A Rate-Optimizing Compiler for Non-malleable Codes Against Bit-wise Tampering and Permutations

Shashank Agrawal* Divya Gupta† Hemanta K. Maji‡ Omkant Pandey§
Manoj Prabhakaran¶

Abstract

A non-malleable code protects messages against a class of tampering functions. Informally, a code is non-malleable if the effect of applying any tampering function on an encoded message is to either retain the message or to replace it with an unrelated message. Two main challenges in this area – apart from establishing the feasibility against different families of tampering – are to obtain *explicit constructions* and to obtain *high-rates* for such constructions.

In this work, we present a compiler to transform low-rate (in fact, zero rate) non-malleable codes against certain class of tampering into an optimal-rate – i.e., rate 1 – non-malleable codes against the same class. If the original code is explicit, so is the new one.

When applied to the family of bit-wise tampering functions, this subsumes (and greatly simplifies) a recent result of Cheraghchi and Guruswami (TCC 2014). Further, our compiler can be applied to non-malleable codes against the class of bit-wise tampering and bit-level permutations. Combined with the rate-0 construction in a companion work, this yields the first explicit rate-1 non-malleable code for this family of tampering functions.

Our compiler uses a new technique for boot-strapping non-malleability by introducing errors, that may be of independent interest.

*University of Illinois, Urbana-Champaign. sagrawl2@illinois.edu.
†University of California, Los Angeles. divyag@cs.ucla.edu.
‡University of California, Los Angeles. hemanta.maji@gmail.com.
§University of Illinois, Urbana-Champaign. omkant@gmail.com.
¶University of Illinois, Urbana-Champaign. mmp@illinois.edu.

Contents

1 Introduction

Non-Malleable Codes have emerged as an object of fundamental interest, at the intersection of coding theory and cryptography. Informally, a code is non-malleable if the message contained in a codeword that has been tampered with is either the original message, or a completely unrelated value. As a relatively new problem, several basic questions are still open. In particular, two main challenges in this area – apart from establishing the feasibility against different families of tampering – are to obtain *explicit constructions* and to obtain *high-rates*[1] for such constructions.

While existential results have been obtained for rate-1 non-malleable codes for very broad classes of tampering functions [CG14a, FMNV14], explicit constructions have turned out to be much harder, in such generality. For the relatively simple class of bit-wise tampering functions introduced in [DPW10], it was only recently that an explicit rate-1 construction was obtained [CG14b]. For the more general class of "split-state" tampering functions, the first construction in [DKO13] encoded only a single bit; in a break-through result, an explicit scheme (of rate 0) was proposed for arbitrary length messages by [ADL14], and more recently, a constant rate construction (for 10 states) was provided in [CZ14].

All the above explicit results relied on the tampering functions being "compartmentalized" — i.e., the codeword is partitioned *a priori* into separate blocks and each block is tampered independently of the others — In a companion paper [AGM+14b], we presented the first instance of an explicit non-malleable code against a class of non-compartmentalized tampering functions. This class consists of functions which can *permute* the bits of a codeword, as well as tamper each bit independently. Apart from being of interest as a natural non-compartmentalized class, non-malleable codes against this class have direct cryptographic applications: in [AGM+14b] it is used to obtain non-malleable string-commitments from non-malleable bit-commitments in a hardware token-based model, with information-theoretic security (or in the standard model under computational assumptions). This application also highlighted the need for explicit constructions, even if randomized constructions are efficient, since the latter calls for a trusted party to carry out the randomized construction.

The construction in [AGM+14b] has 0 rate. In this paper, we present a simple but powerful compiler to transform such a non-malleable code into a rate-1 non-malleable code against the same family. In fact, our compiler is general enough that it can be applied to non-malleable codes against bit-wise tampering too, to improve their rate from 0 to 1. This subsumes (and greatly simplifies) a result of [CG14b].

1.1 Our Contribution

Let $\mathcal{F}^*$ be the class of tampering functions $f : \{0,1\}^N \to \{0,1\}^N$ of the form $f(x) = f_\pi(f_1(x_1), \cdots, f_N(x_N))$, where f_π permutes the indices of its input according to a permutation $\pi : [N] \hookrightarrow [N]$, and each $f_i : \{0,1\} \to \{0,1\}$ is one of the four possible binary functions over $\{0,1\}$.

Our main technical result is the following.

Informal Theorem 1. *There exists a black-box compiler that takes a non-malleable code* NMC_0 *secure against* $\mathcal{F}^*$*, which may have a polynomial blowup in size during encoding (and hence rate*

[1] Rate refers to the asymptotic ratio of the length of a message to the length of its encoding (in bits), as the message length increases to infinity. The best rate possible is 1; if the length of the encoding is super-linear in the length of the message, the rate is 0.

0), and defines a rate-1 non-malleable code NMC_1 *secure against* $\mathcal{F}^*$*. The encoding and decoding algorithms of* NMC_1 *make only black-box calls to the respective functions of* NMC_0 *(on much smaller inputs).*

In fact, we present our compiler as consisting of two black-box components: NMC_0 and a rate-1 binary error-correcting code. (We require the error-correcting code to also have an easy to satisfy privacy requirement.) The encoding and decoding algorithms of NMC_1 make only black-box calls to the encoding, decoding and error-correcting functions of this error-correcting code. An error-correcting code with the requisite properties is easily instantiated using (low-distance) Reed-Solomon codes over a field of characteristic 2.

Instantiating NMC_0 with the non-malleable code of [AGM+14b] (which has rate 0, as the codewords are super-linear in the length of the messages), we get our main result.

Corollary 1. *There exists an explicit and efficient rate-1 non-malleable code against* $\mathcal{F}^*$*.*

We point out that the above result has immediate implications for the class of all bit-wise tampering functions $\mathcal{F}_{\mathsf{BIT}}$ [DPW10]. Non-malleable codes for this family has been studied by [DPW10, CG14b]. We note that $\mathcal{F}_{\mathsf{BIT}}$ is a subset of $\mathcal{F}^*$ in which π is restricted to be the identity permutation. As a consequence, we reproduce a result of [CG14b] as a simple corollary to Corollary 1:

Corollary 2. *There exists an explicit and efficient rate-1 non-malleable code against* $\mathcal{F}_{\mathsf{BIT}}$*.*

In fact, Theorem 1 continues to hold true, without altering the compiler, if $\mathcal{F}^*$ is replaced by $\mathcal{F}_{\mathsf{BIT}}$ (so that NMC_0 and NMC_1 are both secure only against $\mathcal{F}_{\mathsf{BIT}}$). This provides a much simpler alternative to a compiler in [CG14b], and proves Corollary 2 without relying on the recent construction from [AGM+14b].

1.2 Prior Work

Cramer et al. [CDF+08] introduced the notion of arithmetic manipulation detection (AMD) codes, which is a special case of non-malleable codes against tampering functions with a simple algebraic structure; explicit AMD codes with optimal (second order) parameters have been recently provided by [CPX14]. Dziembowski et al. motivated and formalized the more general notion of non-malleable codes in [DPW10]. They showed existence of a constant rate non-malleable code against the class of all bit-wise independent tampering functions.

The existence of rate-1 non-malleable codes against various classes of tampering functions is known. For example, existence of such codes with rate $(1-\alpha)$ was shown against any tampering function family of size $2^{2^{\alpha n}}$; but this scheme has inefficient encoding and decoding [CG14a]. For tampering functions of size $2^{\mathrm{poly}(n)}$, rate-1 codes (with efficient encoding and decoding) exist, and can be obtained efficiently with overwhelming probability [FMVW14].

However, explicit constructions of non-malleable codes have remained elusive, except for some well structured tampering function classes. For the setting where the codeword is partitioned into separate blocks and each block can be tampered arbitrarily but independently, an encoding scheme was proposed in [CKM11]. In the most general such compartmentalized model of tampering, where there are only two compartments (known as the split-state model), an explicit encoding scheme for bits was proposed by [DKO13]. Recently, in a break-through result, an explicit scheme (of rate 0)

was proposed for arbitrary length messages by [ADL14]. A constant rate construction for 10 states was provided in [CZ14].

In the computational setting, there has been a sequence of works on improving the rate of error-correcting codes [Lip94, MPSW05, OPS07, HO08, GS10, CKO14] and constructing non-malleable codes and its variants [LL12, FMNV14].

An explicit rate 1 code for the class of bit-wise independent tampering function was proposed by [CG14b]. Note that a tampering function in this class tampers each bit independently and is subsumed by our work and a companion paper [AGM+14b]. In the construction of [CG14b], they exhaustively search for an encoding scheme (which is guaranteed by [FMVW14]) for messages with logarithmic length. This is a complex procedure (and intuitively obscure) and the compiler which extends the non-malleability to long messages is also complicated. We, on the other hand, begin with a rate 0 code of [AGM+14b] against a more general class of (non-compartmentalized) tampering functions and apply our compiler to obtain rate 1 non-malleable code against the more general class itself.

We remark that preliminary results leading to this work appear in [AGM+14a]. The results of this paper and [AGM+14b] together subsume and significantly extend the results in [AGM+14a].[2]

1.3 Technical Overview

Improved Efficiency via Hybrid Encoding. A recurring theme in cryptographic constructions for improving efficiency (in our setting, efficiency refers to the rate of the code) is a "boot-strapping" or "hybrid" approach. It takes a scheme with strong security (but low efficiency), and combines it with an efficient scheme (with a weak form of security) to obtain an efficient scheme with strong security. Perhaps the most well-known example of this approach in cryptography is that of "hybrid encryption," which improves the efficiency of a (non-malleable) public-key encryption scheme by using it to encrypt a short key for a symmetric-key encryption scheme, and then using the latter to encrypt the actual message (e.g., see [CS03, Kur11]).

