Non-Malleable Codes from Average-Case Hardness: AC^0, Decision Trees, and Streaming Space-Bounded Tampering

Marshall Ball[1], Dana Dachman-Soled[2], Mukul Kulkarni[2], and Tal Malkin[1]

[1] Columbia University
{marshall,tal}@cs.columbia.edu
[2] University of Maryland
danadach@ece.umd.edu, mukul@umd.edu

Abstract. We show a general framework for constructing non-malleable codes against tampering families with average-case hardness bounds. Our framework adapts ideas from the Naor-Yung double encryption paradigm such that to protect against tampering in a class $\mathcal{F}$, it suffices to have average-case hard distributions for the class, and underlying primitives (encryption and non-interactive, simulatable proof systems) satisfying certain properties with respect to the class.

We instantiate our scheme in a variety of contexts, yielding efficient, non-malleable codes (NMC) against the following tampering classes:

- Computational NMC against AC^0 tampering, in the CRS model, assuming a PKE scheme with decryption in AC^0 and NIZK.
- Computational NMC against bounded-depth decision trees (of depth n^ϵ, where n is the number of input variables and constant $0 < \epsilon < 1$), in the CRS model and under the same computational assumptions as above.
- Information theoretic NMC (with no CRS) against a streaming, space-bounded adversary, namely an adversary modeled as a read-once branching program with bounded width.

Ours are the first constructions that achieve each of the above in an efficient way, under the standard notion of non-malleability.

1 Introduction

Non-malleable codes, introduced in the seminal work of Dziembowski, Pietrzak and Wichs [32], are an extension of error-correcting codes. Whereas error-correcting codes provide the guarantee that (if not too many errors occur) the receiver can recover the original message from a corrupted codeword, non-malleable codes are essentially concerned with security. In other words, correct decoding of corrupted codewords is not guaranteed (nor required), but it is instead guaranteed that adversarial corruptions cannot influence the output of the decoding in a way that depends on the original message: the decoding is either correct or *independent* of the original message.

The main application of non-malleable codes is in the setting of tamper-resilient computation (although non-malleable codes have also found connections in other areas of cryptography [24, 23, 37] and theoretical computer science [19]). Indeed, as suggested in the initial work of Dziembowski et al. [32], non-malleable codes can be used to encode a secret state in the memory of a device such that a tampering adversary interacting with the device does not learn anything more than the input-output behavior. Unfortunately, it is impossible to construct non-malleable codes secure against arbitrary tampering, since the adversary can always apply the tampering function that decodes the entire codeword to recover the message m and then re-encodes a related message m'. Thus, non-malleable codes are typically constructed against limited classes of tampering functions $\mathcal{F}$. Indeed, given this perspective, error correcting codes can be viewed as a special case of non-malleable codes, where the class of tampering functions, $\mathcal{F}$, consists of functions which can only modify some fraction of the input symbols. Since non-malleable codes have a weaker guarantee than error correcting codes, there is potential to achieve non-malleable codes against much broader classes of tampering functions $\mathcal{F}$ (including tampering that modifies every bit).

Exploring rich classes of tampering functions. Several works construct non-malleable codes (NMC) against general tampering classes of bounded size, but with non-explicit, existential, or inefficient constructions (cf. [32, 20, 36]). For efficient and explicit constructions, a large body of works construct NMC against bit-wise tampering (cf. [32, 22, 11]), and more generally split-state tampering (cf. [48, 31, 3, 20, 21, 2, 1, 16, 45, 40, 41]), where the adversary can tamper each part of the codeword independently of other parts, as well as NMC against permutations, flipping, and setting bits [5].

A recent line of works is shifting towards considering the construction of NMC against tampering classes $\mathcal{F}$ that correspond to well-studied complexity-theoretic classes, and may also better correspond to tampering attacks in practice. Specifically, Ball et al. [8] construct NMC against local tampering functions including NC^0, and Chattopadhyay and Li [17] construct NMC against AC^0 tampering, but inefficiently (with super-poly size codewords). Additionally, NMC with weaker notions of security are constructed by Faust et al. [33] against space-bounded tampering (in the random-oracle model), and by Chandran et al. [13] for block-wise tampering (where the adversary receives the message in a streaming fashion, block-by-block). We discuss these works in Section 1.3.

In this work, we continue this line of research and consider constructing non-malleable codes against various complexity classes, including: (1) AC^0 tampering, where the tampering function is represented by a polynomial size constant-depth, unbounded fan-in/fan-out circuit, (2) tampering with bounded-depth decision trees, where the tampering function is represented by a decision tree with n variables and depth n^ϵ for $\epsilon < 1$, (3) streaming tampering with quadratic space, where the tampering function is represented by a read-once, bounded-width ($2^{o(n^2)}$) branching program, (4) small threshold circuits: depth d circuits of majority gates with a quasilinear number of wires, (5) fixed polynomial time tampering: randomized turing machines running in time $O(n^k)$ for any fixed

k. Constructing non-malleable codes against a wide array of complexity classes is desirable since in practice, the capabilities of a tampering adversary are uniquely tied to the computational setting under consideration and/or the physical device being used. For example, our motivation for studying AC^0 stems from a setting wherein an attacker has limited *time* to tamper, since the tampering function must complete before race conditions take effect (e.g. before the end of a clock-cycle in a synchronous circuit). AC^0 circuits, which are constant-depth circuits, model such attackers since the propagation delay of a circuit is proportional to the length of the longest path from input to output.

1.1 Our Results

We present general frameworks for constructing non-malleable codes for encoding one and multi-bits against various tampering classes $\mathcal{F}$ for which average case hardness results are known. Our frameworks (one for single-bit and one for multi-bit) include both a generic construction, which requires that certain underlying primitives are instantiated in a suitable way, as well as a proof "template." Our frameworks are inspired by the well-known double-encryption paradigm for constructing CCA2-secure public key encryption schemes [49, 51, 46]. And although we rely on techniques that are typically used in the cryptographic setting, we instantiate our framework for particular tampering classes $\mathcal{F}$ in both the computational setting and in the information theoretic one. For the computational setting, our results rely on computational assumptions, and require a common-reference string (CRS), which the adversary can see before selecting the tampering function (as typical in other NMC works using CRS or random oracles). For the information theoretic setting, our results do not require CRS nor any computational assumption (as the primitives in our framework can be instantiated information theoretically). Our general theorem statements provide sufficient conditions for achieving NMC against a class $\mathcal{F}$. Somewhat informally, the main such condition, especially for the one-bit framework, is that there are sufficiently strong average-case hardness results known for the class $\mathcal{F}$. In particular, we obtain the following results, where all the constructions are efficient and, for the multi-bit NMC, the achieved rate is $1/\operatorname{poly}(m)$ where m is the length of the message being encoded.

- **Constructions for AC^0 tampering:** We obtain computational NMC in the CRS model against AC^0 tampering. Our constructions require public key encryption schemes with decryption in AC^0, which can be constructed e.g. from exponential hardness of learning parity with noise [10], as well as non-interactive zero knowledge (NIZK), which can be constructed in the CRS model from enhanced trapdoor permutations.
  Previous results by Chattopadhyay and Li [17] achieve NMC for AC^0 with information theoretic security (with no CRS), but are inefficient, with super-polynomial rate.
- **Constructions for bounded-depth decision trees:** We obtain computational NMC in the CRS model against tampering with bounded-depth decision trees. Our construction requires the same computational

assumptions as the AC^0 construction above. The depth of the decision tree we can handle is n^ϵ, where n is the number of bits being encoded, and ϵ is any constant. No results for this class were previously known.

– **Constructions for streaming, space-bounded tampering:** We obtain *unconditional* non-malleable codes against streaming, space-bounded tampering, where the tampering function is represented by a read-once, bounded-width branching program. Our construction does not require CRS or computational assumptions.
No NMC results for this standard complexity theoretic class were previously known. However, this tampering class can be viewed as a subset (or the intersection) of the space bounded class considered by Faust et al. [33] (who don't limit the adversary to be streaming), and the block-wise tampering class considered by Chandran et al. [13] (who don't bound the adversary's space, but don't give security in the event that decoding fails). In both cases there cannot be NMC with the standard notion of security, and so those previous works must relax the security requirement (and [33] also relies on a random oracle). In contrast, we achieve standard (in fact, even stronger) notion of NMC, without random oracle (nor CRS, nor any computational assumption) for our class.
– **Additional Constructions:** We also briefly note two additional applications of our paradigm as proof of concept. Both complexity classes can be represented circuits of size $O(n^c)$ for some fixed c, a class which [36] provide non-malleable codes for in the CRS model, *without* computational assumptions. We include these results here, merely to show the applicability of our framework to general correlation bounds; for example strong correlation bounds against $\mathsf{ACC}^0[p]$ or TC^0 are likely immediately lead to non-malleable codes against the same classes using our framework.
  1. Under the same assumptions invoked in the constructions against AC^0 and bounded-depth decision trees we obtain computational NMC in the CRS model against tampering with small threshold circuits: threshold circuits with depth d and $n^{1+\epsilon}$ wires.
  2. Assuming any public key encryption scheme and zk-SNARKs, we obtain computational NMC in the CRS model against tampering by Turing Machines running in time $O(n^k)$, where k is a constant. However, we should note that these codes have weak tampering guarantees: tampering experiments with respect to different messages are only polynomially close to one another.

1.2 Technical Overview

We begin by describing our computational NMC construction (in the CRS model) for one-bit messages secure against tampering in AC^0, which will give the starting point intuition for our results. We then show how the AC^0 construction can be modified to derive a general template for constructing NMC for one-bit messages secure against a wider range of tampering classes $\mathcal{F}$, and discuss various classes $\mathcal{F}$ for which the template can be instantiated. We then discuss

how the template can be extended to achieve NMC for multi-bit messages secure against a wide range of tampering classes $\mathcal{F}$. Finally, we discuss some particular instantiations of our multi-bit template, including our constructions of computational NMC (in the CRS model) against tampering in AC^0 and against bounded-depth decision trees, as well as our *unconditional* NMC (with no CRS) against streaming tampering adversaries with bounded memory.

