Explicit Optimal-Rate 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 various classes of tampering. 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.

Keywords: Non-malleable Codes, Explicit Construction, Information Theoretic, Rate 1.

[*]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.

[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.

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 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] (unpublished). 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

[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.

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 [AGM$^+$14a] is $\tau = (h, h(c))$, where h is a randomly chosen hash function with a short description, and is a (statistical) collision-resistant hash function. 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 the adversary includes the possibility of the adversary creating an entirely new tag τ^* (and a new purported codeword $\widetilde{c}$ obtained by malleating 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 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 $h(\widetilde{c})$, but in general, 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^*.

[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 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 $f^{(1)}$.
    - 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 matches 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 $\{1, \dots, n\}$ 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 $X = (X_1, \dots, X_n)$ and $S = \{i_1, \dots, i_{|S|}\} \subseteq [n]$, we define the random variable $X_S = (X_{i_1}, \dots, X_{i_{|S|}})$. 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.

[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 $1 - (\frac{1}{2})^{k/2}$.

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

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

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

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

negl stands for an (unspecified) negligible function. All logarithms are to the base 2.

We write $f(x_{[n]})$, where the domain of f consists of single elements in the vector x, as a shorthand for the vector $(f(x_1)\dots f(x_n))$.

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 following codeword: $x_{\pi^{-1}(1)}\dots x_{\pi^{-1}(N)} =: x_{\pi^{-1}([N])}$.
2. Family of Bit-Wise Tampering Functions. This class, represented by $\mathcal{F}_{\{0,1\}}$, 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 $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ to consist of tampering functions of the form $f = (f_1, \dots, f_N \in, \pi)$, where $\pi \in \mathcal{S}_N$ and $f_i \in \mathcal{F}_{\{0,1\}}$, and

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

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 $S = (X_0, X_1, \dots, X_m)$ 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 $[m, \ell, t, d]$-error-correcting secret-sharing scheme *if* $\log|\Lambda| = \ell$*, and the following conditions hold:*

1. *t-privacy: $\forall\, s, s' \in \Lambda$, $\forall\, T \subseteq [m]$ such that $|T| \leqslant t$, we have*
$$\mathrm{SD}\left((X_T|X_0 = s), (X_T|X_0 = s')\right) = 0.$$
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., $[m, \ell(m), t(m), d(m)]$-ECSS schemes for each positive integer m. We define the *rate* of such a scheme to be $\lim_{m\to\infty} \frac{\ell(m)}{m}$. 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 $\mathcal{F}$.

2.4 Concentration Bound

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

Lemma 1 (Tail Inequality for Hypergeometric Distribution [Hoe63, Chv79])**.** *Let $c \in (0, 1)$ be a constant, $m, n \in \mathbb{N}$ and $m \in [cn, (1 - c)n]$. Let $X_{[n]} = U_{\binom{[n]}{m}}$, where $\binom{[n]}{m}$ denotes the set of all m-sized subsets of $[n]$. For every $t \in \mathbb{N}$, we have:*
$$\Pr_{x_{[n]}\sim X_{[n]}}\left(\sum_{i\in[t]} x_i = t\left(\frac{m}{n} \pm \varepsilon\right)\right) \leqslant 2\exp\left(-\mathrm{D}_{\mathrm{KL}}\left(\frac{m}{n} + \varepsilon, \frac{m}{n}\right) \cdot t\right) \leqslant 2\exp(-\varepsilon^2 t/3),$$
where $\mathrm{D}_{\mathrm{KL}}(\alpha, \beta) := \alpha \ln \frac{\alpha}{\beta} + (1 - \alpha) \ln \frac{1-\alpha}{1-\beta}$.