The high-level approach in this work, as well as in many works on improving the rate of error-correcting codes and non-malleable codes [GS10, CG14b], fits this template. A basic idea for non-malleable codes in these works involves encoding the message using a high-rate (randomized) code and appending to it a tag that is encoded using an inefficient non-malleable encoding NMC_0. That is, the final codeword has the form $(c, \mathsf{NMC}_0(\tau))$, where c is a (malleable) encoding of the message, and τ consists of some information about c that "binds" c to τ.

Intuitively, the short tag τ should encode some information about the much longer c in a way that makes it hard to change c without changing τ as well, and since the latter is encoded using a non-malleable code, one could hope that the over all code is non-malleable. One such choice of the tag, used in a preliminary version of this result [AGM+14a], is $\tau = (h, h(c))$, where h is a randomly chosen hash function with a short description from a (statistical) collision-resistant hash function family. As shown in [AGM+14a], this suffices for the class of attacks involving permutations (but not allowing the adversary to set/reset the bits). However, when the class of attacks allowed for

[2] In [AGM+14a] a weaker class of tampering functions was considered, which did not contain all bit-wise tampering functions. While a rate-amplification approach for this class was presented there, it was more complicated, and relied on the specifics of the rate-0 construction there. The approach for rate-amplification there breaks down when all bit-wise functions are allowed.

the adversary includes the possibility of the adversary creating an entirely new tag τ^* (and a new purported codeword $\widetilde{c}$ obtained by mauling the original codeword c), this approach fails. This is because, it raises the possibility that the adversary can pick h such that $h(\widetilde{c})$ can be predicted with non-negligible probability, even though the adversary may have some uncertainty about $\widetilde{c}$. While it is plausible that most of the functions h in a function family would lead to unpredictable values of $h(\widetilde{c})$, it appears difficult to rule out there being no such h, or to provide an efficient algorithm for detecting them.

Our Approach: Adding Errors for Non-Malleability. We introduce a novel approach to boot-strapping non-malleability. We first motivate our approach using a loose analogy. Consider a student plagiarising a homework solution, by copying it from an original source, and blindly making a few alterations (without actually comprehending the original solution). The student would try to remove various pieces of identifying information (e.g., change variable names, reorder sentences etc.) and even introduce minor typographical errors, while hoping to make it look approximately correct to the grader. If there is not much variability in correct solutions, then, even if confronted with the original solution, the student will have plausible deniability that she came up with the solution on her own. However, if the original source happened to contain several minor random errors itself (which a grader would have recognized as minor, and ignored), then the chances are that many of them would make their way into the plagiarized solution as well. In this case, it will be unlikely that the student could have introduced those errors on her own, and this will be a strong indication of plagiarization.

While our problem of non-malleability is different, our solution follows the above intuition. Our encoding has the form $\mathsf{Enc}(s) = (\mathsf{ECSS}(s) \oplus e_R, \mathsf{NMC}_0(\tau))$, where now ECSS is a light-weight (rate-1) encoding of s, R is an appropriately sized random subset of indices (say, $|R| = n^\delta$ bits, where $|\mathsf{ECSS}(s)| = n$), e_R is a sparse n-bit vector, with zeros outside of R and uniformly randomly chosen bits in R. The tag τ is a succinct representation of the bits of $\mathsf{ECSS}(s) \oplus e_R$ at the positions in R. Note that $|\tau|$ is much shorter than n (e.g., $O(n^\delta \log n)$), so that (for an appropriate choice of δ), $\mathsf{NMC}_0(\tau)$ will be $o(n)$-bits long. The property we will need from ECSS is that it is an "error-correcting secret-sharing scheme" which is an error-correcting encoding that also behaves as a (ramp) secret-sharing scheme, so that any small subset of the bits in an encoding has values independent of the message it encodes. (Such a code can be readily instantiated using, for example, any linear error-correcting code of appropriate (sub-linear) distance and dual distance.)

To decode (c, σ), the following consistency check is carried out: apply error-correction to obtain a codeword $\widehat{c}$ from c; also decode σ using the decoding of NMC_0 to obtain τ; then ensure that at the locations recorded in τ, c matches the recorded bits, and everywhere else c matches $\widehat{c}$.

In other words, our encoding amounts to adding random errors to the efficiently encoded messages (while allowing error-correction); further, we require this to be accompanied by an "errata" (encoded using a non-malleable code) which lists *all the errors* in the first part. Now, intuitively, if the adversary chooses to create an errata on its own, but creates $\widehat{c}$ by tampering c (i.e., using significantly many bits from c), then it is unlikely that the new errata matches $\widehat{c}$. On the other hand, if the adversary retains the errata from a given codeword, then any significant tampering on c will result in a mismatch; instead, if $\widehat{c}$ is obtained by only lightly tampering c, then, due to the distance of the code the only possibility to obtain a valid encoding is to have $\widehat{c} = c$, and by a simple privacy requirement on the code, the probability of this happening when only a small number of bits in c are involved, is independent of the message.

Formal Analysis. We present a modular proof that the above construction is indeed a non-malleable code against $\mathcal{F}^*$, the family of permutations and bit-wise tampering attacks, by relying on the security of NMC_0 as well as the error-correction and privacy properties of ECSS in a black-box manner. The proof is somewhat simpler, if instead of considering $\mathcal{F}^*$, we considered only $\mathcal{F}_{\mathsf{BIT}}$, the family of bit-wise tampering functions, as in [CG14b] (in which case, NMC_0 need be secure only against this class of tampering functions). Below we sketch this simpler proof, and indicate in footnotes the main points of departure for the full proof.

Formally, we need to argue that for any message s and any admissible attack f, for a randomly constructed codeword $\mathsf{Enc}(s)$, the outcome of $\mathsf{Dec}(f(\mathsf{Enc}(s)))$ is almost identically distributed as a simulated outcome which probabilistically maps f to the original message s, a fixed message distribution M_f, or $\perp$ (with M_f and the probabilities depending only on f). The simulated outcome is defined as follows:

Simulating $\mathsf{Dec}(f(\mathsf{Enc}(s)))$ (Given only f).

Let $\mathcal{L}$ and $\mathcal{R}$ denote the set of indices in our code corresponding to c and $\mathsf{NMC}_0(\tau)$, respectively. Given f, we proceed as follows.

- Define attacks $f^{(1)}$ and $f^{(2)}$ obtained by restricting respectively to $\mathcal{L}$ and $\mathcal{R}$.[3]
- Then, $f^{(2)}$ is simply a bit-wise tampering attack on $\mathsf{NMC}_0(\tau)$. By the security guarantee of NMC_0, we can sample, based only on $f^{(2)}$ (independent of τ), the outcome of $\mathsf{Dec}(f^{(2)}(\mathsf{NMC}_0(\tau)))$ as $\perp$, some string τ^*, or same^* (i.e., τ itself).
- Case simulated outcome of $\mathsf{Dec}(f^{(2)}(\mathsf{NMC}_0(\tau)))$ is $\perp$: Set the simulated outcome of decoding $\mathsf{Dec}_1(f(\mathsf{ECSS}(s), \mathsf{NMC}_0(\tau)))$ to be $\perp$.
- Case simulated outcome of $\mathsf{Dec}(f^{(2)}(\mathsf{NMC}_0(\tau)))$ is τ^*: Let $c' = f^{(1)}(c)$. We consider two sub-cases, depending on the number of bits in c' that are not fixed by the attack $f^{(1)}$ (i.e., the number of bits that depend on the original bit at that position).
  - If the number of bits of c that influence c' is "small," then they can be sampled independent of the message s, by relying on the fact that ECSS is a (ramp) secret-sharing scheme. Then the simulated outcome is obtained by error-correcting c', decoding it and checking for consistency with τ^*.
  - If the number of bits of c' that depend on c is not small, then (following the analogy of plagiarism from above), there is an overwhelming probability that this set of bits contain several random bits and the probability that τ^* correctly records them is negligible. In this case, the simulated outcome is $\perp$.
- Case simulated outcome of $\mathsf{Dec}(f^{(2)}(\mathsf{NMC}_0(\tau)))$ is same^*: In this case, τ remains unchanged. Again we consider two sub-cases, this time depending on the number of untampered bits in the attack $f^{(1)}$.

[3] When permutations are allowed, this is no more possible. Instead, we follow a more elaborate argument in which $f^{(2)}$ is defined after sampling the value of the bits from $\mathcal{L}$ that are moved to $\mathcal{R}$ by the permutation attack. To be able to do this independent of the encoded message, we rely on the error-correcting scheme ECSS being a ramp secret-sharing scheme.

– If there are only a small number of tampered bits in $f^{(1)}$, then $f^{(1)}$ ($\mathsf{ECSS}(s)$) results in a valid codeword iff $f^{(1)}$ has the effect of not altering $\mathsf{ECSS}(s)$. This depends only on the values of the bits of $\mathsf{ECSS}(s)$ which are tampered by $f^{(1)}$ (and τ), which in turn is independent of the message s, due to ECSS being a secret-sharing scheme. Hence we can sample τ and these bits, independent of s. This is used to simulate the outcome being same^* or $\perp$.
– On the other hand, if there are several tampered bits, then we simulate the outcome to be $\perp$.

To argue that the last step results in only a negligible statistical error, we follow an argument similar to the plagiarism argument, but this time relying on the fact that τ is retained as it is, and will have a record of all the actual errors. Consider the set of bits that were tampered by $f^{(1)}$. Except with negligible probability, a significant number of bits in this set would have been recorded in τ. For each such bit, the probability that the tampered bit does not match the bit recorded in τ is at least $\frac{1}{2}$ (it is 1 if the tampering function is a bit-flip, and $\frac{1}{2}$ if it is a set/reset), independent of the other bits. Hence the probability that τ matches all of those bits is negligible.[4] Thus indeed, the outcome of the actual decoding would be $\perp$, except with negligible probability.

2 Preliminaries

We denote the set $\{1, \ldots, n\}$ by $[n]$. Probability distributions are represented by capital letters. The distribution U_S represents a uniform distribution over the set S. Given a distribution X, $x \sim X$ represents that x is sampled according to the distribution X. We shall often use the convention that a realization of a random variable denoted as X will be represented by the variable x.

