption of s^0 and s^1 in the following way.

  1. if $\mathsf{Verify}(\tilde{\mathsf{key}}_1, f_1'(x), f_2'(\mathsf{Tag}(\tilde{\mathsf{key}}_1, x))) = \perp$ then output $\perp$,
  2. otherwise output $\mathcal{A}(\mathsf{Dec}(\tilde{\mathsf{key}}_0, f_1'(x))$.

  This is because for both $i \in \{0,1\}$ we have $\mathsf{out}^{s^i} = \mathsf{Dec}(\tilde{\mathsf{key}}_0, f_1'(x^{s^i}))$.

□

E Proof of Theorem 5.1

Proof of Theorem 5.1. Let $K = 2^k, N = 2^n$. It will be useful for us to think of a function $f : \{0,1\}^n \to \{0,1\}^n$ as a directed graph $G_f = (V, E_f)$ with vertex set $V = \{0,1\}^v$ and edges $E_f = \{(v, \tilde{v}) | v \in V, f(v) = \tilde{v}\}$. Notice that the out-degree is 1 and that there may be some self-loops in G. Notice that a randomly chosen decoding function, essentially labels each vertex $v \in V$ with a uniformly random value $s \in \{0,1\}^k$. For any $s, \tilde{s} \in \{0,1\}^k$, we define the following random variables (over the choice of the decoding function Dec):

- Let $V(s)$ be the set of vertices $v \in V = \{0,1\}^n$ such that $\mathsf{Dec}(v) = s$.
- Let $E_f(s \to \tilde{s})$ be the set of edges $(v, \tilde{v})$ where $v \neq \tilde{v}$ and $\mathsf{Dec}(v) = s, \mathsf{Dec}(\tilde{v}) = \tilde{s}$
- Let $E_f(s \to \mathsf{same}^\star)$ be the set of self-loop edges (v, v) for which $\mathsf{Dec}(v) = s$.
- Let $E_f(* \to \tilde{s}) \stackrel{\text{def}}{=} \bigcup_{s \in \{0,1\}^k} E_f(s \to \tilde{s})$. (we will also use this notation for $\tilde{s} = \mathsf{same}^\star$).
- For $A \subseteq V \cup \{\mathsf{same}^\star\}$, define $E_f(s \to A) \stackrel{\text{def}}{=} \bigcup_{\tilde{s} \in A} E_f(s \to \tilde{s})$, and $E_f(* \to A) \stackrel{\text{def}}{=} \bigcup_{\tilde{s} \in A} E_f(* \to \tilde{s})$.

We use these variables to define a density-function $D_f : \{0,1\}^k \cup \{\mathsf{same}^\star\} \to [0,1]$ by $D_f(s) = |E_f(* \to s)|/N$. Then, for any function f and any $s \in \{0,1\}^k$, we have

$$\mathsf{SD}(\mathsf{StrongNM}_s^f, D_f) = \max_{A \subseteq \{0,1\}^k \cup \{\mathsf{same}^\star\}} \frac{|E_f[s \to A]|}{|V(s)|} - \frac{|E_f(* \to A)|}{N} \tag{5}$$

Therefore, taking probabilities (only) over the choice of the decoding function Dec, let $\mathcal{E}$ be the event that $(\mathsf{Enc}, \mathsf{Dec})$ is not an ε-secure non-malleable code w.r.t. $\mathcal{F}$. Then:

$$\begin{aligned}
\Pr[\mathcal{E}] &\le \Pr[\exists f \in \mathcal{F}, s \in \{0,1\}^k \text{ s.t. } \mathsf{SD}(\mathsf{StrongNM}_s^f, D_f) > \varepsilon] \\
&= \Pr\left[\exists f \in \mathcal{F}, s \in \{0,1\}^k, A \subseteq \{0,1\}^k \cup \{\mathsf{same}^\star\} \text{ s.t. } \frac{|E_f[s \to A]|}{|V(s)|} - \frac{|E_f(* \to A)|}{N} > \varepsilon\right] \\
&\le \sum_{f \in \mathcal{F}} \sum_{s \in \{0,1\}^k} \sum_{A \subseteq \{0,1\}^k \cup \{\mathsf{same}^\star\}} \Pr\left[\frac{|E_f[s \to A]|}{|V(s)|} - \frac{|E_f(* \to A)|}{N} > \varepsilon\right] \qquad (6)
\end{aligned}$$