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 $\mathcal{D}$ 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 $\mathsf{Sim}_{\mathcal{D}(f)}^{(s)}$ as follows.

$$\mathsf{Sim}_{\mathcal{D}(f)}^{(s)} = \mathsf{copy}(\mathcal{D}(f), s),$$

where $\mathsf{copy} : (\{0,1\}^L \cup \{\mathsf{same}^*, \perp\}) \times \{0,1\}^L \to \{0,1\}^L \cup \{\perp\}$ is defined as

$$\mathsf{copy}(\alpha, \beta) = \begin{cases} \beta & \text{if } \alpha = \mathsf{same}^* \\ \alpha & \text{otherwise.} \end{cases}$$

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

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

$(\mathsf{Enc}, \mathsf{Dec})$ 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: $\mathsf{adv}_{\mathsf{Enc},\mathsf{Dec},\mathcal{F}} \leqslant \nu(L)$.

Figure 1: Definition of Non-Malleabile Codes

3 Construction and Proof

In this section, we shall prove our main theorem:

Theorem 3 (Compiler)**.** *If there exists a* $[t^c, t, \nu_0]$ *non-malleable code* NMC_0 *against the tampering class* $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_t$ *and an* $[M, L, T, D]$ *binary error-correcting secret-sharing scheme* ECSS*; then there exists an* $[L, N, \nu_1]$ *non-malleable code against the tampering class* $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ *with* $N = M + M^{1/(1+c)} \log^{2/c} M$ *and* $\nu_1(N) = \mathsf{negl}(M) + \nu_0(N - M)$*, if* $T, D \geqslant M^{1/(1+c)} \log^{2/c} M$.

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.

Note that if there exists a fast decoding algorithm for ECSS scheme up to D errors, then the decoding algorithm of our scheme benefits from it. Further, if $T, D \geqslant M^{1/(1+c)} \log^{2/c} M$, then the simulation goes through with error $\mathsf{negl}(M) + \nu_0(M^{1/(1+c)} \log^{2/c} M)$, where $\nu_0(\cdot)$ is the simulation error of NMC_0 code.

Discussion. Note that our compiler is a fully black-box compiler, as in 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 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 have efficient encoding and decoding procedures.

3.1 Proof of Main Theorem

In this section we shall prove Theorem 3. Suppose we are given an $[M, L, T, D]$-ECSS scheme with $T, D \geqslant M^{1/(1+c)} \log^{2/c} M$, where c is as in the theorem statement. In the proof below, we shall use the following parameters: $N^{(1)} = M$, $p_e = M^{-1/(1+c)}$, $N^{(2)} = M^{1/(1+c)} \log^{2/c} M$, and $\alpha = \beta = M^{1/(1+c)} \log^2 M$.

We shall interpret the codeword produced by Figure 2 as a two-part code. 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. Let $(C^{(1)}|s)$ denote the distribution of the left codeword conditioned on the message being s using the binary error correcting secret sharing scheme ECSS. And $(C^{(2)}|\tau)$ is the distribution of the right codeword when the encoded message is τ using the non-malleable encoding scheme NMC_0.

The definition of non-malleable codes can be interpreted as follows. There exists a simulator Sim which takes as input the tampering function f. Subsequently, it outputs one of the following three events: $\mathsf{Event}_{\mathsf{same}^*}$, $\mathsf{Event}_{\mathsf{fix}}$ and $\mathsf{Event}_{\perp}$. $\mathsf{Event}_{\mathsf{same}^*}$ corresponds to the event that $\mathcal{D}(f)$ has output same^*. $\mathsf{Event}_{\mathsf{fix}}$ corresponds to the event that $\mathcal{D}(f)$ has output s^* using a fixed distribution over the message space. $\mathsf{Event}_{\perp}$ corresponds to the event that $\mathcal{D}(f)$ has output $\perp$. Note that the probability of each of these events is independent of the message s.

To prove Theorem 3, given the simulator Sim_0 for the NMC_0 code, it suffices to construct the simulator Sim_1 using Sim_0. Suppose we are given a tampering function $f \in \mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ and $f \equiv (f_1, \ldots, f_N, \pi)$. Let $\mathcal{L} = [N^{(1)}]$ and $\mathcal{R} = [N] \setminus [N^{(1)}]$. Let $X \subseteq \mathcal{L}$ be the set of all indices $i \in \mathcal{L}$ such that $\pi(i) \in \mathcal{R}$, i.e. the function f takes index from the left code to the right code.

