Explicit Non-malleable Codes Resistant to Permutations and Perturbations

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 message contained in a tampered codeword is either the original message, or a completely unrelated one. Although existence of such codes for various rich classes of tampering functions is known, *explicit* constructions exist only for "compartmentalized" tampering functions: i.e. the codeword is partitioned into *a priori fixed* blocks and each block can *only be tampered independently*. The prominent examples of this model are the family of bit-wise independent tampering functions and the split-state model.

In this paper, for the first time we construct explicit non-malleable codes against a natural class of non-compartmentalized tampering functions. We allow the tampering functions to *permute the bits* of the codeword and (optionally) perturb them by flipping or setting them to 0 or 1. We construct an explicit, efficient non-malleable code for arbitrarily long messages in this model (unconditionally).

We give an application of our construction to non-malleable commitments, as one of the first direct applications of non-malleable codes to computational cryptography. We show that non-malleable *string* commitments can be "entirely based on" non-malleable *bit* commitments. More precisely, we show that simply encoding a string using our code, and then committing to each bit of the encoding using a *CCA-secure bit commitment* scheme results in a non-malleable string commitment scheme. This reduction is unconditional, does not require any extra properties from the bit-commitment such as "tag-based" non-malleability, and works even with physical implementations (which may not imply standard one-way functions). Further, even given a partially malleable bit commitment scheme which allows toggling the committed bit (instantiated, for illustration, using a variant of the Naor commitment scheme under a non-standard assumption on the PRG involved), this transformation gives a fully non-malleable string commitment scheme. This application relies on the non-malleable code being explicit.

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

*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-malleability is a cryptographic notion [DDN91] which requires that an encoding (encryption, commitment etc.) of a message cannot be used to create a valid encoding of a "related" message by a (computationally) crippled adversary. Non-malleable codes [DPW10] is a special case of this idea: here, the encoding is in the form of a single string (rather than an interactive protocol), but the attacker is heavily crippled in that the tampering function it can apply on a codeword must belong to very simple classes (e.g., bit-wise functions). Since the class of tampering functions is simple, one can hope to prove the non-malleability of a code without relying on any computational assumptions.

Nevertheless, it has been a challenge to obtain *explicit constructions* of non-malleable codes for expressive families of attacks. Prior explicit constructions of non-malleable codes rely on the "compartmentalized" structure of the tampering function, i.e. the codeword is partitioned into *a priori fixed* blocks and each block can *only* be tampered independently. The prominent examples of this model are the family of bit-wise independent tampering functions and the split-state model.

In this work, we seek to build explicit non-malleable codes (with efficient encoding and decoding algorithms) for certain non-compartmentalized tampering functions. In particular, we consider bit-permutation attacks composed with arbitrary bit-wise functions.[1] The motivation for choosing this class of attacks comes from the following intriguing question:

Can non-malleable string-commitments be "entirely based" on non-malleable bit-commitments?

To formalize this problem, we may consider an idealized model of bit commitments using physical tokens: to commit a bit to Bob, Alice can create a small physical token which has the bit "locked" inside (and later, she can send him a "key" to open the token). This completely hides the bit from Bob until Alice reveals it to him; on the other hand, Alice cannot change the bit inside the token once she has sent it to Bob. Further, this is a non-malleable bit commitment scheme, in that if Bob plays a man-in-the-middle adversary, and wants to send a commitment to Carol, he can can only send the token from Alice as it is, or create a new token himself, independent of the bit committed to by Alice.

Now, we ask whether, in this model, one can make non-malleable string commitments (relying on no computational assumptions). *This is a question about non-malleable codes in disguise!* Indeed, if we required the commitment protocol to involve just a single round of sending a fixed number of tokens, then a commitment protocol is nothing but a non-malleable encoding of a string into bits, and the class of tampering functions we need to protect against is that of *bit-level permutations* and bit-wise set/reset. Though we presented this string commitment scheme in an idealized setting involving tokens, it can be translated to *a reduction of non-malleable string commitment to* CCA-secure *bit commitment (as defined in [CLP10]).*

[1] An earlier (unpublished) version of this work [AGM+14] considered a restricted class of bit-wise functions, along with bit-permutations. Specifically, it did not allow the bit-wise functions to be constants (i.e., to set a bit to a fixed value), but required that there is at least a constant probability that the output is 0 and that it is 1, if the input is, say, uniform. Such a code is not sufficient to achieve the following commitment result. The code we build in this paper does handle all bit-wise functions (we note that to handle all randomized bit-wise functions, it is enough to handle the four deterministic functions from $\{0,1\}$ to $\{0,1\}$).

As mentioned above, the non-malleable codes we build can withstand a slightly larger class of tampering attacks, which corresponds to the ability of the adversary to apply any set of functions from $\{0,1\}$ to $\{0,1\}$ to the bits stored in the tokens (i.e., set, reset, flip or keep), before applying the permutation attack. (As such, in the above application, we do not actually require the bit commitment scheme to be CCA secure.)

This application also brings out an important aspect of non-malleable codes: whether they are explicit or not. While there indeed is an efficient randomized construction of non-malleable codes that can resist permutations [FMVW14], it will not be suitable in this case, because neither the sender nor the receiver in a commitment scheme can be trusted to pick the code honestly (Bob could play either role), and non-malleable codes are not guaranteed to stay non-malleable if the description of the code itself can be tampered with.

Construction sketch. The focus of our construction is in being able to obtain a clean analysis rather than the best efficiency. Indeed, in on-going work we consider a more efficient construction that builds on the current construction (albeit with a more complex analysis). Our construction consists of four steps, that are sketched below. We present a more detailed overview and further motivation behind these steps in Section 1.3.

∘ We start with a large-alphabet randomized encoding which has a large enough distance and whose positions are t-wise independent for a large enough t (e.g., a "packed secret-sharing scheme" based on the Reed-Solomon code suffices), and make it resistant to permutations by incorporating into each character its position value; i.e., the character at the i^{th} position in a codeword x_i is re-encoded as $\langle i, x_i \rangle$, and allowed to occur at any position in the new codeword.

∘ The above code uses a large alphabet. It is concatenated with a binary inner code that is also resistant to permutations: each character in the outer code's alphabet is mapped to a positive integer (in a certain range) and is encoded by a block of bits whose weight (number of positions with a 1) equals this integer. Note that a permutation may move bits across the different blocks. To resist such attacks, we keep the bits within each block randomly permuted, and also, ensure that a good fraction of the weights do not correspond to a valid block (achieved, for instance, by requiring that the the weight of each block is a multiple of 3[2]), so that blindly mixing together bits from different blocks has *some* probability of creating an invalid block. A careful combinatorial argument can be used to show that, despite dependencies among the blocks caused by a permutation attack, the probability of having all attacked blocks remaining valid decreases multiplicatively with the number of blocks being attacked thus. This, combined with the fact that the outer code has a large distance, ensures that the probability of creating a different valid codeword by this attack is negligible. However, we need to ensure not only that the attack has negligible chance of modifying one codeword into a different valid codeword, but also that the probability of creating an invalid codeword is (almost) independent of the actual message. Roughly, this is based on the large independence of the outer code.

∘ The resulting code is not necessarily resistant to attacks which can set/reset several bits. Towards achieving resistance to such attacks as well, we consider an intermediate 2-phase attack family: here the adversary can set/reset bits at *random positions*, learn which positions were subjected to this

[2]In our actual analysis, we also allow the attacker to flip any subset of bits. This prevents us from having valid weights to be 0 modulo 2, as flipping an even number of positions preserves this parity.

attack, and then specify a permutation attack.[3] To resist such attacks, we encode each bit in the above codeword into a bundle, using an additive secret-sharing. Then, if one or more bits in a bundle are set/reset, all the other bits in the bundle turn uniformly random. Hence, unless the adversary chooses to set/reset a very large number of positions (in which case almost every bundle is touched, and all information about the original message is lost), for every bit which has been set/reset, there will be several that are uniformly random. Now, even though the adversary can apply a permutation to rearrange these random bits (into as few bundles as possible), to ensure that there are only a few bundles with a random bit, the adversary is forced to set/reset at most a few bundles' worth of bits. We note that our actual analysis follows a somewhat different argument, but fits the above intuition.

∘ Finally, the above code is modified as follows: a random permutation over the bits of the code is applied to a codeword; the permutation itself is encoded using a code of large distance, and appended to the above (permuted) codeword. Then it can be shown that a full-fledged attack (involving arbitrary set/reset and permutations) on such a codeword translates to a 2-phase attack of the above kind. Note that we do *not* rely on the permutation itself to be encoded in a non-malleable fashion. Indeed, the adversary can be allowed to learn and modify the encoded permutation *after* it has committed to the set/reset part of its attack on the rest of the codeword; in the 2-phase attack, this is modeled by the fact that the adversary can learn which positions in the codeword were set and reset, before deciding on the permutation attack.

1.1 Prior Work

Cramer et al. [CDF+08] introduced the notion of arithmetic manipulation detection codes, which is a special case of non-malleable codes; AMD codes with optimal 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. 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 $2^{\text{poly}(n)}$, rate 1 codes (with efficient encoding and decoding) exist with overwhelming probability [FMVW14].

On the other hand, explicit constructions of non-malleable codes have remained elusive, except for some well structured tampering function classes. Recently, 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. For a more general compartmentalized model of tampering, in which 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 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].

[3] In the actual analysis, we need to consider a slightly stronger 2-phase attack, in which the adversary can also learn the values of the bits in a small number of positions before specifying a permutation (and flipping a subset of bits).

Note that all known explicit construction of codes against particular tampering function classes heavily relies on the compartmentalized nature of the family of tampering functions, i.e. the codeword can be *a priori* partitioned into pieces such that the tampering function is applied independently to each partition. For example, bit-wise independent tampering functions act on each bit independently; and in the split-state model, the tampering on each state is independent. The class of functions being studied in this paper is one of the most natural classes of tampering functions without the aforementioned "compartmentalization" property.

Codes under computational assumptions. The idea of improving the rate of error-correcting codes by considering computationally limited channels stems from the work of Lipton [Lip94]. Restricting the channels to be computationally efficient allows one to use cryptographic assumptions, for example, Micali et. al. [MPSW05] show how to combine digital signatures with list-decodable codes to go beyond the classical error correction bound for unique decoding. Further constructions in various settings were provided in [OPS07, HO08, GS10, CKO14]. In the setting of non-malleable codes as well, constructions based on computational assumptions have been explored, e.g., in [LL12, FMNV14].

Non-malleable commitments. There is extensive literature on non-malleable commitments starting from the work of Dolev, Dwork and Naor [DDN91] leading to recent constant-round constructions based on one-way functions [Goy11, LP11, GLOV12]. Our application of nonmalleable codes to non-malleable commitments is similar in spirit to the work of Meyers and Shelat [MS09] on the completeness of bit encryption.

Concurrently, and independently of our work, Chandran et al. [CGM+14] relate non-malleable commitments to a new notion of non-malleable codes, called *blockwise non-malleable codes*. Blockwise NM-codes are a generalization of the split-state model where the adversary tampers with one state at a time. Chandran et al. show that block non-malleable codes with t blocks imply non-malleable commitments of $t-1$ rounds. In contrast our work is in the standard setting where there is a single state, and shows that non-malleable codes boost the security of non-malleable commitments (from bits to strings or partial to full).

Application of non-malleable codes to cryptographic constructions. AMD codes have found several applications in information-theoretic cryptography, for secret-sharing, randomness extraction and secure multi-party computation (e.g., [BT07, CDF+08, GIM+10, GIP+14]). However, the more general notion of non-malleable codes have had few other applications, outside of the direct application to protecting the contents of device memories against tampering attacks.

Our application to non-malleable commitment is one of the few instances where non-malleable codes have found an application in a natural cryptographic problem that is not information-theoretic in nature. A similar application appears in the recent independent work of Coretti et al. [CMTV14]. There, a variant of (replayable) CCA-secure string encryption is constructed from CCA-secure bit encryption. While similar in spirit, an important difference between our approach and theirs is that the non-malleable code used in [CMTV14] is still compartmentalized, and as a result, their construction requires the use of a separate cryptographic system (a separate public-key) for each compartment — i.e., each bit position — of the string. In contrast, our non-malleable code is not compartmentalized; this enables us to obtain non-malleable commitment (which does not have the

notion of a public-key) with no a priori bound on the length of the string.[4] Another difference is that since [CMTV14] aims to achieve CCA secure encryption (as opposed to non-malleable encryption), they consider a "continuous" version of non-malleability for codes, but settle for a variant of (replayable) CCA security in which the decryption key self-destructs the first time it is used on an invalid ciphertext. Since our focus is on non-malleability (in which the adversary can output only one set of commitments, or in the case of non-malleable encryption, make only one parallel query to the decryption oracle), we do not encounter these issues, and our solution does not involve any self-destruction.

1.2 Our Contribution

The class of tampering functions which permutes the bits of the codeword is represented by $\mathcal{S}_N$. The set of all tampering functions which allow the adversary to tamper a *bit* by passing it through a channel is denoted by $\mathcal{F}_{\{0,1\}}$; this includes forwarding a bit unchanged, toggling it, setting it to 1, or resetting it to 0. The class of tampering functions which allows the adversary to do apply both: i.e., tamper bits followed by permuting them is represented by: $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$. Our main result is a non-malleable code against this class of tampering functions.

Theorem 1 (Non-malleable Code)**.** *There exists an explicit and efficient non-malleable code for multi-bit messages where the class of tampering functions permits the adversary to forward, toggle, set to 0 or set to 1 each bit of the codeword, followed by permuting the (altered) codeword.*

Our main non-malleable encoding which is robust to $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ relies on a basic encoding scheme. The basic encoding scheme is robust to a weaker class of tampering functions, but it provides slightly stronger security guarantees. More specifically, the basic scheme protects only against $\widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ class, where $\widetilde{\mathcal{F}}_{\{0,1\}}$ is the class of functions which either forward a bit unchanged or toggle it but do not set or reset it. The stronger security guarantee given by basic scheme is that it allows the adversary to *adaptively* choose the tampering function $\widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$. The adversary first specifies n_0 and n_1, i.e. number of indices it wants to reset to 0 and number of indices it wants to set to 1. It is provided a random subset of indices of size n_0 which is all reset to 0; and a (disjoint) random subset of indices of size n_1 which is all set to 1. Given this information, the adversary can adaptively choose the tampering function in $\widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$. Even given this additional power, the adversary cannot tamper the codeword to produce related messages (except with negligible probability).

The security definition achieved by our basic construction is presented in Figure 7. The encoding scheme is provided in Figure 8 and our result can be distilled in the following theorem.

Theorem 2 (2-Phase Non-malleability)**.** *For all L, there exists encoding and decoding scheme* Enc *and* Dec*, respectively, such that, for all n_0, n_1, and for all $n_p \leqslant \log^9 \kappa$, there exists $\mathbb{D}$ such that, for all $f \in \widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ and mapping* map*, the expected simulation error in Figure 7 is* $\mathsf{negl}(\kappa)$.

Contribution to non-malleable commitments. As noted earlier, we consider the question of constructing simple string non-malleable commitments from bit non-malleable commitments. For

[4] Our non-malleable code can also be used to obtain a variant of non-malleable encryption [BS99, PSV06] — which refers to a relaxation of CCA2 secure encryption wherein the adversary can make only one parallel set of decryption queries — for strings of *a priori* unbounded length, using CCA2 bit encryptions. The variant we obtain is analogous to Replayable CCA (RCCA) security [CKN03].

example, if we simply encode the given string and commit to each of its bit using the given non-malleable bit-commitment, does it result in a secure non-malleable string commitment schemes? What are the conditions we need on the underlying bit commitment scheme?

For this question, we are interested in a really simple reduction, as opposed to, e.g. "merely" black-box reductions. Indeed, if we ask for a merely black-box construction we can invoke known (but complex) reductions: a bit commitment scheme (even malleable) implies a one-way function, which in turn imply string commitments in a black box way [GLOV12]. Such reductions are not, what we call *totally* black-box. For example, if we switch to a model where we are given the bit-commitment scheme as a *functionality* which can be executed only a bounded number of times, such as a one-time program [GKR08] or a hardware token [Kat07], then we do not necessarily have standard one-way functions. Therefore, the reduction should avoid assuming additional complexity assumptions such as OWFs or signatures. In fact, for this reason, the reduction should also not rely on using tags and "tag-based" non-malleability [PR05b]. It should work with standard *non-tag-based* non-malleable bit-commitments.

Our reduction actually satisfies these conditions provided that we start with a (non-tag-based) CCA-secure bit-commitment scheme [CLP10]. We show that (perhaps the simplest construction where) if we just commit to each bit of a random codeword of the given string works! This gives us the following theorem:

Theorem 3 (CCA Bit-commitment to Non-malleable String Commitment)**.** *There exists a simple and efficient black-box compiler which, when provided with:*

- *A non-malleable encoding robust to* $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$*, and*
- *A* r*-round (possibly* **non-tag-based***) CCA-secure bit-commitment scheme*

yields a r*-round non-malleable string-commitment scheme.*

We note that the theorem statement is **unconditional**: it does not assume any computational assumption beyond the given non-malleable bit-commitment. In particular, the theorem holds even if the bit-commitment is implemented in a model which does not necessarily imply OWFs. Furthermore, in Section 6, we prove that in fact, the theorem holds even if the bit-commitment is not CCA-secure but only satisfies a much weaker notion which we call *bounded-parallel* security.

Finally, we show the power of our non-malleable codes by demonstrating that even if we start with a seemingly much weaker scheme which allows partial malleability, e.g., it may allow the MIM to toggle the committed bit, our non-malleable codes can "boost" it to full-fledged malleability. See Section 6 for details.

1.3 Technical Overview

We provide a high level overview of our results and main technical highlights.

1.3.1 Basic Encoding Scheme

Our main non-malleable code construction relies crucially on a basic encoding scheme. The basic encoding scheme only guarantees non-malleability against a weaker class of functions than our target tampering function class $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$; but it provides stronger security guarantees which is reminiscent of adaptive choice of the tampering function based on partial leakage on the message encoding.

Our basic encoding scheme protects against permutation and toggling of bits, i.e. $\widetilde{\mathcal{F}}\{0,1\} \circ \mathcal{S}_N$; as well as set/reset of random positions in the codewords. That is, the adversary specifies a message s to be encoded and specifies how many positions it wants to reset to 0 (say, n_0) and how many positions it wants to set to 1 (say, n_1). Following which, these subsets are chosen uniformly at random and provided to the adversary. Suppose I_0 and I_1 corresponding to the indices which are reset and set, respectively. The adversary chooses the tampering function $f \in \widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ which permutes the bits followed by toggling some of them. The security requirement dictates that, over the random choice of I_0 and I_1, the adversary should not be able to produce messages related to the original message s, except with negligible probability.

We provide a short intuitive summary of the experiment in Figure 1. The experiment is formally provided in Figure 7. In the actual experiment, the adversary also gets to see a few bits of encoding itself. This turns out to be completely innocuous and, hence, is excluded in this overview.

Expt:

1. The adversary provides n_0 and n_1; and message s.
2. Let I be a random subset of $[N]$ of size $n_0 + n_1$. Randomly partition I into I_0 and I_1 with size n_0 and n_1, respectively. Provide I_0 and I_1 to the adversary.
3. The adversary provides tampering function f.
4. Encode s to obtain code c. In the code c, reset each index in I_0 and set all indices in I_1 to get c'. Apply tampering function f to c' and obtain tampered codeword $\tilde{c}$. Decode $\tilde{c}$ to obtain $\tilde{s}$.

Security requirement: Either $\tilde{s} = s$ or $\tilde{s}$ is unrelated to s.

Figure 1: Intuitive 2-Phase Non-Malleability Experiment. Formal definition in Figure 7.

Construction Intuition. We systematically develop our basic encoding scheme and provide a high level overview of its security proof for intuitive experiment in Figure 1. We begin with an elementary unary encoding scheme, i.e. where a number $n \in \mathbb{N}$ is encoded with 1^n (suitably padded with 0s and randomized). This scheme resists permutation attacks but this scheme is inefficient (because it has exponentially large output).

Next logical step is to explore concatenation codes based constructions, where an outer code (say Reed-Solomon encoding with share-packing techniques) is concatenated with a suitable non-malleable inner code. It is tempting to attempt to de-randomize the existential results of [FMVW14] to obtain these inner encoding schemes (because Reed-Solomon codes use fields of logarithmic al-

phabet size). This might have worked if the adversary was restricted to use permutations which preserve the block boundaries for Reed-Solomon codes. This is certainly not the case. If we consider the class of all permutations, the size of the tampering family is too large even for small (i.e., constant size) fields. Hence, such a de-randomization is not feasible for our family of tampering functions.