We now *fix* some $f \in \mathcal{F}$, $s \in \{0,1\}^k$, $A \subseteq \{0,1\}^k \cup \{\mathsf{same}^\star\}$ and our goal it to upper bound:

$$\Pr\left[\frac{|E_f[s \to A]|}{|V(s)|} - \frac{|E_f(* \to A)|}{N} > \varepsilon\right] \qquad (7)$$

Let $\delta = \varepsilon/4, t = (\varepsilon/4)(N/K)$ and assume that:

$$|V(s)| \ge (1-\delta)\frac{N}{K} \quad , \quad |E_f(s \to A)| - \frac{|E_f(* \to A)|}{K} \le t \qquad (8)$$

then:

$$\begin{aligned}
\frac{|E_f[s \to A]|}{|V(s)|} - \frac{|E_f(* \to A)|}{N} &\le \frac{|E_f[* \to A]|/K + t}{(1-\delta)N/K} - \frac{|E_f(* \to A)|}{N} \\
&\le \frac{\delta|E_f[* \to A]| + tK}{N(1-\delta)} \\
&\le 2\delta + 2tK/N \le \varepsilon
\end{aligned}$$

where the inequality on the last line follows since $|E_f[* \to A]| \le N$. Therefore we show:

$$\begin{aligned}
(7) \le \Pr[(8) \text{ does not hold}] &\le \Pr\left[|V(s)| < (1-\delta)\frac{N}{K}\right] + \Pr\left[|E_f(s \to A)| - \frac{|E_f(* \to A)|}{K} > t\right] \\
&\le e^{-\delta^2 \frac{N}{2K}} + 2e^{-t^2/8N} \qquad (9) \\
&\le e^{-\frac{\varepsilon^2 N}{32K}} + 2e^{-\frac{\varepsilon^2 N}{128K^2}} \le 3e^{-\frac{\varepsilon^2 N}{128K^2}}
\end{aligned}$$

where we rely on the bounds from Lemma E.1 and Lemma E.2 (presented after this proof) for (9). Now, continuing from (6), we have

$$\begin{aligned}
\Pr[\mathcal{E}] &\le \sum_{f \in \mathcal{F}} \sum_{s \in \{0,1\}^k} \sum_{A \subseteq \{0,1\}^k \cup \{\mathsf{same}^\star\}} 3e^{-\frac{\varepsilon^2 N}{128K^2}} \\
&\le 2^{\log(|\mathcal{F}|) + k + K + 3 - \frac{\varepsilon^2 N}{128K^2}} \qquad (10)
\end{aligned}$$

We now bound (10) by ρ by setting

$$\begin{aligned}
\Pr[\mathcal{E}] \le \rho \quad &\text{follows if} \quad \log(|\mathcal{F}|) + k + K + 3 + \log(1/\rho) < \varepsilon^2 N/128K^2 \\
&\text{follows if} \quad 128K^2/\varepsilon^2(\log(|\mathcal{F}|)kK4\log(1/\rho)) < N \\
&\text{follows if} \quad n > \log(\log(|\mathcal{F}|)) + 3k + \log(k) + 2\log\left(\frac{1}{\varepsilon}\right) + \log\left(\log\left(\frac{1}{\rho}\right)\right) + 9
\end{aligned}$$

where the last inequality is the assumption of our theorem. Therefore, with probability $1-\rho$, $\mathcal{E}$ does not occur and the random function Dec gives rise to a non-malleable code w.r.t. $\mathcal{F}$ with security ε. In particular, some such codes with the above bound exist for any $\rho < 1$. This concludes the proof, up to the bounds from Lemma E.1 and Lemma E.2, which are shown next. □