Ingredients:

1. ECSS be an $[M, L, T, D]$ binary error correcting secret sharing scheme with encoding, error-correcting and reconstruction algorithms $\mathsf{Enc}_{\mathsf{ECSS}}, \mathsf{ECorr}_{\mathsf{ECSS}}, \mathsf{Rec}_{\mathsf{ECSS}}$ respectively.

2. $\mathsf{NMC}_0 = (\mathsf{Enc}_{\mathsf{NMC}_0}, \mathsf{Dec}_{\mathsf{NMC}_0})$ be a $[t^c, t, \nu_0]$ non-malleable encoding scheme against $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_t$, where t will be defined below.

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

1. $\mathsf{Enc}(s) = (c^{(1)}, c^{(2)})$, where $N^{(1)} := \left|c^{(1)}\right| = M$ and $N^{(2)} := \left|c^{(2)}\right| = t$, defined as follows:

2. Sample $E = \{i_1, \dots, i_B\}$, as follows. Let $\delta = c/(1+c)$, $p_e = (N^{(1)})^\delta / N^{(1)}$ and $B = 2p_e N^{(1)}$. Let $x \in \{0,1\}^{N^{(1)}}$ be a vector where each index $i \in [N^{(1)}]$ is set to 1 with probability p_e; otherwise it is set to 0. Let $E = \{i : i \in [N^{(1)}] \text{ and } x_i = 1\}$; if $|E| > B = 2p_e N^{(1)}$, then truncate E to first B indices and if $|E| < B$, then repeat elements in E to increase its size to B.

3. $c^{(1)}$ is defined as follows: Let $a^{(1)} \sim \mathsf{Enc}_{\mathsf{ECSS}}(s)$. For $i \in [N^{(1)}]$, define $c_i^{(1)} = a_i^{(1)}$, if $i \notin E$. Otherwise, $c_i^{(1)} = a_i^{(1)} + e_i$, where $e_i \sim U_{\{0,1\}}$.

4. Let $\tau = (E, c_E^{(1)})$, where $c_E^{(1)} = (c_{i_1}^{(1)}, \cdots, c_{i_B}^{(1)})$. Let $|\tau|$ represent the bit length of τ. Let $t = |\tau|^{1/c}$.

5. Let $c^{(2)} \sim \mathsf{Enc}_{\mathsf{NMC}_0}(\tau)$.

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

1. Interpret $c \equiv (c^{(1)}, c^{(2)})$ of length $N^{(1)}$ and $N^{(2)}$, respectively.

2. Let $\tau = \mathsf{Dec}_{\mathsf{NMC}_0}(c^{(2)})$. If $\tau = \perp$, output $\perp$ and halt.

3. Let $\hat{c} = \mathsf{ECorr}_{\mathsf{ECSS}}(c^{(1)})$. If $\hat{c} = \perp$, output $\perp$ and halt.

4. Let $s = \mathsf{Rec}_{\mathsf{ECSS}}(\hat{c})$. If $s = \perp$, output $\perp$ and halt.

5. Interpret τ as (E, z_E), where positions in z_E are indexed by elements in E. Let c' be defined as follows: for $i \in [N^{(1)}]$, $c'_i = \hat{c}_i$ if $i \notin E$ and otherwise $c'_i = z_i$. If $c' \neq c^{(1)}$, output $\perp$. Else, output s.

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

Note that in our scheme the probability that there are more than $2p_e N^{(1)}$ erroneous indices is $\mathsf{negl}(M)$. So, we always assume that τ has exactly $2p_e N^{(1)}$ erroneous indices (by padding, if necessary). We shall view the generation of the tag as, first, generating the indices where the codeword $c^{(1)}$ will be sampled and then copying those entries into the tag.

Step 1. Sample $c_X^{(1)} \sim C_X^{(1)} \equiv (C_X^{(1)}|s)$. This distribution is independent of s because the privacy of the share-packing scheme using ECSS is $T \geqslant N^{(2)} \geqslant |X|$ and the error is generated independently of s.