For a joint variable $X = (X_1, \ldots, X_n)$ and $S = \{i_1, \ldots, i_{|S|}\} \subseteq [n]$, we define the random variable $X_S = (X_{i_1}, \ldots, X_{i_{|S|}})$, where $|S|$ denotes the size of the set S. We use a similar notation for vectors as well, for example x_S represents the vector restricted to indices in the set S. For a function $f(\cdot)$, the random variable $Y = f(X)$ represents the following distribution: sample $x \sim X$ and output $f(x)$. For a randomized algorithm A, we write $A(z)$ to denote the distribution of the output of A on an input z.

The statistical distance between two distributions S and T over a finite sample space I is defined as:

$$\mathrm{SD}\left(S, T\right) := \frac{1}{2} \sum_{i \in I} \left| \Pr_{x \sim S}[x = i] - \Pr_{x \sim T}[x = i] \right|.$$

The hamming distance between two vectors $c, c' \in \{0,1\}^m$ is given by

$$\mathrm{HD}(c, c') := \left| \{ i \in [m] | c_i \neq c'_i \} \right|.$$

A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every positive polynomial $\mathrm{poly}(\cdot)$ and all sufficiently large n, $f(n) \leqslant 1/\mathrm{poly}(n)$. We use $\mathsf{negl}(M)$ to denote an (unspecified) negligible function in M.

[4] When we allow permutations, some amount of correlation can exist between the different tampered bits. For example, if two bits that are recorded got swapped with each other, the probability of not having an error is $\frac{1}{2}$ and not $\frac{1}{4}$. But this is the most extreme example: if k bits recorded in τ have been tampered with, we show that the error probability is at least $1 - (\frac{1}{2})^{k/2}$.

Lastly, all logarithms in this paper would be to the base 2.

2.1 Classes of Tampering Functions

We shall consider the following basic tampering function classes.

1. Family of Permutations. Let $\mathcal{S}_N$ denote the set of all permutations $\pi : [N] \to [N]$. Given an input codeword $x_{[N]} \in \{0,1\}^N$, tampering with function $\pi \in \mathcal{S}_N$ yields the codeword: $x_{\pi^{-1}(1)} \ldots x_{\pi^{-1}(N)}$.
2. Family of Bit-Wise Tampering Functions. This class, represented by $\mathcal{F}_{\mathsf{BIT}}$, contains the following four functions, for a single bit input: a) $f(x) \mapsto x$, b) $f(x) \mapsto 1 \oplus x$, c) $f(x) \mapsto 0$, and d) $f(x) \mapsto 1$. These functions are, respectively, called *forward*, *toggle*, *reset* and *set* functions.

We define a more complex tampering function class $\mathcal{F}_{\mathsf{BIT}} \circ \mathcal{S}_N$ to consist of tampering functions of the form $f = (f_1, \ldots, f_N, \pi)$, where $\pi \in \mathcal{S}_N$ and $f_i \in \mathcal{F}_{\mathsf{BIT}}$, and

$$f(x_{[N]}) = f_{\pi^{-1}(1)}(x_{\pi^{-1}(1)}) \ldots f_{\pi^{-1}(N)}(x_{\pi^{-1}(N)}).$$

That is, to apply f to x, first we apply f_i to each position x_i, and apply the permutation π to the resulting string. Our main result provides an efficient rate-1 non-malleable code against this class.

2.2 Error-Correcting Secret-Sharing Scheme

In this section, we define error-correcting secret-sharing schemes that will be used in our construction.

Definition 1 (Error-Correcting Secret-Sharing Scheme (ECSS))**.** *Let $S = (X_0, X_1, \ldots, X_M)$ be a joint distribution over $\Lambda \times \{0,1\}^M$, such that the support of X_0 is all of Λ. (The random variable X_0 represents the secret being shared and X_i for $i \in [M]$ represents the i-th share.)*

We say that S is an $[M, L, T, D]$-error-correcting secret-sharing scheme *if $\log|\Lambda| = L$, and the following conditions hold:*

1. *T-privacy: $\forall\ s, s' \in \Lambda$, $\forall\ J \subseteq [M]$ such that $|J| \leqslant T$, we have*
$$\mathrm{SD}\left((X_J|X_0 = s), (X_J|X_0 = s')\right) = 0.$$
2. *D-error-correction: For any two distinct $c, c' \in \mathsf{Supp}(X_{[M]})$, the hamming distance between them $\mathrm{HD}(c, c') > 2d$, where $\mathsf{Supp}(X_{[M]})$ denotes the support of distribution $X_{[M]}$.*
3. *Reconstruction: For any $s, s' \in \Lambda$ such that $s \neq s'$, we have*
$$\mathrm{SD}\left((X_{[M]}|X_0 = s), (X_{[M]}|X_0 = s')\right) = 1.$$

In the remainder of the paper, by an ECSS scheme, we shall implicitly refer to a family of ECSS schemes indexed by M, i.e., $[M, L(M), T(M), D(M)]$-ECSS schemes for each positive integer M. We define the *rate* of such a scheme to be $\lim_{M\to\infty} \frac{L(M)}{M}$. We will be interested in *efficient* ECSS schemes. For this, we define three algorithms associated with such a scheme.

- $\mathsf{Enc}_{\mathsf{ECSS}}(s)$: This is a randomized algorithm that takes $s \in \Lambda$ as input and outputs a sample from the distribution $(X_{[M]}|X_0 = s)$.
- $\mathsf{ECorr}_{\mathsf{ECSS}}(\tilde{c})$: This algorithm takes a $\tilde{c} \in \{0,1\}^M$ as input, and outputs a $c \in \mathsf{Supp}(X_{[M]})$ such that $\mathrm{HD}(c, \tilde{c}) \leqslant D$. If such a c does not exist, it outputs $\perp$.
- $\mathsf{Rec}_{\mathsf{ECSS}}(c)$: This algorithm takes a $c \in \{0,1\}^M$ as input, and outputs a secret $s \in \Lambda$ such that $c \in \mathsf{Supp}(X_{[M]}|X_0 = s)$. If such a secret does not exist, it outputs $\perp$.

Note that the uniqueness of the output of algorithms $\mathsf{ECorr}_{\mathsf{ECSS}}$ and $\mathsf{Rec}_{\mathsf{ECSS}}$ is guaranteed by the D-error-correction and reconstruction properties respectively. An ECSS scheme is said to be *efficient* if the three algorithms defined above run in time bounded by a polynomial in M.

2.3 Non-malleable codes

In Figure 1 we present the definition of an $[N, L, \nu]$-non-malleable code against a family of tampering functions $\mathcal{F}$.

Let $\mathcal{F}$ be a set of functions of the form $f : \{0,1\}^N \to \{0,1\}^N$. Consider two mappings $\mathsf{Enc} : \{0,1\}^L \to \{0,1\}^N$ (possibly randomized) and $\mathsf{Dec} : \{0,1\}^N \to \{0,1\}^L \cup \{\perp\}$.

For $f \in \mathcal{F}$ and $s \in \{0,1\}^L$, define a random variable $\mathsf{Tamper}_f^{(s)}$ over $\{0,1\}^L \cup \{\perp\}$ as follows:

$$\mathsf{Tamper}_f^{(s)} = \mathsf{Dec}(f(\mathsf{Enc}(s))).$$

Let Sim be a map from $\mathcal{F}$ to distributions over the sample space $\{0,1\}^L \cup \{\mathsf{same}^*, \perp\}$. For $f \in \mathcal{F}$ and $s \in \{0,1\}^L$, define the random variable $\mathsf{Copy}_{\mathsf{Sim}(f)}^{(s)}$ as follows.

$$\mathsf{Copy}_{\mathsf{Sim}(f)}^{(s)} = \begin{cases} s & \text{if } \mathsf{Sim}(f) = \mathsf{same}^* \\ \mathsf{Sim}(f) & \text{otherwise.} \end{cases}$$

The simulation error (or, advantage) is defined to be:

$$\mathsf{adv}_{\mathsf{Enc},\mathsf{Dec},\mathcal{F}} := \inf_{\mathsf{Sim}} \max_{\substack{s \in \{0,1\}^L \\ f \in \mathcal{F}}} \mathrm{SD}\left(\mathsf{Tamper}_f^{(s)}, \mathsf{Copy}_{\mathsf{Sim}(f)}^{(s)}\right)$$

$(\mathsf{Enc}, \mathsf{Dec})$ is called an $[N, L, \nu]$-*non-malleable code against* $\mathcal{F}$ if the following conditions hold:

- Correctness: $\forall s \in \{0,1\}^L$, $\Pr[\mathsf{Dec}(\mathsf{Enc}(s)) = s] = 1$.
- Non-Malleability: $\mathsf{adv}_{\mathsf{Enc},\mathsf{Dec},\mathcal{F}} \leqslant \nu(N)$.

We say that the coding scheme is *efficient* if Enc and Dec run in time bounded by a polynomial in N.

Figure 1: Definition of Non-Malleable Codes

2.4 Concentration Bound

The following concentration bound will be useful in our proof. Below, we write $a \neq b \pm \varepsilon$ to mean $a \notin [b-\varepsilon, b+\varepsilon]$.

Lemma 1 (Tail Inequality for Hypergeometric Distribution [Hoe63, Chv79]). *Let $c \in (0, 1/2)$ be a constant, $m, n \in \mathbb{N}$ and $m \in [cn, (1-c)n]$. Let X be a random variable distributed uniformly over all n-bit strings with exactly m 1s. For every $t \in \mathbb{N}$, we have*

$$\Pr_{x \sim X}\left[\sum_{i \in [t]} x_i \neq t\left(\frac{m}{n} \pm \varepsilon\right)\right] \leqslant 2\exp\left(-\mathrm{D}_{\mathrm{KL}}\left(\frac{m}{n}+\varepsilon, \frac{m}{n}\right)\cdot t\right) \leqslant 2\exp(-\varepsilon^2 t/3),$$

where $\mathrm{D}_{\mathrm{KL}}(\alpha, \beta) := \alpha \ln \frac{\alpha}{\beta} + (1-\alpha)\ln\frac{1-\alpha}{1-\beta}$.