The starting point: Computational NMC against AC^0 *for one-bit messages.* The idea is to use a very similar paradigm to the Naor and Yung paradigm for CCA1 encryption [49] (later extended to achieve CCA2 [51, 46]), using double encryption with simulation-sound NIZK. The main observation is that using the tableau method, we can convert *any* NIZK proof system with polynomial verification into a NIZK proof system with a verifier in AC^0.

We also need a PKE scheme with perfect correctness and decryption in AC^0(this can be constructed using the transformation of Dwork et al. [30] on top of the scheme of Bogdanov and Lee [10]).

We now sketch (a slightly simplified version of) the NM encoding scheme:

The CRS will contain a public key PK for an encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as above, and a CRS crs for a NIZK. For $b \in \{0,1\}$, Let $\mathcal{D}_b$ denote the distribution over $x_1, \ldots, x_n \in \{0,1\}^n$ such that $x_1, \ldots, x_n$ are uniform random, conditioned on the parity of the bits being equal to b.

To encode a bit b**:**

1. Randomly choose bits $x_1, \ldots, x_n$ from $\mathcal{D}_b$
2. Compute $c_1 \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(x_1), \ldots, c_n \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(x_n)$ and $c \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(b)$.
3. Compute n NIZK proofs $\pi_1, \ldots, \pi_n$ that $c_1, \ldots, c_n$ are encryptions of bits $x_1, \ldots, x_n$.
4. Compute a NIZK proof π that there exists a bit b' such that the plaintexts underlying $c_1, \ldots, c_n$ are in the support of $\mathcal{D}_{b'}$ and b' is the plaintext underlying c.
5. Compute tableaus $T_1, \ldots, T_n$ of the computation of the NIZK verifier on $\pi_1, \ldots, \pi_n$.
6. Compute a tableau T of the computation of the NIZK verifier on proof π.
7. Output $(c_1, \ldots, c_n, c, T, (x_1, T_1), \ldots, (x_n, T_n))$.

To decode $(c_1, \ldots, c_n, c, T, (x_1, T_1), \ldots, (x_n, T_n))$**:**

1. Check the tableaus $T_1, \ldots, T_n, T$.
2. If they all accept, output the parity of $x_1, \ldots, x_n$.

In the proof we will switch from an honest encoding of b to a simulated encoding and from an honest decoding algorithm to a simulated decoding algorithm. At each point we will show that the decodings of tampered encodings stay the same. Moreover, if, in the final hybrid, decodings of tampered encodings depend on b, we will use this fact to build a circuit in AC^0, whose output is correlated with the parity of its input, reaching a contradiction. In more detail, in the first hybrid we switch to simulated proofs. Then we switch $c_1, \ldots, c_n, c$,

in the "challenge" encoding to encryptions of garbage $c'_1, \ldots, c'_n, c'$, and next we switch to an alternative decoding algorithm *in* AC^0 , which requires the trapdoor SK (corresponding to the public key PK which is contained in the CRS).

Alternative Decoding Algorithm:

To decode $(c_1, \ldots, c_n, c, T, (x_1, T_1), \ldots, (x_n, T_n))$**:**

1. check the tableaus $T_1, \ldots, T_n, T$
2. If it accepts, output the decryption of c using trapdoor SK.

In the final hybrid, the simulator will not know the parity of $x_1, \ldots, x_n$ in the challenge encoding and will have received precomputed $T_1^0, T_1^1, \ldots, T_n^0, T_n^1, T$ as non-uniform advice, where T is a simulated proof of the statement "the plaintexts underlying $c'_1, \ldots, c'_n$ and the plaintext underlying c' have the same parity" and for $i \in [n], \beta \in \{0,1\}, T_i^\beta$ is a simulated proof of the statement "c'_i is an encryption of the bit β".

We will argue by contradiction that if the decoding of the tampered encoding is correlated with the parity of $x_1, \ldots, x_n$ then we can create a circuit whose output is correlated with the parity of its input in AC^0 . Specifically, the AC^0 circuit will have the crs, SK, precomputed $c'_1, \ldots, c'_n, c', T, T_1^0, T_1^1, \ldots, T_n^0, T_n^1$ and adversarial tampering function f hardwired in it. It will take $x_1, \ldots, x_n$ as input. It will compute the simulated encoding in AC^0 by selecting the correct tableaus: $T_1^{x_1}, \ldots, T_n^{x_n}$ according to the corresponding input bit. It will then apply the adversarial tampering function (in AC^0), perform the simulated decoding (in AC^0) and output a guess for the parity of $x_1, ..x_n$ based on the result of the decoding. Clearly, if the decoding in the final hybrid is correlated with parity, then we have constructed a distribution over AC^0 circuits such that w.h.p. over choice of circuit from the distribution, the output of the circuit is correlated with the parity of its input. This contradicts known results on the hardness of computing parity in AC^0 .

A general template for one-bit NMC. The above argument can be used to derive a template for the construction/security proof of NMC against more general classes $\mathcal{F}$. The idea is to derive a high-level sequence of hybrid distributions and corresponding *minimal* requirements for proving the indistinguishability of consecutive hybrids. We can now instantiate the tampering class $\mathcal{F}$, "hard distributions" $(\mathcal{D}_0, \mathcal{D}_1)$, encryption scheme and NIZK proof in any way that satisfies these minimal requirements. Note that each hybrid distribution is a distribution over the output of the tampering experiment. Therefore, public key encryption and NIZK against arbitrary PPT adversaries may be too strong of a requirement. Indeed, it is by analyzing the exact security requirements needed to go from one hybrid to the other that (looking ahead) we are able to remove the CRS and all computational assumptions from our construction of NMC against streaming adversaries with bounded memory. In addition, we can also use our template to obtain constructions (in the CRS model and under computational assumptions) against other tampering classes $\mathcal{F}$.

Extending the template to multi-bit NMC. The construction for AC^0 given above and the general template do not immediately extend to multi-bit messages. In particular, encoding m bits by applying the parity-based construction bit-by-bit fails, even if we use the final proof T to "wrap together" the encodings of multiple individual bits. The problem is that the proof strategy is to entirely decode the tampered codeword and decide, based on the results, whether to output 0 or 1 as the guess for the parity of some $x_1, \ldots, x_n$. But if we encode many bits, $b_1, \ldots, b_m$, then the adversary could maul in such a way that the tampered codeword decodes to $b'_1, \ldots, b'_m$ where each of b'_i is *individually* independent of the parity of the corresponding $x^i_1, \ldots, x^i_n$, but taken as a whole, the entire output may be correlated. As a simple example, the attacker might maul the codeword so that it decodes to $b'_1, \ldots, b'_m$ that are uniform subject to satisfying $b'_1 \oplus \cdots \oplus b'_m = b_1 \oplus \cdots \oplus b_m$. Clearly, there is a correlation here between the input and output, but we cannot detect this correlation in AC^0, since detecting the correlation itself seems to require computing parity!

In the case of parity (and the class AC^0), the above issue can be solved by setting m sufficiently small (but still polynomial) compared to n. We discuss more details about the special case of parity below. However, we would first like to explain how the general template must be modified for the multi-bit case, given the above counterexample. Specifically, note that the difficulty above comes into play only in the final hybrid. Thus, we only need to modify the final hybrid slightly and require that for any Boolean function F over m variables, it must be the case that the composition of F with the simulated decoding algorithm is in a computational class that still cannot distinguish between draws $x_1, \ldots, x_n$ from $\mathcal{D}_0$ or $\mathcal{D}_1$. While the above seems like a strong requirement, we show that by setting m much smaller than n, we can still obtain meaningful results for classes such as AC^0 and bounded-depth decision trees.

Multi-bit NMC against AC^0. If we want to encode m bits, for each of the underlying encodings $i \in [m]$, we will use $n :\approx m^3$ bits: $\boldsymbol{x}^i = x^i_1, \ldots, x^i_n$. To see why this works, we set up a hybrid argument, where in each step we will fix all the underlying encodings except for a single one: $\boldsymbol{x} = x_1, \ldots, x_n$, which we will switch from having parity 0 to having parity 1. Therefore, we can view C—the function computing the output of the tampering experiment in this hybrid—to be a function of variables $\boldsymbol{x} = x_1, \ldots, x_n$ only (everything else is constant and "hardwired"). For $i \in [m]$, let C_i denote the i-th output bit of C. We use $\mathsf{PAR}(\boldsymbol{x})$ to denote the parity of $\boldsymbol{x}$.

Now, for any Boolean function F over m variables, consider $F(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$, where we are simply taking an arbitrary Boolean function F of the decodings of the individual bits. Our goal is to show that $F(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ is not correlated with parity of $\boldsymbol{x}$. Consider the Fourier representation of $F(y_1, \ldots, y_m)$. This is a linear combination of parities of the input variables $y_1, \ldots, y_m$, denoted $\chi_S(y_1, \ldots, y_m)$, for all subsets $S \in \{0,1\}^m$. (See here [27]).

On the other hand, $F(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ is a Boolean function over $n \approx m^3$ variables (i.e. a linear combination over parities of the input variables $x_1, \ldots, x_n$, denoted $\chi_{S'}(x_1, \ldots, x_n)$, for all subsets $S' \in \{0,1\}^n$). A representation of $F(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ can be obtained by taking each term $\hat{F}(S)\chi_S(y_1, \ldots, y_m)$ in the Fourier representation of F and composing with $C_1, \ldots, C_m$ to obtain the term $\hat{F}(S)\chi_S(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$. Since, by well-known properties of the Fourier transform, $|\hat{F}(S)| \leq 1$, we can get an upper bound on the correlation of $F(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ and $\mathsf{PAR}(\boldsymbol{x})$, by summing the correlations of each function $\chi_S(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ and $\mathsf{PAR}(\boldsymbol{x})$. Recall that the correlation of a Boolean function g with $\mathsf{PAR}(\boldsymbol{x})$ is by definition, exactly the Fourier coefficient of g corresponding to parity function $\chi_{[n]}$. Thus, to prove that the correlation of $\chi_S(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ and $\mathsf{PAR}(\boldsymbol{x})$ is low, we use the fact that $\chi_S(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ can be computed by a (relatively) low depth circuit. To see this, note that each C_i is in AC^0 and so has low depth, moreover, since S has size at most m, we only need to compute parity over m variables, which can be done in relatively low depth when $m \ll n$. We now combine the above with Fourier concentration bounds for low-depth circuits [52]. Ultimately, we prove that for each S, the correlation of $\chi_S(C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ and $\mathsf{PAR}(\boldsymbol{x})$, is less than $1/2^{m(1+\delta)}$, where δ is a constant between 0 and 1. This means that we can afford to sum over all 2^m terms in the Fourier representation of F and still obtain negligible correlation.