Step 2. Given $c_X^{(1)}$, define the function $f^{(2)} \in \mathcal{F}_{\{0,1\}} \circ \mathcal{S}_{N^{(2)}}$ as follows. First, $f^{(2)}$ includes the restriction of function f which takes both inputs and outputs only from $\mathcal{R}$. Second, we consider the restriction of f which take indices in $\mathcal{L}$ to $\mathcal{R}$, i.e. restriction of f which begin from the set of indices i such that $i \in X$. For any such $i \in X$, the value of $c_X^{(1)}$ the output of $\pi(f_i(c_i^{(1)}))$ is either 0 or 1; so we interpret it as f_{reset} or f_{set} in $f^{(2)}$, respectively.

Step 3. Run Sim_0 with function $f^{(2)}$ to obtain the event $\mathsf{Event}_{\mathsf{same}^*}$, $\mathsf{Event}_{\mathsf{fix}}$ or $\mathsf{Event}_{\perp}$.

Step 4. If $\mathsf{Event}_{\perp}$ occurs, then we (as Sim_1) output $\mathsf{Event}_{\perp}$ and stop.

Step 5. (Else) If $\mathsf{Event}_{\mathsf{fix}}$ occurs, then consider the following case analysis. Suppose Sim_0 outputs τ^* as the tampered tag.

Let $\overline{X} = [N^{(1)}] \setminus X$. Define S as the set of all indices $i \in \overline{X}$ such that $\pi(i) \in \mathcal{L}$ and $f_i \in \{f_{\mathsf{forward}}, f_{\mathsf{toggle}}\}$. That is, S is the set of all indices which are mapped from left code into the left code by the tampering function f using f_{forward} or f_{toggle} on them.

Step 5.a. If $|S| \leqslant \alpha$, then do the following. Since, Sim_0 outputs τ^*, this implies that the probability of an encoding of τ being consistent with τ^* and $f^{(2)}$ does not depend on τ. This means that sampling τ consistent with prior values (i.e. s, $c_X^{(1)}$, f and tampered tag being an encoding of τ^*) is equivalent to sampling random τ unconditionally.

So, sample τ randomly. Sample a codeword $c^{(2)}$ which is consistent with $c_X^{(1)}$, f and decoding of $\tilde{c}^{(2)}$ begin τ^*. Note that these samplings are independent of s. Let $E \subseteq \mathcal{L}$ be the indices in the original left codeword which have errors and thus are already fixed by τ, i.e. τ fixed $c_E^{(1)}$.

Sample $c_S^{(1)} \sim (C_S^{(1)}|c_X^{(1)}, \tau) \equiv (C_S^{(1)}|c_X^{(1)}, \tau, s)$. This is independent of s because $T \geqslant N^{(2)} + (2p_e N^{(1)}) + \alpha \geqslant |X| + |E| + |S|$.

Note that now $\tilde{c}^{(1)}$ is fixed, because any index $i \in \mathcal{L}$ whose value is not always 0 or 1 in the tampered codeword either has $\pi^{-1}(i) \in \mathcal{R}$ or $\pi^{-1}(i) \in S$. Check whether $\tilde{c}^{(1)}$ is a valid codeword and is consistent with τ^*. If check passes, then we output $\mathsf{Event}_{\mathsf{fix}}$ with associated message s^*, which is the decoded message of $\tilde{c}^{(1)}$; otherwise we output $\mathsf{Event}_{\perp}$. Stop the simulator.[5]

Step 5.b. If $|S| > \alpha$, then do the following. Output $\mathsf{Event}_{\perp}$ and stop.

[5] *Remark:* If there exists a valid s^*, then this value s^* is always identical across different values of τ^* because our choice of (error correctible) distance of the share-packing scheme is $D > N^{(2)} + (2p_e N^{(1)}) + \alpha \geqslant |Y| + |E| + |\pi(S)|$, where Y is the set of indices $i \in \mathcal{L}$ such that $\pi^{-1}(i) \in \mathcal{R}$ (i.e. those indices which are brought from right code to the left code by f). So, there is at most one valid left message s^* because the distance of the left code is too large.