3 Construction and Proof

In this section, we shall prove our main theorem:

Theorem 3 (Compiler). *Suppose there exists a $[t', t, \nu_0]$ non-malleable code NMC_0 against the tampering class $\mathcal{F}_{\mathsf{BIT}} \circ \mathcal{S}_{t'}$, with $t' \leqslant t^d$ for some constant $d \geqslant 1$, and an $[M, L, T, D]$ binary error-correcting secret-sharing scheme ECSS. Then there exists an $[N, L, \nu_1]$ non-malleable code NMC_1 against the tampering class $\mathcal{F}_{\mathsf{BIT}} \circ \mathcal{S}_N$ with $N \leqslant M + M^{d/(d+1)}\log^{2d} M$ and $\nu_1(N) = \mathsf{negl}(M) + \nu_0(N-M)$, if $T, D \geqslant 2M^{d/(d+1)}\log^{2d} M$. Further, if NMC_0 and ECSS are efficient, then NMC_1 is also efficient.*

Note that above $N = M(1+o(1))$. If $M = L(1+o(1))$, then the code is a rate-1 code. Also, $\mathrm{poly}(N) < N - M < N$, and hence if ν_0 is a negligible function, so is ν_1.

Our compiler is described in Figure 2. When properly instantiated (see Section 3.2) we can obtain our main results Corollary 1 and Corollary 2.

Discussion. Note that our compiler is a fully black-box compiler, in that both the components are used in fully black-box manner and the security of our compiler directly reduces to the security of its components in a black-box manner. Further, the output code is explicit if both ECSS and NMC_0 are explicit; and encoding and decoding of the new code is efficient if both its components are efficient.

3.1 Proof of Main Theorem

To achieve the parameters stated in the proof, we shall use the following parameters in our construction in Figure 2: $T, D \geqslant 2M^{d/(d+1)}\log^{2d} M$, $N^{(1)} = M$, and $p_e = M^{-d/(d+1)}$. From the construction, we get $|\tau| \leqslant B(\log M + 1) = 2p_e M \log 2M$. Using this we get, $N^{(2)} \leqslant |\tau|^d \leqslant 2^d M^{d/(d+1)}\log^d 2M < M^{d/(d+1)}\log^{2d} M$.

We shall interpret the codeword produced by Figure 2 as a two-part codeword. The *left-part* is a share-packing based on the ECSS and the *right-part* has the non-malleable encoding of τ using NMC_0. For ease of notation, let $\mathcal{L} = [N^{(1)}]$ and $\mathcal{R} = [N] \setminus \mathcal{L}$.

Ingredients:

1. An $[M, L, T, D]$ binary error correcting secret sharing scheme ECSS with encoding, error-correcting and reconstruction algorithms $\mathsf{Enc}_{\mathsf{ECSS}}, \mathsf{ECorr}_{\mathsf{ECSS}}$ and $\mathsf{Rec}_{\mathsf{ECSS}}$ respectively.
2. A $[t', t, \nu_0]$ non-malleable coding scheme NMC_0 against $\mathcal{F}_{\mathsf{BIT}} \circ \mathcal{S}_{t'}$ with encoding and decoding algorithms $\mathsf{Enc}_{\mathsf{NMC}_0}$ and $\mathsf{Dec}_{\mathsf{NMC}_0}$ respectively, with $t' \leqslant t^d$ for some constant $d \geqslant 1$. We require that $T, D \geqslant 2M^{d/(d+1)} \log^{2d} M$.

$\mathsf{Enc}(s \in \{0,1\}^L)$:

1. Let $N^{(1)} = M$, $p_e = (N^{(1)})^{-d/(d+1)}$ and $B = 2p_e N^{(1)}$. Sample a random subset $E = \{i_1, \ldots, i_B\}$ of $[N^{(1)}]$ of size B.
2. Draw $a^{(1)} \sim \mathsf{Enc}_{\mathsf{ECSS}}(s)$. For $i \in [N^{(1)}]$, define
$$c_i^{(1)} = \begin{cases} a_i^{(1)} & \text{if } i \notin E \\ a_i^{(1)} + e_i & \text{otherwise,} \end{cases}$$
where $e_i \sim U_{\{0,1\}}$.
3. Let $\tau = (E, c_E^{(1)})$, where $c_E^{(1)} = (c_{i_1}^{(1)}, \cdots, c_{i_B}^{(1)})$.
4. Draw $c^{(2)} \sim \mathsf{Enc}_{\mathsf{NMC}_0}(\tau)$. Note that $N^{(2)} := |c^{(2)}| \leqslant |\tau|^d$.
5. Output $(c^{(1)}, c^{(2)})$ as the codeword.

$\mathsf{Dec}(\tilde{c} \in \{0,1\}^N)$:

1. Interpret $\tilde{c}$ as $(\tilde{c}^{(1)}, \tilde{c}^{(2)})$ of length $N^{(1)}$ and $N^{(2)}$, respectively.
2. Let $\tau^* = \mathsf{Dec}_{\mathsf{NMC}_0}(\tilde{c}^{(2)})$.
3. $\mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$:
   (a) If $\tau^* = \perp$, output $\perp$ and halt.
   (b) Let $a^* = \mathsf{ECorr}_{\mathsf{ECSS}}(\tilde{c}^{(1)})$. If $a^* = \perp$, output $\perp$ and halt.
   (c) Let $\tilde{s} = \mathsf{Rec}_{\mathsf{ECSS}}(a^*)$. If $\tilde{s} = \perp$, output $\perp$ and halt.
   (d) Interpret τ^* as (E^*, r^*), where positions in r^* are indexed by elements in E^*. More formally, if $E^* = \{i_1, \ldots, i_B\}$, then $r^* = (r_{i_1}^*, \ldots, r_{i_B}^*)$. Let c^* be defined as follows: for $i \in [N^{(1)}]$, $c_i^* = a_i^*$ if $i \notin E^*$, and otherwise $c_i^* = r_i^*$. If $c^* \neq \tilde{c}^{(1)}$, output $\perp$; else, output $\tilde{s}$.

Figure 2: Compiler for Rate-1 Non-Malleable Code

Consider the $[t', t, \nu_0]$ non-malleable coding scheme NMC_0 against $\mathcal{F}' = \mathcal{F}_{\mathsf{BIT}} \circ \mathcal{S}_{t'}$ with encoding and decoding algorithms $\mathsf{Enc}_{\mathsf{NMC}_0}$ and $\mathsf{Dec}_{\mathsf{NMC}_0}$ respectively. For $\tau \in \{0,1\}^t$ and $f' \in \mathcal{F}'$, the random

variable $\mathsf{Tamper}^{(\tau)}_{f',\mathsf{NMC}_0}$ over $\{0,1\}^t \cup \{\perp\}$ with respect NMC_0 is given by

$$\mathsf{Tamper}^{(\tau)}_{f',\mathsf{NMC}_0} = \mathsf{Dec}_{\mathsf{NMC}_0}(f'(\mathsf{Enc}_{\mathsf{NMC}_0}(\tau))).$$

The non-malleability of NMC_0 guarantees that there exists a map Sim_0 from $\mathcal{F}'$ to distributions over the sample space $\{0,1\}^t \cup \{\mathsf{same}^*, \perp\}$ such that

$$\mathrm{SD}\left(\mathsf{Tamper}^{(\tau)}_{f',\mathsf{NMC}_0}, \mathsf{Copy}^{(\tau)}_{\mathsf{Sim}_0(f')}\right) \leqslant \nu_0(t').$$

for all $\tau \in \{0,1\}^t$ and $f' \in \mathcal{F}'$. See Figure 1 for the definition of non-malleable codes. We use an additional subscript in the notation of Tamper function to distinguish it from the one we define below for NMC_1.

Fix a tampering function $f = (f_1, \ldots, f_N, \pi) \in \mathcal{F}^* = \mathcal{F}_{\mathsf{BIT}} \circ \mathcal{S}_N$ and a message s. The random variable $\mathsf{Tamper}^{(s)}_{f,\mathsf{NMC}_1}$ over $\{0,1\}^L \cup \{\perp\}$ for the non-malleable coding scheme NMC_1 described in Figure 2 (with encoding and decoding functions Enc and Dec respectively) is given by

$$\mathsf{Tamper}^{(s)}_{f,\mathsf{NMC}_1} = \mathsf{Dec}(f(\mathsf{Enc}(s))).$$

Our goal is to show that there exists a map Sim_1 from $\mathcal{F}^*$ to distributions over the sample space $\{0,1\}^L \cup \{\mathsf{same}^*, \perp\}$ such that

$$\mathrm{SD}\left(\mathsf{Tamper}^{(s)}_{f,\mathsf{NMC}_1}, \mathsf{Copy}^{(s)}_{\mathsf{Sim}_1(f)}\right) \leqslant \nu_1(N).$$

(It is easy to see that NMC_1 is correct.)

We provide a description of Sim_1 in the next section and show that $\mathsf{Copy}^{(s)}_{\mathsf{Sim}_1(f)}$ is statistically close to $\mathsf{Tamper}^{(s)}_{f,\mathsf{NMC}_1}$ after that.