We amalgamate the two techniques mentioned above, i.e. concatenation of Reed-Solomon encodings with (a slight variation of) the unary encoding, to obtain our first candidate construction. Although, it does not fully resolve our problem; but we include this because some of its salient features are useful in our final basic encoding.

The first encoding scheme is the following. Encode the message s using Reed-Solomon encoding (with good distance and privacy) using share packing techniques. Suppose the outer codewords is $g_1, \ldots, g_n$. Now, encode each $x_i = \langle i, g_i \rangle$ with a suitable unary scheme. The encoding of x_i is the characteristic vector of a random subset of $[m]$ with weight $2c + 2x_i$, where c is a suitable constant and m is sufficiently large.

Note that this encoding scheme is safe against permutations which preserve the block boundaries (because x_i has the index i encoded within it).

Example 1. Now consider a tampering function f which swaps one bit from encoding of $\langle 1, g_1 \rangle$ and $\langle 2, g_2 \rangle$; and does not tamper any other positions. Due to the high distance of the outer encoding scheme, either the tampered code is invalid or it is identical to the original codeword. Since, the outer code also has very high privacy, the probability that the set of first two tampered outer codewords is identical to the set of first two initial outer codewords is independent of the message s.

Example 2. Now consider a tampering function f which creates each inner codeword by accumulating bits from every inner codeword of the input code. In this case, we would like to use the fact that each bit in the inner code is unpredictable, i.e. it has constant probability of being 0 or 1. Copying unpredictable bits across block boundaries should not yield valid inner codewords, because $\mod 2$ is highly sensitive. But this turns out to be subtle because these bits are not completely independent; but are correlated. We show that the intuition is partially correct (Lemma 8). By careful analysis we exhibit that the parities of at least half of the inner codewords of the tampered code are independent (see Lemma 4 and this bound is optimal). This suffices to show that all inner codewords in the tampered code are valid only with negligible probability.

Example 3. Consider any of the examples mentioned above and a tampering function which toggles an even number of bits in an inner codeword. Such a tampering function would maintain the parity $0 \mod 2$. It turns out that, instead of using an unary scheme which encodes with $0 \mod 2$ weight strings, if we use strings of weight $0 \mod 3$, our encoding scheme is robust to permutation and bit toggles as well. In this case, we can show that if a tampering function toggles non-zero number of bits in any inner codeword, then its parity $\mod 3$ is constant unpredictable.

Example 4. Next, we explore whether this encoding scheme is also non-malleable against an adversary who can set/reset random positions followed by permuting the remaining bits. We present the main bottleneck tampering function. Suppose we have a tampering function which uses $n_0 + n_1 = nm - m$. After a random set of indices is set/reset, we choose our permutation f as follows. It moves around the bits which were set/reset to explicitly write down the first $(n-1)$

inner codewords. Then, it copies remaining bits from the input code to the final inner codeword.

The problem with this example is that the tampered code can either be invalid or it is valid and is identical to what is defined by the explicitly written down inner codewords (because the tampering function set/reset nearly the whole codeword). But the probability of the final inner codeword of the tampered code being consistent is not independent of the message s (because it is obtained by copying bits from a large number of inner codewords of the input code).

So, we employ the following trick. We additively share each bit of the inner encoding scheme. Now the bits being copied are uniformly random unless f can choose all bits which specify the additive secret share of one bit. By suitably choosing parameters we can make the probability of this event negligibly small.

Our proof shows that this encoding scheme remains non-malleable against the attacks described in the first three examples. There is a variant of example 2 (say, Example 2′) which creates a technical bottleneck for this particular encoding scheme. The tampering function preserves block boundaries of the outer codeword; but permutes entries within each inner encoding. We want to claim that if such a permutation mixes the additive shares of different bits, then the inner codeword becomes invalid with constant probability. This is shown in Lemma 13.

Figure 8 formally explains our basic-encoding scheme and its security proof is included in Section 4 and Appendix C. The case analysis proceeds slightly differently from presentation above; but it is instructive to have these five working examples while understanding the proof.

1.3.2 Main Non-malleable Code Construction

Our main non-malleable coding scheme resistant against the class of attacks $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ is built on top of the basic coding scheme. In order to encode a message s, we choose a random permutation σ. The codeword consists of two parts: the first part is the basic encoding of s with σ applied on it, and the second part is a Reed-Solomon encoding of σ with high distance. Intuitively, applying a random permutation ensures that setting/resetting bits in the main codeword results in random positions begin modified in the basic codeword, exactly the kind of attack basic code can handle.

In order to show that the main coding scheme is resistant against a tampering function $f \in \mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$, we prove that we can construct an attack on the 2-phase non-malleability of basic coding scheme such that the distribution of modified message $\tilde{s}$ in the main coding scheme is statistically close to that in the basic coding scheme. This implies that if $\tilde{s}$ is independent of s in the latter case, the same holds in the former case as well.

At a high level the proof proceeds as follows. First of all, if a large number of bits in the left part of the main codeword is moved to the right, we claim that the resulting right codeword is invalid with high probability. This follows from the fact that the left codeword has a large number of bits that are distributed uniformly at random and the right codeword has high distance. Therefore, we assume that only a small number of bits move across the two parts of the codeword.

Given a tampering function f, we can find out how many bits are being set to 0 and 1 in the left part of the main codeword, and at what positions. Let these numbers be n_0 and n_1, and the positions be $\hat{I}_0$ and $\hat{I}_1$. We can imagine that a basic codeword is created and a random set of n_0 indices

are set to 0 and another random set of indices are set to 1; call these sets I_0 and I_1 respectively. Now, given I_0 and I_1, we can choose a permutation σ at random such that σ maps positions in I_b to $\hat{I}_b$, for $b \in \{0,1\}$. Therefore, effectively, both the left part of the main codeword and the basic codeword are being modified in the same way.

The main coding scheme's encoding and decoding procedures are more complex though. If tampering leads to an invalid right codeword, we would like to modify the basic codeword in such a way that decoding fails. Towards this, we must first see how f modifies the right codeword. Observe that some bits from the left codeword may be moved to the right, but their number is small. We would like to 'see' those bits in order to make sure the right codeword is modified in a consistent way. This requires the 2-phase non-malleability experiment to reveal a small part of the codeword, and allow the adversary to base its attack on that (which it does).

If we find that the tampered right codeword is valid, we could decode it to obtain a permutation $\tilde{\sigma}$, which may be different from σ. The decoding procedure of main coding scheme would apply the inverse of $\tilde{\sigma}$ to the left part of the codeword, and then the run the decoding algorithm of the basic coding scheme, to recover a message $\tilde{s}$. Our proof shows how one can construct a tampering function in the smaller class $\widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_m$ (where m denotes the size of the basic codeword) such that the decoding of the basic codeword also produces $\tilde{s}$.

1.3.3 Non-malleable String Commitments from CCA-secure Bit Commitments

We now present a brief overview of the main ideas used in proving Theorem 3. Recall that we need to show that given a CCA-secure bit-commitment that is not necessarily tag-based, committing to each bit of the codeword individually, results in a non-malleable string commitment scheme.

The proof that this scheme works is not straightforward because the adversary can simply permute the commitment protocols and succeed in committing a related codeword. Although the codewords are immune to permutations, and become invalid, this holds only when the permutations *do not depend* on the codeword. In this case, the adversary selects a permutations *after* seeing the codeword (although in committed form). This prevents us from directly applying the non-malleability of our codeword.

To resolve this issue, we make use of the fact that the scheme is CCA-secure. In CCA-secure bit commitments, roughly speaking, the hiding of the commitments holds even in the presence of a decommitment oracle $\mathcal{O}$ which reruns the values committed to by the adversary to $\mathcal{O}$ provided that they are not a copy of the challenge commitment.

At a high level, given such an $\mathcal{O}$, we use it to extract the permutation applied by the adversary on the input commitment. Due to the hiding of commitment, we can be sure that the distribution of such permutations is computationally independent of the committed bit. We then rely on the fact that a CCA-secure bit commitment is actually *concurrent* non-malleable [PR05a] and therefore the permutations are computationally independent of the entire string committed to on left. This defines a distribution over permutations, from which we can sample given $\mathcal{O}$, even before the challenge commitment is given. Thus we the adversary succeeds in committing a related string, it means that a permutations from the distribution so defined succeeds in mauling our codewords, which is not possible.

The actual argument works with a general adversary (instead of permutations), and extracts a tampering function $f \in \mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$. Also, we do not appeal to the concurrent non-malleability of CCA-secure scheme in our proof, and give a direct self-contained proof.

We remark that the reduction is non-trivial only because we are working in the standard **non-tag-based** setting. Otherwise, in case of tags, one can simply sign the entire transcript using the tags and obtain a NM string commitment. In case of *bit* commitments, tag-based non-malleability is a stronger requirement than the standard (non-tag-based) non-malleability.[5]

As noted earlier, observe that to extract the tampering function f, our reduction does not require "full fledged" CCA security. The reduction can make all of its commitments to the oracle in parallel and there is an a-priori known bound on how many such bit-commitments are sent to $\mathcal{O}$ (all in parallel). Constructing Bit-commitment schemes in this model is significantly easier than the full-fledged concurrency: e.g., constant round constructions under standard assumptions are possible in this model [Pas04, GLP+12] whereas full-fledged CCA requires at least $\widetilde{O}(\log n)$ rounds [CLP10, GLP+12] or non-standard assumptions [PPV08]. We call this the *bounded-parallel* security and our reduction works for this weaker notion as well.

2 Preliminaries

We denote the set $\{1, \ldots, n\}$ by $[n]$. If $a \in [b-\varepsilon, b+\varepsilon]$, then we represent it as: $a = b \pm \varepsilon$. For a set S, the set of all k-subsets of S is represented by $\binom{S}{k}$; and the set of all subsets of S is represented by 2^S. For a function f, if $f(i) = j$, then we represent it by $i \mapsto_f j$.

Probability distributions are represented by bold capital alphabets, for example $\mathbf{X}$. The distribution $\mathbf{U}_S$ represents a uniform distribution over the set S. Given a distribution $\mathbf{X}$, $x \sim \mathbf{X}$ represents that x is sampled according to the distribution $\mathbf{X}$. And, for a set S, $x \xleftarrow{\$} S$ is equivalent to $x \sim \mathbf{U}_S$.

For a joint variable $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$ and $S = \{i_1, \ldots, i_{|S|}\} \subseteq [n]$, we define the random variable $\mathbf{X}_S = (\mathbf{X}_{i_1}, \ldots, \mathbf{X}_{i_{|S|}})$. 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 $\mathbf{Y} = f(\mathbf{X})$ represents the following distribution: Sample $x \sim \mathbf{X}$; and output $f(x)$. Further, $f(x_{[n]})$ represents the vector $f(x_1) \ldots f(x_n)$. For example, $i + [n] = \{i+1, \ldots, i+n\}$.

The statistical distance between two distributions $\mathbf{S}$ and $\mathbf{T}$ over a finite sample space I is defined as:

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

For a pair $z = \langle x, y \rangle$, define $\mathsf{first}(z) := x$ and $\mathsf{second}(z) := y$.

[5] Pass and Rosen [PR05b] argue that for *string* commitments, the two notions are equivalent since one can simply commit the tag as part of the string, if there are no tags. Since we only have *bit* commitments, this does not work. The tag-based approach also requires the additional assumption that it is possible sign the transcript. This may be undesirable in models such as hardware tokens.

2.1 Classes of Tampering Functions

We shall consider the following set of tampering functions.

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 Fundamental Channels. The set of fundamental channels over $\{0,1\}$, represented as $\mathcal{F}_{\{0,1\}}$, contains the following binary channels f: a) $f(x) = x$, b) $f(x) = 1 \oplus x$, c) $f(x) = 0$, or d) $f(x) = 1$. These channels are, respectively, called *forward*, *toggle*, *reset* and *set* functions.
3. Family of Sensitive Channels. The set of *sensitive* functions $\widetilde{\mathcal{F}}_{\{0,1\}}$ contains only forward and toggle channels. In other words, tampering involves XOR-ing an N-bit input string with a fixed N-bit string.

We can define more complex tampering function classes by composition of these function classes. For example, composition of $\mathcal{S}_N$ with $\mathcal{F}_{\{0,1\}}$ yields the following class of tampering functions. For any $\pi \in \mathcal{S}_N$ and $f_1, \dots, f_N \in \mathcal{F}_{\{0,1\}}$, it transforms a codeword $x_{[N]}$ into:

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

This class is represented by: $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$. Our main result provides an efficient non-malleable code against the tampering class $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$.

$\mathsf{Expt}_{\mathsf{Enc},\mathsf{Dec},\mathcal{F},\{\mathcal{D}_f\}_{f\in\mathcal{F}}}(1^\kappa)$:
Let $\mathsf{Enc} : \{0,1\}^L \to \mathcal{C} \subseteq \{0,1\}^N$ be an encoding scheme (possibly randomized); and $\mathsf{Dec} : \{0,1\}^N \to \{0,1\}^L \cup \{\perp\}$ be its corresponding decoding scheme. Let $\mathcal{F}$ be the class of permissible tampering functions. Any $f \in \mathcal{F}$ maps elements in $\mathcal{C}$ to elements in $\{0,1\}^N$. The distribution $\mathcal{D}_f$ is over the sample space $\{0,1\}^N \cup \{\mathsf{same}^*, \perp\}$, for every $f \in \mathcal{F}$.

For $f \in \mathcal{F}$ and $s \in \{0,1\}^L$, define $\mathsf{Tamper}_f^{(s)}$ as the following random variable over the sample space $\{0,1\}^L \cup \{\perp\}$: Let $c \leftarrow \mathsf{Enc}(s; r)$, for random r. Let $\tilde{c} := f(c)$. Output $\mathsf{Dec}(\tilde{c})$.

For $f \in \mathcal{F}$ and $s \in \{0,1\}^L$, define the random variable $\mathsf{Sim}_{\mathcal{D}_f}^{(s)}$ as follows: Sample $a \sim \mathcal{D}_f$. If $a = \mathsf{same}^*$, then output s; otherwise output a.

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

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

Figure 2: Non-Malleability Experiment

3 Building Blocks

In this section, we define encoding schemes (equivalently, secret sharing schemes) relevant to our construction.

Definition 1 (Secret Sharing Scheme)**.** *Consider alphabet sets $\Lambda_0, \Lambda_1, \ldots, \Lambda_m$ and a joint distribution $\mathbf{S} = (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_m)$ over the space $\Lambda_0 \times \Lambda_1 \times \cdots \times \Lambda_m$. The random variable $\mathbf{X}_0$ represents the secret being shared and $\mathbf{X}_i$ for $i \in [m]$ represents the i-th share. For $s \in \Lambda_0$ and set $T = \{i_1, \ldots, i_\ell\}$, the conditional distribution $(\mathbf{X}_T | \mathbf{X}_0 = s)$ is defined as the conditional distribution of $(\mathbf{X}_{i_1}, \ldots, \mathbf{X}_{i_\ell})$ under $\mathbf{S}$ when $\mathbf{X}_0 = s$. We define the following properties of the secret sharing schemes.*

1. *t-independence: For any $s \in \Lambda_0$, $T \subseteq [n]$ such that $|T| \leqslant t$, we have*

$$\mathrm{SD}\left((\mathbf{X}_T | \mathbf{X}_0 = s), \mathbf{U}_{\Lambda_T}\right) = 0$$

2. *t-privacy: For any $s_1, s_2 \in \Lambda_0$, $T \subseteq [n]$ such that $|T| \leqslant t$, we have*

$$\mathrm{SD}\left((\mathbf{X}_T | \mathbf{X}_0 = s_1), (\mathbf{X}_T | \mathbf{X}_0 = s_2)\right) = 0$$

3. *r-reconstruction: For any $s_1, s_2 \in \Lambda_0$, $T \subseteq [n]$ such that $|T| \geqslant r$, we have*

$$\mathrm{SD}\left((\mathbf{X}_T | \mathbf{X}_0 = s_1), (\mathbf{X}_T | \mathbf{X}_0 = s_2)\right) = 1$$

Consider a secret sharing scheme with r-reconstruction. Then any two different secrets s, s' have at least $m - r + 1$ different shares. Hence, we define the *distance* for this secret sharing scheme to be $m - r + 1$.

Secret Sharing Schemes. Below, we describe some secret sharing schemes which are relevant to our construction.

1. Basic Secret Sharing scheme using Reed-Solomon codes $\mathbf{X}^{(\mathsf{RS},n,k,\ell,\mathbb{F})}$. This is a generalization of Massey secret sharing scheme [Mas95] and is commonly referred to as the "share-packing technique" for Reed-Solomon codes. This is an $[n, k]$ code over a field $\mathbb{F}$, such that $|\mathbb{F}| \geqslant n + \ell$. Let $\{f_{-\ell}, \ldots, f_{-1}, f_1, \ldots, f_n\} \subseteq \mathbb{F}$. The secret sharing of message $(s_1, \ldots, s_\ell) \in \mathbb{F}^\ell$ is done by choosing a random polynomial $p(\cdot)$ of degree $< k$ conditioned on $(p(f_{-1}), \ldots, p(f_{-\ell})) = (s_1, \ldots, s_\ell)$. The shares $\{y_1, \ldots, y_n\}$ are evaluations of $p(\cdot)$ at $\{f_1, \ldots, f_n\}$ respectively. The formal description of the secret sharing scheme is provided in Figure 3. The field $\mathbb{F}$ will generally have characteristic 2 and this scheme will be used in our main construction presented in Section 5.

   The encoding has $(k - \ell)$-privacy (in fact, $(k - \ell)$ independence) and distance $d = n - k + 1$.

2. Secret Sharing scheme using Reed-Solomon codes $\mathbf{X}^{(\mathsf{aRS},n,k,\ell,\mathbb{F})}$. Consider any $[n + \ell, k]$ code over finite field $\mathbb{F}$ such that $|\mathbb{F}| \geqslant n + \ell$. Given a message $s \in \mathbb{F}^\ell$, the secret sharing is performed as follows: Sample a random Reed-Solomon code conditioned on the fact that its first ℓ elements are identical to the message s. Let $y_1, \ldots, y_n$ be the remaining elements in the codeword. The shares are defined to be $\langle 1, y_1 \rangle, \ldots, \langle n, y_n \rangle$. It is known that efficient encoding

Secret Sharing Scheme $\mathbf{X}^{(\mathsf{RS},n,k,\ell,\mathbb{F})}$:

(a) Sample space: $\Lambda_0 = \mathbb{F}^\ell$, $\Lambda_1 = \cdots = \Lambda_n = \mathbb{F}$.

(b) Conditions: $|\mathbb{F}| \geqslant n + \ell$ and $n \geqslant k \geqslant \ell$.

(c) Joint Distribution $(\mathbf{X}_0, \ldots, \mathbf{X}_n)$ is defined via the following sampling procedure: We assume that $\{f_{-\ell}, \ldots, f_{-1}, f_1, \ldots, f_n\} \subseteq \mathbb{F}$.

i. Pick a random polynomial: $p(x) = \sum_{i=0}^{k-1} a_i x^i$, where $a_i \xleftarrow{\$} \mathbb{F}$ and $i \in \{0\} \cup [k-1]$.

ii. Define $x_0 = (p(f_{-1}), \ldots, p(f_{-\ell})) \in \mathbb{F}^\ell$.

iii. Define $x_i = p(f_i)$, for $i \in [n]$.

iv. Output $(x_0, x_{[n]})$.

Efficient Encoding and Decoding. Efficient sampling property for $\mathbf{X}^{(\mathsf{RS},n,k,\ell,\mathbb{F})}$ follows from the efficiency of Lagrange interpolation.

Figure 3: Basic Reed-Solomon based Secret Sharing.

Secret Sharing Scheme $\mathbf{X}^{(\mathsf{aRS},n,k,\ell,\mathbb{F})}$:

(a) Sample space: $\Lambda_0 = \mathbb{F}^\ell$, $\Lambda_1 = \cdots = \Lambda_n = [n] \times \mathbb{F}$.

(b) Conditions: $|\mathbb{F}| \geqslant n + \ell$ and $n \geqslant k \geqslant \ell$.

(c) Joint Distribution $(\mathbf{X}_0, \ldots, \mathbf{X}_n)$ is defined via the following sampling procedure: We assume that $\{f_{-\ell}, \ldots, f_{-1}, f_1, \ldots, f_n\} \subseteq \mathbb{F}$.

i. Pick a random polynomial: $p(x) = \sum_{i=0}^{k-1} a_i x^i$, where $a_i \xleftarrow{\$} \mathbb{F}$ and $i \in \{0\} \cup [k-1]$.

ii. Define $x_0 = (p(f_{-1}), \ldots, p(f_{-\ell})) \in \mathbb{F}^\ell$.

iii. Define $x_i = \langle i, p(f_i) \rangle$, for $i \in [n]$.

iv. Output $(x_0, x_{[n]})$.