The argument for correctness of this simulation is provided below. We shall argue that this simulation step incurs only $\mathsf{negl}(M)$ simulation error. Our proof shall show that: For every original left codeword and for random tag τ, the tampered codeword is valid with negligible probability.

Compute S as above and define $V = \pi(S)$.

Since Sim_0 outputs τ^*, this implies that the probability of an encoding of τ being consistent with τ^* and $f^{(2)}$ does not depend on τ. This means that the following two distributions are identical: a) Random τ conditioned on being consistent with $c^{(1)}_X$, tampering function f and the tampered tag being an encoding of τ^*, and b) Random τ. So, sample τ (unconditionally).

The following argument is over randomly chosen τ.

Let Z be those indices $i \in \mathcal{L}$ such that $\pi^{-1}(i) \in \mathcal{R}$, i.e. f brings the index from the right codeword to the left codeword. Let G_{τ^*} be the set of all indices corresponding to τ^*. Let F_τ be the indices $i \in \mathcal{L}$ such that $\pi(j) = i$ and j is an erroneous index according to original tag τ.

Consider two tags τ_1 and τ_2 in the original codeword with corresponding set of error indices being E_1 and E_2, respectively. Pick any $c^{(1)}_{\mathcal{L}\setminus(X\cup E_1\cup E_2)}$ consistent with prior sampled values. If for two different $\tau = \tau_1$ and $\tau = \tau_2$ the tampered codeword is valid, then the $\tilde{a}^{(1)}$ must be identical for both these cases. This is because $D \geqslant N^{(2)} + 3(2p_eN^{(1)}) \geqslant |Z| + (|G_{\tau^*}| + |F_{\tau_1}| + |F_{\tau_2}|)$; and the tampered codeword is identical at all indices $i \in \mathcal{L} \setminus (Z \cup G_{\tau^*} \cup F_{\tau_1} \cup F_{\tau_2})$ when the tag is either τ_1 or τ_2.

Consider a fixed $a^{(1)}$. Let S' be a subset of S of size $\log^2 M/p_e$, where $T \geqslant N^{(2)} + (2p_eN^{(1)}) + (\log^2 M/p_e) \geqslant |X| + |E| + |S'|$ and E is the set of erroneous indices according to τ.

Over the choices of random τ, we have the following property. With probability $1 - \mathsf{negl}(M)$, the number of indices in S' which correspond to erroneous indices according to τ is $\Omega(p_e\,|S'|) = \Omega(\log^2 M)$ (by Lemma 1). Suppose $\pi^{-1}(i)$ is an index in a erroneous field according to τ. Recall that $\tilde{a}_i = a_i$; but $\tilde{c}_i$ is inconsistent with τ^* with probability $1/2$ (because $\tilde{c}_i \oplus \tilde{a}_i$ is a uniform bit). Therefore, the probability that all errors at indices $\pi(S')$ are consistent with τ^* it at most $2^{-\Omega(p_e|S'|)} = \mathsf{negl}(M)$.

Step 6. (Else) If $\mathsf{Event}_{\mathsf{same}^*}$ occurs, then consider the following case analysis.

Let V be the set of indices $i \in \mathcal{L}$ such that $f_{\pi^{-1}(i)} \neq f_{\mathsf{forward}}$ or $\pi^{-1}(i) \neq i$. Define $\overline{V} = \mathcal{L} \setminus V$. So $\overline{V}$ is the set of all indices $i \in \mathcal{L}$ such that $f_i = f_{\mathsf{forward}}$ and $\pi(i) = i$. The remaining indices are contained in V. Note that $X \subseteq V$. Define $n_{\mathsf{non\text{-}id}} = |V|$. Consider the following case analysis.