3.1.1 Description of simulator

Given $f = (f_1, \ldots, f_N, \pi) \in \mathcal{F}^*$, we define the following set of indices in $[N]$ which will be used in the description of the simulator (and later in the proof):

- Let $X \subseteq \mathcal{L}$ be the set of all indices which move from left to right as a result of applying the permutation π, i.e., $X = \{i \in \mathcal{L} \mid \pi(i) \in \mathcal{R}\}$.
- Similarly, let $Y = \{i \in \mathcal{R} \mid \pi(i) \in \mathcal{L}\}$ be the indices which move from right to left. Note that $|X| = |Y|$.
- Let $\overline{X} = \mathcal{L} \setminus X$.
- Let $J \subseteq \overline{X}$ such that for all $i \in J$, $f_i \in \{f_{\mathsf{forward}}, f_{\mathsf{toggle}}\}$. In other words, J is the set of all indices on the left which are mapped into the left codeword using f_{forward} or f_{toggle}.
- Let $V \subseteq \overline{X}$ such that for all $i \in V$, either $\pi(i) \neq i$ or $f_i \neq f_{\mathsf{forward}}$.
- Let $\overline{V} = \overline{X} \setminus V$. In other words, $\overline{V}$ denotes the set of indices on the left which are not tampered.

Observe that only the bits at indices X in $c^{(1)}$ affect the tampered right codeword $\tilde{c}^{(2)}$. Hence, given $c^{(1)}_X$, we can construct a tampering function $f^{(2)} \in \mathcal{F}_{\mathsf{BIT}} \circ \mathcal{S}_{t'}$ which acts on the right codeword. Let $\rho : Y \to X$ be an arbitrary bijection from Y to X. The function $f^{(2)} = (f^{(2)}_1, \ldots, f^{(2)}_{N^{(2)}}, \pi^{(2)})$ is given by

$$f^{(2)}_{i-N^{(1)}} = \begin{cases} f_i & \text{if } i \in \mathcal{R} \setminus Y \\ f_{\mathsf{reset}} & \text{if } i \in Y \text{ and } f_{\rho(i)}(c^{(1)}_{\rho(i)}) = 0 \\ f_{\mathsf{set}} & \text{if } i \in Y \text{ and } f_{\rho(i)}(c^{(1)}_{\rho(i)}) = 1 \end{cases} \tag{1}$$

$$\pi^{(2)}(i - N^{(1)}) = \begin{cases} \pi(i) - N^{(1)} & \text{if } i \in \mathcal{R} \setminus Y \\ \pi(\rho(i)) - N^{(1)} & \text{otherwise} \end{cases}$$

for $i \in \mathcal{R}$.

Let $\mathcal{D}_{\mathsf{ECSS}}$ be the distribution of $\mathsf{Enc}_{\mathsf{ECSS}}(s')$ for a random $s' \in \{0,1\}^L$. We are now ready to describe how the distribution $\mathsf{Sim}_1(f)$ is generated:

(**1**) Set τ, $c^{(1)}_X$ and $f^{(2)}$ as follows:

First sample E as described in Step 1 of Figure 2. Let $\tau = (E, r)$ where $r \sim U_{\{0,1\}^B}$ (i.e., r is a random bit-string of length B). The bits in r are indexed by the indices in E. Also, sample $a^{(1)} \sim \mathcal{D}_{\mathsf{ECSS}}$. Together τ and $a^{(1)}$ is used to determine $c^{(1)}_X$: for $i \in X$, let $c^{(1)}_i = a^{(1)}_i$ if $i \notin E$; otherwise, $c^{(1)}_i = r_i$. Finally, define $f^{(2)}$ using $c^{(1)}_X$, as described in Equation 1. (If the number of error indices in X, i.e. $|X \cap E|$, is larger than $O(\log^2 M)$, output $\perp$ and stop. Using Lemma 1, we can show that this happens with only $\mathsf{negl}(M)$ probability.)

(**2**) Draw $\theta \sim \mathsf{Sim}_0(f^{(2)})$. Let $\tau^* = \mathsf{Copy}^{(\tau)}_{\theta}$.

(**3**) Set $c^{(2)}$ as follows:

$c^{(2)}$ is drawn from the output distribution of $\mathsf{Enc}_{\mathsf{NMC}_0}(\tau)$ conditioned on $\mathsf{Dec}_{\mathsf{NMC}_0}(f^{(2)}(c^{(2)})) = \tau^*$ [5]. If no such codeword exists, then the simulator fails.

(**4**) θ obtained in Step 2 could be an element in $\{0,1\}^t$, same^* or $\perp$. Let $\mathsf{Event}_{\mathsf{fix}}$, $\mathsf{Event}_{\mathsf{same}^*}$ and $\mathsf{Event}_{\perp}$ denote the corresponding events. Do the following based on which event takes place:

(a) Case $\mathsf{Event}_{\perp}$: Output $\perp$ and stop.

(b) Case $\mathsf{Event}_{\mathsf{fix}}$: We have the following two sub cases, based on the size of the set J. Let $\alpha = M^{d/(d+1)} \log^2 M$.

i. Case $|J| > \alpha$: Output $\perp$ and stop.

ii. Case $|J| \leqslant \alpha$: Extend the definition of $c^{(1)}_X$ using $a^{(1)}, r$ from Step 1 to $c^{(1)}_J$ as well: i.e., for $i \in J$, define $c^{(1)}_i = a^{(1)}_i$ if $i \notin E$; otherwise, $c^{(1)}_i = r_i$. Combined with $c^{(2)}$ defined in the previous step, this defines a unique $\tilde{c}^{(1)}$ because for any $i \in \mathcal{L}$, $\tilde{c}^{(1)}_i$ is either 0 or 1 or has $\pi^{-1}(i) \in J \cup Y$. Output $\mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$ and stop (where Dec^* is as defined in Step 3 of the description of Dec, see Figure 2).

(c) Case $\mathsf{Event}_{\mathsf{same}^*}$: Define $n_{\mathsf{non\text{-}id}} := |V|$. We have the following two cases based on $n_{\mathsf{non\text{-}id}}$. Let $\beta = N^{(2)} + M^{d/(d+1)} \log^2 M$.

[5] It may not be possible to do this reverse sampling step efficiently. However, an efficient simulation is implied by the existence of an inefficient simulation; see Remark 1 in [CG14b].

i. Case $n_{\mathsf{non\text{-}id}} > \beta$: Output $\perp$ and stop.

ii. Case $n_{\mathsf{non\text{-}id}} \leqslant \beta$: Extend the definition of $c_X^{(1)}$ using $a^{(1)}, r$ from Step 1 to $c_V^{(1)}$ as well: i.e., for $i \in V$, define $c_i^{(1)} = a_i^{(1)}$ if $i \notin E$; otherwise, $c_i^{(1)} = r_i$. Combined with $c^{(2)}$ defined in the previous step, this defines a unique $\tilde{c}_{V\cup X}^{(1)}$ because for any $i \in V \cup X$, $\pi^{-1}(i) \in V \cup Y$. Output same^* if $\tilde{c}_{V\cup X}^{(1)} = c_{V\cup X}^{(1)}$, else output $\perp$.

3.1.2 Hybrids and their indistinguishability

Recall that we want to show that the distributions $\mathsf{Tamper}_{f,\mathsf{NMC}_1}^{(s)}$ and $\mathsf{Copy}_{\mathsf{Sim}_1(f)}^{(s)}$ are statistically close to each other. Towards this, we first define four intermediate hybrids, (described below, and summarized, for quick reference, in Figure 3), and show that for every $f \in \mathcal{F}^*$ and $s \in \{0,1\}^L$,

$$\mathsf{Tamper}_{f,\mathsf{NMC}_1}^{(s)} \equiv \mathsf{Hyb0}_f^{(s)} \equiv \mathsf{Hyb1}_f^{(s)} \equiv \mathsf{Hyb2}_f^{(s)} \approx \mathsf{Hyb3}_f^{(s)} \approx \mathsf{Copy}_{\mathsf{Sim}_1(f)}^{(s)}.$$

Of these, the first equality is because the experiment defining $\mathsf{Hyb0}_f^{(s)}$ is identical to that defining $\mathsf{Tamper}_{f,\mathsf{NMC}_1}^{(s)}$, but restated in a convenient form for comparison with the following hybrids. The second equality relies on the privacy of the ECSS code. The third equality follows since in defining $\mathsf{Hyb2}_f^{(s)}$ we merely change the order in which two random variables are sampled in $\mathsf{Hyb1}_f^{(s)}$, taking care to not change their distributions. The statistical difference between $\mathsf{Hyb2}_f^{(s)}$ and $\mathsf{Hyb3}_f^{(s)}$ will be bounded by $\nu_0(N-M)$ using the non-malleability of NMC_0. Finally, we upper bound the statistical difference between $\mathsf{Hyb3}_f^{(s)}$ and $\mathsf{Copy}_{\mathsf{Sim}_1(f)}^{(s)}$ by $\mathsf{negl}(M)$, relying on the privacy and distance of ECSS and the "errata" technique. This will show that the statistical distance between $\mathsf{Tamper}_{f,\mathsf{NMC}_1}^{(s)}$ and $\mathsf{Copy}_{\mathsf{Sim}_1(f)}^{(s)}$ is at most $\mathsf{negl}(M) + \nu_0(N-M) = \nu_1(N)$, thus proving Theorem 3.

We now discuss the above steps in detail.

Tamper vs. Hybrid 0: Our first claim, which is easy to verify, is that $\mathsf{Tamper}_f^{(s)}$ is identically distributed to $\mathsf{Hyb0}_f^{(s)}$ defined below.

1. Sample E as described in Step 1 of Figure 2. Let $\tau = (E, r)$ where $r \sim U_{\{0,1\}^B}$. (The bits in r are indexed by the indices in E.) Sample $a^{(1)} \sim \mathsf{Enc}_{\mathsf{ECSS}}(s)$. For $i \in \mathcal{L}$, let $c_i^{(1)} = a_i^{(1)}$ if $i \notin E$; otherwise, $c_i^{(1)} = r_i$. With $c^{(1)}$ defined, construct a tampering function $f^{(2)}$ as described before.
2. Sample $c^{(2)} \sim \mathsf{Enc}_{\mathsf{NMC}_0}(\tau)$. Let $\tilde{c}^{(2)} = f^{(2)}(c^{(2)})$.
3. Let $\tau^* = \mathsf{Dec}_{\mathsf{NMC}_0}(\tilde{c}^{(2)})$.
4. For $i \in \overline{X} \cup Y$, let $\tilde{c}_{\pi(i)}^{(1)} = f_i(c_i)$, where $\overline{X} = \mathcal{L} \setminus X$. Note that this completely defines $\tilde{c}^{(1)}$ because $\pi(\overline{X} \cup Y) = \mathcal{L}$[6]. Output $\mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$.