Lemma E.1. *For any $\delta > 0$, $\Pr\left[|V(s)| < (1-\delta)\frac{N}{K}\right] \leq e^{-\delta^2 \frac{N}{2K}}$.*

Proof. Follows directly by the (multiplicative) Chernoff bound, since each vertex $v \in V$ is labeled with a uniformly random value from $\{0,1\}^k$. □

Lemma E.2. *Let $W = |E_f(s \to A)| - \frac{|E_f(* \to A)|}{K}$. Then for any $t \geq 0$, $\Pr[W \geq t] \leq 2e^{-t^2/8N}$.*

Proof. For each edge $e = (v, \tilde{v})$, we define a random variable

$$C_e = \begin{cases} 0 & \text{if } e \notin E_f(* \to A) \\ 1 - 1/K & \text{if } e \in E_f(s \to A) \\ (-1/K) & \text{otherwise (if } e \in E_f(* \to A) \setminus E_f(s \to A)) \end{cases} \tag{11}$$

Then $W = \sum_{e \in E} C_e$. We wish to prove that W is closely centered around 0. Unfortunately, the variables C_e are not independent. Our analysis is broken up into three claims. First we show that in any subset $U \subseteq E$ which doesn't contain *non-trivial cycles* (cycles with more than 1 edge), the sum of C_e over U forms a martingale. Second, we show that E can be partitioned into two such subsets $E = U_1 \cup U_2$. Lastly we show that these two properties imply that W is closely centered at 0, using Azuma's inequality on martingales.

Claim E.1. *For any subset $U \subseteq E$ of edges which does not contain a non-trivial cycle (i.e. a cycle consisting of more than 1 edge), there is an ordering $e_1, \dots, e_{|U|}$ of the edges in U such that the random variables $W_i = \sum_{j=1}^{i} C_{e_j}$ form a Martingale.*

Proof. We can define a topological sort $v_1, \dots, v_{|V|}$ on the vertices V so that, for all edges $e = (v_i, v_j) \in U$, $i \geq j$. We can extend the ordering on vertices to one on edges by identifying each edge with its starting vertex (this is unique since the out-degree is 1). Think of the function Dec as being chosen by assigning the values $\mathsf{Dec}(v_1), \mathsf{Dec}(v_2), \dots, \mathsf{Dec}(v_{|V|})$ in order, uniformly at random. Let $e_k = (v_i, v_j)$ be some edge in U. Then, the choice of $\mathsf{Dec}(v_1), \dots, \mathsf{Dec}(v_{i-1})$ completely determines the values $C_{e_1}, \dots, C_{e_{k-1}}$ and hence also the values of $W_1, \dots, W_{k-1}$. We consider two cases.

$v_i = v_j$: In this case, if $\mathsf{same}^\star \notin A$ then C_{e_k} is always 0. If $\mathsf{same}^\star \in A$ then e_k is always in $E_f(* \to A)$, and is also in $E_f(s \to A)$ iff $\mathsf{Dec}(v_i)$ is labeled with s, which occurs with probability $1/K$. Therefore, $\mathsf{E}[C_{e_k} \mid C_{e_1}, \dots, C_{e_{k-1}}] = 0$.

$v_i \neq v_j$: In this case, the value $\mathsf{Dec}(v_j)$ is completely determined at this point since $i > j$. If $\mathsf{Dec}(v_j) \notin A$ then C_{e_k} is always 0. Otherwise, it takes on the value $1 - 1/K$ with probability $1/K$ (the probability the $\mathsf{Dec}(v_i) = s$) and $-1/K$ with probability $(1 - 1/K)$. Therefore $\mathsf{E}[C_{e_k} \mid C_{e_1}, \dots, C_{e_{k-1}}] = 0$.