Efficient Encoding and Decoding. Efficient sampling property for $\mathbf{X}^{(\mathsf{aRS},n,k,\ell,\mathbb{F})}$ follows from the efficiency of Lagrange interpolation. In fact, it is also efficient to sample $x_S \sim (\mathbf{X}_S | \mathbf{X}_T = x_T)$, for any $S, T \subseteq \{0\} \cup [n]$.

Figure 4: Augmented Reed-Solomon based Secret-Sharing Scheme.

and decoding exist using Lagrange interpolation. For formal description of this scheme refer to Figure 4.

The encoding scheme has $(k-\ell)$-privacy and distance $d = n - k + 1$.

3. Balanced unary secret sharing scheme $\mathbf{X}^{(\mathsf{unary},m,F,p)}$. Set $m := 3p\,|F|$, where F is the message space. Assume that there exists a bijection $\mathsf{map}\colon F \to \mathbb{Z}_{|F|}$. Given a message $s \in \mathbb{Z}_{|F|}$, the secret sharing is performed as follows. Sample a random set S of $[m]$ of weight $\lceil m/3 \rceil + ps$. The shares are defined to be the characteristic vector of set S. Note that this scheme has efficient encoding and decoding. For a formal description refer to Figure 5.

   For any $s \in F$ and any set S used for encoding s, the total weight of the final shares lie in $[m/3, 2m/3]$. Hence, the name *balanced unary* secret sharing scheme.

Secret Sharing Scheme $\mathbf{X}^{(\mathsf{unary},m,F,p)}$:

(a) Sample space: $\Lambda_0 = \mathbb{Z}_{|F|}$, $\Lambda_1, \ldots, \Lambda_m = \{0,1\}$.

(b) Condition: $m = 3p\,|F| + 1$.

(c) Joint Distribution $(\mathbf{X}_0, \ldots, \mathbf{X}_m)$ is defined via the following sampling procedure: We assume that there exists a bijection from the set F to $\mathbb{Z}_{|F|}$, say $f \mapsto \mathsf{map}(f)$.

   i. Pick $f \xleftarrow{\$} F$.

   ii. Define $x_0 = \mathsf{map}(f)$.

   iii. Sample $x_{[m]} \xleftarrow{\$} \binom{[m]}{\lceil m/3 \rceil + px_0}$. Here $x_{[m]}$ is the characteristic vector of the sampled subset.

   iv. Output $(x_0, x_{[m]})$.

Efficient Encoding and Decoding. It is easy to see that there exists efficient encoding and decoding schemes. In fact, it is also efficient to sample $x_S \sim (\mathbf{X}_S | \mathbf{X}_T = x_T)$, for any $S, T \subseteq \{0\} \cup [m]$.

Figure 5: Balanced Unary Secret-Sharing Scheme.

4. Additive secret sharing scheme $\mathbf{X}^{(\mathsf{add},u,(G,+))}$. Let $(G,+)$ be a Abelian group. Let $y_1, \ldots, y_u$ be random elements in G. Define $y_0 = y_1 + \cdots + y_u$. We define the secret as y_0 and the shares as $y_1, \ldots, y_u$. The formal description is provided in Figure 6.

   The encoding scheme is $(u-1)$-independent and has distance $d = 1$.

Definition 2 (Concatenation Codes.)**.** *Consider two encoding schemes, the outer encoding scheme* $\mathbf{X}^{(\mathsf{out})} = (\mathbf{X}_0^{(\mathsf{out})}, \mathbf{X}_1^{(\mathsf{out})}, \ldots, \mathbf{X}_n^{(\mathsf{out})})$ *over* $\Lambda_0 \times \Lambda \times \cdots \times \Lambda$ *and the inner encoding scheme* $\mathbf{X}^{(\mathsf{in})} = (\mathbf{X}_0^{(\mathsf{in})}, \mathbf{X}_1^{(\mathsf{in})}, \ldots, \mathbf{X}_m^{(\mathsf{in})})$ *over* $\Lambda \times \Lambda' \times \cdots \times \Lambda'$*. The concatenation of the outer code with the inner code is defined as the joint distribution* $\mathbf{X}^{(\mathsf{concat})} = (\mathbf{X}_0^{(\mathsf{concat})}, \mathbf{X}_1^{(\mathsf{concat})}, \ldots, \mathbf{X}_{nm}^{(\mathsf{concat})})$ *over* $\Lambda_0 \times \Lambda' \times \cdots \times \Lambda'$*. Given a secret* $s \in \Lambda_0$*, sample* $\mathbf{x}_{[nm]} \sim (\mathbf{X}_{[nm]}^{(\mathsf{concat})} | \mathbf{X}_0^{(\mathsf{concat})} = s)$ *as follows: Sample* $\mathbf{x}_{[n]}^{(\mathsf{out})} \sim \left(\mathbf{X}_{[n]}^{(\mathsf{out})} \;\middle|\; \mathbf{X}_0^{(\mathsf{out})} = s\right)$*. Next, for each* $i \in [n]$*, sample* $\mathbf{x}_{(i-1)m+[m]} \sim \left(\mathbf{X}_{[m]}^{(\mathsf{in})} \;\middle|\; \mathbf{X}_0^{(\mathsf{in})} = \mathbf{x}_i^{(\mathsf{out})}\right)$*. Output* $\mathbf{x}_{[nm]}$*.*

Secret Sharing Scheme $\mathbf{X}^{(\mathsf{add},u,(G,+))}$:

(a) Sample space: $\Lambda_0 = G$, $\Lambda_1, \ldots, \Lambda_u = G$.

(b) Condition: G is Abelian.

(c) Joint Distribution $(\mathbf{X}_0, \ldots, \mathbf{X}_u)$ is defined via the following sampling procedure:

i. Pick $x_1, \ldots, x_u \xleftarrow{\$} G$.

ii. Define $x_0 = \sum_{i \in [u]} x_i$.

Efficient Encoding and Decoding. It is easy to see that there exists efficient encoding and decoding schemes. In fact, it is also efficient to sample $x_S \sim (\mathbf{X}_S | \mathbf{X}_T = x_T)$, for any $S, T \subseteq \{0\} \cup [u]$.

Figure 6: Additive Secret-Sharing Scheme.

We represent $\mathbf{X}^{(\mathsf{concat})} = \mathbf{X}^{(\mathsf{out})} \circ \mathbf{X}^{(\mathsf{in})}$ *as the concatenation of* $\mathbf{X}^{(\mathsf{out})}$ *with* $\mathbf{X}^{(\mathsf{in})}$.

Encoding and decoding procedures for concatenation codes are defined naturally using corresponding procedures for inner and outer encoding schemes. Note that the final encoding and decoding procedures are efficient if the corresponding procedures are efficient for inner and outer schemes.

Moreover, we emphasize that we do not focus on error correcting codes. In particular, if any of inner or outer decoding procedures fails, we output $\perp$ as the decoding of the overall code.

Suppose we have a codeword $c_{[n]}$ over $\mathbb{F}^n$ then c_i is referred to as the i-th *element* of the codeword. Now, consider a concatenation code where each element c_i is further encoded using an inner code over some field $(\mathbb{F}')^m$. The resultant codeword is $d_{[mn]} \in (\mathbb{F}')^{mn}$. The i-th *block* in $d_{[mn]}$ corresponds to the encoding of the i-th element of $c_{[n]}$.

For example, we shall consider the following concatenated secret sharing scheme. Let $\mathbf{X}^{(\mathsf{in})}$ be concatenation of outer code $\mathbf{X}^{(\mathsf{unary},m,\mathbb{F},3)}$ and inner code $\mathbf{X}^{(\mathsf{add},u,(\mathbb{GF}(2),\oplus))}$. Let $\mathbf{X}^{(\mathsf{out})}$ be $\mathbf{X}^{(\mathsf{aRS},n,k,\ell,\mathbb{F})}$. We shall consider the concatenation of $\mathbf{X}^{(\mathsf{out})}$ with $\mathbf{X}^{(\mathsf{in})}$.

4 Basic Encoding Scheme

Before we present our non-malleable encoding scheme against the class of tampering functions $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$, we present our basic encoding scheme. This encoding scheme can be interpreted as a non-malleable code against a weaker class of tampering functions $\widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ but with additional security properties beyond what non-malleability prescribes. These additional security properties are summarized via the experiment presented in Figure 7; we refer to this property as "2-Phase Non-malleability" property. Intuitively, the adversary gets additional information about the codeword in the first phase before it gets to choose the tampering function in second phase. More precisely, the 2-Phase experiment is as follows:

1. Adversary sends message s and $n_0, n_1, n_p \in [N]$.

2. The challenger picks an index set $I \xleftarrow{\$} \binom{[N]}{n_0+n_1+n_p}$ and randomly partitions I into I_0, I_1 and I_p of size n_0, n_1 and n_p, respectively. It picks $c = \mathsf{Enc}(s; r)$, where r is uniformly randomly chosen. Then it sends (I_0, I_1, I_p, c_{I_p}) to the adversary.

3. Adversary sends a tampering function $f \in \widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$.

The security proof of our main construction presented in Section 5 reduces to the 2-phase non-malleability proof of our basic scheme.

$\mathsf{Expt}_{\mathsf{Enc},\mathsf{Dec},n_0,n_1,n_p,\mathbb{D}}(1^\kappa)$:
Let $\mathsf{Enc}\colon \{0,1\}^L \to \mathcal{C} \subseteq \{0,1\}^N$ be an encoding scheme (possibly randomized); and $\mathsf{Dec}\colon \{0,1\}^N \to \{0,1\}^L \cup \{\perp\}$ be its corresponding decoding scheme. Let $n_0, n_1, n_p \in [N]$. Let $\mathcal{F}$ be the class of tampering functions. $\mathbb{D}$ is a collection of distributions explained below; and map is a mapping explained below.

For $s \in \{0,1\}^L$, define $\mathsf{Tamper}^{(s)}_{\mathsf{map}}$ as the following random variable: Pick $I \xleftarrow{\$} \binom{[N]}{n_0+n_1+n_p}$. Randomly partition I into I_0, I_1 and I_p of size n_0, n_1 and n_p, respectively. Let $c = \mathsf{Enc}(s; r)$, where r is uniformly randomly chosen. Let c' be defined by the following string: For $i \in [N]$,

$$c'_i = \begin{cases} 0, & i \in I_0 \\ 1, & i \in I_1 \\ c_i, & \text{otherwise} \end{cases}.$$

Let $(I_0, I_1, I_p, c_{I_p}) \mapsto_{\mathsf{map}} f$ be the mapping defined by map, where $f \in \mathcal{F}$. Let $\widetilde{c} = f(c')$. Output $\mathsf{Dec}(\widetilde{c})$.
Now, define $\mathsf{Tamper}^{(s)}_{I_0,I_1,I_p,c_{I_p},f}$ as the random variable $\mathsf{Tamper}^{(s)}_{\mathsf{map}}$ conditioned on I_0, I_1, I_p, c_{I_p} and $(I_0, I_1, I_p, c_{I_p}) \mapsto_{\mathsf{map}} f$.

$\mathbb{D}$ is a collection of distributions indexed by $(I_0, I_1, I_p, c_{I_p}, f)$ and each distribution $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ is over the sample space $\{0,1\}^L \cup \{\mathsf{same}^*, \perp\}$. For any $s \in \{0,1\}^L$, define the random variable $\mathsf{Sim}^{(s)}_{\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}}$, for $(I_0, I_1, I_p, c_{I_p}) \mapsto_{\mathsf{map}} f$, as follows: Sample $a \sim \mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$. If $a = \mathsf{same}^*$, then output s; otherwise output a.

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

$$\mathsf{adv}_{\mathsf{Enc},\mathsf{Dec},n_0,n_1,n_p,\mathcal{F},\mathbb{D}} := \max_{\substack{s\in\{0,1\}^L \\ \text{Mapping: } \mathsf{map}}} \mathop{\mathbb{E}}_{\substack{I_0,I_1 \\ I_p,c_{I_p}}} \left[\mathrm{SD}\left(\mathsf{Tamper}^{(s)}_{I_0,I_1,I_p,c_{I_p},f}, \mathsf{Sim}^{(s)}_{\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}} \right) \right]$$

Figure 7: 2-Phase Non-Malleability Experiment

Construction. As a high level, our encoding scheme is a concatenation code (see Definition 2) which does the following: Given a message s, it samples an outer code according to augmented Reed-Solomon code based secret sharing (see Figure 4). Then for each outer code element, it samples

an inner codeword which itself is a concatenation code using balanced unary secret sharing scheme (see Figure 5) and additive sharing scheme (see Figure 6).

The choice of parameters of our scheme are as follows: Let κ be the statistical security parameter. Given a message $s \in \{0,1\}^L$, let $\ell = \frac{L}{2 \lg L}$. Define $n := 3\ell$ and $k = 2\ell$. Let $\mathbb{F}$ be a finite field of characteristic 2 such that $|\mathbb{F}| \geqslant 4\ell$. Define $F := [n] \times \mathbb{F}$ and $u := \omega(\log \kappa)$. Then we define the following two secret sharing schemes.

1. $\mathbf{X}^{(\mathsf{out})} := \mathbf{X}^{(\mathsf{aRS}, n, k, \ell, \mathbb{F})}$.
2. $\mathbf{X}^{(\mathsf{in})} := \mathbf{X}^{(\mathsf{unary}, m, \mathbb{F}, 3)} \circ \mathbf{X}^{(\mathsf{add}, u, (\mathbb{GF}(2), \oplus))}$.

Our basic encoding scheme is defined by $\mathbf{X}^{(\mathsf{basic})} = \mathbf{X}^{(\mathsf{out})} \circ \mathbf{X}^{(\mathsf{in})}$. Formally, our scheme is described in Figure 8.

Let κ be the statistical security parameter. Let $\{0,1\}^L$ be the message space.
Let $\mathbb{F}$ be a characteristic 2 field such that: Let $\ell = \frac{L}{2 \lg L}$. Let $|\mathbb{F}| \geqslant 4\ell$. Let $n = 3\ell$ and $k = 2n/3 = 2\ell$. Let $\mathbf{X}^{(\mathsf{out})}$ be the secret sharing scheme $\mathbf{X}^{(\mathsf{aRS}, n, k, \ell, \mathbb{F})}$ (see Figure 4).

Let $F = [n] \times \mathbb{F}$ and $u = \omega(\log \kappa)$. Consider the secret sharing schemes $\mathbf{X}^{(\mathsf{unary}, m, F, 3)}$ (see Figure 5) and $\mathbf{X}^{(\mathsf{add}, u, (\mathbb{GF}(2), +))}$ (see Figure 6). Define the $\mathbf{X}^{(\mathsf{in})}$ as the secret sharing scheme which is the following concatenation (see Definition 2) code: $\mathbf{X}^{(\mathsf{unary}, m, F, 3)} \circ \mathbf{X}^{(\mathsf{add}, u, (\mathbb{GF}(2), +))}$.

Define $\mathbf{X}^{(\mathsf{basic})}$ as: $\mathbf{X}^{(\mathsf{out})} \circ \mathbf{X}^{(\mathsf{in})}$.

$\mathsf{Enc}_{\mathsf{basic}}(s \in \mathbb{F}^\ell)$:

1. Output $c_{[umn]} \sim (\mathbf{X}^{(\mathsf{basic})}_{[umn]} | \mathbf{X}^{(\mathsf{basic})}_0 = s)$.

$\mathsf{Dec}_{\mathsf{basic}}(c_{[umn]} \in \{0,1\}^{umn})$:

1. Decode $c_{[umn]}$ by decoding algorithm corresponding to $\mathbf{X}^{(\mathsf{basic})}$ code.

Figure 8: Basic Non-malleable Code achieving 2-Phase Non-malleability.

We emphasize that we can use Algebraic-Geometric codes instead of Reed-Solomon codes in our outer code to improve the rate of our code by a logarithmic factor, which is not the emphasis of this paper. We forgo this optimization for ease of presentation of the main ideas of our construction.

Useful Terminology. We shall visualize our inner code $\mathbf{X}^{(\mathsf{in})}$ as a two-dimensional object, where each *column* represents the additive secret shares of a bit in the unary encoding scheme.

4.1 Proof of Theorem 2

In this section we prove the result for $n_p = 0$ and $\mathcal{F} = \mathcal{S}_N$. The main result is obtained by slight modification of the arguments provided below; and are presented in Appendix C.

The proof of Theorem 2 will crucially rely on the notion of "equivalence of codes" and "dirty (inner) codewords," as defined below.

Equivalence of Codewords for our Scheme. We need the concept of equivalence of two codewords $g^{(\mathsf{basic})}_{[umn]}$ and $h^{(\mathsf{basic})}_{[umn]}$ which are equivalent if each block encodes identical outer codeword element.[6] Formally defined as follows:

Inner codes. Consider two inner codewords, $g^{(\mathsf{in})}_{[um]}$ and $h^{(\mathsf{in})}_{[um]}$. We say that $g^{(\mathsf{in})}_{[um]}$ and $h^{(\mathsf{in})}_{[um]}$ are equivalent codes if they encode the same message according to the inner code $\mathbf{X}^{(\mathsf{in})}$.

Non-Malleable codes. Consider two codewords $g^{(\mathsf{basic})}_{[umn]}$ and $h^{(\mathsf{basic})}_{[umn]}$. We say that $g^{(\mathsf{basic})}_{[umn]} \cong h^{(\mathsf{basic})}_{[umn]}$ if the following holds. Define $g^{(\mathsf{in})}_i = g^{(\mathsf{basic})}_{(i-1)um+[um]}$ for all $i \in [n]$. Similarly, define $h^{(\mathsf{in})}_i = h^{(\mathsf{basic})}_{(i-1)um+[um]}$ for all $i \in [n]$. Then, there exists a $\pi : [n] \to [n]$ such that for all $i \in [n]$, $g^{(\mathsf{in})}_i \cong h^{(\mathsf{in})}_{\pi(i)}$.

Dirty Inner Codewords. We say that an inner codeword in $\tilde{c}$ is dirty if:

1. The codeword partially receives its bits from one inner codeword. To clarify, it can be the case that it receives bits from more than one inner codewords, or it receives bits bits from some inner codeword and some of its bits are obtained from $I_0 \cup I_1$.
2. (The codeword receives all its bits from one inner codeword but) The permutation within the inner codeword is not column preserving. That is, there exists a column which receives bits from more than one column of the same inner codeword.

We shall show that if an inner codeword partially receives bits from an inner codeword, then decoding of that codeword fails with constant probability (see Lemma 9 Case 2.). On the other hand, if the codeword receives all its bits from one single inner codeword and the permutation is not column preserving, then decoding of that codeword also fails with constant probability (see Lemma 9 Case 1.). Both these results rely on the high independence of $\mathbf{X}^{(\mathsf{add})}$ scheme and the fact that a single bit in $\mathbf{X}^{(\mathsf{unary})}$ is 0 or 1 with constant probability. This will show that any dirty inner codeword is invalid with a constant probability. The total number of dirty inner codewords are represented by n_{dirty}.

There are only two ways to prepare inner codeword such that it is valid with probability 1. Based on these, we have following two kinds of inner codewords.

The inner codewords which are not dirty fall in two categories described below.

[6] Note that we only insist that $g^{(\mathsf{basic})}_{[umn]}$ and $h^{(\mathsf{basic})}_{[umn]}$ not only encode the same message s but also every outer codeword element is identical.

Fixed Inner Codewords. We say that an inner codeword is *completely fixed* if all its bits are obtained from bits in $I_0 \cup I_1$. That is, the whole codeword is explicitly written down using bits from $I_0 \cup I_1$. The number of such inner codewords is represented by n_{fixed}.

Copied Codewords. We say that an inner codeword is completely copied if it is generated by copying one whole inner codeword and (possibly) performing column preserving permutations. The number of such inner codewords is represented by n_{copy}.

Note that $n = n_{\mathsf{dirty}} + n_{\mathsf{fixed}} + n_{\mathsf{copy}}$.

4.2 Key Steps of Proof

Now we begin with our case analysis. We shall explain how $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ will be determined for various cases. We refer to the term $\mathrm{SD}\left(\mathsf{Tamper}^{(s)}_{I_0,I_1,I_p,c_{I_p},f}, \mathsf{Sim}^{(s)}_{\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}}\right)$ in Figure 7 as the simulation error. The expectation of simulation error over random choices of I_0, I_1, I_p and c_{I_p} is referred to as expected simulation error.

The threshold value $\log^{10}\kappa$ chosen below for analysis is arbitrary; any suitable $\mathrm{poly}\log\kappa$ will suffice.