[6]Here there is a slight abuse of notation: $\pi(S)$ for any $S \subseteq [N]$ should be interpreted as the set $\{\pi(i) \mid i \in S\}$.

Hybrid 0:

τ

(1) $c^{(1)} \mid \tau, s$

$f^{(2)} \mid c^{(1)}, f$

(2) $c^{(2)} \mid \tau$

(3) $\tau^* = \mathsf{Dec}_{\mathsf{NMC}_0}(f^{(2)}(c^{(2)}))$

(4) $\tilde{c}^{(1)} \mid c^{(1)}, c^{(2)}, f$

$\mathsf{Hyb0}_f^{(s)} = \mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$

Hybrid 1:

τ

(1) $(\star)$ $c_X^{(1)} \mid \tau$

$(\star)$ $f^{(2)} \mid c_X^{(1)}, f$

(2) $c^{(2)} \mid \tau$

(3) $\tau^* = \mathsf{Dec}_{\mathsf{NMC}_0}(f^{(2)}(c^{(2)}))$

$(\star)$ $c^{(1)} \mid c_X^{(1)}, \tau, s$

(4) $\tilde{c}^{(1)} \mid c^{(1)}, c^{(2)}, f$

$\mathsf{Hyb1}_f^{(s)} = \mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$

Hybrid 2:

τ

(1) $c_X^{(1)} \mid \tau$

$f^{(2)} \mid c_X^{(1)}, f$

(2) $(\star)$ $\tau^* = \mathsf{Dec}_{\mathsf{NMC}_0}(f^{(2)}(\mathsf{Enc}_{\mathsf{NMC}_0}(\tau)))$

(3)$(\star)$ $c^{(2)} \mid \tau, \mathsf{Dec}_{\mathsf{NMC}_0}(f^{(2)}(c^{(2)})) = \tau^*$

$c^{(1)} \mid c_X^{(1)}, \tau, s$

(4) $\tilde{c}^{(1)} \mid c^{(1)}, c^{(2)}, f$

$\mathsf{Hyb2}_f^{(s)} = \mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$

Hybrid 3:

τ

(1) $c_X^{(1)} \mid \tau$

$f^{(2)} \mid c_X^{(1)}, f$

(2) $(\star)$ $\theta \sim \mathsf{Sim}_0(f^{(2)})$

$(\star)$ $\tau^* = \mathsf{Copy}_\theta^{(\tau)}$

(3) $c^{(2)} \mid \tau, \mathsf{Dec}_{\mathsf{NMC}_0}(f^{(2)}(c^{(2)})) = \tau^*$

$c^{(1)} \mid c_X^{(1)}, \tau, s$

(4) $\tilde{c}^{(1)} \mid c^{(1)}, c^{(2)}, f$

$\mathsf{Hyb3}_f^{(s)} = \mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$

Figure 3: For each hybrid experiment, the order in which the random variables are sampled is defined below. Notation $a \mid b$ denotes that a is sampled conditioned on b. The definition of the random variables is as in the description of the simulator. All the hybrids are parametrized by a function $f \in \mathcal{F}^*$ and a message $s \in \{0,1\}^L$. Items with a $(\star)$ before them indicate differences from the previous hybrid.

Note that in sampling $\mathsf{Tamper}^{(s)}_{f,\mathsf{NMC}_1} = \mathsf{Dec}(f(\mathsf{Enc}(s)))$, the same steps as above are carried out, but in a different order: first $(c^{(1)}, c^{(2)})$ are sampled, then $(\tilde{c}^{(1)}, \tilde{c}^{(2)})$ are obtained, then τ^* is generated, and finally $\mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$ is output.

Hybrid 0 vs. Hybrid 1: Recall that $\mathcal{D}_{\mathsf{ECSS}}$ is the distribution of $\mathsf{Enc}_{\mathsf{ECSS}}(s')$ for a random $s' \in \{0,1\}^L$. In $\mathsf{Hyb1}^{(s)}_f$, Steps 1 and 4 change as described below (and the other two steps stay the same):

1. Sample $\tau = (E, r)$ in the same way as before. Sample $a^{(1)} \sim \mathcal{D}_{\mathsf{ECSS}}$. For $i \in X$, define $c^{(1)}_i = a^{(1)}_i$ if $i \notin E$; otherwise, $c^{(1)}_i = r_i$. Define $f^{(2)}$ using $c^{(1)}_X$.

4. Compute rest of $c^{(1)}$ consistent with $c^{(1)}_X$, τ and s. More formally, sample $b^{(1)}$ from the distribution $\mathsf{Enc}_{\mathsf{ECSS}}(s)$ conditioned on bits at indices X being $a^{(1)}_X$. For $i \in \overline{X}$, define $c^{(1)}_i = b^{(1)}_i$ if $i \notin E$; otherwise, $c^{(1)}_i = r_i$. Compute $\tilde{c}^{(1)}$ as described before and output $\mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$.

Effectively, $\mathsf{Hyb1}^{(s)}_f$ splits the sampling of $c^{(1)}$ into two parts. Only bits at indices in X are computed in Step 1, which is enough to define $f^{(2)}$. In order to show that $\mathsf{Tamper}^{(s)}_f$ and $\mathsf{Hyb1}^{(s)}_f$ are identically distributed, we must prove that $c^{(1)}_X$ can be sampled without knowledge of s as described in modified Step 1. This is indeed the case, since $|X| \leqslant N^{(2)} \leqslant T$, the privacy parameter of ECSS.

Hybrid 1 vs. Hybrid 2: We now define $\mathsf{Hyb2}^{(s)}_f$ which is a slightly different way of interpreting $\mathsf{Hyb1}^{(s)}_f$. Observe that the Steps 2 and 3 in $\mathsf{Hyb1}^{(s)}_f$ generate a τ^* from the distribution $\mathsf{Tamper}^{(\tau)}_{f^{(2)},\mathsf{NMC}_0}$. In $\mathsf{Hyb2}^{(s)}_f$, we use this distribution to modify the following steps (rest of the steps remain unchanged):

2. Let $\tau^* \sim \mathsf{Tamper}^{(\tau)}_{f^{(2)},\mathsf{NMC}_0}$. Recall that $\mathsf{Tamper}^{(\tau)}_{f^{(2)},\mathsf{NMC}_0} = \mathsf{Dec}_{\mathsf{NMC}_0}(f^{(2)}\ (\mathsf{Enc}_{\mathsf{NMC}_0}(\tau)))$.

3. Sample a random codeword $c^{(2)} \sim \mathsf{Enc}_{\mathsf{NMC}_0}(\tau)$ such that $\mathsf{Dec}_{\mathsf{NMC}_0}(f^{(2)}(c^{(2)})) = \tau^*$.

It is clear that $\mathsf{Hyb1}^{(s)}_f$ is identically distributed to $\mathsf{Hyb2}^{(s)}_f$.

Hybrid 2 vs. Hybrid 3: The next hybrid – $\mathsf{Hyb3}^{(s)}_f$ – is same as $\mathsf{Hyb2}^{(s)}_f$ but with one difference:

- In Step 2, draw $\theta \sim \mathsf{Sim}_0(f^{(2)})$. Let $\tau^* = \mathsf{Copy}^{(\tau)}_\theta$.

In order to show that $\mathsf{Hyb2}^{(s)}_f$ and $\mathsf{Hyb3}^{(s)}_f$ are statistically close, we use the non-malleability of NMC_0. Consider an adversary $\mathcal{A}$ who, when given f and s as inputs, runs Steps 1 of $\mathsf{Hyb2}^{(s)}_f$ (or $\mathsf{Hyb3}^{(s)}_f$) to obtain τ and $f^{(2)}$, and sends them to a challenger. The challenger replies back with either $\mathsf{Tamper}^{(\tau)}_{f^{(2)},\mathsf{NMC}_0}$ or $\mathsf{Copy}^{(\tau)}_{\mathsf{Sim}_0(f^{(2)})}$; let τ^* denote this response. When adv receives τ^*, it runs Steps 3 and 4 of $\mathsf{Hyb2}^{(s)}_f$ (or $\mathsf{Hyb3}^{(s)}_f$), and outputs whatever the output of Step 4 is. It is easy to see that

if challenger responds with $\mathsf{Tamper}^{(\tau)}_{f^{(2)},\mathsf{NMC}_0}$, then output of $\mathcal{A}$ is identically distributed to $\mathsf{Hyb2}^{(s)}_f$, and otherwise it is identically distributed to $\mathsf{Hyb3}^{(s)}_f$. We know that for all $f^{(2)} \in \mathcal{F}'$ and $\tau \in \{0,1\}^t$, $\mathrm{SD}\left(\mathsf{Tamper}^{(\tau)}_{f^{(2)},\mathsf{NMC}_0}, \mathsf{Copy}^{(\tau)}_{\mathsf{Sim}_0(f^{(2)})}\right) \leqslant \nu_0(t')$. Hence, $\mathrm{SD}\left(\mathsf{Hyb2}^{(s)}_f, \mathsf{Hyb3}^{(s)}_f\right) \leqslant \nu_0(t') = \nu_0(N-M)$.

Hybrid 3 vs. Simulator: The final step in the proof is to show that the distributions $\mathsf{Hyb3}^{(s)}_f$ and $\mathsf{Copy}^{(s)}_{\mathsf{Sim}_1(f)}$ are statistically close. $\mathsf{Copy}^{(s)}_{\mathsf{Sim}_1(f)}$ is generated by the simulator as defined in Section 3.1.1, except that in Step 4-(c)-ii, "output same^*" is replaced with "output s." Then we note that the first three steps in the experiment defining $\mathsf{Hyb3}^{(s)}_f$ and that defining $\mathsf{Copy}^{(s)}_{\mathsf{Sim}_1(f)}$ (or $\mathsf{Sim}_1(f)$) are identical[7].