Multi-bit NMC against bounded-depth decision trees. Our result above extends to bounded-depth decision trees by noting that (1) If we apply a random restriction (with appropriate parameters) to input $x_1, \ldots, x_n$ then, w.h.p. the AC^0 circuit used to compute the output of the tampering experiment collapses to a bounded-depth decision tree of depth $m^\varepsilon - 1$; (2) on the other hand, again choosing parameters of the random restriction appropriately, $\mathsf{PAR}(x_1, \ldots, x_n)$ collapses to parity over at least $m^{1+\varepsilon}$ variables; (3) any Boolean function over m variables can be computed by a decision tree of depth m; (4) the composition of a depth-$m^\varepsilon - 1$ decision tree and depth-m decision tree yields a decision tree of depth at most $(m^\varepsilon - 1)(m) < m^{1+\varepsilon}$. Finally, we obtain our result by noting that decision trees of depth less than $m^{1+\varepsilon}$ are uncorrelated with parity over $m^{1+\varepsilon}$ variables.

Unconditional NMC (with no CRS) against bounded, streaming tampering. Recently, Raz [50] proved that learning parity is hard for bounded, streaming adversaries. In particular, this gives rise to hard distributions $\mathcal{D}_b, b \in \{0,1\}$ such that no bounded, streaming adversary can distinguish between the two. $\mathcal{D}_b$ corresponds to choosing a random parity χ_S, outputting random examples $(\boldsymbol{x}, \chi_S(\boldsymbol{x}))$ and then outputting $\boldsymbol{x}^*$ such that $\chi_S(\boldsymbol{x}^*)$ is equal to b. The above also yields an unconditional, "parity-based" encryption scheme against bounded, streaming adversaries. Note, however, that in order to decrypt (without knowledge of the secret key), we require space beyond the allowed bound of the adversary. Given the above, we use $\mathcal{D}_b, b \in \{0,1\}$ as the hard distributions in our construction and use the parity-based encryption scheme as the "public key encryption scheme" in our construction. Thus, we get rid of the public key in

the CRS (and the computational assumptions associated with the public key encryption scheme).

To see why this works, note that in the hybrid where we require semantic security of the encryption scheme, the decryption algorithm is not needed for decoding (at this point the honest decoding algorithm is still used). So essentially we can set the parameters for the encryption scheme such that the output of the Tampering experiment in that hybrid (which outputs the decoded value based on whether $x_1, .., x_n$ is in the support of $\mathcal{D}_0$ or $\mathcal{D}_1$) can be computed in a complexity class that is too weak to run the decryption algorithm. On the other hand, we must also consider the later hybrid where we show that the output of the tampering experiment can be computed in a complexity class that is too weak to distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$. In this hybrid, we do use the alternate decoding procedure. But now it seems that we need decryption to be contained in a complexity class that is too weak to decide whether $x_1, \ldots, x_n$ is in the support of $\mathcal{D}_0$ or $\mathcal{D}_1$, while previously we required exactly the opposite! The key insight is that since we are in the streaming model and since (1) the simulated ciphertexts $(c'_1, \ldots, c'_n, c')$ in this hybrid contain no information about $x_1, \ldots, x_n$ and (2) the simulated ciphertexts precede $x_1, \ldots, x_n$, the output of the tampering function in blocks containing ciphertexts *does not depend on* $x_1, \ldots, x_n$ *at all*. So the decryption of the tampered ciphertexts can be given as non-uniform advice, instead of being computed on the fly, and we avoid contradiction.

In order to get rid of the CRS and computational assumption for the NIZK, we carefully leverage some additional properties of the NMC setting and the streaming model. First, we consider cut-and-choose based NIZK's (based on MPC-in-the-head), where the Verifier is randomized and randomly checks certain locations or "slots" in the proof to ensure soundness. Specifically, given a Circuit-SAT circuit C and witness w, the prover will secret share $w := w_1 \oplus \cdots \oplus w_\ell$ and run an MPC protocol among ℓ parties (for constant ℓ), where party P_i has input w_i and the parties are computing the output of $C(w_1 \oplus \cdots \oplus w_\ell)$. The prover will then "encrypt" each view of each party in the MPC protocol, using the parity-based encryption scheme described above and output this as the proof. This is then repeated λ times (where λ is security parameter). The Verifier will then randomly select two parties from each of the λ sets, decrypt the views and check that the views correspond to the output of 1 and are consistent internally and with each other.

We next note that in our setting, the NIZK simulator can actually know the randomness used by the Verifier. This is because the simulated codeword and the decoding are done by the same party in the NMC security experiment. Therefore, the level of "zero-knowledge" needed from the simulation of the NIZK is in-between honest verifier and malicious. This is because the adversary can still use the tampering function to "leak" information from the unchecked slots of the proof to the checked slots, while a completely honest verifier would learn absolutely nothing about the unchecked slots. In order to switch from a real proof to a simulated proof, we fill in unchecked slots one-by-one with parity-based encryptions of garbage. We must rely on the fact that a bounded, streaming

adversary cannot distinguish real encryptions from garbage encryptions in order to argue security. Specifically, since we are in the bounded streaming model, we can argue that the adversary can only "leak" a small amount of information from the unchecked slots to the checked slots. This means that the entire output of the experiment can be simulated by a bounded, streaming adversary, which in turn means that the output of the experiment must be indistinguishable when real, unchecked encodings are replaced with encodings of garbage. Arguing simulation soundness, requires a similar argument, but more slots are added to the proof and slots in an honest proof are only filled if the corresponding position in the bit-string corresponding to the statement to be proven is set to 1. We encode the statement in such a way that if the statement changes, the adversary must switch an unfilled slot to a filled slot. Intuitively, since the bounded streaming attacker can only carry over a small amount of information from previous slots, this will be as difficult as constructing a new proof from scratch.

1.3 Related Work

The notion of NMC was formalized by Dziembowski, Pietrzak and Wichs [32]. Split state classes of tampering functions introduced by Liu and Lysyanskaya [48], have subsequently received much attention with a sequence of improvements achieving reduced number of states, improved rate, or other desirable features [31, 3, 18, 2, 7, 1, 42, 16, 45, 40, 41]. Recently [6, 8] gave efficient constructions of non-malleable codes for "non-compartmentalized" tampering function classes.

Faust et.al [36] presented a construction of efficient NMC in CRS model, for tampering function families $\mathcal{F}$ with size $|\mathcal{F}| \leq 2^{\mathsf{poly}(n)}$, where n is the length of codeword. The construction is based on t-wise independent hashing for t proportional to $\log |\mathcal{F}|$. This gives information-theoretically secure NMC resilient to tampering classes which can be represented as poly-size circuits. While [36] construction allows adaptive selection of tampering function $f \in \mathcal{F}$ after the t-wise independent hash function h (CRS) is chosen, the bound on the size of F needs to be fixed *before* h is chosen. In particular, this means that the construction does not achieve security against the tampering functions $f \in \mathsf{AC}^0$ in general, since AC^0 contains *all* poly-size and constant depth circuit families, but rather provides tamper resilience against *specific* families in AC^0 (ACC^0, etc.) Cheraghchi and Guruswami [20] in an independent work showed the existence of information theoretically secure NMC against tampering families $\mathcal{F}$ of size $|\mathcal{F}| \leq 2^{2^{\alpha n}}$ with optimal rate $1 - \alpha$. This paper gave the first characterization of the rate of NMC, however the construction of [20] is inefficient for negligible error.

Ball et.al [8] gave a construction of efficient NMC against n^{δ}-local tampering functions, for any constant $\delta > 0$. Notably, this class includes NC^0 tampering functions, namely constant depth circuits with bounded fan-in. It should be noted however, that the results of [8] do not extend to tampering adversaries in AC^0, since even for a low depth circuit in AC^0, any single output bit can depend on *all* input bits, thus violating the n^{δ}-locality constraint.

In a recent work, Chattopadhyay and Li [17] gave constructions of NMC based on connections between NMC and seedless non-malleable extractors. One of their results is an efficient NMC against t-local tampering functions, where the decoding algorithm for the NMC is deterministic (in contrast, the result in [8] has randomized decoding). The locality parameters of the NMC in [17] are not as good as the one in [8], but better than the deterministic-decoding construction given in the appendix of the full version of [8]. Additionally, [17] also present a NMC against AC^0 tampering functions. However, this NMC results in a codeword that is super-polynomial in the message length, namely inefficient.

A recent work by Faust et.al [33] considered larger tampering classes by considering space bounded tampering adversaries in random oracle model. The construction achieves a new notion of *leaky* continuous non-malleable codes, where the adversary is assumed to learn some bounded $\log(|m|)$ bits of information about the underlying message m. However, this result is not directly comparable to ours as the adversarial model we consider is a that of standard non-malleability (without leakage), and for a subset of this tampering class (streaming space-bounded adversary) we achieve information theoretic security without random oracles.

Chandran et.al [13] considered another variant of non-malleable codes, called *block-wise* non-malleable codes. In this model, the codeword consists of number of blocks and the adversary receives the codeword block-by-block. The tampering function also consists of various function f_is, where each f_i can depend on codeword blocks $c_1, \ldots, c_i$ and modifies c_i to c'_i. It can be observed that standard non-malleability cannot be achieved in this model since, the adversary can simply wait to receive all the blocks of the codeword and then decode the codeword as part of last tampering function. Therefore, [13] define a new notion called non-malleability with replacement which relaxes the non-malleability requirement and considers the attack to be successful only if the tampered codeword is *valid and related* to the original message.

Other works on non-malleable codes include [34, 21, 14, 4, 38, 12, 26, 35, 2, 15, 42, 25, 29]. We guide the interested reader to [39] and [48] for a discussion of various models for tamper and leakage resilience.