So, in both cases $\mathsf{E}[C_{e_k} \mid C_{e_1}, \dots, C_{e_{k-1}}] = 0$ and $\mathsf{E}[W_k | W_1, \dots, W_{k-1}] = W_{k-1}$ as we wanted to show. □

Claim E.2. *For any directed graph $G = (V, E)$ where each vertex has out-degree one, there is a partition of E into to components U_1, U_2 such that neither component contains a non-trivial cycle.*

Proof. Since the out-degree is one, each edge participates in at most one non-trivial cycle. Therefore, we can break up each non-trivial cycle separately, by placing half the edges into U_1 and the other half into U_2. □

(continuing the proof of Lemma E.2) Let U_1, U_2 be a partition of E as in Claim E.2. Let $W_1 = \sum_{e \in U_1} C_e$, $W_2 = \sum_{e \in U_2} C_e$ so that $W = W_1 + W_2$. Then, by Claim E.1 we have the martingales $W_1^{(1)}, W_1^{(2)}, \dots, W_1^{(|U_1|)} = W_1$ and $W_2^{(1)}, W_2^{(2)} \dots, W_2^{(|U_2|)} = W_2$. So

$$\Pr[W \geq t] \leq \Pr[W_1 \geq t/2] + \Pr[W_2 \geq t/2] \leq e^{-t^2/8|U_1|} + e^{-t^2/8|U_2|} \leq 2e^{-t^2/8N}$$

where we use Azuma's inequality to bound the two probabilities (and note that $|U_1|, |U_2| \leq N$). □

F Proof of Theorem 6.1

Proof of Theorem 6.1. By the definition of non-malleability, for each $f \in \mathcal{F}$ there exists a distribution D_f such that

$$\left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \text{Output: } \tilde{s}. \end{array} \right\} \approx \left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^\star, \text{ and } \tilde{s} \text{ otherwise.} \end{array} \right\}$$

We use these distributions to define the simulator $\mathcal{S}$ for the tamper simulatability definition. Recall that $\mathcal{S}$ has black-box access to the adversary $\mathcal{A}$, the functionality $G(\cdot,\cdot)$ and the tampering function $f(\cdot)$ queried by $\mathcal{A}$. The simulation proceeds by running $\mathcal{A}$ where, in each round:

1. When the adversary $\mathcal{A}$ issues a Tamper command with function $f \in \mathcal{F}$ the simulator samples $\tilde{s} \leftarrow D_f$ (this can be done efficiently give oracle access to $f(\cdot)$, by the definition to non-malleability).

2. When the adversary issues an $\mathsf{Execute}$ command with input x:

   - If $\tilde{s} = \mathsf{same}^\star$, then the simulator $\mathcal{S}$ forwards the $\mathsf{Execute}(x)$ query to its system $\langle G, s \rangle$ and forwards the output y to $\mathcal{A}$.
   - If $\tilde{s} \neq \mathsf{same}^\star$ the simulator $\mathcal{S}$ goes into "Overwritten Mode" (step 3).

3. "Overwritten Mode": The simulator $\mathcal{S}$ uses the modified state $\tilde{s}$ to perfectly simulate the system $\langle G^{\mathsf{Enc},\mathsf{Dec}}, \mathsf{Enc}(\tilde{s}) \rangle$ for all Tamper and $\mathsf{Execute}$ queries in all future rounds. Note that this is possible, using oracle access to $G(\cdot,\cdot)$ and to the tampering functions $f(\cdot)$.

To show the indistinguishability of simulation, we notice that, in each round *prior to and including* the round where the simulation enters "overwritten mode", if the starting state in that round is s, the modification function is f and the the input queried by the adversary is x then (by the definition of non-malleability):

$$\left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} = \mathsf{Dec}(\tilde{c}) \\ \text{Output: } G(x, \tilde{s}). \end{array} \right\} \approx \left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \text{Output: } G(x,s) \text{ if } \tilde{s} = \mathsf{same}^\star, \text{ and } G(x,\tilde{s}) \text{ otherwise.} \end{array} \right\}$$

where the distributions represent the output y seen by the adversary, and the updated "effective state" s of the system in the $\mathsf{TamperInteract}$ game (on the left-hand side) and in the simulation within the $\mathsf{BBInteract}$ game (on the right hand side). Also, in each round *after* the simulation enters "overwritten mode" it acts exactly the same as the $\mathsf{TamperInteract}$ game. These two facts, together with a hybrid argument over the number of rounds, prove the indistinguishability of the simulation. □