Case 1. $n_{\mathsf{dirty}} \geqslant \log^{10}\kappa$. In this case, we have $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ output $\perp$ with probability 1. The simulation error in this case is $\mathsf{negl}(\kappa)$ because the probability that the tampered codeword has all valid inner encodings is $\leqslant \xi^{n_{\mathsf{dirty}}/2} = \mathsf{negl}(\kappa)$, where $\xi \in (0,1)$ is a constant (by Lemma 9).

Case 2. $n_{\mathsf{dirty}} < \log^{10}\kappa$. We shall show that it is highly unlikely (over random choices of I_0 and I_1) that $n_0 + n_1 \geqslant \log^{13}\kappa$ and $n_0 + n_1 \leqslant N - um\log^{10}\kappa$ and still we get this case. In particular, it is $\mathsf{negl}(\kappa)$ (see Lemma 17).

So, henceforth, we can assume that either $n_0 + n_1 < \log^{13}\kappa$ or $n_0 + n_1 > N - um\log^{10}\kappa$; at an expense of $\mathsf{negl}(\kappa)$ additive term in expected simulation error.

Case 2.1. $n_0 + n_1 \leqslant \log^{13}\kappa$. In this case, the tampering function copies most of the inner codewords into the tampered codeword, because $n_{\mathsf{copy}} = n - n_{\mathsf{dirty}} - n_{\mathsf{fixed}} \geqslant n - \log^{10}\kappa - \log^{13}\kappa/um$. So, the tampered codeword can either be invalid or (if valid) equivalent to the original codeword (because distance of the outer codeword $\gg \Theta(\log^{13}\kappa)$). Now the probability that the completely fixed and dirty codewords are each identical to their counterparts in the input codeword does not depend the message s because the privacy of the outer codeword is $\gg n_{\mathsf{dirty}} + n_{\mathsf{fixed}}$. This probability σ can be computed exhaustively and does not depend on the message s. So, $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ outputs same^* with probability σ; otherwise outputs $\perp$.

Case 2.2. $n_0 + n_1 > N - um\log^{10}\kappa$. Since the $n_0 + n_1$ is large and n_{dirty} is small, this implies that most inner-codewords have been completely fixed, because $n_{\mathsf{fixed}} = n - n_{\mathsf{dirty}} - n_{\mathsf{copy}} > n - \log^{10}\kappa - n_{\mathsf{copy}} > n - 2\log^{10}\kappa$. In this case, the tampered code word is either invalid or (if valid)

equivalent to the codeword consistent with the fixed inner codewords (due to high distance of the outer encoding scheme).

First check whether the fixed blocks can define a valid outer codeword. If not, then set $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ to output $\perp$ with probability 1. Again, simulation error in this case is 0.

Otherwise, we are in the case when the set of completely fixed inner codewords fixes the outer codeword. Let its outer codeword be g^* and the message corresponding to it be s^*. We say that (I_0, I_1) is *good* if it contains at least one bit from each column. Since we have $n_0 + n_1 > N - um \log^{10} \kappa$, we have $\Pr[(I_0, I_1) \text{ is good}] = 1 - \mathsf{negl}(\kappa)$ (by Lemma 18).

If (I_0, I_1) is not good, then we define $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ to output $\perp$ with probability 1. The simulation error for particular (I_0, I_1) can be at most 1; but this incurs additional $\mathsf{negl}(\kappa)$ expected simulation error over the choices of (I_0, I_1).

If (I_0, I_1) is good, then we need to check whether the set of remaining inner codewords of the tampered codeword are equivalent to the set of inner codewords of g^*. Note that if (I_0, I_1) is good then all bits of the original codeword restricted to $[N]\setminus(I_0 \cup I_1)$ are independent uniform random bits (because all proper subsets of columns have independent uniform bits). This can be exhaustively computed starting from uniformly random bits. Define this probability to be σ. Clearly, this probability is independent of the message s. We define $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ to output s^* with probability σ; and output $\perp$ with probability $1 - \sigma$. The simulation error in this case is 0.

There exists $\mathbb{D}$ such that the expected simulation error for our encoding scheme is $\mathsf{negl}(\kappa)$, for all n_0, n_1 and mapping function map, when $\mathcal{F} = \mathcal{S}_N$ and $n_p = 0$. For the full proof of Theorem 2 see Appendix C.

5 Main Construction

Our main encoding scheme is described in Figure 9. In order to show that it is resistant against the class of tampering attacks $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$, we first define two new random variables for the 2-phase non-malleability experiment:

$$\mathsf{Tamper}^{(s)}_{n_0,n_1,n_p,\mathsf{map}} := \mathop{\mathbb{E}}_{\substack{I_0,I_1\\ I_p,c_{I_p}}} \mathsf{Tamper}^{(s)}_{I_0,I_1,I_p,c_{I_p},f},$$

$$\mathcal{D}_{n_0,n_1,n_p,\mathsf{map}} := \mathop{\mathbb{E}}_{\substack{I_0,I_1\\ I_p,c_{I_p}}} \mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}.$$

In the above, given n_0, n_1, n_p and s, I_0, I_1, I_p and c_{I_p} are chosen as described in Figure 7. If we assume that the length of the codeword produced by the basic encoding scheme is $N^{(1)}$, then the mapping function is of the form:

$$\mathsf{map} : \binom{[N^{(1)}]}{n_0} \times \binom{[N^{(1)}]}{n_1} \times \binom{[N^{(1)}]}{n_p} \times \{0,1\}^{n_p} \to \widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_{N^{(1)}},$$

and $f = \mathsf{map}(I_0, I_1, I_p, c_{I_p})$. If a coding scheme is secure against the 2-phase non-malleability attack, then we can show that for all $n_0, n_1, n_p, \mathsf{map}$, there exists a distribution $\mathcal{D}_{n_0,n_1,n_p,\mathsf{map}}$ such that for

all s:

$$\mathrm{SD}\left(\mathsf{Tamper}^{(s)}_{n_0,n_1,n_p,\mathsf{map}}, \mathsf{Sim}^{(s)}_{\mathcal{D}_{n_0,n_1,n_p,\mathsf{map}}}\right) \leqslant \mathsf{negl}(\kappa). \tag{1}$$

We are now ready to state our lemma, whose proof is provided in Section 5.1.

Lemma 1. *For every $f \in \mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$, there exist n_0, n_1, n_p ($n_0 + n_1 + n_p \leqslant N^{(1)}$) and map (as defined above) such that,*

$$\mathrm{SD}\left(\mathsf{Tamper}^{(s)}_{f}, \mathsf{Tamper}^{(s)}_{n_0,n_1,n_p,\mathsf{map}}\right) \leqslant \mathsf{negl}(\kappa), \tag{2}$$

where $\mathsf{Tamper}^{(s)}_{f}$ is defined w.r.t the main encoding scheme (Figure 9) and $\mathsf{Tamper}^{(s)}_{n_0,n_1,n_p,\mathsf{map}}$ is defined w.r.t. the basic encoding scheme (Figure 8).

It is easy to see how this lemma implies Theorem 1. Let the n_0, n_1, n_p and map provided by the above lemma for a particular f be denote by $n_0^{(f)}, n_1^{(f)}, n_p^{(f)}$ and $\mathsf{map}^{(f)}$. For every $f \in \mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$, define $\mathcal{D}^*_f$ to be $\mathcal{D}_{n_0^{(f)},n_1^{(f)},n_p^{(f)},\mathsf{map}^{(f)}}$. By (1) and (2), it follows that:

$$\mathrm{SD}\left(\mathsf{Tamper}^{(s)}_{f}, \mathsf{Sim}^{(s)}_{\mathcal{D}^*_f}\right) \leqslant \mathsf{negl}(\kappa).$$

5.1 Proof of Lemma 1

Let $f_{\mathsf{set}}, f_{\mathsf{reset}}, f_{\mathsf{toggle}}$ and f_{forward} be functions which map an input bit $b \in \{0,1\}$ to an output bit as follows: $f_{\mathsf{set}}(b) = 1$, $f_{\mathsf{reset}}(b) = 0$, $f_{\mathsf{toggle}}(b) = 1 - b$ and $f_{\mathsf{forward}}(b) = b$. Consider any tampering function $f = (f_1, \ldots, f_N, \pi) \in \mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$, where $f_i \in \{f_{\mathsf{set}}, f_{\mathsf{reset}}, f_{\mathsf{toggle}}, f_{\mathsf{forward}}\}$ for $i \in [1, N]$ and $\pi \in \mathcal{S}_N$. Let $s \in \{0,1\}^L$ be a message that we would like to encode. We know that the codeword $c_{[N]}$ generated by Enc consist of two parts $c^{(1)}_{[N^{(1)}]}$ (left) and $c^{(2)}_{[N^{(2)}]}$ (right). Define $\mathcal{L} = [N^{(1)}]$ and $\mathcal{R} = \{N^{(1)} + 1, \ldots, N^{(2)}\}$. Further define the following sets of indices:

$$\hat{I}_0 = \{i \in \mathcal{L} \mid \pi(i) \in \mathcal{L} \text{ and } f_i = f_{\mathsf{reset}}\},$$

$$\hat{I}_1 = \{i \in \mathcal{L} \mid \pi(i) \in \mathcal{L} \text{ and } f_i = f_{\mathsf{set}}\},$$

$$\hat{I}_p = \{i \in \mathcal{L} \mid \pi(i) \in \mathcal{R}\}.$$

Note that given the tampering function f, the sets $\hat{I}_0, \hat{I}_1$ and $\hat{I}_p$ can be easily computed, and that these sets are disjoint.

First consider the case when $|\hat{I}_p| \geqslant \omega(\log \kappa)$, i.e., a substantial number of bits are being moved from the left part of the codeword to the right. Let $\tilde{c}^{(2)}_{[N^{(2)}]}$ denote the right part of the codeword after applying the tampering function f, i.e., $\tilde{c}^{(2)}_i := f_{\pi^{-1}(N^{(1)}+i)}(c_{\pi^{-1}(N^{(1)}+i)})$ for $i \in [N^{(2)}]$. We know that $\tilde{c}^{(2)}_{[N^{(2)}]}$ consists of $d+\ell-1$ elements of the field $\mathbb{F}$. We refer to these elements as blocks. A block is dirty iff a bit on the left part of the codeword is moved to this block through the permutation π. We know that there are at least $\omega(\log \kappa)$ and at most $N^{(1)} = d - 1$ dirty blocks (because the size of left codeword is $N^{(1)}$). Consider any ℓ blocks which are not dirty. If these blocks do not lie on an $\ell - 1$ degree polynomial, then the decoding algorithm of $\mathbf{X}^{(\mathsf{RS})}$ would output $\perp$. On the other

Let $\{0,1\}^L$ be the message space and $\{0,1\}^N$ be the codeword space. Let $N^{(1)} = umn$ be the size of codeword output by $\mathsf{Enc}_{\mathsf{basic}}$, and $d = N^{(1)} + 1$. Let $\mathbb{F}$ be a finite field of characteristic 2. Let $\ell = \left\lceil \frac{N^{(1)} \log N^{(1)}}{\log_2 |\mathbb{F}|} \right\rceil$, $n = d + \ell$ and $k = \ell$. Note that a permutation in $\mathcal{S}_{N^{(1)}}$ can be represented using $N^{(1)} \log N^{(1)}$ bits.

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

1. Choose a permutation $\sigma \in \mathcal{S}_{N^{(1)}}$ at random.
2. Let $c^{(1)}_{[N^{(1)}]} \sim \sigma(\mathbf{X}^{(\mathsf{basic})}_{[N^{(1)}]} | \mathbf{X}^{(\mathsf{basic})}_0 = s)$.
3. Let $c^{(2)}_{[N^{(2)}]} \sim \left(\mathbf{X}^{(\mathsf{RS},n,k,\ell,\mathbb{F})}_{[n]} | \mathbf{X}^{(\mathsf{RS},n,k,\ell,\mathbb{F})}_0 = \sigma\right)$. Here we interpret σ as an element in $\mathbb{F}^\ell$; and the shares $\in \mathbb{F}^n$ as an element in $\{0,1\}^{N^{(2)}}$.
4. Output $c_{[N]} := (c^{(1)}_{[N^{(1)}]}, c^{(2)}_{[N^{(2)}]})$.

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

1. Let $\left(\tilde{c}^{(1)}_{[N^{(1)}]}, \tilde{c}^{(2)}_{[N^{(2)}]}\right) \equiv \tilde{c}_{[N]}$.
2. Decode $\tilde{c}^{(2)}_{[N^{(2)}]}$ by the decoding algorithm of $\mathbf{X}^{(\mathsf{RS})}$ to obtain a permutation $\tilde{\sigma}$.
3. Output $\tilde{s} \in \{0,1\}^L$ obtained by decoding $\tilde{\sigma}^{-1}(\tilde{c}^{(1)}_{[N^{(1)}]})$ according to the decoding algorithm of $\mathbf{X}^{(\mathsf{basic})}$. (If either of the decoding algorithms fail, output $\perp$.)

Figure 9: Main Non-malleable Code

hand, suppose that the non-dirty blocks do lie on such a polynomial. In this case, the remaining blocks should have a fixed value. We know that at least $\omega(\log \kappa)$ of them are dirty, and that any $\omega(\log \kappa)$ bits in the left codeword are uniformly distributed. Hence, the probability that the dirty blocks will have the desired fixed value is negligible. Therefore, when $|\hat{I}_p| \geqslant \omega(\log \kappa)$, $\mathsf{Tamper}_f^{(s)}$ is $\perp$ with all but negligible probability irrespective of s. In this case, we set n_0, n_1, n_p and map so that $\mathsf{Tamper}^{(s)}_{n_0,n_1,n_p,\mathsf{map}}$ is $\perp$ with high probability too.[7]

On the other hand if $|\hat{I}_p| \leqslant \omega(\log \kappa)$, set $n_0 = |\hat{I}_0|, n_1 = |\hat{I}_1|, n_p = |\hat{I}_p|$. Let $I_0, I_1, I_p \subseteq \mathcal{L}$ be three random, disjoint sets of indices of sizes n_0, n_1 and n_p respectively. Also, let $c^*_{[N^{(1)}]}$ be a codeword sampled randomly from $\mathsf{Enc}_{\mathsf{basic}}(s)$. (In the following, we would only use $c^*_{I_p}$ to define map.) Choose a random permutation $\sigma \in \mathcal{S}_{N^{(1)}}$ such that $\sigma(I_b) = \hat{I}_b$ for $b \in \{0,1\}$ and $\sigma(I_p) = \hat{I}_p$. Draw $c^{(2)}_{[N^{(2)}]}$ as described in Figure 9. Observe that the distribution of $(\sigma(c^*_{[N^{(1)}]}), c^{(2)}_{[N^{(2)}]})$ is identical to the output of Enc algorithm.

In order to obtain the tampered codeword $\tilde{c}^{(2)}_{[N^{(2)}]}$, we define a tampering function $g = (g_1, \ldots, g_{N^{(2)}}, \pi_R)$ which operates on the right part of the codeword. Let $\hat{W} := \{i \in \mathcal{R} \mid \pi(i) \in \mathcal{L}\}$ be the indices in the right codeword that move to the left. Recall that $\hat{I}_p$ is the set of indices that move the opposite way, i.e., from left to right. Since π is a permutation, we know that $|\hat{W}| = |\hat{I}_p|$. We want to define a permutation π_R restricted to the indices in $\mathcal{R}$. However, since there are bits that move across, we first let μ be an arbitrary bijection from $\hat{W}$ to $\hat{I}_p$. Then $\pi_R : \mathcal{R} \to \mathcal{R}$ is given by

$$\pi_R(i) = \begin{cases} \pi(i) & \text{if } i \notin \hat{W} \\ \pi(\mu(i)) & \text{if } i \in \hat{W}. \end{cases}$$

Note that since $\mu(i) \in \hat{I}_p$, $\pi(\mu(i)) \in \mathcal{R}$. We define $g_1, \ldots, g_{N^{(2)}}$ as follows:

$$g_i(b) = \begin{cases} f_j & \text{if } j \notin \hat{W} \\ f_{\mathsf{set}} & \text{if } j \in \hat{W} \text{ and } f_{\mu(j)}(c^*_{\sigma^{-1}(\mu(j))}) = 1 \\ f_{\mathsf{reset}} & \text{if } j \in \hat{W} \text{ and } f_{\mu(j)}(c^*_{\sigma^{-1}(\mu(j))}) = 0, \end{cases}$$

where $j = N^{(1)} + i$. Observe that $\mu(j) \in \hat{I}_p$, and hence $\sigma^{-1}(\mu(j)) \in I_p$.

Apply $g_1, \ldots, g_{N^{(2)}}$ followed by π_R to $c^{(2)}_{[N^{(2)}]}$ to obtain the tampered codeword $\tilde{c}^{(2)}_{[N^{(2)}]}$. If $\tilde{c}^{(2)}_{[N^{(2)}]}$ is not a valid codeword, set $\mathsf{map}(I_0, I_1, I_p, c^*_{I_p}) = \hat{f}$, where $\hat{f} \in \widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_{N^{(1)}}$ such that $\mathsf{Dec}_{\mathsf{basic}}(\hat{f}(c^*_{N^{(1)}})) = \perp$ with high probability. Otherwise, let $\tilde{\sigma}$ be the decoded permutation. Set $\mathsf{map}(I_0, I_1, I_p, c^*_{I_p}) = f^*$, where $f^* = (f^*_1, \ldots, f^*_{N^{(1)}}, \pi^*)$ is defined as follows. Let $\pi_L : \mathcal{L} \to \mathcal{L}$ be a permutation given by

$$\pi_L(i) = \begin{cases} \pi(i) & \text{if } i \notin \hat{I}_p \\ \pi(\mu^{-1}(i)) & \text{if } i \in \hat{I}_p. \end{cases}$$

[7] It is easy to note that there exists tampering functions which always produce invalid codes.

Set $\pi^* := \tilde{\sigma}^{-1} \circ \pi_L \circ \sigma$. On the other hand, $f_1^*, \ldots, f_{N^{(1)}}^*$ are given by:

$$f_i^* = \begin{cases} f_{\mathsf{set}} & \text{if } i \in I_1 \\ f_{\mathsf{reset}} & \text{if } i \in I_0 \\ f_{\mathsf{forward}} & \text{if } i \in I_p \text{ and } c_i^* = f_{\mu^{-1}(\sigma(i))}(c^{(2)}_{\mu^{-1}(\sigma(i))-N^{(1)}}) \\ f_{\mathsf{toggle}} & \text{if } i \in I_p \text{ and } c_i^* \neq f_{\mu^{-1}(\sigma(i))}(c^{(2)}_{\mu^{-1}(\sigma(i))-N^{(1)}}) \\ f_{\sigma(i)} & \text{otherwise.} \end{cases}$$

Note that $f_{\sigma(i)} \in \{f_{\mathsf{forward}}, f_{\mathsf{toggle}}\}$.