2 Definitions

Where appropriate, we interpret functions $f : S \to \{\pm 1\}$ as boolean functions (and vice-versa) via the mapping: $0 \leftrightarrow 1$ and $1 \leftrightarrow -1$. The support of vector $\boldsymbol{x}$ is the set of indices i such that $x_i \neq 0$. A *bipartite graph* is an undirected graph $G = (V, E)$ in which V can be partitioned into two sets V_1 and V_2 such that $(u, v) \in E$ implies that either $u \in V_1$ and $v \in V_2$ or $v \in V_1$ and $u \in V_2$.

Non-Malleable Codes In this section we define the notion of *non-malleable codes* and its variants. In this work, we assume that the decoding algorithm of the non-malleable code may be *randomized* and all of our generic theorems are stated for this case. Nevertheless, only our instantiation for the streaming adversary

(refer Section 7 in full version [9])requires a randomized decoding algorithm, while our other instantiations enjoy deterministic decoding. We note that the original definition of non-malleable codes, given in [32], required a deterministic decoding algorithm. Subsequently, in [8], an alternative definition that allows for randomized decoding was introduced. We follow here the definition of [8]. Please see [8] for a discussion on why deterministic decoding is not necessarily without loss of generality in the non-malleable codes setting and for additional motivation for allowing randomized decoding.

Definition 1 (Coding Scheme). *Let $\Sigma, \widehat{\Sigma}$ be sets of strings, and $\kappa, \widehat{\kappa} \in \mathbb{N}$ be some parameters. A coding scheme consists of two algorithms (E, D) with the following syntax:*

- *The encoding algorithm* (perhaps randomized) *takes input a block of message in Σ and outputs a codeword in $\hat{\Sigma}$.*
- *The decoding algorithm* (perhaps randomized) *takes input a codeword in $\hat{\Sigma}$ and outputs a block of message in Σ.*

We require that for any message $m \in \Sigma$, $\Pr[\mathsf{D}(\mathsf{E}(m)) = m] = 1$, where the probability is taken over the choice of the encoding algorithm. In binary settings, we often set $\Sigma = \{0,1\}^{\kappa}$ and $\widehat{\Sigma} = \{0,1\}^{\widehat{\kappa}}$.

We next provide definitions of non-malleable codes of varying levels of security. We present general, game-based definitions that are applicable even for NMC that are in a model with a CRS, or that require computational assumptions. The corresponding original definitions of non-malleability, appropriate for an unconditional setting without a CRS, can be obtained as a special case of our definitions when setting $\mathsf{crs} = \perp$ and taking $\mathcal{G}$ to include all computable functions. These original definitions are also presented in Appendix A.1 of the full version [9].

Definition 2 (Non-malleability). *Let $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ be a coding scheme. Let $\mathcal{F}$ be some family of functions. For each attacker A, $m \in \Sigma$, define the tampering experiment $\mathsf{Tamper}^{\Pi,\mathcal{F}}_{A,m}(n)$:*

1. Challenger samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$ and sends crs to A.
2. Attacker A sends the tampering function $f \in \mathcal{F}$ to the challenger.
3. Challenger computes $c \leftarrow \mathsf{E}(\mathsf{crs}, m)$.
4. Challenger computes the tampered codeword $\tilde{c} = f(c)$ and computes $\tilde{m} = \mathsf{D}(\mathsf{crs}, \tilde{c})$.
5. Experiment outputs $\tilde{m}$.

Fig. 1: Non-Malleability Experiment $\mathsf{Tamper}^{\Pi,\mathcal{F}}_{A,m}(n)$

We say the coding scheme $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is non-malleable against tampering class $\mathcal{F}$ and attackers $A \in \mathcal{G}$, if for every $A \in \mathcal{G}$ there exists a PPT

simulator Sim *such that for any message* $m \in \Sigma$ *we have,*

$$\mathsf{Tamper}^{\Pi,\mathcal{F}}_{A,m}(n) \approx \mathbf{Ideal}_{\mathsf{Sim},m}(n)$$

where $\mathbf{Ideal}_{\mathsf{Sim},m}(n)$ *is an experiment defined as follows,*

1. Simulator Sim has oracle access to adversary A and outputs $\tilde{m} \cup \{\mathsf{same}^*\} \leftarrow \mathsf{Sim}^{A(\cdot)}(n)$.
2. Experiment outputs m if Sim outputs same^* and outputs $\tilde{m}$ otherwise.

Fig. 2: Non-Malleability Experiment $\mathbf{Ideal}_{\mathsf{Sim},m}(n)$

Definition 3 (Strong Non-malleability). *Let* $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *be a coding scheme. Let* $\mathcal{F}$ *be some family of functions. For each attacker* $A \in \mathcal{G}$*,* $m \in \Sigma$*, define the tampering experiment* $\mathsf{StrongTamper}^{\Pi,\mathcal{F}}_{A,m}(n)$*:*

1. Challenger samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$ and sends crs to A.
2. Attacker A sends the tampering function $f \in \mathcal{F}$ to the challenger.
3. Challenger computes $c \leftarrow \mathsf{E}(\mathsf{crs}, m)$.
4. Challenger computes the tampered codeword $\tilde{c} = f(c)$.
5. Compute $\tilde{m} = \mathsf{D}(\mathsf{crs}, \tilde{c})$.
6. Experiment outputs same^* if $\tilde{c} = c$, and $\tilde{m}$ otherwise.

Fig. 3: Strong Non-Malleability Experiment $\mathsf{StrongTamper}^{\Pi,\mathcal{F}}_{A,m}(n)$

We say the coding scheme $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *is strong non-malleable against tampering class* $\mathcal{F}$ *and attackers* $A \in \mathcal{G}$ *if we have*

$$\mathsf{StrongTamper}^{\Pi,\mathcal{F}}_{A,m_0}(n) \approx \mathsf{StrongTamper}^{\Pi,\mathcal{F}}_{A,m_1}(n)$$

for any $A \in \mathcal{G}$*,* $m_0, m_1 \in \Sigma$*.*

We now introduce an intermediate variant of non-malleability, called *Medium Non-malleability*, which informally gives security guarantees "in-between" strong and regular non-malleability. Specifically, the difference is that the experiment is allowed to output same^* only when some predicate g evaluated on $(c, \tilde{c})$ is set to true. Thus, strong non-malleability can be viewed as a special case of medium non-malleability, by setting g to be the identity function. On the other hand, regular non-malleability does not impose restrictions on when the experiment is allowed to output same^*. Note that g cannot be just any predicate in order for the definition to make sense. Rather, g must be a predicate such that if g evaluated on $(c, \tilde{c})$ is set to true, then (with overwhelming probability over the random coins of D) $\mathsf{D}(\tilde{c}) = \mathsf{D}(c)$.

Definition 4 (Medium Non-malleability). *Let* $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *be a coding scheme. Let* $\mathcal{F}$ *be some family of functions.*

Let $g(\cdot,\cdot,\cdot,\cdot)$ be a predicate such that, for each attacker $A \in \mathcal{G}$, $m \in \Sigma$, the output of the following experiment, $\mathsf{Expt}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ is 1 with at most negligible probability:

1. Challenger samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$ and sends crs to A.
2. Attacker A sends the tampering function $f \in \mathcal{F}$ to the challenger.
3. Challenger computes $c \leftarrow \mathsf{E}(\mathsf{crs}, m)$.
4. Challenger computes the tampered codeword $\tilde{c} = f(c)$.
5. Challenger samples $r \leftarrow U_\ell$.
6. Experiment outputs 1 if $([g(\mathsf{crs}, c, \tilde{c}, r) = 1] \wedge [\mathsf{D}(\mathsf{crs}, \tilde{c}; r) \neq m])$.

Fig. 4: The experiment corresponding to the special predicate g

For g as above, each $m \in \Sigma$, and attacker $A \in \mathcal{G}$, define the tampering experiment $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ *as shown in figure 5:*

1. Challenger samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$ and sends crs to A.
2. Attacker A sends the tampering function $f \in \mathcal{F}$ to the challenger.
3. Challenger computes $c \leftarrow \mathsf{E}(\mathsf{crs}, m)$.
4. Challenger computes the tampered codeword $\tilde{c} = f(c)$.
5. Challenger samples $r \leftarrow U_\ell$ and computes $\tilde{m} = \mathsf{D}(\mathsf{crs}, \tilde{c}, r)$.
6. Experiment outputs same^* if $g(\mathsf{crs}, c, \tilde{c}, r) = 1$, and $\tilde{m}$ otherwise.

Fig. 5: Medium Non-Malleability Experiment $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m,g}(n)$

We say the coding scheme $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is medium non-malleable against tampering class $\mathcal{F}$ and attackers $A \in \mathcal{G}$ if we have

$$\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m_0,g}(n) \approx \mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m_1,g}(n)$$

for any $A \in \mathcal{G}$, $m_0, m_1 \in \Sigma$.

We next recall some standard definitions of public-key encryption (PKE), pseudorandom generator (PRG), and non-interactive zero knowledge proof systems with simulation soundness in sections 2.2, and 2.3 of the full version [9].