Before proceeding further, we restate Step 4 of $\mathsf{Hyb3}^{(s)}_f$ here for the convenience of the reader:

- Sample $c^{(1)}_{\overline{X}}$ consistent with $c^{(1)}_X$, τ and s. Compute $\tilde{c}^{(1)}$ as follows: for $i \in \overline{X} \cup Y$, let $\tilde{c}^{(1)}_{\pi(i)} = f_i(c_i)$. Combined with $c^{(2)}$ from the previous step, this completely defines $\tilde{c}^{(1)}$ because $\pi(\overline{X} \cup Y) = \mathcal{L}$. Output $\mathsf{Dec}^*(\tilde{c}^{(1)}, \tau^*)$.

Consider the following case analysis based on the events defined by the value of θ. For each case, we show that the output of $\mathsf{Copy}^{(s)}_{\mathsf{Sim}_1(f)}$ is same as that of $\mathsf{Hyb3}^{(s)}_f$, except with probability $\mathsf{negl}(M)$.

1. Case $\mathsf{Event}_\perp$: In this case $\tau^* = \perp$, so $\mathsf{Hyb3}^{(s)}_f$ outputs $\perp$, just like $\mathsf{Copy}^{(s)}_{\mathsf{Sim}_1(f)}$ does.

2. Case $\mathsf{Event}_{\mathsf{fix}}$: We have the following two cases based on $|J|$. Recall that $J = \{i \in \mathcal{L} \mid \pi(i) \in \mathcal{L} \text{ and } f_i \in \{f_{\mathsf{forward}}, f_{\mathsf{toggle}}\}\}$.

- Case $|J| > \alpha$: In this case, we show that $\Pr[\mathsf{Hyb3}^{(s)}_f \neq \perp] \leqslant \mathsf{negl}(M)$ over randomly chosen $\tau = (E, r)$ (conditioned on $c^{(1)}_X$). First, we define some notation. Let $Z \subseteq \mathcal{L}$ be set of indices $i \in \mathcal{L}$ such that $\pi^{-1}(i) \in \mathcal{R}$, i.e., Z is the set of indices on the left which come from the right. Let $\tau^* = (E^*, r^*)$, where the bits in r^* are indexed by the indices in E^*. (We know that $\tau^* \in \{0,1\}^t$ since $\theta \in \{0,1\}^t$). Let F_τ be the indices $i \in \mathcal{L}$ such that $\pi(j) = i$ and $j \in E$, i.e., j is an erroneous index according to original tag τ.

  Fix any ECSS codeword $a^{(1)}$ for the left consistent with $c^{(1)}_X$. We first show that, irrespective of the value of τ, there exists at most one ECSS codeword a^* such that the tampered left codeword $\tilde{c}^{(1)}$ is consistent with a^* and τ^*. More precisely, for any two possible values of τ — τ_1 and τ_2 — let $\hat{c}^{\tau_1}$ and $\hat{c}^{\tau_2}$ be the corresponding values of $\tilde{c}^{(1)}$. Suppose $\mathsf{Dec}^*(\hat{c}^{\tau_1}, \tau^*) \neq \perp$ and $\mathsf{Dec}^*(\hat{c}^{\tau_2}, \tau^*) \neq \perp$. Let $\mathsf{ECorr}_{\mathsf{ECSS}}(\hat{c}^{\tau_1}) = a^*_1$ and $\mathsf{ECorr}_{\mathsf{ECSS}}(\hat{c}^{\tau_2}) = a^*_2$. Then, we argue that there is an element $c \in \{0,1\}^M$ such that $\mathrm{HD}(a^*_1, c), \mathrm{HD}(a^*_2, c) \leqslant D$. Hence, by the correctness of $\mathsf{ECorr}_{\mathsf{ECSS}}$, we have $a^*_1 = a^*_2$.

[7] There is one minor difference in Step 1 though: $\mathsf{Sim}_1(f)$ outputs $\perp$ if the number of error indices in X is large, whereas $\mathsf{Hyb3}^{(s)}_f$ doesn't. However, this only adds a negligible amount of error. (See description of $\mathsf{Sim}_1(f)$ for more details.)

To prove this, we shall let $c = \hat{c}^{\tau_1}$. Then $\mathrm{HD}(a_1^*, c) \leqslant B < D$, since $\mathsf{Dec}^*(c, \tau^*) \neq \perp$ (note that in the last step of Dec^* this is ensured). Similarly, $\mathrm{HD}(a_2^*, \hat{c}^{\tau_2}) \leqslant B$. Then,

$$\mathrm{HD}(a_2^*, c) \leqslant \mathrm{HD}(a_2^*, \hat{c}^{\tau_2}) + \mathrm{HD}(c, \hat{c}^{\tau_2}) \leqslant B + 2B + N^{(2)},$$

where we used the fact that, since $\hat{c}^{\tau_1}_{\mathcal{L}\setminus Z}$ and $\hat{c}^{\tau_2}_{\mathcal{L}\setminus Z}$ are derived from the same value of $a^{(1)}$ by adding at most B errors each, followed by applying a tampering function that cannot increase the hamming distance, their hamming distance is at most $2B$, and $|Z| \leqslant N^{(2)}$. By the choice of our parameters, $3B + N^{(2)} \leqslant D$, as required.

Thus, $\mathsf{Hyb3}_f^{(s)} \neq \perp$ only if $\mathsf{ECorr}_{\mathsf{ECSS}}(\tilde{c}^{(1)}) = a^*$, for a value a^* which is fixed independent of τ. Further, since $\tau^* = (E^*, r^*)$ is fixed, there is a unique c^* as in the description of Dec^* in Figure 2, and it must be the case that $\tilde{c}^{(1)} = c^*$.

Next, we show that over randomness of τ (conditioned on $a^{(1)}, c_X^{(1)}$), the probability that tampered codeword $\tilde{c}^{(1)} = c^*$ is negligible in M.

Over randomness of τ, we know that number of error indices in J w.r.t. τ are $\Omega(\log^2 M)$ with $1 - \mathsf{negl}(M)$ probability (see Lemma 1 and recall that the number of error indices in X is only $O(\log^2 M)$). Let these error indices be G.

Note that for a random τ, the value of erroneous indices outside X is uniform. This implies that each bit in $c_G^{(1)}$ is independent uniform bit (even after fixing $a^{(1)}$, $c_X^{(1)}$ and all other bits of $c_G^{(1)}$). Moreover, by definition of set J, for any $i \in G$, we have $\pi(i) \in \mathcal{L}$ (since $J \cap X = \emptyset$) and the bit $\tilde{c}^{(1)}_{\pi(i)} = f_i(c_i^{(1)})$ is a uniform random bit (since $f_i \in \{f_{\mathsf{forward}}, f_{\mathsf{toggle}}\}$). Hence $\Pr[\tilde{c}^{(1)}_{\pi(i)} = c^*_{\pi(i)}] = 1/2$, even conditioned on $\tilde{c}^{(1)}_{\pi(j)} = c^*_{\pi(j)}$ for $j \in G \setminus \{i\}$. Therefore, probability that $\tilde{c}^{(1)}_{\pi(G)} = c^*_{\pi(G)}$ is at most $2^{-\Omega(\log^2 M)} = \mathsf{negl}(M)$.

- Case $|J| \leqslant \alpha$: In this case, $\mathsf{Copy}^{(s)}_{\mathsf{Sim}_1(f)}$ and $\mathsf{Hyb3}_f^{(s)}$ behave in the same way except the manner in which they generate $c_J^{(1)}$ (rest of $c^{(1)}$ is not important). In the former case, $c_J^{(1)}$ is sampled from $\mathcal{D}_{\mathsf{ECSS}}$ (conditioned on $c_X^{(1)}$), while in the latter case it is sampled from $\mathsf{Enc}_{\mathsf{ECSS}}(s)$ (again conditioned on $c_X^{(1)}$). This, however, makes no difference because T, the privacy parameter of ECSS, is at least $N^{(2)} + \alpha \geqslant |X| + |J|$.

3. Case $\mathsf{Event}_{\mathsf{same}^*}$: Since $\theta = \mathsf{same}^*$, we know that $\tau^* = \tau$. We have the following two cases based on $n_{\mathsf{non\text{-}id}} = |V|$. Recall that $V = \{i \in \mathcal{L} \mid \pi(i) \in \mathcal{L} \text{ and } (\pi(i) \neq i \text{ or } f_i \neq f_{\mathsf{forward}})\}$.

- Case $n_{\mathsf{non\text{-}id}} > \beta$: In this case, we show that $\Pr[\mathsf{Hyb3}_f^{(s)} \neq \perp] \leqslant \mathsf{negl}(M)$ over a random choice of $\tau = (E, r)$ (conditioned on $c_X^{(1)}$). In fact, we will show that for any ECSS codeword $a^{(1)}$ for the left (consistent with $c_X^{(1)}$), $\mathsf{Hyb3}_f^{(s)} = \perp$ with $1 - \mathsf{negl}(M)$ probability. Fix an ECSS codeword $a^{(1)}$ for the left consistent with $c_X^{(1)}$. Rest of the analysis will be over the randomness of τ, conditioned on $c_X^{(1)}$ and $a^{(1)}$.