We show that given $\tilde{c}^{(2)}_{[N^{(2)}]}$ decodes to $\tilde{\sigma}$, $\tilde{\sigma}^{-1}(f(\sigma(c^*_{[N^{(1)}]}))) = f^*(c^*_{[N^{(1)}]})$. This would imply that $\mathsf{Dec}_{\mathsf{basic}}(f^*(c^*_{[N^{(1)}]}))$ exactly matches with

$$\mathsf{Dec}(f(\mathsf{Enc}(s)) = \mathsf{Dec}^*_{\mathsf{basic}}(\tilde{\sigma}^{-1} f(\mathsf{Enc}(s)) = \mathsf{Dec}^*_{\mathsf{basic}}(\tilde{\sigma}^{-1}(f(\sigma(c^*_{[N^{(1)}]}))))).$$

In the following, we use $h_1 h_2 \ldots h_n(x)$ as a shorthand for $h_1(h_2(\ldots h_n(x)\ldots))$ for clarity, where $h_1, \ldots, h_n$ are functions. We also let $m = N^{(1)}$. Let $c^{*(\sigma, f_1, \ldots, f_m)}$ be the codeword obtained after applying σ followed by $f_1, \ldots, f_m$ on $c^*_{[m]}$. We know that for $i \in [1, m]$,

$$c_i^{*(\sigma, f_1, \ldots, f_m)} = \begin{cases} 0 & \text{if } \sigma^{-1}(i) \in I_0 \\ 1 & \text{if } \sigma^{-1}(i) \in I_1 \\ f_i(c^*_{\sigma^{-1}(i)}) & \text{otherwise.} \end{cases}$$

Further, when we apply the permutation π, we get a codeword $c^{*(\sigma,f)}_{[m]}$ such that for $i \in [1, m]$,

$$c_i^{*(\sigma,f)} = \begin{cases} 0 & \text{if } \sigma^{-1}\pi^{-1}(i) \in I_0 \\ 1 & \text{if } \sigma^{-1}\pi^{-1}(i) \in I_1 \\ f_{\pi^{-1}(i)}(c^{(2)}_{\pi^{-1}(i)-m}) & \text{if } \sigma^{-1}\mu\pi^{-1}(i) \in I_p \\ f_{\pi^{-1}(i)}(c^*_{\sigma^{-1}\pi^{-1}(i)}) & \text{otherwise,} \end{cases}$$

where μ is the bijection from $\hat{W}$ to $\hat{I}_p$. Using π_L, we can rewrite the above as follows:

$$c_i^{*(\sigma,f)} = \begin{cases} 0 & \text{if } \sigma^{-1}\pi_L^{-1}(i) \in I_0 \\ 1 & \text{if } \sigma^{-1}\pi_L^{-1}(i) \in I_1 \\ f_{\pi^{-1}(i)}(c^{(2)}_{\pi^{-1}(i)-m}) & \text{if } \sigma^{-1}\pi_L^{-1}(i) \in I_p \\ f_{\pi_L^{-1}(i)}(c^*_{\sigma^{-1}\pi_L^{-1}(i)}) & \text{otherwise.} \end{cases}$$

Finally, applying $\tilde{\sigma}^{-1}$, we get a codeword $c^{*(\sigma,f,\tilde{\sigma}^{-1})}_{[m]}$ such that

$$c_i^{*(\sigma,f,\tilde{\sigma}^{-1})} = \begin{cases} 0 & \text{if } \sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i) \in I_0 \\ 1 & \text{if } \sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i) \in I_1 \\ f_{\pi^{-1}\tilde{\sigma}(i)}(c^{(2)}_{\pi^{-1}\tilde{\sigma}(i)-m}) & \text{if } \sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i) \in I_p \\ f_{\pi_L^{-1}\tilde{\sigma}(i)}(c^*_{\sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i)}) & \text{otherwise.} \end{cases}$$

On the other hand, when f^* is applied on $c^*_{[m]}$, we get a codeword $\tilde{c}^*_{[m]}$ such that

$$\tilde{c}^*_i = \begin{cases} 0 & \text{if } \sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i) \in I_0 \\ 1 & \text{if } \sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i) \in I_1 \\ f_{\mu^{-1}\sigma\sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i)}(c^{(2)}_{\mu^{-1}\sigma\sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i)-m}) & \text{if } \sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i) \in I_p \\ f_{\sigma\sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i)}(c^*_{\sigma^{-1}\pi_L^{-1}\tilde{\sigma}(i)}) & \text{otherwise.} \end{cases}$$

Simplifying using the fact that $\pi_L^{-1}(i) = \mu\pi^{-1}(i)$ for $i \in \pi_L(\hat{I}_p)$, we can easily see that for all $i \in [m]$, $c_i^{*(\sigma,f,\tilde{\sigma}^{-1})} = \tilde{c}^*_i$. This completes the proof.

Observe that though the function map constructed above is randomized, one can easily show that there must exist a deterministic function which satisfies (2).

6 Application to Non-malleable Commitments

6.1 Non-malleability Definitions

Let $\langle C, R\rangle$ denote a statistically binding and computationally hiding commitment scheme. We assume w.l.o.g. that the scheme has a non-interactive reveal phase: committer simply sends the value v along with decommitment information d, and it is verified using a function $\mathsf{open}(c, v, d)$ where c is the commitment transcript. We also assume that the scheme is efficiently checkable: to accept or reject, R applies a public function on the transcript c.

CCA-security. We consider an adversary A who has access to a decommitment oracle $\mathcal{O}$. $\mathcal{O}$ participates with A in many *concurrent* sessions of (the commit phase of) $\langle C, R\rangle$; at the end of each session, if the session is accepting, $\mathcal{O}$ returns the (unique) value committed by A in that session; otherwise it returns $\perp$.[8] For a bit b and auxiliary input z, let $\mathsf{BIT}_b(\langle C, R\rangle, \mathcal{O}, A, n, z)$ denote the output of the following experiment: on input $(1^n, z)$, $A^{\mathcal{O}}$ receives a commitment to b while simultaneously interacting with $\mathcal{O}$. If $A^{\mathcal{O}}$ sends a commitment to $\mathcal{O}$ whose transcript is identical to that of the left commitment, experiment aborts and outputs $\perp$; otherwise it outputs whatever $A^{\mathcal{O}}$ outputs.

Definition 3 (CCA-secure Bit Commitments, [CLP10])**.** *Let $\langle C, R\rangle$ be a bit commitment scheme and $\mathcal{O}$ be a decommitment oracle for it. We say that $\langle C, R\rangle$ is* CCA-secure w.r.t. *$\mathcal{O}$, if for every PPT A, every $z \in \{0,1\}^*$ it holds that:*

$$\mathsf{BIT}_0(\langle C, R\rangle, \mathcal{O}, A, n, z) \quad \approx_c \quad \mathsf{BIT}_1(\langle C, R\rangle, \mathcal{O}, A, n, z).$$

We say that $\langle C, R\rangle$ is CCA-secure if there exists a decommitment oracle $\mathcal{O}'$ such that $\langle C, R\rangle$ is CCA-secure w.r.t. $\mathcal{O}'$.

[8] If there is more than one possible decommitment, $\mathcal{O}$ returns any one of them. Note that since $\langle C, R\rangle$ is efficiently checkable, and the session is accepting, such a valid decommitment always exists. In addition, note that since we only have statistical binding, this value is unique except with negligible probability.

An analogous version of the definition which considers many concurrent executions on left (instead of just one), is known to be equivalent to the current definition via a simple hybrid argument [PR05a].

Bounded parallel security. Let $t(n)$ be an arbitrary polynomial. We say that an adversary A defined above is a *t-bounded-parallel* adversary w.r.t. $\mathcal{O}$ if it makes at most $k \leqslant t(n)$ commitments to $\mathcal{O}$ and all k sessions are executed in parallel.

Definition 4 (t-Bounded-parallel CCA-secure Bit Commitments)**.** *We say that $\langle C, R\rangle$ is t-bounded-parallel CCA-secure if it is CCA-secure (satisfies Definition 3) w.r.t. all t-bounded-parallel adversaries.*

Non-malleable string commitment. For a bit b and auxiliary input z, let $\mathsf{STR}_b(\langle C, R\rangle, \mathcal{O}, A, n, z)$ denote the output of the following experiment: on input $(1^n, z)$, A adaptively chooses two *strings* (v_0, v_1) of length n, and receives a commitment to v_b while simultaneously A also interacts with a receiver, attempting to commit to some value. Define $\widetilde{v}$ to be the value contained in the right commitment.[9]. If A's commitment transcript on right is either not accepting, or identical to the transcript on left, then output a special symbol $\perp$; if $\widetilde{v} = v$, output a special symbol same^*; otherwise, output $\widetilde{v}$.[10]

Definition 5 (Non-malleable String Commitments)**.** *We say that $\langle C, R\rangle$ is a non-malleable string commitment if for every PPT A and every $z \in \{0,1\}^*$ it holds that*

$$\mathsf{STR}_0(\langle C, R\rangle, A, n, z) \quad \approx_c \quad \mathsf{STR}_1(\langle C, R\rangle, A, n, z).$$

6.2 Non-malleable Sting Commitments from Non-malleable Bit-commitments

Construction. Given a bit commitment scheme $\langle C, R\rangle$, we construct a string commitment scheme $\langle C', R'\rangle$ for $\{0,1\}^n$ as follows. Let $\mathsf{nm\text{-}code}$ be a non-malleable coding scheme for messages of length n that is robust to $\mathcal{F} := \mathcal{F}_{0,1} \circ \mathcal{S}_N$, and let $t(n)$ denote the length of the codewords for some fixed polynomial t. Let Enc and Dec be encoding and decoding algorithms. To commit to a string $v \in \{0,1\}^n$, C' generates a random codeword $w \leftarrow \mathsf{Enc}(v)$, and commits to each bit of w independently, and in parallel using the bit-commitment protocol $\langle C, R\rangle$. The receiver checks that no two bit-commitment transcripts, out of t such transcripts, are identical. If the check fails, or if any of the bit-commitment transcripts are not accepting, the receiver rejects; otherwise it accepts the commitment. To decommit to v, the receiver sends v along with decommitment information for each bit of w denoted by (w_i, d_i) for every $i \in [t]$; the receiver accepts v if and only if all recommitments verify and the resulting codeword decodes to v.

We now prove Theorem 3 by proving that this reduction results in a non-malleable string commitment scheme.

[9] Note that $\widetilde{v}$ is unique w.h.p. and there exists $\widetilde{d}$ s.t. $\mathsf{open}(\widetilde{c}, \widetilde{v}, \widetilde{d}) = 1$ where $\widetilde{c}$ is the right commitment.

[10] Following [DDN91], this definition allows MIM to commit to the same value. It is easy to prevent MIM from committing the same value generically in case of string commitments: convert the scheme to tag based by appending the tag with v, and then sign the whole transcript using the tag.

Proof. To prove the theorem, we prove that scheme $\langle C', R' \rangle$ satisfies Definition 5. We observe that the reduction is unconditional and preserves round complexity. Note that in our setting $N = t(n)$, and we use N and t interchangeably. We now show that distributions STR_0 and STR_1 are computationally indistinguishable.

We first describe a procedure that defines a distribution over tampering functions $f \in \mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ based on the A's behavior. Let $\mathcal{O}$ be a decommitment oracle for $\langle C, R \rangle$. The procedure $\mathcal{E}^{\mathcal{O}}$ has access to $\mathcal{O}$, takes a bit b as input, and works as follows.

Procedure $\mathcal{E}^{\mathcal{O}}(b)$: Procedure incorporates A, n, z, and initiates an execution of A on $(1^n, z)$. It forwards all right interactions of A to $\mathcal{O}$, and interacts on left as follows. When A sends (v_0, v_1), $\mathcal{E}$ computes codewords w_0, w_1 corresponding to these values, and then commits to codeword w_b using $\langle C', R' \rangle$. When A sends its last message on the right, $\mathcal{E}$ checks that all $t(n)$ bit-commitments on right are accepting and that no two of them have identical transcripts. If the test fails, it outputs $\perp$; otherwise, it outputs a tampering function f constructed as follows (we view f as a table of t entries, where each entry contains either a unique number in $[t]$ or a string in $\{\mathsf{set}, \mathsf{reset}\}$):[11]

1. For every right transcript i that is not an exact copy of any of the t transcripts on left, $\mathcal{E}$ forwards the last message of this session to $\mathcal{O}$ and receives a decommitment to either 0 or 1. If the received value is 0, it writes reset in the i-th position of f, and if the value is 1, it writes set. (In the unlikely event $\mathcal{O}$ does not return a bit, $\mathcal{E}$ aborts.)
2. For every right transcript i that is a copy of a unique left transcript j, $\mathcal{E}$ writes the number j in i-th position of f.

Let $f_b := \mathcal{E}^{\mathcal{O}}(b)$. Let w_b and $\widetilde{w_b}$ denote the distributions of codewords on left-side and right-side (committed by A) in the execution of $\mathcal{E}^{\mathcal{O}}(b)$. We note that these distribution are well defined and only depend on A, n, z (and $\mathcal{O}$ as well, but any valid oracle will yield statistically similar distributions). By construction, since there are no repeated transcripts, $f_b \in \mathcal{F}_{0,1} \circ \mathcal{S}_N$ w.h.p. for all $b \in \{0, 1\}$ (if A does not abort). Further, by definition:

$$\widetilde{w_b} = f_b(w_b).$$

Now consider the experiment defining distribution STR_b. We observe that, by construction, the execution of this experiment is identical to the internal execution of $\mathcal{E}^{\mathcal{O}}(b)$, and therefore if we let $\widetilde{v_b}$ the value in the commitment on right, then $\widetilde{v_b}$ is distributed identically to $\mathsf{Dec}(f_b(w_b))$ where Dec is the decoding algorithm of $\mathsf{nm\text{-}code}$.

For two bits a, b, define the following game $G^{\mathcal{O}}(a, b)$:

Game $G^{\mathcal{O}}(a, b)$: The game first proceeds exactly as the execution of $\mathcal{E}^{\mathcal{O}}(a)$ and obtains a tampering function distributed as f_a; it then applies f_a to w_b and if $f_a(w_b) = w_b$ it reruns same^*, otherwise it returns the decoding of the resulting codeword: i.e., the message $\underline{X_{a,b} := \mathsf{Dec}(f_a(w_b))}$ where (w_0, w_1) are two codewords sampled in the beginning (when following steps of $\mathcal{E}^{\mathcal{O}}(a)$).

[11] For this reduction it is not necessary for the $\mathsf{nm\text{-}code}$ to be robust against toggle.

Observe that STR_b is distributed identically to $X_{b,b}$, and we need to show that $X_{0,0} \approx_c X_{1,1}$. We do this in two steps: we first show that $X_{0,0} \approx_c X_{1,0}$ and $X_{0,1} \approx_c X_{1,1}$; then we show that $X_{0,1} \approx_c X_{1,0}$.

Lemma 2. *$X_{0,0} \approx_c X_{1,0}$ and $X_{0,1} \approx_c X_{1,1}$.*

Proof. We prove that $X_{0,0} \approx_c X_{1,0}$. The proof of the second part will be identical to that of the first part. The proof follows from the CCA-security of the bit-commitment scheme.

Formally, to prove that $X_{0,0} \approx_c X_{1,0}$, we construct a sequence of t hybrids $\{H_i\}_{i \in t}$, each of which has access to the oracle $\mathcal{O}$, as follows. $H_0^{\mathcal{O}}$ is identical to game $G_{0,0}^{\mathcal{O}}$ in which permutation f_0 is applied to w_0.

Hybrid $H_i^{\mathcal{O}}$: H_i proceeds identically to H_{i-1} except that it constructs its codewords differently. When A produces v_0, v_1, H_i defines string v_i^* as follows: first i bits of v_i^* are identical to the first i bits of v_1, and the rest $t - i$ bits are identical to the last $t - i$ bits of v_0. It then sets: $w_0^i = \mathsf{Enc}(v_i^*)$ and then commits to w_0^i (instead of w_0—a codeword of v_0). The rest of the execution proceeds identically to H_{i-1}, and hybrid constructs a tampering function, denoted f_0^i, exactly as procedure $\mathcal{E}$ and then outputs $\mathsf{Dec}(f_0^i(w_1))$.

Observe that $H_t^{\mathcal{O}}$ is identical to $G^{\mathcal{O}}(0,1)$, and f_0^t is identical to f_1; output of H_t is identical to $X_{1,0}$. We claim that for every $i \in [t]$: $H_{i-1} \approx_c H_i$. If this is not true for some i, then we can construct an adversary $A_{\mathsf{bit}}^{\mathcal{O}}$ against the CCA security of $\langle C, R \rangle$ as follows. $A_{\mathsf{bit}}^{\mathcal{O}}$ interacts with an outside challenger as follows. It starts and executes identically to hybrid H_{i-1} for all commitments on left and right except for the commitment corresponding to the i-th bit on left side. Instead, it receives this commitment from the outside challenger and plays it as its own i-th commitment. In the end, $A_{\mathsf{bit}}^{\mathcal{O}}$ outputs $\mathsf{Dec}(f_0^i(w_0))$. W.l.o.g. let i-th bits of v_0 and v_1 be 0 and 1 respectively: then, if the challenger commits to b, A_{bit}'s execution is identical to H_{i-b+1}. It follows that the two hybrids must be indistinguishable. And since there are only polynomially many hybrids, we have that $X_{0,0} \approx_c X_{1,0}$. The claim for the second part follows in a near identical fashion, with a simple change of variables. □

Lemma 3. *$X_{1,0} \approx_c X_{0,1}$.*

Proof. The proof of this part will make use of the non-malleability of $\mathsf{nm\text{-}code}$, in addition to the CCA-security of the bit-commitment scheme. Before proceeding further, let us note that f_0 and f_1 are well defined distribution, and in particular can be sampled (though inefficiently) using procedure $\mathcal{E}^{\mathcal{O}}(0)$ and $\mathcal{E}^{\mathcal{O}}(1)$.

To prove the claim, we define $Y_{0,0}$ to be the output of the following experiment which uses the oracle $\mathcal{O}$: the experiment samples f_0 using A and $\mathcal{O}$ (e.g., following $\mathcal{E}^{\mathcal{O}}(0)$); it then randomly samples an *independent* codeword w_0^* for the value v_0 and returns the decoding of $f_0(w_0^*)$.

We first claim that due to the CCA security of $\langle C, R \rangle$, it holds that $X_{1,0} \approx_c Y_{0,0}$. By computational hiding of the commitment scheme, we have that $X_{1,0} \approx_c Y_{0,0}$. First observe that in $X_{1,0}$, f_1 is applied on an *independent codeword* w_0 of the (adversarially chosen value v_0); the same holds in $Y_{0,0}$ but with function f_0 independent of the codeword of v_0. Therefore, the indistinguishability of these two variables follows in identical fashion as the proof of the previous lemma.

Finally, we claim that $Y_{0,0} \approx_s X_{0,1}$ (statistically close). In both cases, f_0 is applied to an independently generated codeword; note that if f_0 does not change the codeword, the distributions return symbol same^*. Therefore, even though f_0 is applied to codewords that correspond to different messages in two hybrids, the output is same^* in case the tampered codewords do not change, and $\perp$ otherwise with high probability. By non-malleability of our code, both of these distributions are statistically close to a simulated distribution that only depends (on the fixed distribution of) f_0. Therefore we have $X_{1,0} \approx_c Y_{0,0} \approx_s X_{0,1}$. □

This completes the proof of the theorem. □

The proof the theorem below is nearly identical to the proof of Theorem 3: it follows from observing that the MIM in the proof has to make only t commitments to $\mathcal{O}$ and all of them can be done in parallel. Let t be a polynomial bounding the length of the codewords of our scheme.

Theorem 4 ($t(n)$-Bounded-parallel Bit-commitment to Non-malleable String Commitment)**.** *There exists a simple and efficient black-box compiler which, when provided with:*

- *A non-malleable encoding robust to $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$, and*
- *A r-round (possibly* **non-tag-based***) $t(n)$-bounded-parallel bit-commitment scheme*

yields a r-round non-malleable string-commitment scheme.

6.3 From Partial Non-malleability to Full Non-malleability

We now show that our non-malleable codes can help upgrade a protocol which has only partial non-malleability to one with full non-malleability. This section is at an informal level where we try to demonstrate our ideas through an example only.

Consider Naor's 2-round commitment based on *adaptive* PRGs as our starting point. Let $G : \{0,1\}^n \to \{0,1\}^{3n}$ be a pseudorandom generator (PRG) where n is the security parameter. In Naor's scheme, the receiver sends a random string $r \in \{0,1\}^{3n}$; to commit to 0, the committer sends $z = G(s)$ and to commit to 1 it sends $z \oplus r$ where $\oplus$ denotes bitwise exclusive-or and $s \in \{0,1\}^n$ is a random seed. If the PRG is an *adaptive* PRG[12] then this scheme is a *partially* non-malleable bit commitment scheme: it allows A to toggle the committed bit, but (probably) not much more. This is because, given access to the inversion oracle $\mathcal{O}_G$, A can learn the value it commits to on "right"; if the scheme were not non-malleable, it compromises the hiding of commitment on "left", contradicting the adaptive security of the PRG G. Although the scheme is not non-malleable, it has some flavor of non-malleability which can be coupled with non-malleable codes to get full security.

We only aim to achieve a slightly weaker definition of non-malleability, called *non-malleability w.r.t. replacement* [Goy11] (building upon [Wee10]). Informally, this is the same as usual definition of

[12]Roughly speaking, following [PPV08, KMO10], G is said to be adaptively secure if no PPT adversary A can tell if $y = G(s)$ for a random $s \in \{0,1\}^n$ or y is uniform *even if* A has access to a special *inversion* oracle $\mathcal{O}_G$; on query a string z of length $3n$, the oracle tells whether z is in the range of G or not. A is not allowed to query the challenge string y.

non-malleability except that whenever the A sends a commitment for which no valid decommitment exists, the definition considers A to have "admitted defeat." When this happens, the definition allows a "simulator" to *replace* the invalid value (denoted by $\perp$) by *any* arbitrary value of its choice (hat helps maintain an indistinguishable distribution. Such weaker definitions suffice for a large class of applications of non-malleability [Goy11, Wee10].

There are 4 ways in which A can "admit defeat" as above. These cases are listed below, and we say that A acts as a defeat channel on the received commitment:

(1) when A receives commitment to a bit 0 on left, it commits to 0 on right, but if it receives commitment to 1, it commits to $\perp$; we denote this by $\mathsf{defeat}_{0\to 0,1\to\perp}$.

(2) opposite of the first case, denoted by $\mathsf{defeat}_{1\to 1,0\to\perp}$.