Definition 5 (Non-Interactive Simulatable Proof System). *A tuple of probabilistic polynomial time algorithms $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ is a* non-interactive simulatable proof system *for language $L \in NP$ with witness relation W if $(\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ have the following syntax:*

- $\mathsf{CRSGen}^{\mathsf{NI}}$ *is a randomized algorithm that outputs $(\mathsf{crs}^{\mathsf{NI}}, \tau_{\mathsf{sim}})$.*
- *On input crs, $x \in L$ and witness w such that $W(x, w) = 1$, $\mathsf{P}^{\mathsf{NI}}(\mathsf{crs}, x, w)$ outputs proof π.*
- *On input crs, x, π, $\mathsf{V}^{\mathsf{NI}}(\mathsf{crs}, x, \pi)$ outputs either 0 or 1.*
- *On input crs, τ_{sim} and $x \in L$, $\mathsf{Sim}^{\mathsf{NI}}(\mathsf{crs}, \tau_{\mathsf{sim}}, x)$ outputs simulated proof π'.*

Completeness: We require the following completeness property: For all $x \in L$, and all w such that $W(x,w) = 1$, for all strings $\mathsf{crs}^{\mathsf{NI}}$ of length $\mathrm{poly}(|x|)$, and for all adversaries $\mathcal{A}$ we have

$$\Pr\left[\begin{array}{c}(\mathsf{crs}^{\mathsf{NI}}, \tau_{\mathsf{Sim}}) \leftarrow \mathsf{CRSGen}^{\mathsf{NI}}(1^n); (x,w) \leftarrow \mathcal{A}(\mathsf{crs}^{\mathsf{NI}}); \\ \pi \leftarrow \mathsf{P}^{\mathsf{NI}}(\mathsf{crs}^{\mathsf{NI}}, x, w) : \mathsf{V}^{\mathsf{NI}}(\mathsf{crs}^{\mathsf{NI}}, x, \pi) = 1\end{array}\right] \geq 1 - \mathsf{negl}(n)$$

Soundness: We say that Π^{NI} enjoys soundness against adversaries $\mathcal{A} \in \mathcal{G}$ if: For all $x \notin L$, and all adversaries $\mathcal{A} \in \mathcal{G}$:

$$\Pr\left[\begin{array}{c}(\mathsf{crs}^{\mathsf{NI}}, \tau_{\mathsf{Sim}}) \leftarrow \mathsf{CRSGen}^{\mathsf{NI}}(1^n); \\ (x,\pi) \leftarrow \mathcal{A}(\mathsf{crs}^{\mathsf{NI}}) : \mathsf{V}^{\mathsf{NI}}(\mathsf{crs}^{\mathsf{NI}}, x, \pi) = 0\end{array}\right] \geq 1 - \mathsf{negl}(n)$$

The security properties that we require of Π^{NI} will depend on our particular non-malleable code construction as well as the particular class, $\mathcal{F}$, of tampering functions that we consider. The exact properties needed are those that will arise from Theorems 2 and 4. In subsequent sections, we will show how to construct non-interactive simulatable proof systems satisfying these properties.

Proof Systems for Circuit SAT We now consider proof of knowledge systems for Circuit SAT, where the prover and/or verifier have limited computational resources.

Definition 6 (Proof of Knowledge Systems for Circuit SAT with Computationally Bounded Prover/Verifier). *For a circuit C, let $\mathcal{L}(C)$ denote the set of strings x such that there exists a witness w such that $C(x,w) = 1$. For a class $\mathcal{C}$, let $\mathcal{L}(\mathcal{C})$ denote the set $\{\mathcal{L}(C) \mid C \in \mathcal{C}\}$. $\Pi = (\mathsf{P}, \mathsf{V})$ is a Circuit SAT proof system for the class $\mathcal{L}(\mathcal{C})$ with prover complexity $\mathcal{D}$ and verifier complexity $\mathcal{E}$ if the following are true:*

- *For all $C \in \mathcal{C}$ and all valid inputs (x,w) such that $C(x,w) = 1$, $\mathsf{P}(C,\cdot,\cdot)$ can be computed in complexity class $\mathcal{D}$.*
- *For all $C \in \mathcal{C}$, $\mathsf{V}(C,\cdot,\cdot)$ can be computed in complexity class $\mathcal{E}$.*
- *Completeness: For all $C \in \mathcal{C}$ and all (x,w) such that $C(x,w) = 1$, we have $\mathsf{V}(C, x, \mathsf{P}(C,x,w)) = 1$*
- *Extractability: For all (C,x,π), if $\Pr_r[\mathsf{V}(C,x,\pi;r) = 1]$ is non-negligible, then given (C,x,π) it is possible to efficiently extract w such that $C(x,w) = 1$.*

We construct Circuit SAT proof systems for the class $\mathcal{L}(\mathsf{P}/\mathsf{poly})$ with verifier complexity AC^0 in section 2.4 of full version [9]. We also construct Circuit SAT proof systems for the class. $\mathcal{L}(\mathsf{P}/\mathsf{poly})$ with streaming verifier in section 2.4 of full version [9].

Given the above, we have the following theorem:

Theorem 1. *Assuming the existence of same-string, weak one-time simulation sound NIZK with deterministic verifier, there exists same-string, weak one-time simulation sound NIZK with verifier in* AC^0 .

We also recall some definitions and results related to boolean analysis and present them next. in section 2.5 of full version [9].

Computational Model for Streaming Adversaries In this section we discuss the computational model used for analysis of the streaming adversaries. This model is similar to the one used in [50].

General Streaming Adversaries. The input is represented as a stream $S_1, \ldots, S_\ell$, where for $i \in [\ell]$, each $S_i \in \{0,1\}^B$, where B is the block length. We model the adversary by a *branching program*. A branching program of length ℓ and width w, is a directed acyclic graph with the vertices arranged in $\ell+1$ layers such that no layer contains more than w vertices. Intuitively, each layer represents a time step of computation whereas, each vertex in the graph corresponds to the potential memory state learned by the adversary. The first layer (layer 0) contains a single vertex, called the *start vertex*, which represents the input. A vertex is called *leaf* if it has out-degree 0, and represents the output (the learned value of x) of the program. Every non-leaf vertex in the program has exactly 2^B outgoing edges, labeled by elements $S \in \{0,1\}^B$, with exactly one edge labeled by each such S, and all the edges from layer $j-1$ going to vertices in layer j. Intuitively, these edges represent the computation on reading S_i as streaming input. The stream $S_1, \ldots, S_\ell$, therefore, define a computation-path in the branching program.

We discuss the streaming branching program adversaries, and streaming adversaries for learning parity in section 2.6 of full version [9].

3 Generic Construction for One-Bit Messages

In this section we present the generic construction for encoding a single bit.

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a public key encryption scheme with perfect correctness (see Definition 7 in [9]). Let $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ be a non-interactive simulatable proof system with soundness against adversaries $\mathcal{A} \in \mathcal{G}$ (see Definition 5). Note that in the CRS model, we implicitly assume that all algorithms take the CRS as input, and for simplicity of notation, sometimes do not list the CRS as an explicit input.

$\mathsf{CRSGen}(1^n)$:

1. Choose $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$.
2. Choose $[(\mathsf{crs}_i^{\mathsf{NI}}, \tau_{\mathsf{sim}}^i)]_{i\in\{0,\ldots n\}} \leftarrow \mathsf{CRSGen}^{\mathsf{NI}}(1^n)$. Let $\overrightarrow{\mathsf{crs}}^{\mathsf{NI}} := [\mathsf{crs}_i^{\mathsf{NI}}]_{i\in\{0,\ldots n\}}$ and let $\overrightarrow{\tau}_{\mathsf{sim}} := [\tau_{\mathsf{sim}}^i]_{i\in\{0,\ldots n\}}$
3. Output $\mathsf{crs} := (\mathrm{PK}, \overrightarrow{\mathsf{crs}}^{\mathsf{NI}})$.

Languages. We define the following languages:

- $\mathcal{L}_i^\beta$: For $i \in [n]$, $\beta \in \{0,1\}$, $s := (\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \in \mathcal{L}_i^\beta$ iff the i-th ciphertext $c_i := k_i \oplus \beta$ (where $\boldsymbol{c} = c_1, \ldots, c_n$) and the i-th encryption $\hat{k}_i$ (where $\hat{\boldsymbol{k}} = \hat{k}_1, \ldots, \hat{k}_{n+1}$) is an encryption of k_i under PK (where PK is hardwired into the language).
- $\mathcal{L}$: $s := (\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \in \mathcal{L}$ iff $(x_1, \ldots, x_n)$ is in the support of D_b where:
  1. For $i \in [n]$, $x_i := c_i \oplus k_i$
  2. $b := c \oplus k_{n+1}$
  3. $\hat{\boldsymbol{k}}$ is an encryption of $k_1, \ldots, k_{n+1}$ under PK (where PK is hardwired into the language).

$\mathsf{E}(\mathsf{crs}, b)$:

1. Sample $\boldsymbol{x} \leftarrow D_b$, where $\boldsymbol{x} = x_1, \ldots, x_n$.
2. Choose an n+1-bit key $\boldsymbol{k} = k_1, \ldots, k_n, k$ uniformly at random. For $i \in [n]$, compute $\hat{k}_i \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k_i)$ and compute $\hat{k}_{n+1} \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k)$. Let $\hat{\boldsymbol{k}} := \hat{k}_1, \ldots, \hat{k}_{n+1}$.
3. Compute $c_1 := k_1 \oplus x_1, \ldots, c_n := k_n \oplus x_n$. Let $\boldsymbol{c} := c_1, \ldots, c_n$.
4. Compute $c := b \oplus k$.
5. For $i \in [n]$, compute a non-interactive, simulatable proof T_i proving $s := (\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \in \mathcal{L}_i^{x_i}$ relative to $\mathsf{crs}_i^{\mathsf{NI}}$.
6. Compute a non-interactive, simulatable proof T proving $s := (\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \in \mathcal{L}$ relative to $\mathsf{crs}_0^{\mathsf{NI}}$.
7. Output $\mathsf{CW} := (\hat{\boldsymbol{k}}, c_1, \ldots, c_n, c, T, x_1, T_1, .., x_n, T_n)$.

$\mathsf{D}(\mathsf{crs}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := (\hat{\boldsymbol{k}}, c_1, \ldots, c_n, c, T, x_1, T_1, .., x_n, T_n)$
2. Check that V^{NI} outputs 1 on all proofs $T_1, .., T_n, T$, relative to the corresponding CRS.
3. If yes, output b such that $x_1 ... x_n$ is in the support of D_b. If not, output 0.

Fig. 6: Non-malleable code $(\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$, secure against $\mathcal{F}$ tampering.

$\mathsf{E}_1(\mathsf{crs}, \overrightarrow{\mathcal{T}}_{\mathsf{sim}}, r, b)$:

1. Sample $\boldsymbol{x} \leftarrow D_b$, where $\boldsymbol{x} = x_1, \ldots, x_n$.
2. Choose an n+1-bit key $\boldsymbol{k} = k_1, \ldots, k_n, k$ uniformly at random. For $i \in [n]$, compute $\hat{k}_i \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k_i)$ and compute $\hat{k}_{n+1} \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k)$. Let $\hat{\boldsymbol{k}} := \hat{k}_1, \ldots, \hat{k}_{n+1}$.
3. Compute $c_1 := k_1 \oplus x_1, \ldots, c_n := k_n \oplus x_n$. Let $\boldsymbol{c} := c_1, \ldots, c_n$.