  Recall that while sampling τ in Step 1, we begin by sampling a random set of error indices E. We consider the size of the set $V \cap E \setminus Z$, where $Z = \{i \in \mathcal{L} | \pi^{-1}(i) \in \mathcal{R}\}$. Over the random choice of E, we have $|V \cap E \setminus Z| = \Omega(\log^2 M)$ with $1 - \mathsf{negl}(M)$ probability. This follows from Lemma 1, because $|V \setminus Z| \geqslant \beta - N^{(2)} = M^{d/(d+1)} \log^2 M$ and the number of error indices in

X is only $O(\log^2 M)$. We identify a set $U \subseteq V \cap E \setminus Z$, such that $|U| \geqslant |V \cap E \setminus Z|/2$ and for each $i \in U$, $\Pr[\tilde{c}_i^{(1)} = c_i^{(1)}] \leqslant 1/2$, even conditioned on $\tilde{c}_{i'}^{(1)} = c_{i'}^{(1)}$, for any set of $i' \in U \setminus \{i\}$. We build U iteratively: initialize $U = \emptyset$ and $W = V \cap E \setminus Z$. Pick any index $i \in W$, and let $j = \pi^{-1}(i)$. Update U to $U \cup \{i\}$ and W to $W \setminus \{i, j\}$, and repeat this until W is empty.

Clearly, $|U| \geqslant |V \cap E \setminus Z|/2$. To verify the other property of U, note that at any step, when we pick $i \in W$, and let $j = \pi^{-1}(i)$, we have either 1) $j = i$ and $f_i \in \{f_{\mathsf{toggle}}, f_{\mathsf{reset}}, f_{\mathsf{set}}\}$ or 2) $j \neq i$ (since $i \in V$) but $j \in \mathcal{L}$ (since $i \notin Z$). Thus $\tilde{c}_i^{(1)} = f_i(c_i^{(1)})$ for $f_i \in \{f_{\mathsf{toggle}}, f_{\mathsf{reset}}, f_{\mathsf{set}}\}$, or $\tilde{c}_i^{(1)} = f_j(c_j^{(1)})$ for $j = \pi^{-1}(i) \notin U$. Since $i \in E$, each bit $c_i^{(1)}$ is uniformly random and hence $\Pr[\tilde{c}_i^{(1)} = c_i^{(1)}] \leqslant 1/2$. (This probability is 0 if $\pi(i) = i$ and $f_i = f_{\mathsf{toggle}}$; otherwise it is exactly $1/2$.) Further, the conditions $\tilde{c}_i^{(1)} = c_i^{(1)}$ for $i \in U$ are independent of each other, since each such condition involves either $c_i^{(1)}$ alone or a pair $(c_i^{(1)}, c_j^{(1)})$ such that $c_j^{(1)}$ never occurs in any other condition.

Thus, $\Pr[\tilde{c}_E^{(1)} = c_E^{(1)}] \leqslant \Pr[\tilde{c}_U^{(1)} = c_U^{(1)}] \leqslant 1/2^{|V \cap E \setminus Z|/2} \leqslant \mathsf{negl}(M)$. But to be a valid left codeword, it has to be the case that $\tilde{c}_E^{(1)} = c_E^{(1)}$, since $\tau^* = \tau$, which records $c_E^{(1)}$. Hence, $\Pr[\mathsf{Hyb3}_f^{(s)} \neq \perp] \leqslant \mathsf{negl}(M)$.

- Case $n_{\mathsf{non\text{-}id}} \leqslant \beta$: We know that most of the indices on the left have been copied identically to the tampered codeword. We use this to argue that if the tampered codeword $\tilde{c}^{(1)}$ is valid w.r.t. τ then $\tilde{c}^{(1)} = c^{(1)}$, and that the probability of this happening depends only on a small number of bits in $c^{(1)}$ and hence independent of the message s. More formally, we have the following.

  Let $a^{(1)}$ be an ECSS encoding of s consistent with $c_X^{(1)}$. The maximum number of errors in the tampered codeword $\tilde{c}^{(1)}$ could be $|V| + |X| + |E| \leqslant \beta + N^{(2)} + 2p_e N^{(1)} \leqslant D$, where D is the error-correction radius of the ECSS code. Hence, $a^* = \mathsf{ECorr}_{\mathsf{ECSS}}(\tilde{c}^{(1)})$, computed in the algorithm Dec^* by $\mathsf{Hyb3}_f^{(s)}$, must be equal to $a^{(1)}$. Further, since $\tau^* = \tau$, $c^* = c^{(1)}$. Therefore, unless $c^{(1)} = \tilde{c}^{(1)}$, the output of $\mathsf{Hyb3}_f^{(s)}$ would be $\perp$.

  Since indices in $\overline{V}$ have been copied identically, $c^{(1)} = \tilde{c}^{(1)}$ iff $c_{V \cup X}^{(1)} = \tilde{c}_{V \cup X}^{(1)}$. This is the exact check which Sim_1 performs, by sampling $c_{V \cup X}^{(1)}$ itself. Note that Sim_1 generates $c_{V \cup X}^{(1)}$ from the same distribution as in the experiment for $\mathsf{Hyb3}_f^{(s)}$, since the privacy parameter T of ECSS is at least $N^{(2)} + \beta \geqslant |X| + |V|$. Hence, in this case, Sim_1 outputs same^* with the same probability as $\mathsf{Hyb3}_f^{(s)} = s$; otherwise, both the random variables are $\perp$.

This completes the proof.

3.2 Instantiations

We shall use the following results:

1. There exists a $[t', t, \nu_0]$ non-malleable code NMC_0 against $\mathcal{F}_{\mathsf{BIT}} \circ \mathcal{S}_{t'}$, where $t' \leqslant t \cdot \mathrm{polylog}\, t$ and $\nu_0(\cdot)$ is a suitable negligible function. This follows from the work of [AGM+14b]. For simplicity, we set $d = 2$ so that we have $t' < t^d$.

2. Using Shamir's secret-sharing scheme [Sha79], with a standard share-packing technique [BM85, FY92], we can obtain an efficient $[M, L, T, D]$ ECSS such that: $M = n\varphi$, $L = \ell\varphi$ and $T = D = (n-\ell)/2$, where $\varphi = 2\log n$. We can choose $n - \ell = n^{3/4}$ so that $T, D = \widetilde{\Theta}(M^{3/4}) \geqslant 2M^{d/(d+1)} \log^{2d} M$, for $d = 2$.

   The secret sharing scheme is formally described as follows. We choose a field $\mathbb{F}$ of characteristic 2 such that $\varphi = |\mathbb{F}| \geqslant n^2$. Pick arbitrary elements in the field $\mathbb{F}$ and name them $S = \{-\ell, \ldots, -1, 1, \ldots, n\}$. The secret sharing scheme is constructed by picking a random $\ell + (n-\ell)/2$ degree polynomial $p(\cdot)$ and evaluating it at elements in S. The share X_0 is the concatenation of the evaluations of $p(\cdot)$ at $\{-\ell, \ldots, -1\}$ and interpreting it as a binary string. The share X_i, for $i \in [M]$, is the ith bit in the concatenation of evaluations of $p(\cdot)$ at $\{1, \ldots, n\}$.

   Note that we can also use Algebraic Geometric code [Gop81, GS96] based share-packing techniques [CC06] over constant size field with characteristic 2. But we forgo this optimization for ease of presentation and simplicity of the resulting code.

Using the above choices for NMC_0 and ECSS, we get the following rate of NMC_1 in Theorem 3.

$$\frac{L}{N} \geqslant 1 - N^{-1/4}\text{polylog}\, N.$$

This directly yields Corollary 1 and Corollary 2.

References

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *STOC*, pages 774–783, 2014.

[AGM+14a] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes resistant to permutations. Cryptology ePrint Archive, Report 2014/316, 2014. http://eprint.iacr.org/.

[AGM+14b] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes resistant to permutations and perturbations. Cryptology ePrint Archive, Report 2014/841, 2014. http://eprint.iacr.org/.

[BM85] G.R. Blakley and Catherine Meadows. Security of ramp schemes. In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 242–268. Springer Berlin Heidelberg, 1985.

[CC06] Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO*, pages 521–536, 2006.

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

[CG14a] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In Moni Naor, editor, *ITCS*, pages 155–168. ACM, 2014.

[CG14b] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *TCC*, 2014.

[Chv79] Vasek Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285 – 287, 1979.

[CKM11] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: Built-in tamper resilience. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 740–758. Springer, 2011.

[CKO14] Nishanth Chandran, Bhavana Kanukurthi, and Rafail Ostrovsky. Locally updatable and locally decodable codes. In Yehuda Lindell, editor, *TCC*, volume 8349 of *Lecture Notes in Computer Science*, pages 489–514. Springer, 2014.

[CPX14] Ronald Cramer, Carles Padró, and Chaoping Xing. Optimal algebraic manipulation detection codes, 2014.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[CZ14] Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. Electronic Colloquium on Computational Complexity, Report 2014/102, 2014. http://eccc.hpi-web.de/.

[DKO13] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 239–257. Springer, 2013.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS*, pages 434–452. Tsinghua University Press, 2010.

[FMNV14] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *TCC*, pages 465–488, 2014.

[FMVW14] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In *EUROCRYPT*, pages 111–128, 2014.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 699–710. ACM, 1992.

[Gop81] Valerii Denisovich Goppa. Codes on algebraic curves. In *Soviet Math. Dokl*, pages 170–172, 1981.

[GS96] Arnaldo Garcia and Henning Stichtenoth. On the asymptotic behaviour of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248–273, 1996.

[GS10] Venkatesan Guruswami and Adam Smith. Codes for computationally simple channels: Explicit constructions with optimal rate. In *FOCS*, pages 723–732. IEEE Computer Society, 2010.

[HO08] Brett Hemenway and Rafail Ostrovsky. Public-key locally-decodable codes. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 126–143. Springer, 2008.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):pp. 13–30, 1963.

[Kur11] Kaoru Kurosawa. Hybrid encryption. In *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 570–572. 2011.

[Lip94] Richard J. Lipton. A new approach to information theory. In *STACS*, pages 699–708, 1994.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2012.

[MPSW05] Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal error correction against computationally bounded noise. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

[OPS07] Rafail Ostrovsky, Omkant Pandey, and Amit Sahai. Private locally decodable codes. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 2007.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11), November 1979.