(3) "toggle" variant of case (1) where A commits to 1 if it receives a commitment to 0 and $\perp$ in the other case; this is denoted by $\mathsf{defeat}_{0\to 1,1\to\perp}$.

(4) opposite of (3), denoted by: $\mathsf{defeat}_{1\to 0,0\to\perp}$.

We need a non-malleable code which, in addition to tolerating permutation, $\mathsf{set}, \mathsf{reset}, \mathsf{toggle}$ attacks, also tolerates these all four defeat attacks described above. More precisely, let $\mathcal{F}^*$ be a class of tampering functions where every function $f \in \mathcal{F}^*$ is fully specified by a string of n' entries where each entry either contains a unique number $i \in [n']$ or an entry from $\{\mathsf{set}, \mathsf{reset}, \mathsf{toggle}_j\}$ or an entry from $\mathsf{defeat} := \{\mathsf{defeat}_{0\to 0,1\to\perp}, \mathsf{defeat}_{1\to 1,0\to\perp}, \mathsf{defeat}_{0\to 1,1\to\perp}, \mathsf{defeat}_{1\to 0,0\to\perp}\}$. Note that only the last requirement is an additional requirement in $\mathcal{F}^*$ when compared to $\mathcal{F} \circ \mathcal{S}_N$. Here n' is the length of the codeword for n-bit strings. When an entry contains a number i, will write copy_i instead of just i to mean that it is a copy of i-th bit of the original codeword. To summarize, $f \in \mathcal{F}^*$ is then described by n' actions where $\mathsf{action} \in \{\mathsf{set}, \mathsf{reset}\} \cup \{\mathsf{copy}_i\}_{i\in[n']} \cup \{\mathsf{toggle}_i\}_{i\in[n']} \cup \mathsf{defeat}$.

On input a codeword $w \in \{0,1\}^{n'}$, the output $w' = f(w)$ corresponding to $f := \{\mathsf{action}_i\}_{i=1}^{n}$ is defined as before: for every $i \in [n']$, if $\mathsf{action}_i = \mathsf{set}$, then $w'_i = 1$; if $\mathsf{action}_i = \mathsf{reset}$, then $w'_i = 0$; if $\mathsf{action}_i = \mathsf{copy}_j$, then $w'_i = w_j$; if $\mathsf{action}_i = \mathsf{toggle}_j$, then $w'_i = 1 - w_j$; finally, if $\mathsf{action}_i \in \mathsf{defeat}$ bit w'_i is defined to be either 0 or $\perp$, according to items (1)–(4) above, depending on the value of w_j and the type of the "defeat" action.

To commit to a string x of length, say n, our commitment scheme first encodes x using a non-malleable code that is secure against $\mathcal{F}^*$. It then commits to each bit of the resulting codeword using Naor's bit commitment scheme (instantiated using an adaptive PRG G). These commitments can be done in parallel. The receiver accepts a string commitment as follows. Let (r_i, t_i) be the two messages of i-th bit commitment; by construction $t_i = z_i$ or $z_i \oplus r_i$ for some string z_i in the range of G. Note that z_i is always well defined for honestly generated commitments. Define set $S_i := \{z_i, z_i \oplus r_i\} = \{t_i, t_i \oplus r_i\}$. Then, the receiver accepts the commitment if for all distinct i, i' it holds that $S_i \cap S_{i'} = \emptyset$. It is easy to check that this holds with high probability for honestly created commitments.

We claim that the above scheme is a non-malleable string commitment scheme. In fact, we only show that, as before, the attack by an adversary A translates to a tampering attack f on the underlying codeword for some $f \in \mathcal{F}^*$. Further, this f can be extracted with the help of the inversion oracle

for G, say $\mathcal{O}_G$. Thereafter, one can apply the arguments as in the previous section to argue full non-malleability.[13]

We show how to construct f given $c, c' \leftarrow A(c)$ and access to the inversion oracle $\mathcal{O}_G$ without compromising the hiding property of left commitment c. Let $c = \{c_i\}_{i\in[n']} = \{(r_i, t_i)\}_{i\in[n']}$ be the commitment given to A on left and $c' = \{c'_j\}_{j\in[n']} = \{(r'_j, t'_j)\}_{j\in[n']}$. Our goal is to construct $f \in \mathcal{F}^*$ *without* violating the hiding of the commitments on left. To do this, we have to be careful to not query the oracle $\mathcal{O}_G$ on any value which might violate the hiding of commitments on left. Indeed, A might carefully select string r_i sent on left, or, t'_j sent on right which reveal useful information via the answers of $\mathcal{O}_G$. Therefore, we will ensure that we never query such strings. These strings of interest are: t'_j, $t'_j \oplus r'_j$, $t'_j \oplus r_i$, $t'_j \oplus r'_j \oplus r_i$ for every j and i in $[n']$.[14]

Recall that we defined set $S_i = \{t_i, t_i \oplus r_i\}$ for commitments on left; define $S'_j = \{t_j, t_j \oplus r_j\}$ analogously for commitments on right. Further recall that $S_i \cap S_{i'} = \emptyset$ for all distinct i, i' w.h.p. for honestly generated commitments, and the commitment on right is accepted if and only if $S'_j \cap S'_{j'} = \emptyset$ for all distinct j, j'. Observe that except with negligible probability, it holds that for every j there do not exist distinct indices i, i' (corresponding to left commitments) such that $S'_j \cap S_i \neq \emptyset$ *and* $S'_j \cap S_{i'} \neq \emptyset$. This is because S_i and $S_{i'}$ do not intersect, and therefore if the claim were false, we must have, w.l.o.g., $t'_j \in S_i$ and $t'_j \oplus r_j \in S_{i'}$; but the later can only happen with negligible probability since r_j and $t_{i'}$ are honestly generated.

This allows us to define the parent of every right commitment j as follows. If there exists an i-th on left such that $S_i \cap S'_j \neq \emptyset$, define $\mathsf{parent}(j) = i$. Note that by the argument above, for every j, if there exists a parent then such a parent is *unique* with high probability.

Given c', we now construct $f = \{\mathsf{action_j}\}_{j\in[n']}$ as follows. For every j on the right:

1. if $\mathsf{parent}(j)$ does not exist, query the oracle $\mathcal{O}_G$ on all strings in the set $X_j := S'_j \cup (\cup_i S_i \oplus r_j)$.[15] It is easy to check that the j-th commitment sent by A on right appears in X_j; let b_j be the bit committed to in this commitment. Since all strings in X_j are sent to $\mathcal{O}_G$, the value of b_j is also known. Define $\mathsf{action}_j = \mathsf{set}$ if $b_j = 0$ and reset otherwise. Observe that no strings in set X_j are likely to appear in any of the sets S_i on left, for all i, with high probability.
2. if $\mathsf{parent}(j) = i$, define action_j as follows.
   (a) IF $r_i = r'_j$ then: (1) if $t'_j = t_i$, define $\mathsf{action}_j = \mathsf{copy}_i$, (2) if $t'_j = t_i \oplus r_i$ let , define $\mathsf{action}_j = \mathsf{toggle}_i$;
   (b) ELSE: (in this case it will be one of the defeat channels as follows:) (1) if $t'_j = t_i$ then $\mathsf{action}_j = \mathsf{defeat}_{0\to 0,1\to\perp}$, (2) if $t'_j \oplus r'_j = t_i \oplus r_i$ then $\mathsf{action}_j = \mathsf{defeat}_{0\to\perp,1\to 1}$, (3) if $t'_j = t_i \oplus r_i$ then $\mathsf{action}_j = \mathsf{defeat}_{0\to\perp,1\to 0}$, (4) if $t'_j \oplus r'_j = t_i$ then $\mathsf{action}_j = \mathsf{defeat}_{0\to 1,1\to\perp}$.

This completes the description of our f. Therefore, with the help of the inversion oracle, attack on the outer commitment has been translated to an attack on the inner codeword w. Further, note

[13] We note that our objective here is not to obtain a string NM commitment from adaptive PRGs; they are already known. We merely want to demonstrate that non-malleable codes can boost partial NM to full NM.

[14] Of course, A is also free to commit to its own values by sending appropriate strings in/out of the range of G; however, such strings will be "easy cases": they will simply translate to *fix types of attacks* on the underlying codeword, as we describe shortly.

[15] $S_i \oplus r_j$ means the set where each element of S_i is XORed with r_j.

that in defining this attack $f \in \mathcal{F}^*$, no string that belong to any of the left sets S_i for all i were queried to the inversion oracle. Therefore, it is possible to learn f with the help of the oracle *without* compromising the hiding of left commitment to w. Therefore, it holds that for every two messages x_0, x_1, if we let w_0, w_1 their respective non-malleable codewords sampled uniformly, c_0, c_1 honestly generated commitments to w_o, w_1 respectively, $c'_b \leftarrow A(c_b)$ the commitment produced by A on input c_b for $b \in \{0,1\}$, and $f_b \in \mathcal{F}^*$ the tampering functions corresponding to c'_b, then by the adaptive security of G it holds that: $f_0 \approx_c f_1$. This is the crux of the argument in the previous section, and now one can proceed in the same manner as in previous section.

Remark. We note that although we do not specifically talk about the defeat channels in our analysis, our code is actually resistant to these classes of attacks as well. Consider any defeat channel, modeled as a deterministic function from $\{0,1\}$ to $\{0,1,\perp\}$. The adversary now applies a tampering which first applies set, reset, toggle, defeat actions to each bit and then applies a permutation from $\mathcal{S}_N$. The reason our code still resists against this enhanced class of attacks is that if the adversary applies a function involving $\perp$ to a large number of bits, then w.h.p., there will be at least one $\perp$ in the final decoding (since, our final code actually has a large independence, if we encode the permutation using such a code). On the other hand, if he applies such functions to only a small number of places, the probability that it will result in a $\perp$ is independent of the message. And conditioned on it not resulting in a $\perp$, the effect of the attack is the same as that of applying an attack which never sends any bit to $\perp$.

References

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

[AGM+14] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes resistant to permutations. In *IACR Eprint*, 2014.

[BS99] Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In *CRYPTO*, pages 519–536, 1999.

[BT07] Anne Broadbent and Alain Tapp. Information-theoretic security without an honest majority. In *ASIACRYPT*, pages 410–426, 2007.

[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.

[CGM+14] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Blockwise non-malleable codes, 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.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO*, pages 565–582, 2003.

[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.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 541–550. IEEE Computer Society, 2010.

[CMTV14] Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes, 2014.

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

[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/.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552. ACM, 1991.

[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.

[GIM+10] S. Dov Gordon, Yuval Ishai, Tal Moran, Rafail Ostrovsky, and Amit Sahai. On complete primitives for fairness. In *TCC*, pages 91–108, 2010.

[GIP+14] Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 495–504, 2014.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *FOCS*, pages 51–60, 2012.

[GLP+12] Vipul Goyal, Huijia Lin, Omkant Pandey, Rafael Pass, and Amit Sahai. Round-efficient concurrently composable secure computation via a robust extraction lemma. Cryptology ePrint Archive, Report 2012/652, 2012. http://eprint.iacr.org/.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 695–704. ACM, 2011.

[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.

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, pages 115–128, 2007.

[KMO10] Eike Kiltz, Payman Mohassel, and Adam O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 673–692. Springer, 2010.

[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.

[LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 705–714. ACM, 2011.

[Mas95] James Lee Massey. Some applications of coding theory in cryptography. In *Codes and Ciphers: Cryptography and Coding IV*, pages 33–47, 1995.

[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.

[MS09] Steven Myers and Abhi Shelat. Bit encryption is complete. In *FOCS*, pages 607–616. IEEE Computer Society, 2009.

[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.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 232–241. ACM, 2004.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 57–74. Springer, 2008.

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 563–572. IEEE Computer Society, 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 533–542. ACM, 2005.

[PSV06] Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *CRYPTO*, pages 271–289, 2006.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *FOCS*, pages 531–540. IEEE Computer Society, 2010.

A Ensuring Independence

Definition 6 (Weighted Bipartite Graph)**.** *Let $G = (V_L, V_R, W)$ be a weighted bipartite graph with partite sets V_L and V_R and a symmetric weight function $W : V_L \times V_R \to \{0\} \cup \mathbb{N}$, such that $W(u, v) = W(v, u)$. The weight on the edge between nodes $u \in V_L$ and $v \in V_R$ is denoted by $w_{u,v}$.*

Let $n_L = |V_L|$ and $n_R = |V_R|$. Without loss of generality we assume that $V_L = [n_L]$ and $V_R = [n_R]$. The graph G is *connected* if there exists a path from every $u \in V_L$ to $v \in V_R$ via edges with positive weights.

The *left degree* of a vertex $u \in V_L$ is denoted by $\deg_L(u) := \sum_{j \in V_R} w_{u,j}$. Similarly, the *right degree* of a vertex $v \in V_R$ is denoted by $\deg_R(v) := \sum_{i \in V_L} w_{i,v}$. An m-regular bipartite graph has $\deg_L(u) = m = \deg_R(v)$, for all $u \in V_L$ and $v \in V_R$. Note that for an m-regular graph, $n_L = n_R$.

An *ordering* of the right partite set is defined by a permutation $\pi : [n_R] \to [n_R]$ on V_R.

Let $G = (V_L, V_R, W)$ be a weighted connected bipartite graph. An edge (u, v) is *k-blue* w.r.t. an ordering π if $w_{u,v} > 0$ and the following conditions are satisfied:

1. $\sum_{j \in V_R : \pi(j) \leqslant \pi(v)} w_{u,j} < \deg_L(u)$
2. $\sum_{j \in V_R : \pi(j) < \pi(v)} w_{u,j} < \deg_L(u) - k$

And edge (u, v) is *k-red* w.r.t. an ordering π if $w_{u,v} > 0$ and it is not k-blue w.r.t. the ordering π. Further, a node $v \in V_R$ is *k-blue* w.r.t. an ordering π if there exists an edge incident on it which is k-blue w.r.t. π.

We emphasize that the classification of an edge as k-blue or k-red edge depends on the ordering π of the nodes in V_R.

Property 1. *For all $u \in V_L$ there exists $v_1 \neq v_2$ such that $w_{u,v_1} > 0$ and $w_{u,v_2} > 0$.*

Observation 1. *Given a weighted bipartite graph G satisfying Property 1 and an ordering π, for any $u \in V_L$ define $\mathsf{first}_\pi(u)$ as the unique $v^* \in V_R$ such that $w_{u,v^*} > 0$ and $\forall v \in V_R$, if $w_{u,v} > 0$ then $\pi(v^*) \leqslant \pi(v)$. Note that if $\deg_L(u) > k$ then the edge (u, v^*) is always k-blue w.r.t. π.*

Claim 1. *Let G be a weighted bipartite graph satisfying Property 1 such that $\deg_L(u) \geqslant 2k$ for all $u \in V_L$. If an edge (u, v) is k-red w.r.t. an ordering $\pi = (\pi_1, \ldots, \pi_{n_R})$ then (u, v) is k-blue w.r.t. $\pi_{\mathsf{rev}} := (\pi_{n_R}, \ldots, \pi_1)$.*

Proof. We have the following two cases for edge (u, v).

1. $\sum_{j \in V_R : \pi(j) \leqslant \pi(v)} w_{u,j} = \deg_L(u)$: Note that $v^*_u = v$ w.r.t. π_{rev}. Since G satisfies Property 1, (u, v) is k-blue w.r.t. π_{rev} by Observation 1.
2. $\sum_{j \in V_R : \pi(j) < \pi(v)} w_{u,j} \geqslant \deg_L(u) - k$: Since $\deg_L(u) \geqslant 2k$, $\sum_{j \in V_R : \pi(j) \geqslant \pi(v)} w_{u,j} \leqslant k$. Thus, $\sum_{j \in V_R : \pi_{\mathsf{rev}}(j) < \pi_{\mathsf{rev}}(v)} w_{u,j} < k < \deg_L(u) - k$. Hence, (u, v) is k-blue w.r.t. π_{rev}. □

Lemma 4. *Let G be a connected weighted bipartite graph satisfying Property 1 such that $\deg_L(u) \geqslant 2k$ for all $u \in V_L$. There exists an ordering π such that at least $n/2$ vertices in V_R are k-blue w.r.t. the ordering π.*

In particular, $\pi \in \{\pi_1 = (1, \ldots, n_R), \pi_2 = (n_R, \ldots, 1)\}$.

Proof. Here we will prove the second statement in the theorem. More precisely, we will show that a vertex $v \in V_R$ is k-blue w.r.t. at least π_1 or π_2. In particular, wlog if $v \in V_R$ is not k-blue w.r.t. π_1, then it is k-blue w.r.t. π_2. Since v is not k-blue in π_1, all the edges incident on v are k-red. Since G is connected, there is at least one edge incident on v. By Claim 1, this edge is k-blue w.r.t. π_2. Hence, v is k-blue w.r.t. π_2.

The lemma follows by an averaging argument. □

B Unpredictability

Given a distribution $\mathcal{D}$ over a set S and a function $f : S \to R$, define the distribution $f(\mathcal{D})$ over set R by the following sampling procedure: Sample $x \sim \mathcal{D}$. Output $f(x)$.

Definition 7 (δ-Balanced)**.** *A distribution $\mathcal{D}$ over a set S is δ-balanced if*

$$\forall s \in S,\ \left(\Pr_{x\sim\mathcal{D}}(x = s) > 0\right) \implies \left(\Pr_{x\sim\mathcal{D}}(x = s) \in [\delta, 1-\delta]\right)$$

Definition 8 (α-Unpredictability)**.** *A distribution $\mathbf{D}$ over sample space $\mathcal{S}$ is said to be α-*unpredictable*, if there exists $s_0, s_1 \in \mathcal{S}$ such that $\Pr_{s\sim\mathbf{D}}[s = s_0]$, $\Pr_{s\sim\mathbf{D}}[s = s_1] \geqslant \alpha$ and $s_0 \neq s_1$.*

Definition 9 (Weight, Density, Dense, Sparse)**.** *For an n-bit binary string $x_{[n]}$, its* weight *(represented as $\mathsf{wt}(x_{[n]})$) is the number of 1s in it. Its* density *is defined to be $\mathsf{wt}(x_{[n]})/n$. It is α-*dense *if its density is at least α; and it is α-*sparse *if its density is at most α.*

Now consider the weighted bipartite graph G as described in the Appendix A. Moreover, let G be a m-regular bipartite graph with $n_L = n = n_R$ and $\deg_L(u) = \deg_R(v) = m$ for all $u \in V_L, v \in V_R$. Next, we label the vertices in V_L by elements in Λ_0. More precisely, let $\mathsf{map} : V_L \to \Lambda_0$ mapping vertices in V_L to Λ_0. The label on $u \in V_L$ is denoted by $\mathsf{map}(u)$. For the rest of the analysis, fix any labeling map for the vertices V_L.

We emphasize that the analysis holds for any arbitrary labeling.

For each $u \in V_L$, $\mathsf{map}(u)$ is encoded using the encoding scheme $\mathbf{X}^{(\mathsf{unary},m,\Lambda_0,3)}$ (see Figure 5). Note that $m = 9|\Lambda_0|$. Also, we will choose the parameter used in red/blue labeling of edges and vertices in Appendix A as $k = m/2$.

Next, $[m]$ is sequentially partitioned into $S_{u,1}, \ldots, S_{u,n}$ such that $[m] = S_{u,1} \cup \ldots \cup S_{u,n}$ and $|S_{u,j}| = w_{u,j}, \forall j \in [n]$. For any vertex $u \in V_L$, sample $\mathbf{x}_{u,[m]} \sim (\mathbf{X}_{[m]}|\mathbf{X}_0 = m(u))$. For $j \in [n_R]$ with $w_{u,j} = 0$, define $\mathbf{B}_{u,j} = 0$. For each $j \in [n_R]$ with $w_{u,j} > 0$, we define $\mathbf{B}_{u,j} = \sum_{i\in S_{u,j}} \mathbf{X}_{u,i}$. In other words, $\mathbf{B}_{u,j}$ is the random variable representing $\mathsf{wt}(\mathbf{X}_{u,S_{u,j}})$.

Now consider an ordering π of V_R such that the number of vertices which are k-blue w.r.t. π are at least $n/2$. Such a ordering π is guaranteed to exist by Lemma 4. Let the set of k-blue vertices w.r.t. π be $\mathsf{S}^{\pi}_{\mathsf{blue}}$ such that $|\mathsf{S}^{\pi}_{\mathsf{blue}}| \geqslant n/2$. We share process these vertices in ascending order as induced by π, i.e. a vertex $v \in \mathsf{S}^{\pi}_{\mathsf{blue}}$ is processed before $v' \in \mathsf{S}^{\pi}_{\mathsf{blue}}$ if and only if $\pi(v) < \pi(v')$.