4. Compute $c := b \oplus k$.
5. For $i \in [n]$, use τ^i_{sim} and r to simulate a non-interactive proof T'_i proving $(\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \in \mathcal{L}^{x_i}_i$, relative to $\mathsf{crs}^{\mathsf{NI}}_i$.
6. Use τ^0_{sim} and r to simulate a non-interactive proof T' proving $(\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \in \mathcal{L}$, relative to $\mathsf{crs}^{\mathsf{NI}}_0$.
7. Output $\mathsf{CW} := (\hat{\boldsymbol{k}}, c_1, \ldots, c_n, c, T', x_1, T'_1, .., x_n, T'_n)$.

Fig. 7: Encoding algorithm with simulated proofs.

$\mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, b)$:

1. Sample $\boldsymbol{x} \leftarrow D_b$, where $\boldsymbol{x} = x_1, \ldots, x_n$.
2. Choose $c'_1, \ldots, c'_n$ uniformly at random. Let $\boldsymbol{c'} := c'_1, \ldots, c'_n$.
3. Choose c' uniformly at random.
4. Set $\boldsymbol{k'} = c'_1, \ldots, c'_n, c'$. For $i \in [n]$, compute $\hat{k}'_i \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k'_i)$ and compute $\hat{k}'_{n+1} \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k')$. Let $\hat{\boldsymbol{k}}' := \hat{k}'_1, \ldots, \hat{k}'_{n+1}$.
5. For $i \in [n]$, use τ^i_{sim} and r to simulate a non-interactive proof T'_i proving $(\hat{\boldsymbol{k'}}, \boldsymbol{c'}, c) \in \mathcal{L}^{x_i}_i$, relative to $\mathsf{crs}^{\mathsf{NI}}_i$.
6. Use τ^0_{sim} and r to simulate a non-interactive proof T' proving $(\hat{\boldsymbol{k'}}, \boldsymbol{c'}, c) \in \mathcal{L}$, relative to $\mathsf{crs}^{\mathsf{NI}}_0$.
7. Output $\mathsf{CW} := (\hat{\boldsymbol{k}}', c'_1, \ldots, c'_n, c', T', x_1, T'_1, .., x_n, T'_n)$.

Fig. 8: Encoding algorithm with simulated proofs and encryptions.

$\mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := (\hat{\boldsymbol{k}}, c_1, \ldots, c_n, c, T, x_1, T_1, .., x_n, T_n)$,
2. Output $\mathsf{Decrypt}_{\mathrm{SK}}(\hat{k}_{n+1})$.

Fig. 9: Extracting procedure Ext**.**

$\mathsf{D}'(\mathsf{crs}, k, \mathsf{CW})$:

1. Parse $\mathsf{CW} := (\hat{\boldsymbol{k}}, c_1, \ldots, c_n, c, T, x_1, T_1, .., x_n, T_n)$,
2. Check that V^{NI} outputs 1 on all proofs $T_1, .., T_n, T$, relative to the corresponding CRS,
3. If not, output 0. Otherwise, output $b := k \oplus c$.

Fig. 10: Alternate decoding procedure D'**, given additional extracted key** k **as input.**

$g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$:

1. Parse $\mathsf{CW} = (\hat{\boldsymbol{k}}, \boldsymbol{c}, c, T, x_1, T_1, .., x_n, T_n)$, $\mathsf{CW}^* = (\hat{\boldsymbol{k}}^*, \boldsymbol{c}^*, c^*, T^*, x_1^*, T_1^*, .., x_n^*, T_n^*)$.
2. If (1) V^{NI} outputs 1 on all proofs $T^*, T_1^*, .., T_n^*$, relative to the corresponding CRS; and (2) $(\hat{\boldsymbol{k}}, \boldsymbol{c}, c) = (\hat{\boldsymbol{k}}^*, \boldsymbol{c}^*, c^*)$, then output 1. Otherwise output 0.

Fig. 11: The predicate $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$.

Let $\Psi(p, c, x, y, r, z)$ be defined as a function that takes as input a predicate p, and variables c, x, y, r, z. If $p(c, x, y, r) = 1$, then Ψ outputs 0. Otherwise, Ψ outputs z.

Theorem 2. *Let (E, D), E_1, E_2, Ext, D' and g be as defined in Figures 6, 7, 8, 9, 10 and 11. Let $\mathcal{F}$ be a computational class. If, for every adversary $\mathcal{A} \in \mathcal{G}$ outputting tampering functions $f \in \mathcal{F}$, all of the following hold:*

Simulation of proofs.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1]$,
2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_0); r_0)) \approx \Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1))$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_0, r_1 are sampled uniformly at random, $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, 0)$ and $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$.

Simulation of Encryptions.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$,
2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1)) \approx \Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2))$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_1, r_2 are sampled uniformly at random, $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$ and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$.

Simulation Soundness.

$$\Pr\left[\begin{matrix}\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \\ \wedge g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0\end{matrix}\right] \leq \mathsf{negl}(n),$$

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2 is sampled uniformly at random and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, 0)$.

Hardness of D_b relative to Alternate Decoding.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$,
2. $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathrm{SK}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2, r_3 are sampled uniformly at random, $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$ and $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, 1)$.

Then the construction presented in Figure 6 is a non-malleable code for class $\mathcal{F}$ against adversaries $\mathcal{A} \in \mathcal{G}$.

Proof (Proof of Theorem 2). We take g to be the predicate that is used in the $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ tampering experiment. We must argue that for every $m \in \{0,1\}$ and every attacker $A \in \mathcal{G}$ the output of the experiment $\mathsf{Expt}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ is 1 with at most negligible probability.

Assume towards contradiction that for some $A \in \mathcal{G}$ the output of the experiment is 1 with non-negligible probability. Then this means that the probability in the last line of experiment $\mathsf{Expt}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ that $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 1 \wedge \mathsf{D}(\mathsf{crs}, \mathsf{CW}^*; r) \neq m$ is non-negligible. Parse $\mathsf{CW} = (\hat{\boldsymbol{k}}, \boldsymbol{c}, c, T, x_1, T_1, .., x_n, T_n)$, $\mathsf{CW}^* = (\hat{\boldsymbol{k}}^*, \boldsymbol{c}^*, c^*, T^*, x_1^*, T_1^*, .., x_n^*, T_n^*)$.

Recall that $\mathsf{D}(\mathsf{crs}, \mathsf{CW}; r) = m$. Thus, if the above event occurs, it means that $\mathsf{D}(\mathsf{crs}, \mathsf{CW}; r) \neq \mathsf{D}(\mathsf{crs}, \mathsf{CW}^*; r)$. But since $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 1$, it means that V^{NI} outputs 1 on all proofs $T^*, [T_i^*]_{i\in[n]}$ and $(\hat{\boldsymbol{k}}, \boldsymbol{c}, c) = (\hat{\boldsymbol{k}}^*, \boldsymbol{c}^*, c^*)$.
This, in turn, means that there must be some bit x_i, x_i^* that CW and CW^* differ on. But note that by assumption $c_i = c_i^*$. Due to the fact that CW is well-formed and perfect correctness of the encryption scheme, it must mean that $c_i^* \notin \mathcal{L}_i^{x_i^*}$. But recall that by assumption, proof T_i^* verifies correctly. This means that soundness is broken by $A \in \mathcal{G}$. This contradicts the security of the proof system Π^{NI}.

Next, recall that we wish to show that for any adversary $A \in \mathcal{G}$ outputting tampering function $\{\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,0,g}\}_{k\in\mathbb{N}} \approx \{\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,1,g}\}_{k\in\mathbb{N}}$
To do so we consider the following hybrid argument:

Hybrid 0: The real game, $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,0,g}$, relative to g, where the real encoding $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, 0)$ and the real decoding oracle D are used.
Hybrid 1: Replace the encoding from the previous game with $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$ where r_1 is chosen uniformly at random and g, D use random coins r_1.
Hybrid 2: Replace the encoding from the previous game with $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$, where r_2 is chosen uniformly at random and g, D use random coins r_2.
Hybrid 3: Replace the decoding from the previous game, with $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathsf{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$. where r_2 is chosen uniformly at random and g, E_2 use random coins r_2.
Hybrid 4: Same as Hybrid 3, but replace the encoding with $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, 1)$, where r_3 is chosen uniformly at random and g, D' use random coins r_3.

Now, we prove our hybrids are indistinguishable.

Claim. Hybrid 0 is computationally indistinguishable from Hybrid 1.

Proof. The claim follows immediately from the **Simulation of proofs** property in Theorem 2.

Claim. Hybrid 1 is computationally indistinguishable from Hybrid 2.

Proof. The claim follows immediately from the **Simulation of Encryptions** property in Theorem 2.

Claim. Hybrid 2 is computationally indistinguishable from Hybrid 3.

Proof. This claim follows from the fact that (1) if $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 1$, then the experiment outputs same^* in both Hybrid 2 and Hybrid 3; and (2) the probability that $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 0$ and the output of the experiment is different in Hybrid 2 and Hybrid 3 is at most negligible, due to the **Simulation Soundness** property in Theorem 2.

Claim. Hybrid 3 is computationally indistinguishable from Hybrid 4.

Proof. This follows from the fact that (1) for $\gamma \in \{2, 3\}$ if $g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1$ then $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma)$ always outputs 0 and so

$$\begin{aligned}&\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma)\\ &\qquad \equiv \Psi(g, \mathsf{crs}, \mathsf{CW}_\gamma, f(\mathsf{CW}_\gamma), r_\gamma, \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma));\end{aligned}$$

and (2) the **Hardness of D_b relative to Alternate Decoding** property in Theorem 2.

4 One-Bit NMC for AC^0

In this section, we show that our generic construction yields efficient NMC for AC^0 in the CRS model, when each of the underlying primitives is appropriately instantiated.