Step 6.a. If $n_{\mathsf{non\text{-}id}} \leqslant \beta$ then do the following. Sample τ consistent with $c^{(1)}_X$ and the tampered tag is an encoding of τ itself. Due to the non-malleable property of NMC_0, we have that this sampling is identical to sampling τ (unconditionally) at random. Sample $c^{(2)}$ which is an encoding of τ and is consistent with f, $c^{(1)}_X$ and the tampered right codeword is an encoding of τ. Let E be the set of erroneous indices according to τ.

Note that the erroneous indices are identical in the original and the tampered codeword. So, we have $c^{(1)}_{\overline{V}\setminus E} = \tilde{c}^{(1)}_{\overline{V}\setminus E} = a^{(1)}_{\overline{V}\setminus E} = \tilde{a}^{(1)}_{\overline{V}\setminus E}$. Since, $D > \beta + (2p_eN^{(1)}) \geqslant |V| + |E|$ the tampered codeword is valid and consistent with tag τ if and only if it is identical to the original codeword.

Since $T \geqslant \beta \geqslant |V|$, we can sample $c^{(1)}_{V\setminus E} \sim (C^{(1)}_{V\setminus E}|c^{(1)}_X, \tau) \equiv (C^{(1)}_{V\setminus E}|c^{(1)}_X, \tau, s)$. Next we check whether $\tilde{c}^{(1)}_{V\setminus E} = c^{(1)}_{V\setminus E}$ and it is consistent with the tag τ. If yes, we output $\mathsf{Event}_{\mathsf{same}^*}$; otherwise we output $\mathsf{Event}_\perp$. Stop the simulator.

Step 6.b. If $n_{\textsf{non-id}} > \beta$ then output $\mathsf{Event}_\perp$ and exit the simulator. We shall show that this step incurs negligible simulation error. Our proof shall show that: For random left codeword (consistent with prior sampled value) and random error indices, the probability that the tag is consistent with the left codeword is negligible.

Let $V' \subseteq V$ be a subset of size $\log^2 M/p_e$. Pick a random error index E. Over the choice of erroneous indices, we have $|E \cap V'| = \Omega(\log^2 M)$ with $1 - \mathsf{negl}(M)$ probability (by Lemma 1).

Consider any index $i \in V' \cap E$. Let $j = \pi^{-1}(i)$. If $j = i$ then $f_i = f_{\mathsf{toggle}}$. Now, with probability at least $1/2$ we have $\tilde{c}^{(1)}_i \neq c^{(1)}_i$ (because, $T \geqslant \Theta(\log^2 M/p_e)$). Update $V' \cap E$ to $(V' \cap E) \setminus \{i, j\}$ and repeat this until $V' \cap E$ is empty.

The probability that: there exist an index $i \in E$ such that $\tilde{c}^{(1)}_i$ is a mismatch with the tag value (i.e. $c^{(1)}_i$), is at least $1 - (1/2)^{|V' \cap E|/2} = 1 - \mathsf{negl}(M)$.

3.2 Instantiations

We shall use the following results:

1. Using Reed-Solomon codes and share-packing techniques, we can construct an $[M, L, T, D]$ ECSS such that: $M = n\varphi$, $L = \ell\varphi$ and $T = D = (n-\ell)/2 = M^{1/(1+c)} \log^{2/c} M$, where $\varphi = 2\log n$. This helps us get the following rate of NMC_1 in Theorem 3.

$$\frac{L}{N} \geqslant 1 - \frac{\log^{4/c} n}{n^{c/(1+c)}} \geqslant 1 - \frac{\text{polylog } N}{N^{c/(1+c)}}$$

   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. The quantitative improvement in rate is not significant to merit usage of such a complex code.

2. There exists $[t', t = t'\text{polylog } t', \nu_0]$ non-malleable code NMC_0 against $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_t$, where $\nu_0(\cdot)$ is a suitable negligible function. This follows from the work of [AGM+14b]. Using this construction as NMC_0 in Theorem 3, the rate of NMC_1 is rate L/N is:

$$\frac{L}{N} \geqslant 1 - N^{-1/2}\text{polylog } N$$

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. Under Submission, Preliminary version available on request, 2014. www.cs.ucla.edu/~hmaji/.

[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.

[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.