Consider the next vertex $v \in \mathsf{S}^{\pi}_{\mathsf{blue}}$ w.r.t. π. Then there is an edge, say (u, v), incident on v which is k-blue w.r.t. π. We shall analyze the distribution on vertex v given the fixings of all the edges to vertices $v' \in V_R$ such that $\pi(v') < \pi(v)$. Let $\mathbf{Y}_v = \cup_{i \in V_L} \cup_{j \in V_R : \pi(j) < \pi(v)} \mathbf{X}_{S_{i,j}}$. Also, let all the edges incident on v apart from (u, v) be $\mathbf{G}_v = \cup_{i \neq u} \mathbf{X}_{S_{i,v}}$.

We partition the outgoing edges from vertex u into three sets $F, S, T \subset [m]$ as follows: $F = \cup_{j:\pi(j)<\pi(v)} S_{u,j}$, $S = S_{u,v}$ and $T = \cup_{j:\pi(j)>\pi(v)} S_{u,j}$. Note that by definition of a k-blue edge, $T \neq \emptyset$.

Figure 10 shows the various sets of edges defined above.

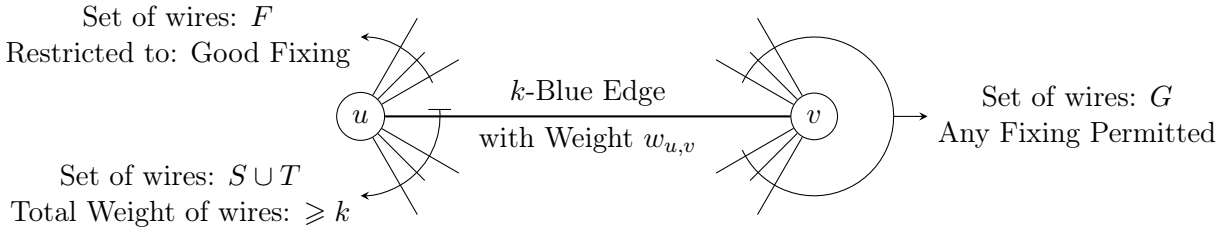

Figure 10: Argument about "Why are Blue Edges Unpredictable?"

For the analysis below, we begin by making the following observation. Though we need to condition the analysis of edge (u, v) on $(\mathbf{Y}_v, \mathbf{G}_v)$, it is sufficient to condition on $\mathbf{X}_F, \mathbf{G}_v$. In particular, we claim the following:

Claim 2. $(\mathbf{B}_{u,v} \bmod 3 | \mathbf{Y}_v = y_v, \mathbf{G}_v = g_v) \equiv (\mathbf{B}_{u,v} \bmod 3 | \mathbf{X}_F = x_F, \mathbf{G}_v = g_v)$, *where* x_F *is restriction of* y_v *to the set* F.

Let $\mathbf{Z}_v$ be the random variable bit string at the node v in this graph. Let $\mathbf{P}_v = \mathsf{wt}(\mathbf{Z}_v) \bmod 3$. In order to show that $\mathbf{Z}_v$ is a valid encoding according to Mod_{Λ_0} conditioned on $\mathbf{Y}_v$ with at most a constant probability, we do the following: We show that $(\mathbf{P}_v | \mathbf{Y})$ is β-unpredictable for some constant $\beta \in (0, 1)$ (Lemma 7). In this direction, we first prove that $(\mathbf{B}_{u,v} \bmod 3 | \mathbf{X}_F, \mathbf{G}_v)$ is α-unpredictable for some constant $\alpha \in (0, 1)$ (Lemma 6).

We prove these lemmas conditioned on the fact that $\mathbf{X}_F$ comes from a good distribution. Hence, we begin by defining a good fixing for $\mathbf{X}_F$. A good fixing intuitively means that even after setting the edges from the vertex u which go to prior vertices, there are a sufficient number of both $0s$ and $1s$ in $\mathbf{X}_{S_{u,v} \cup T}$. More precisely, we define it as follows:

Definition 10. *(c-Good Fixing) Let* x^*_F *be a fixing for the variable* $\mathbf{X}_F$. *We say that* x^*_F *is a c-good fixing if for all* $x \in \mathsf{Supp}(\mathbf{X}_{S \cup T} | \mathbf{X}_F = x^*_F)$, x *is c-dense and* $(1 - c)$*-sparse.*

We emphasize that above definition is independent of the weight of the edge (u, v).

Lemma 5. *Sample* $\mathbf{x}_{u,[m]} \sim (\mathbf{X}_1, \ldots, \mathbf{X}_m | \mathbf{X}_0 = \mathsf{map}(u))$. *Then,* $\forall c \in (0, 1/3)$, $\exists \nu = \mathsf{negl}(\kappa)$ *such that* $\Pr[\mathbf{x}_{u,F}$ *is a c-good fixing*$] \geqslant 1 - \nu$, *where the probability is taken over the randomness of the sampling procedure.*

Proof. This follows from Lemma 11 by noting that $\mathsf{wt}(\mathbf{x}_{u,[m]}) \in [m/3, 2m/3]$ and $|S_{u,v}| + |T| > m/2$. □

Lemma 6 (Unpredictability of $\mathbf{B}_{u,v} \bmod 3$)**.** *Let $c \in (0,1)$ be a constant such that x^*_F be a c-good fixing for $\mathbf{X}_F$. Let $\mathbf{G} = g_v$. Then, there exists a constant $\alpha > 0$ such that $(\mathbf{B}_{u,v} \bmod 3 | \mathbf{X}_F = x^*_F, \mathbf{G}_v = g_v)$ is α-unpredictable.*

Proof. Since x^*_F is a c-good fixing, $\mathbf{X}_{S_{u,v} \cup T}$ is c-dense and $(1-c)$-sparse. By Lemma 14, there exists a constant $\alpha > 0$ such that $(\mathbf{B}_{u,v} \bmod 3 | \mathbf{X}_F = x^*_F, \mathbf{G}_v = g_v)$ is α-unpredictable.

□

Lemma 7 (Unpredictability of $(\mathbf{P}_v | \mathbf{Y}_v = y_v)$)**.** *There exists a constant $\beta \in (0,1)$ and $\nu = \mathsf{negl}(\kappa)$ such that $(\mathbf{P}_v | \mathbf{Y}_v)$ is β-unpredictable with probability $1 - \nu(\kappa)$.*

Proof. Sample $\mathbf{x}_{u,[m]} \sim (\mathbf{X}_1, \ldots, \mathbf{X}_m | \mathbf{X}_0 = \mathsf{map}(u))$. Then by Lemma 5, $\forall c \in (0, 1/3)$, $\exists \nu = \mathsf{negl}(\kappa)$ such that $\Pr[\mathbf{x}_{u,F}$ is a $c-$good fixing$] \geqslant 1 - \nu$. We call this a good event. Now, given such a c-good fixing x^*_F and any fixing g_v of $\mathbf{G}_v$, by Lemma 6, $(\mathbf{B}_{u,v} \bmod 3 | \mathbf{X}_F = x^*_F, \mathbf{G}_v = g_v)$ is α-unpredictable for a constant $\alpha \in (0,1)$. This implies that $(\mathbf{B}_{u,v} \bmod 3 | \mathbf{Y}_v = y_v, \mathbf{G}_v = g_v)$ is α-unpredictable (Claim 2). Hence, $(\mathbf{P}_v | \mathbf{Y}_v = y_v, \mathbf{G}_v = g_v)$ is α-unpredictable.

Since there are $\binom{3}{2}$ pairs of possible parity values, by an averaging argument over g_v, $(\mathbf{P}_v | \mathbf{Y}_v = y_v)$ is β-unpredictable for $\beta = \alpha \binom{3}{2}^{-1}$ conditioned on the good event. □

Lemma 8. $\Pr[\forall j \in V_R,\ \mathbf{P}_j \equiv 0 \bmod 3] \leqslant (1-\gamma)^{n_R/2}$ *for some constant $\gamma \in (0,1)$ with probability $1 - \mathsf{negl}(\kappa)$. Further, this holds even for a graph with multiple connected components.*

Proof.

$$\begin{aligned}
\Pr[\forall j \in V_R,\ \mathbf{P}_j \equiv 0 \bmod 3] &\leqslant \Pr[\forall j \in \mathsf{S}^{\pi}_{\mathsf{blue}},\ \mathbf{P}_j \equiv 0 \bmod 3] \\
&= \prod_{j \in \mathsf{S}^{\pi}_{\mathsf{blue}}} \Pr[\mathbf{P}_j \equiv 0 \bmod 3 | \mathbf{Y}_j = y_j] \\
&\leqslant (1 - \beta + \nu(\kappa))^{|\mathsf{S}^{\pi}_{\mathsf{blue}}|},
\end{aligned}$$

where $\nu = \mathsf{negl}(\kappa)$. The last inequality follows from Lemma 7. The lemma follows by noting that $|\mathsf{S}^{\pi}_{\mathsf{blue}}| \geqslant n_R/2$.

Note that the above analysis was done for a given mapping map. Thus, analysis of each components can be done independently. Hence, this analysis extends naturally to any bipartite graph G. □

B.1 Unpredictability of Dirty Inner Codewords

Lemma 9. *Consider the definition of "dirty codewords" as in Section 4.1. If there are n_{dirty} inner codewords which are dirty, then the probability that all of them are valid inner codewords is $\leqslant \Theta(1)^{n_{\mathsf{dirty}}/2}$.*

Proof. We consider the following exhaustive cases.

Case 1. Suppose a dirty inner codeword receives all its bits from one inner codeword but the permutation is not column preserving. Consider only those columns which are not column preserving; discard the rest. Then we construct a bipartite graph, where left partite set represents the input columns; and the right partite set represents the tampered columns. The XOR of every entry in the input column is the value associated with that input column. Now we can apply Lemma 13.

If there exists a connected component of size $\geqslant 3$, then the parity of the whole inner codeword mod 3 is (constant) unpredictable. So, the inner codeword is invalid with constant probability.

If all connected components are of size 2, and both left nodes have associated values $(0,0)$ or $(1,1)$, then the parity mod 3 of the right vertices is also constant unpredictable. Further, with constant probability (over the randomness of $\mathbf{X}^{(\mathsf{unary})}$) the value associated with these two left columns is $(0,0)$ or $(1,1)$, respectively.

This shows that with constant probability the tampered inner codeword is invalid.

Case 2. Now we consider the permutations which mix multiple inner codewords.

Let us assume that the permutation is column preserving. In this case, we can directly apply Lemma 8 to conclude that the probability of all tampered inner encodings are valid is $\Theta(1)^{n_{\mathsf{dirty}}/2}$.

If the permutation is not column preserving, then argument of Lemma 8 goes through again, because every blue edge continues to be constant unpredictable even if it copies a part of the column (due to high independence of $\mathbf{X}^{(\mathsf{add})}$). □

Lemma 10. *Consider the definition of "dirty codewords" as in Appendix C. If there are n_{dirty} inner codewords which are dirty, then the probability that all of them are valid inner codewords is $\leqslant \xi^{n_{\mathsf{dirty}}/2}$ for some constant $\xi \in (0,1)$.*

Proof. We consider the following exhaustive cases.

Case 1. Suppose a dirty codeword is such that it is a column preserving copy of one inner codeword, but there exists a column which has an odd number of toggle gates. This attack corresponds to flipping certain bits in the inner encoding scheme, i.e., balanced unary encoding scheme. By Lemma 16 the parity of this inner codeword mod 3 is constant unpredictable. Moreover, this inner codeword fails independently of all the inner codewords.

Case 2. Suppose a dirty inner codeword receives all its bits from one inner codeword but the permutation is not column preserving. First, consider only those columns which are not column preserving; discard the rest. Then we construct a bipartite graph, where left partite set represents the input columns; and the right partite set represents the tampered columns. The XOR of every entry in the input column is the value associated with that input column. If that column has odd number of toggle gates, we flip the input value of the left vertex. Now we can apply Lemma 13.

If there exists a connected component of size $\geqslant 3$, then the parity of the whole inner codeword mod 3 is (constant) unpredictable. So, the inner codeword is invalid with constant probability.

If all connected components are of size 2, and both left nodes have associated values $(0,0)$ or $(1,1)$, then the parity mod 3 of the right vertices is also constant unpredictable. Further, with constant probability (over the randomness of $\mathbf{X}^{(\mathsf{unary})}$) the value associated with these two left columns is $(0,0)$ or $(1,1)$, respectively. Finally, if there are some other columns which have been copied in a column preserving manner but toggle gates have been applied to them, still it is easy to see that the parity of the whole inner codeword is constant unpredictable. This shows that with constant probability the tampered inner codeword is invalid. Note that in case also, the dirty codeword fails independent of the other codewords.

Case 3. Now we consider the permutations which mix multiple inner codewords and such connected components. Let us assume that the permutation is column preserving. In this case, we can directly apply Lemma 8 to conclude that the probability of all tampered inner encodings are valid is $\Theta(1)^{n_{\mathsf{dirty}}/2}$. Note that a blue edge remains unpredictable even when some bits are toggled by Lemma 16.

If the permutation is not column preserving, then argument of Lemma 8 goes through, because every blue edge continues to be constant unpredictable even if it copies a part of the column (due to high independence of $\mathbf{X}^{(\mathsf{add})}$). □

C Remaining part of Theorem 2

In this section, we give a formal proof of Theorem 2. First we give a high level overview about how we need to change the proof given in the Section 4.1 for the case when $n_p = 0$ and $\mathcal{F} = \mathcal{S}_N$ to handle $n_p \neq 0$ and $\mathcal{F} = \widetilde{\mathcal{F}} \circ \mathcal{S}_N$.

We begin defining "dirty inner codewords" for this setting. An inner codeword is dirty if:

1. The inner codeword receives its bits partially from one inner codeword.
2. (The codeword receives all its bits from one inner codeword but) The permutations within the inner codeword is not column preserving.
3. (The codeword receives all its bits from one inner codeword and it is column preserving but) There exists a column which has odd number of toggle gates.

C.1 First Generalization

First we shall generalize the proof in Section 4.1 to $\widetilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ but still have $n_p = 0$.

Now, we show that the analysis of Section 4 needs to change only slightly. Moreover, the analysis does not need to change for the case when the number of dirty codewords is small. Below, we highlight the changes for the case when dirty is large. We need to show that each dirty inner

codeword fails with a constant probability. Moreover, given that the number of dirty codewords is "large", we need that large number of these fail independently, so that the total failure probability is $1-\mathsf{negl}(\kappa)$. Below, we describe why each of the dirty codewords described satisfies these conditions.

1. The analysis of Lemma 8 continues to hold identically because an unpredictable (blue) edge when toggled remains unpredictable. So, the first case of definition of dirty codewords can be taken care of identically as in Section 4.1.
2. The analysis of Lemma 13 continues to hold as well because toggling a wire is equivalent to toggling the value at its left vertex. Unpredictability still holds and we can show that the parity $\mod 3$ is unpredictable. So, the second case of dirty codeword definition mentioned above is taken care of.
3. For the third case of how inner codewords become dirty, we need a new lemma. The new lemma (see Lemma 16)[16] says that, even if one column has odd number of toggle gates, the inner codeword becomes invalid with constant probability. Thus, our analysis of Section 4.1 continues to hold in this setting.

We use these cases to prove Lemma 10 which we use in our full proof below.

C.2 Second Generalization

Now, we generalize the proof for $n_p \leqslant \log^9 \kappa$. In Case 1 of the proof, where $n_{\mathsf{dirty}} \geqslant \log^{10} \kappa$, the tampering function can reduce the number of dirty inner codewords by at most n_p. But $n_{\mathsf{dirty}} - n_p = \Theta(\log^{10} \kappa)$, so that part of the proof still goes through.

Now, for Case 2 of the proof, we have $n_{\mathsf{dirty}} < \log^{10} \kappa$. We can again interpret c_{I_p} as additional small number of set/resets. Since, the privacy and distance of the outer codewords are $\gg \log^{13} \kappa$ an additional number of $\log^9 \kappa$ set/resets do not create any bottlenecks for Case 2.1. or Case 2.2.

Now we give a formal proof of Theorem 2.

C.3 Formal Proof of Theorem 2

Refer to the above definition of dirty codewords. We now define the copied and fixed codewords for this scenario.

Fixed Inner Codewords. We say that an inner codeword is *completely fixed* if all its bits are obtained from bits in $I_0 \cup I_1$. That is, the whole codeword is explicitly written down using bits from $I_0 \cup I_1$. Note that some of these bits might have been toggled. The number of such inner codewords is represented by n_{fixed}.

[16] This result holds not only for toggle-channels but a more general class of channels called non-constant channels.

Copied Codewords. We say that an inner codeword is completely copied if it is generated by copying one whole inner codeword and (possibly) performing column preserving permutations. Also, even number of toggles might have been applied to any of these columns. The number of such inner codewords is represented by n_{copy}.

Note that, as before, $n = n_{\mathsf{dirty}} + n_{\mathsf{fixed}} + n_{\mathsf{copy}}$.

For this proof we also define peeked codewords as follows:

Peeked Inner Codewords. We say that an inner codeword is a peeked codeword if one of its bits has been copied from I_p. Let n_{peek} be the number of such codewords. Note that $n_{\mathsf{peek}} \leqslant n_p$.

C.3.1 Key Steps of Proof

Now we begin with our case analysis. We shall explain how $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ will be determined for various cases. We refer to the term $\mathrm{SD}\left(\mathsf{Tamper}^{(s)}_{I_0,I_1,I_p,c_{I_p},f}, \mathsf{Sim}^{(s)}_{\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}}\right)$ in Figure 7 as the simulation error. The expectation of simulation error over random choices of I_0, I_1, I_p and c_{I_p} is referred to as expected simulation error.

The threshold value $\log^{10}\kappa$ chosen below for analysis is arbitrary; any suitable $\mathrm{poly}\log\kappa$ will suffice.

Case 1. $n_{\mathsf{dirty}} \geqslant \log^{10}\kappa$. In this case, we have $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ output $\perp$ with probability 1.

The simulation error in this case is $\mathsf{negl}(\kappa)$ as follows: Consider the set of dirty codewords which do not contain any bit from I_p. Let n'_{dirty} be the number of such codewords. Then, $n'_{\mathsf{dirty}} \geqslant n_{\mathsf{dirty}} - n_{\mathsf{peek}} \geqslant \log^{10}/2$. Now, the probability that the tampered codeword has all valid inner encodings is $\leqslant \xi^{n'_{\mathsf{dirty}}/2} = \mathsf{negl}(\kappa)$, where $\xi \in (0,1)$ is a constant (by Lemma 10).

Case 2. $n_{\mathsf{dirty}} < \log^{10}\kappa$. We shall show that it is highly unlikely (over random choices of I_0 and I_1) that $n_0 + n_1 \geqslant \log^{13}\kappa$ and $n_0 + n_1 \leqslant N - um\log^{10}\kappa$ and still we get this case. In particular, it is $\mathsf{negl}(\kappa)$ (see Lemma 17).

So, henceforth, we can assume that either $n_0 + n_1 < \log^{13}\kappa$ or $n_0 + n_1 > N - um\log^{10}\kappa$; at an expense of $\mathsf{negl}(\kappa)$ additive term in expected simulation error.

Case 2.1. $n_0 + n_1 \leqslant \log^{13}\kappa$. In this case, the tampering function copies most of the inner codewords into the tampered codeword, because $n_{\mathsf{copy}} = n - n_{\mathsf{dirty}} - n_{\mathsf{fixed}}$. Define n'_{copy} as the number of codewords copied but not peeked. Then, $n'_{\mathsf{copy}} \geqslant n_{\mathsf{copy}} - n_{\mathsf{peek}} \geqslant n - n_{\mathsf{dirty}} - n_{\mathsf{fixed}} - n_{\mathsf{peek}} \geqslant n - \log^{10}\kappa - \log^{13}\kappa/um - \log^{9}\kappa \geqslant n - 2\log^{13}\kappa$. So, the tampered codeword can either be invalid or (if valid) equivalent to the original codeword (because distance of the outer codeword $\gg 2\log^{13}\kappa$). Now the probability that the completely fixed and dirty codewords are each identical to their counterparts in the input codeword does not depend the message s because the privacy of the outer

codeword is $\gg 2\log^{13}\kappa$. This probability σ can be computed exhaustively and does not depend on the message s. So, $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ outputs same^* with probability σ; otherwise outputs $\perp$.

Case 2.2. $n_0 + n_1 > N - um\log^{10}\kappa$. Since the $n_0 + n_1$ is large and n_{dirty} is small, this implies that most inner-codewords have been completely fixed. Define n'_{fixed} as number of codewords which are fixed but not peeked. Then, $n'_{\mathsf{fixed}} \geqslant n - n_{\mathsf{dirty}} - n_{\mathsf{copy}} - n_{\mathsf{peek}} > n - \log^{10}\kappa - n_{\mathsf{copy}} - \log^9\kappa > n - 3\log^{10}\kappa$. In this case, the tampered code word is either invalid or (if valid) equivalent to the codeword consistent with the fixed inner codewords (due to high distance of the outer encoding scheme).