Theorem 3. *$\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ (presented in Figure 6) is a one-bit, computational, non-malleable code in the CRS model, secure against every PPT adversary $\mathcal{A}$ outputting tampering functions $f \in \mathsf{AC}^0$, if the underlying components are instantiated in the following way:*

- *$\mathcal{E} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is a public key encryption scheme with perfect correctness and decryption in AC^0 .*
- *$\Pi^{\mathsf{NI}} := (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ is a same-string, weak one-time simulation-sound NIZK with verifier in AC^0 .*
- *For $b \in \{0, 1\}$, D_b is the distribution that samples bits $x_1 \dots x_n$ uniformly at random, conditioned on $x_1 \oplus \cdots \oplus x_n = b$.*

Note that given Theorem 1, proof systems Π^{NI} as above exist, under the assumption that same-string, weak one-time simulation-sound NIZK with (arbitrary polynomial-time) deterministic verifier exists. Such NIZK can be constructed in the CRS model from enhanced trapdoor permutations [51]. Public key encryption with perfect correctness and decryption in AC^0 can be constructed by applying the low-decryption-error transformation of Dwork et al. [30] to the (reduced decryption error) encryption scheme of Bogdanov and Lee [10]. Refer to section 4 of the full version [9] for additional details.

Proof (Proof of theorem 3). To prove the theorem, we need to show that for every PPT adversary $\mathcal{A}$ outputting tampering functions $f \in \mathcal{F}$, the necessary properties from Theorem 2 hold. We next go through these one by one.

Simulation of proofs.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1]$,
2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_0); r_0)) \approx \Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1))$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_0, r_1 are sampled uniformly at random, $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, 0)$ and $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$.
This follows immediately from the zero-knowledge property of $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$.

Simulation of Encryptions.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$,
2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1)) \approx \Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2))$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_1, r_2 are sampled uniformly at random, $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$ and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$.
This follows immediately from the fact that $\boldsymbol{c}, c$ and $\boldsymbol{c}', c'$ are identically distributed when generated by E_1 versus E_2 and from the semantic security of the public key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

Simulation Soundness.

$$\Pr\left[\begin{array}{c}\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \\ \wedge g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0\end{array}\right] \leq \mathsf{negl}(n),$$

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2 is sampled uniformly at random and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, 0)$.
Note that $g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0$ only if either of the following is true: (1) V^{NI} did not output 1 on all tampered proofs $T^*, T_1^*, \ldots, T_n^*$ in $f(\mathsf{CW}_2)$; or (2) the first 3 elements of CW_2 and $f(\mathsf{CW}_2)$ are not identical (i.e., $(\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \neq (\hat{\boldsymbol{k}^*}, \boldsymbol{c^*}, c^*)$). Now in case (1), both $\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2)$, and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ output 0. This is contradiction to the claim that $\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$. In case (2), the extractor $\mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2))$ outputs $k_{n+1}^* := \mathsf{Decrypt}_{\mathrm{SK}}(\hat{k^*}_{n+1})$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ outputs $b^* = c^* \oplus k_{n+1}^*$. Now, if $\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ but V^{NI} outputs 1 on all tampered proofs $T^*, T_1^*, \ldots, T_n^*$ in $f(\mathsf{CW}_2)$ then one-time simulation soundness of $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ does not hold.

Hardness of D_b relative to Alternate Decoding.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$,
2. $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2, r_3 are sampled uniformly at random, $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$ and $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, 1)$. Let $\boldsymbol{X}$ denote a random variable where $\boldsymbol{X}$ is sampled from D_0 with probability $1/2$ and $\boldsymbol{X}$ is sampled from D_1 with probability $1/2$ and let random variable CW denote the output of E_2 when $\boldsymbol{X}$ replaces $\boldsymbol{x}$.

To show (1), assume $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$ and $\Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$ differ by a non-negligible amount. This implies that takes as input $\boldsymbol{X}$, hardwires all other random variables, and outputs 1 in the case that $g(\mathsf{crs}, \mathsf{CW}, f(\mathsf{CW}), r) = 1$ and 0 otherwise, implying that it has non-negligible correlation to the parity of its input $\boldsymbol{X}$. We will show that the above can be computed by an AC^0 circuit with input $\boldsymbol{X}$, thus contradicting Theorem 2 from [9] which says that an AC^0 circuit has at most negligible correlation with parity of its input $\boldsymbol{X}$, denoted $\mathcal{P}(\boldsymbol{X})$.

We construct the distribution of circuits $\mathcal{C}^1_{\mathcal{F}}$, and $C \sim \mathcal{C}^1_{\mathcal{F}}$ is drawn as:

1. Sample $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.
2. Sample tampering function $f \leftarrow \mathcal{A}(\mathsf{crs})$.
3. Sample $\boldsymbol{c}', c'$ uniformly at random.
4. Set $\boldsymbol{k}' = c'_1, \ldots, c'_n, c$. For $i \in [n]$, compute $\hat{k}'_i \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k'_i)$ and compute $\hat{k}'_{n+1} \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k')$.
5. Sample r uniformly at random.
6. Sample simulated proofs $[T_i'^{\beta}]_{\beta \in \{0,1\}, i \in [n]}$ and T' (as described in Figure 8).
7. Output the following circuit C that has the following structure:
   - **hardwired variables:** crs, SK, f, $\hat{\boldsymbol{k}}'$, $\boldsymbol{c}'$, c', r, $[T_i'^{\beta}]_{\beta \in \{0,1\}, i \in [n]}$.
   - **input:** $\boldsymbol{X}$.
   - **computes and outputs:** $g(\mathsf{crs}, \mathsf{CW}, f(\mathsf{CW}), r)$.

   Note that given all the hardwired variables, computing CW is in AC^0 since all it does is, for $i \in [n]$, select the correct simulated proof $T_i'^{x_i}$ based on the corresponding input bit x_i. Additionally, f in AC^0 and g in AC^0 , since bit-wise comparison is in AC^0 and V^{SAT} is in AC^0 . Thus, the entire circuit is in AC^0 .

To show (2), assume $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$ have non-negligible statistical distance. This implies that a circuit that takes as input $\boldsymbol{X}$, hardwires all other random variables, and outputs $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$ has non-negligible correlation to the parity of $\boldsymbol{X}$. We will show that $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$ can be computed by an AC^0 circuit with input $\boldsymbol{X}$, thus contradicting Theorem 2 from [9], which says that an AC^0 circuit has at most negligible correlation with the parity of its input $\boldsymbol{X}$, denoted $\mathcal{P}(\boldsymbol{X})$.

We construct the distribution of circuits $\mathcal{C}^2_{\mathcal{F}}$, and $C \sim \mathcal{C}^2_{\mathcal{F}}$ is drawn as:

1. Sample $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.
2. Sample tampering function $f \leftarrow \mathcal{A}(\mathsf{crs})$.
3. Sample $\boldsymbol{c}', c'$ uniformly at random.
4. Set $\boldsymbol{k}' = c'_1, \ldots, c'_n, c$. For $i \in [n]$, compute $\hat{k}'_i \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k'_i)$ and compute $\hat{k}'_{n+1} \leftarrow \mathsf{Encrypt}_{\mathrm{PK}}(k')$.

5. Sample r uniformly at random.
6. Sample simulated proofs $[T_i^{'\beta}]_{\beta\in\{0,1\},i\in[n]}$ and T' (as described in Figure 8).
7. Output the following circuit C that has the following structure:
   - **hardwired variables:** crs, SK, f, $\hat{\boldsymbol{k}}'$, $\boldsymbol{c}'$, c', r, $[T_i^{'\beta}]_{\beta\in\{0,1\},i\in[n]}$.
   - **input:** $\boldsymbol{X}$.
   - **computes and outputs:** $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$.

   Note that $\mathsf{Ext} \in \mathsf{AC}^0$ since decryption for $\mathcal{E} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ in AC^0 . Moreover, as above, given all the hardwired variables, computing CW is in AC^0 since all it does is, for $i \in [n]$, select the correct simulated proof $T_i^{'x_i}$ based on the corresponding input bit x_i. Additionally, f in AC^0 and D' is in AC^0 , since xor of two bits is in AC^0 and V^{SAT} is in AC^0 . Thus, the entire circuit is in AC^0 .

Analysis for more tampering classes is presented in section 4.1 of full version [9]

5 Construction for Multi-Bit Messages

The construction for encoding multi-bit messages is similar to that for encoding a single bit, presented in section 3. The construction repeats the procedure for encoding single bit m times, for encoding m-bit messages and binds it with a proof T.

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a public key encryption scheme with perfect correctness (see Definition 7 in [9]). Let $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ be a non-interactive simulatable proof system with soundness against adversaries $\mathcal{A} \in \mathcal{G}$ (see Definition 5). Note that in the CRS model, we implicitly assume that all algorithms take the CRS as input, and for simplicity of notation, sometimes do not list the CRS as an explicit input.

$\mathsf{CRSGen}(1^n)$:

1. Choose $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^n)$.
2. Choose $[\mathsf{crs}^{\mathsf{NI}}_{i,j}, \tau^{i,j}_{\mathsf{sim}}]_{(i,j)=(0,0),i\in[m],j\in[n]} \leftarrow \mathsf{CRSGen}^{\mathsf{NI}}(1^n)$. Let $\overrightarrow{\mathsf{crs}}^{\mathsf{NI}} := [\mathsf{crs}^{\mathsf{NI}}_{i,j}]_{(i,j)=(0,0),i\in[m],j\in[n]}$ and let $\overrightarrow{\tau}_{\mathsf{sim}} := [\tau^{i,j}_{\mathsf{sim}}]_{(i,j)=(0,0),i\in[m],j\in[n]}$
3. Output $\mathsf{crs} := (\mathrm{PK}, \overrightarrow{\mathsf{crs}}^{\mathsf{NI}})$.

Languages. We define the following languages:

– $\mathcal{L}^{\beta}_{i,j}$: For $i \in [m], j \in [n]$, $\beta \in \{0,1\}$, $s := ([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}^{\beta}_{i,j}$ iff the (i,j)-th ciphertext $c^i_j := k^i_j \oplus \beta$ (where $\overline{\boldsymbol{c}} = [c^i_j]_{i\in[m],j\in[n]}$) and the (i,j)-th encryption $\hat{k}^i_j$ (where $\hat{\boldsymbol{k}}^i = \hat{k}^i_1, \ldots, \hat{k}^i_{n+1}$) is an encryption of k^i_j under PK (where PK is hardwired into the language).