First check whether the fixed blocks can define a valid outer codeword. If not, then set $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ to output $\perp$ with probability 1. Again, simulation error in this case is 0.

Otherwise, we are in the case when the set of completely fixed inner codewords fixes the outer codeword. Let its outer codeword be g^* and the message corresponding to it be s^*. We say that (I_0, I_1) is *good* if it contains at least one bit from each column. Since we have $n_0 + n_1 > N - um\log^{10}\kappa$, we have $\Pr[(I_0, I_1) \text{ is good}] = 1 - \mathsf{negl}(\kappa)$ (by Lemma 18).

If (I_0, I_1) is not good, then we define $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ to output $\perp$ with probability 1. The simulation error for particular (I_0, I_1) can be at most 1; but this incurs additional $\mathsf{negl}(\kappa)$ expected simulation error over the choices of (I_0, I_1).

If (I_0, I_1) is good, then we need to check whether the set of remaining inner codewords of the tampered codeword are equivalent to the set of inner codewords of g^*. Note that if (I_0, I_1) is good then all bits of the original codeword restricted to $[N]\setminus(I_0 \cup I_1)$ are independent uniform random bits (because all proper subsets of columns have independent uniform bits). This can be exhaustively computed starting from uniformly random bits. Define this probability to be σ. Clearly, this probability is independent of the message s. We define $\mathcal{D}_{I_0,I_1,I_p,c_{I_p},f}$ to output s^* with probability σ; and output $\perp$ with probability $1 - \sigma$. The simulation error in this case is 0.

There exists $\mathbb{D}$ such that the expected simulation error for our exceeding scheme is $\mathsf{negl}(\kappa)$, for all n_0, n_1 and for all $n_p < \log^9\kappa$ and mapping function map, when $\mathcal{F} = \widetilde{\mathcal{F}} \circ \mathcal{S}_N$.

D Mathematical Tools

In this section we prove some useful mathematical tools relevant for our results.

Lemma 11 (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 $\mathbf{X}_{[n]} = \mathbf{U}\binom{[n]}{m}$. For every $t \in \mathbb{N}$, we have:*

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

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

$$\Pr_{x_{[n]} \sim \mathbf{X}_{[n]}} \left(\sum_{i \in [t]} x_i \in [(c-\varepsilon)t, (1-c-\varepsilon)t] \right) \leqslant 2 \exp(-\varepsilon^2 t/3)$$

Lemma 12 (Coupon Collector Problem: Concentration)**.** *Suppose there are n coupons. Let $X_{n,t}$ be the random variable representing the number of tries needed to obtain t unique coupons (where coupons are picked uniformly at random with replacement). Then, we have the following concentration bound:*

$$\Pr[X_{n,t} \geqslant b] \leqslant \frac{e^t}{(n/t)^{b-t}}$$

Proof. Let G_p be a geometric random variable, i.e. outputs i with probability $q^{i-1}p$, where $q = 1-p$ and $p \in (0,1)$.

Let X_i be the random variable representing the number of samples needed to see the i-th unique coupon after the $(i-1)$-th coupon has already been sampled. Note that X_i is identical to G_{p_i}, where $p_i = \left(1 - \frac{i-1}{n}\right)$. We define $q_i = \frac{i-1}{n}$.

Now, we have $X_{n,t} = \sum_{i=1}^{t} X_i$. So, we have:

$$\Pr\left[X_{n,t} \geqslant b\right] = \Pr\left[\exp\left(\lambda X_{n,t}\right) \geqslant \exp(\lambda b)\right], \text{ where } \lambda > 0$$

Note that, when $q \exp(\lambda) < 1$, we have:

$$\begin{aligned}
\mathbb{E}\left[\exp(\lambda G_p)\right] &= \sum_{i \geqslant 1} \exp(\lambda i) q^{i-1} p \\
&= \frac{p \exp(\lambda)}{1 - q \exp(\lambda)}
\end{aligned}$$

Therefore, we have:

$$\begin{aligned}
\mathbb{E}\left[\exp\left(\lambda X_{n,t}\right)\right] &= \prod_{i=1}^{t} \mathbb{E}\left[\exp\left(\lambda X_i\right)\right] \\
&= \prod_{i=1}^{t} \frac{p_i \exp(\lambda)}{1 - q_i \exp(\lambda)}
\end{aligned}$$

Note that, we need $\exp(\lambda) < 1/q_i$, for all $i \in [t]$. This implies that $\exp(\lambda) < n/(t-1)$. We use $\lambda = \lambda^*$ such that $\exp(\lambda^*) = n/t$.

Now, for $\lambda = \lambda^*$, we have:

$$\begin{aligned}
\mathbb{E}\left[\exp\left(\lambda X_{n,t}\right)\right] &= \prod_{i=1}^{t} \left(\frac{np_i}{t - nq_i}\right) \\
&= \frac{n(n-1)\ldots(n-t+1)}{t(t-1)\ldots 1} = \binom{n}{t} \\
\exp(\lambda b) &= \left(\frac{n}{t}\right)^b
\end{aligned}$$

By Markov inequality, we have:

$$\begin{aligned}\Pr\left[\exp\left(\lambda X_{n,t}\right) \geqslant \exp(\lambda b)\right] &\leqslant \frac{\mathbb{E}\left[\exp\left(\lambda X_{n,t}\right)\right]}{\exp(\lambda b)} \\ &\leqslant \binom{n}{t} \Big/ \left(\frac{n}{t}\right)^b \\ &\leqslant \left(\frac{en}{t}\right)^t \Big/ \left(\frac{n}{t}\right)^b = e^t (n/t)^{-(b-t)}\end{aligned}$$

□

Lemma 13 (Re-randomization Lemma)**.** *Let $G = ((L, R), E)$ be an undirected bipartite graph with partite sets L and R. Corresponding to every node $u \in L$, we have $a_u \in \{0, 1\}$. Corresponding to every node $u \in L$ and $v \in N(u)$, we have random variables $Z_{u,v}$ such that they are independent uniform variables on $\{0, 1\}$ and the only constraint on them is:*

$$\sum_{v \in N(u)} Z_{u,v} = a_u$$

Define $Y_v = \sum_{u \in N(v)} Z_{u,v}$. Consider a connected component in G with node set $\{u_1, \ldots, u_s\}$ in the left partite set and $\{v_1, \ldots, v_t\}$ in the right partite set. Then, $Y_{v_1}, \ldots, Y_{v_t}$ are independent uniform random variables, except $\sum_{i \in [t]} Y_{v_i} = \sum_{i \in [s]} a_{u_i}$.

Proof. We shall prove the result by induction on the number of nodes in the left partite set. Suppose $|L| = 1$, then this result is trivially true.

Suppose this is true for all bipartite graphs with $|L| < n$.

Now suppose $|L| = n$. Remove one of the vertices and apply the induction hypothesis on the remaining graph G'. Suppose the removed vertex is u^*.

For the inductive step, we perform a second induction on $d_L(u^*)$. If $d_L(u^*) = 1$ then we have $Z_{v,u^*} = a_{u^*}$, where v be the unique vertex in $N(u^*)$. There are two cases to consider: a) v lies in a connected component of G', or b) v is a connected only to u. In both these cases, it is easy to see that the induction hypothesis is true.

Suppose $|N(u^*)| = 2$. In this case, we have following cases: a) both vertices lie in one component of G', b) both vertices lie in different components G', c) both vertices do not lie in any component of G', or d) one of the vertices lies in a component of G' while another does not lie in any component of G'. The only interesting case is (b); for all other cases it is easy to see that the induction hypothesis holds trivially. For case (b), Suppose that v_1 and v_2 are neighbors of u^*. Suppose v_1 lies in the first component of G' and v_2 lies in the second component. Note that Z_{u^*,v_1} is uniformly random, so all Y_v in the first component become uniformly random. Next, note that $Z_{u^*,v_2} = a_{u^*} + Z_{u^*,v_1}$. So, Y_{v_2} in G' gets added with Z_{u^*,v_2}. The updated Y_vs are all uniformly random except that: $\sum_{v \in R'} Y_v = \sum_{u \in L'} a_u$, where L' and R' are the left- and right- partite sets of the union of first and second connected components.

Suppose $|N(u^*)| > 2$, then it follows from the fact that the first $|N(u^*)| - 1$ random variables are uniformly and independently random. So, this case is directly similar to the $|N(u^*)| = 2$ case. □

Lemma 14 (Unpredictability Lemma)**.** *Let $c \in (0,1)$ be a constant, $m, n \in \mathbb{N}$ and $m \in [cn, (1-c)n]$. Define $\mathbf{X}_{[n]} = \mathbf{U}_{\binom{[n]}{m}}$. Let $p \in \mathbb{N}$ be a constant. Given any $t \in [n-1]$, let $\mathsf{parity}_{n,m,t,p}$ be the random variable: $\sum_{i\in[t]} X_i \mod p$. Then, there exists a constant $\mu \in (0,c)$ such that $\mathsf{parity}_{n,m,t,p}$ is μ-unpredictable.*

Proof. We consider two cases.

Case $(n-t+1) \geqslant n/2$. A random sample of $\mathbf{X}_{[n]}$, satisfies the following condition with $1-\nu(n)$ probability (where, $\nu(n) = \mathsf{negl}(n)$): The random variable $\mathbf{X}_{[n]\setminus[t-1]}$ is $(c-\varepsilon)$-dense and $(1-c+\varepsilon)$-sparse (by Lemma 11), for any constant $\varepsilon \in (0,1)$. Lets call this a good event. This implies that the random variable $\mathbf{X}_t$ is $(c-\varepsilon)$-balanced. Therefore, conditioned on a good event, $\mathsf{parity}_{n,m,t,p}$ is $(c-\varepsilon)$-unpredictable.

Since there are $\binom{p}{2}$ pairs of parity values, by an averaging argument, $\mathsf{parity}_{n,m,t,p}$ is $(c-\varepsilon)(1-\nu(n))\binom{p}{2}^{-1}$-unpredictable. Any constant $\mu < (c-\varepsilon)\binom{p}{2}^{-1}$ suffices.

Case $(n-t+1) < n/2$. This implies that $t-1 > n/2$. With $1-\nu(n)$ probability, where $\nu(n) = \mathsf{negl}(n)$, $\mathbf{X}_{[t+1]}$ is $(c-\varepsilon)$-dense and $(1-c+\varepsilon)$-sparse (by Lemma 11), for any constant $\varepsilon \in (0,1)$. Lets call this a good event. This implies that the random variable $\mathbf{X}_{t+1}$ is $(c-\varepsilon)$-balanced. Therefore, conditioned on a good event, parity of last $(n-t)$ bits is $(c-\varepsilon)$-unpredictable. Which implies that conditioned on a good event, $\mathsf{parity}_{n,m,t,p}$ is $(c-\varepsilon)$-unpredictable.

Consequently, $\mathsf{parity}_{n,m,t,p}$ is $(c-\varepsilon)(1-\nu(n))\binom{p}{2}^{-1}$-unpredictable. Any constant $\mu < (c-\varepsilon)\binom{p}{2}^{-1}$ suffices. □

Lemma 15 (Unpredictability of Dirty Matchings)**.** *Let $c \in (0,1)$ be a constant, $m, n \in \mathbb{N}$ and $m \in [cn, (1-c)n]$. Define $\mathbf{X}_{[n]} = \mathbf{U}_{\binom{[n]}{m}}$. Consider non-constant channels $f_{i_1}, \ldots, f_{i_z}$, $\mathcal{I} = \{i_1, \ldots, i_z\} \subseteq [n]$. Define $\mathbf{Y}$ as follows: For all $i \in [n]$, define $Y_i = f_i(X_i)$ if $i \in \mathcal{I}$, else $Y_i = X_i$. Consider the random variable $\mathbf{P}_Y = \sum_{i\in[n]} Y_i \mod 3$. Then there exists a constant $\mu \in (0,1)$ such that $\mathbf{P}_Y$ is μ-unpredictable.*

Proof. A channel f is "confusing" if there exists $b \in \{0,1\}$ such that $\mathsf{Supp}(f(b)) = \{0,1\}$.

Case 1. There exists a confusing channel in $\{f_{i_1}, \ldots, f_{i_z}\}$. Without loss of generality assume that f_1 is confusing and $\mathsf{Supp}(f(0)) = \{0,1\}$. We have $\Pr[X_1 = 0] \in [c, 1-c]$. Fix a setting of $\mathbf{X}$ conditioned on $X_1 = 0$. Fix the internal randomness of all remaining channels. This fixes $Y_2 + \cdots + Y_n \mod 3$. Now, we have $\Pr(Y_1 = 0)$ and $\Pr(Y_1 = 1)$ are at least a constant conditioned on these fixings. So, overall we can set $\mu = \Pr(X_1 = 0) \cdot \min\{\Pr[Y_1 = 0 | X_1 = 0], \Pr[Y_1 = 1 | X_1 = 0]\}$.

Case 2. There are no confusing channels. This implies that all channels are "toggle" channels. Without loss of generality assume that $\{i_1, \ldots, i_z\} = [z]$. If f_i is a toggle then note that $Y_i = 1 - X_i$ over $\mathbb{Z}_3$. So,

$$\begin{aligned} Y_1 + \cdots + Y_n \mod 3 &= z - (X_1 + \cdots + X_z) + (X_{z+1} + \cdots + X_n) \mod 3 \\ &= z + m - 2(X_1 + \cdots + X_z) \mod 3 \\ &= z + m + (X_1 + \cdots + X_z) \mod 3 \end{aligned}$$

Note that $X_1 + \cdots + X_z$ is μ unpredictable, for some constant μ Lemma 14. Hence, $Y_1 + \cdots + Y_n \mod 3$ is μ unpredictable. □

Lemma 16 (Unpredictability w.r.t. Channels)**.** *Let $c \in (0,1)$ be a constant, $m, n \in \mathbb{N}$ and $m \in [cn, (1-c)n]$. Define $\mathbf{X}_{[n]} = \mathbf{U}_{\binom{[n]}{m}}$. For any $t \in [n-1]$, let $f_1, \ldots, f_t$ be non-constant channels; and define $Y_i = f_i(X_i)$, for all $i \in [t]$. Let $\mathsf{parity}_{n,m,t,3}$ be the random variable: $\sum_{i \in [t]} Y_i \mod 3$. Then, there exists a constant $\mu \in (0, c)$ such that $\mathsf{parity}_{n,m,t,3}$ is μ-unpredictable.*

Further, for $t = n$ and $(n-m) \neq 0 \mod 3$, if there exists f_i such that it is not the identity mapping (i.e. it is not $f(b) = b$), then $\Pr[\mathsf{parity}_{n,m,t,3} \neq 0]$ is at least a constant.

Proof. When $t \in [n-1]$, the proof follows by combining the proofs of Lemma 14 and Lemma 15. We just show the proof for the case $(n - t + 1) \geqslant n/2$. The final case $(n - t + 1) < n/2$ follows analogously.

A channel f is "confusing" if there exists $b \in \{0,1\}$ such that $\mathsf{Supp}(f(b)) = \{0,1\}$.

Case 1. Suppose there exists a confusing channel in $\{f_1, \ldots, f_t\}$. Without loss of generality assume that f_t is confusing and $\mathsf{Supp}(f(0)) = \{0,1\}$.

We know that with probability $1 - \mathsf{negl}(n)$, the random variable $\mathbf{X}_{[n]\setminus[t-1]}$ is $(c-\varepsilon)$-dense and $(1-c+\varepsilon)$-sparse (by Lemma 11), for any constant $\varepsilon \in (0,1)$. Let us call this a good event. This implies that the random variable $\mathbf{X}_t$ is $(c-\varepsilon)$ balanced. Therefore, conditioned on a good event $\Pr[\mathbf{X}_t = 0] \geqslant (c - \varepsilon)$.

Conditioned on $\mathbf{X}_t = 0$ and the good event, choose a fixing of $\mathbf{X}_{[n]}$ and also fix the internal randomness of all channels $\{f_1, \ldots, f_{t-1}\}$. Now, it is clear that $\mathsf{parity}_{n,m,t,3}$ is at least $(c-\varepsilon) \cdot \min\{\Pr[Y_t = 0 | X_t = 0], \Pr[Y_t = 1 | X_t = 0]\} - \mathsf{negl}(n)$.

Case 2. Suppose there are no confusing channels; that is all channels are toggle channels. Again we condition on the good event mentioned above and mimic the proof of Lemma 15 for the corresponding case.

Let us consider the case of $t = n$. Suppose there exists a confusing channel. Without loss of generality assume that f_1 is a confusing channel with $\mathsf{Supp}(f(0)) = \{0,1\}$. Then we know that $\Pr[X_1 = 0] \in [c, 1-c]$. By fixing a choice of $X_{2,\ldots,n}$ and internal randomness of $f_2, \ldots, f_t$ we get that $\mathsf{parity}_{n,m,t,3}$ is constant unpredictable.

Suppose all channels are toggles. In this case, we use the fact that $Y_i = 1 - X_i$, for each $i \in [t]$.

Now we have:

$$\sum_{i\in[n]} Y_i \mod 3 = \sum_{i\in[t]} 1 - X_i \mod 3$$
$$= (n-m) \mod 3$$

□

D.1 Some useful Results

Lemma 17. *If $n_0+n_1 \geqslant \log^{13}\kappa$ and $n_0+n_1 \leqslant N - um\log^{10}\kappa$, then $n_{\mathsf{dirty}} \geqslant \log^{10}\kappa$, with probability $1-\mathsf{negl}(\kappa)$.*

Proof. Let $\widehat{I}$ be the number of inner codewords which have at least one index reset or set. We consider the following cases:

1. If $n_0+n_1 \geqslant \log^{13}\kappa$ and $n_0+n_1 < n\log^2\kappa$. Then $\widehat{I} \geqslant \Theta((n_0+n_1)/\log^2\kappa)$ with probability $1-\mathsf{negl}(\kappa)$.[17] Next, observe that $n_{\mathsf{dirty}} = \widehat{I} - n_{\mathsf{fixed}} \geqslant \Theta\left(\frac{n_0+n_1}{\log^2\kappa}\right) = \omega(\log^{10}\kappa)$.

2. If $n_0+n_1 \geqslant n\log^2\kappa$ but $n_0+n_1 \leqslant N - um\log^{10}\kappa$: In this case, we have $\widehat{I} = n$, with probability $1-\mathsf{negl}(\kappa)$. Next, observe that $n_{\mathsf{dirty}} = \widehat{I} - n_{\mathsf{fixed}} \geqslant n - (n - \log^{10}\kappa) = \log^{10}\kappa$, with probability $1-\mathsf{negl}(\kappa)$. □

Lemma 18 (Load Balancing)**.** *If $n_0+n_1 \geqslant N - um\log^{10}\kappa$ then the probability that there exists a column none of whose indices are in $I_0 \cup I_1$ is $1-\mathsf{negl}(\kappa)$, if a) $u=\omega(1)$, and b) $n/\log^{10}\kappa = \kappa^{\Theta(1)}$.*

Proof. We have $|I_0 \cup I_1| \geqslant N - um\log^{10}\kappa$. Consider the number of indices which are *not* set/rest (call them *holes*): $H \leqslant um\log^{10}\kappa$. There are mn columns. Expected number of holes per column: $\leqslant u\frac{\log^{10}\kappa}{n}$. We want to show that the probability of a column receiving u holes is very low. So, we consider the general load balancing problem below. The probability of receiving u holes in on column is less than the probability of receiving at least u balls in one bin, when H balls are thrown into mn bins.

Consider the problem of throwing α balls into η bins. The probability that the first bin contains at least τ balls is at most $\binom{\alpha}{\tau}\eta^{-\tau}$. So, the probability p that there exists a bin with at least τ balls is at most $\eta\binom{\alpha}{\tau}\eta^{-\tau}$, by union bound. Let $\mu = \alpha/\eta$ be the average number of balls per bin, then $p \leqslant n(e\mu/\tau)^{\tau}$.

In our case, we have $(e\mu/\tau) = (1/\eta)^{\Theta(1)}$. Thus, it suffices for $\tau = \omega(1)$ to make $p = \mathsf{negl}(\kappa)$.

Consequently, the probability that one of our columns has u holes when the expected number of holes is $u\log^{10}\kappa/n$ is $\mathsf{negl}(\kappa)$, if a) $u=\omega(1)$, and b) $n/\log^{10}\kappa = \kappa^{\Theta(1)}$. □

[17] See Lemma 12. If the number of samples $b \geqslant t + \frac{t+\omega(\log\kappa)}{\ln(n/t)}$, then we get t unique coupons with $1-\mathsf{negl}(\kappa)$ probability.