– $\mathcal{L}$: $s := ([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}$ iff For each $i \in [m]$, $(x_1^i, \ldots, x_n^i)$ is in the support of D_{b^i} where:
  1. For $i \in [m], j \in [n]$, $x_j^i := c_j^i \oplus k_j^i$
  2. $b^i := c^i \oplus k_{n+1}^i$ (where $\overline{c} := c^1, \ldots, c^m$)
  3. $\hat{\boldsymbol{k}}^i$ is an encryption of $k_1^i, \ldots, k_{n+1}^i$ under PK (where PK is hardwired into the language).

$\mathsf{E}(\mathsf{crs}, \boldsymbol{b} := b^1, \ldots, b^m)$:

1. Sample $\overline{\boldsymbol{x}} := \boldsymbol{x}^1, \ldots, \boldsymbol{x}^m \leftarrow D_{\boldsymbol{b}}$, where for $i \in [m]$, $\boldsymbol{x}^i = x_1^i, \ldots, x_n^i$.
2. Choose an $m \cdot (n+1)$-bit key $\overline{\boldsymbol{k}} := [\boldsymbol{k}^i]_{i\in[m]} = [k_1^i, \ldots, k_n^i, k^i]_{i\in[m]}$ uniformly at random. For $i \in [m], j \in [n+1]$, compute $\hat{k}_j^i \leftarrow \mathsf{Encrypt}(\text{PK}, k_j^i)$. For $i \in [m]$, let $\hat{\boldsymbol{k}}^i := \hat{k}_1^i, \ldots, \hat{k}_{n+1}^i$.
3. For $i \in [m], j \in [n]$, compute $c_j^i := k_j^i \oplus x_j^i$. Let $\overline{\boldsymbol{c}} := [c_j^i]_{i\in[m],j\in[n]}$.
4. For $i \in [m]$, compute $c^i := k^i \oplus b^i$. Let $\overline{c} := [c^i]_{i\in[m]}$.
5. For $i \in [m], j \in [n]$, compute a non-interactive, simulatable proof T_j^i proving $([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}_{i,j}^{x_j^i}$ relative to $\mathsf{crs}_{i,j}^{\mathsf{NI}}$.
6. Compute a non-interactive, simulatable proof T proving $([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}$ relative to $\mathsf{crs}_{0,0}^{\mathsf{NI}}$.
7. Output $\mathsf{CW} := ([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}, T, [(x_j^i, T_j^i)]_{i\in[m],j\in[n]})$.

$\mathsf{D}(\mathsf{crs}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := ([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}, T, [(x_j^i, T_j^i)]_{i\in[m],j\in[n]})$
2. Check that V^{NI} outputs 1 on all proofs $[T_j^i]_{i\in[m],j\in[n]}, T$, relative to the corresponding CRS.
3. If yes, output $[b^i]_{i\in[m]}$ such that $x_1^i ... x_n^i$ is in the support of D_{b^i}. If not, output $\mathbf{0}$.

Fig. 12: Non-malleable code $(\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$**, secure against** $\mathcal{F}$ **tampering.**

$\mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, \boldsymbol{b} := b^1, \ldots, b^m)$:

1. Sample $\overline{\boldsymbol{x}} := \boldsymbol{x}^1, \ldots, \boldsymbol{x}^m \leftarrow D_b$, where for $i \in [m]$, $\boldsymbol{x}^i = x_1^i, \ldots, x_n^i$.
2. Choose an $m \cdot (n+1)$-bit key $\overline{\boldsymbol{k}} := [\boldsymbol{k}^i]_{i\in[m]} = [k_1^i, \ldots, k_n^i, k^i]_{i\in[m]}$ uniformly at random. For $i \in [m], j \in [n+1]$, compute $\hat{k}_j^i \leftarrow \mathsf{Encrypt}(\text{PK}, k_j^i)$. For $i \in [m]$, let $\hat{\boldsymbol{k}}^i := \hat{k}_1^i, \ldots, \hat{k}_{n+1}^i$.
3. For $i \in [m], j \in [n]$, compute $c_j^i := k_j^i \oplus x_j^i$. Let $\overline{\boldsymbol{c}} := [c_j^i]_{i\in[m],j\in[n]}$.
4. For $i \in [m]$, compute $c^i := k^i \oplus b^i$. Let $\overline{c} := [c^i]_{i\in[m]}$.

5. For $i \in [m], j \in [n]$, simulate, using $\tau_{\mathsf{sim}}^{i,j}$ and r, a non-interactive proof $T_j^{'i}$ proving $s := ([\hat{\boldsymbol{k}}^{i}]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}_{i,j}^{x_j^i}$, relative to $\mathsf{crs}_{i,j}^{\mathsf{NI}}$.
6. Simulate, using $\tau_{\mathsf{sim}}^{0,0}$ and r, a non-interactive proof T' proving $s := ([\hat{\boldsymbol{k}}^{i}]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}$, relative to $\mathsf{crs}_{0,0}^{\mathsf{NI}}$.
7. Output $\mathsf{CW} := ([\hat{\boldsymbol{k}}^{i}]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}, T', [(x_j^i, T_j^{'i})]_{i\in[m],j\in[n]})$.

Fig. 13: Encoding algorithm with simulated proofs.

$\mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, \boldsymbol{b} := b^1, \ldots, b^m)$:

1. Sample $\overline{\boldsymbol{x}} := \boldsymbol{x}^1, \ldots, \boldsymbol{x}^m \leftarrow D_b$, where for $i \in [m]$, $\boldsymbol{x}^i = x_1^i, \ldots, x_n^i$.
2. Choose $[c_j^{'i}]_{i\in[m],j\in[n]}$ uniformly at random. Let $\overline{\boldsymbol{c}}' := [c_j^{'i}]_{i\in[m],j\in[n]}$.
3. Choose $[c^{'i}]_{i\in[m]}$ uniformly at random. Let $\overline{c}' := [c^{'i}]_{i\in[m]}$.
4. Set the $m \cdot (n+1)$-bit key $\overline{\boldsymbol{k}}' := [\boldsymbol{k}^{'i}]_{i\in[m]} = [c_1^{'i}, \ldots, c_n^{'i}, c^{'i}]_{i\in[m]}$. For $i \in [m], j \in [n+1]$, compute $\hat{k}_j^{'i} \leftarrow \mathsf{Encrypt}(\mathrm{PK}, k_j^{'i})$. For $i \in [m]$, let $\hat{\boldsymbol{k}}^{'i} := \hat{k}_1^{'i}, \ldots, \hat{k}_{n+1}^{'i}$.
5. For $i \in [m], j \in [n]$, simulate, using $\tau_{\mathsf{sim}}^{i,j}$ and r, a non-interactive proof $T_j^{'i}$ proving $s := ([\hat{\boldsymbol{k}}^{i}]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}_{i,j}^{x_j^i}$, relative to $\mathsf{crs}_{i,j}^{\mathsf{NI}}$.
6. Simulate, using $\tau_{\mathsf{sim}}^{0,0}$ and r, a non-interactive proof T' proving $s := ([\hat{\boldsymbol{k}}^{i}]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}$, relative to $\mathsf{crs}_{0,0}^{\mathsf{NI}}$.
7. Output $\mathsf{CW} := ([\hat{\boldsymbol{k}}^{'i}]_{i\in[m]}, \overline{\boldsymbol{c}}', \overline{c}', T', [(x_j^i, T_j^{'i})]_{i\in[m],j\in[n]})$.

Fig. 14: Encoding algorithm with simulated proofs and encryptions.

$\mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := ([\hat{\boldsymbol{k}}^{i}]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}, T, [(x_j^i, T_j^i)]_{i\in[m],j\in[n]})$,
2. Output $[\mathsf{Decrypt}(\mathrm{SK}, \hat{k}_{n+1}^i)]_{i\in[m]}$.

Fig. 15: Extracting procedure Ext**.**

$\mathsf{D}'(\mathsf{crs}, [k^i]_{i\in[m]}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := ([\hat{\boldsymbol{k}}^{i}]_{i\in[m]}, , \overline{\boldsymbol{c}}, \overline{c}, T, [(x_j^i, T_j^i)]_{i\in[m],j\in[n]})$,
2. Check that V^{NI} outputs 1 on all proofs $[T_j^i]_{i\in[m],j\in[n]}, T$, relative to the corresponding CRS,
3. For $i \in [m]$, output $b^i := k^i \oplus c^i$.

Fig. 16: Alternate decoding procedure D'**, given additional extracted key** $[k^i]_{i\in[m]}$ **as input.**

$g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$:

1. Parse $\mathsf{CW} = ([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}, T, [(x_j^i, T_j^i)]_{i\in[m],j\in[n]})$, $\mathsf{CW}^* = ([\hat{\boldsymbol{k}}^{*i}]_{i\in[m]}, \overline{\boldsymbol{c}}^*, \overline{c}^*, T^*, [(x_j^{*i}, T_j^{*i})]_{i\in[m],j\in[n]})$.
2. If (1) V^{NI} outputs 1 on all proofs $T^*, [T_j^{*i})]_{i\in[m],j\in[n]}$, relative to the corresponding CRS; and (2) $([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) = ([\hat{\boldsymbol{k}}^{*i}]_{i\in[m]}, \overline{\boldsymbol{c}}^*, \overline{c}^*)$, then output 1. Otherwise output 0.

Fig. 17: The predicate $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$.

Let $\Psi(p, c, x, y, r, z)$ be defined as a function that takes as input a predicate p, and variables c, x, y, r, z. If $p(c, x, y, r) = 1$, then Ψ outputs the m-bit string $\mathbf{0}$. Otherwise, Ψ outputs z.

Theorem 4. *Let (E, D), E_1, E_2, Ext, D' and g be as defined in Figures 12, 13, 14, 15, 16 and 17. Let $\mathcal{F}$ be a computational class. If, for every pair of m-bit messages $\boldsymbol{b}_0, \boldsymbol{b}_1$ and if, for every adversary $\mathcal{A} \in \mathcal{G}$ outputting tampering functions $f \in \mathcal{F}$, all of the following hold:*

Simulation of proofs.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1]$,
2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_0); r_0)) \approx \Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1))$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_0, r_1 are sampled uniformly at random, $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, \boldsymbol{b}_0)$ and $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, \boldsymbol{b}_0)$.

Simulation of Encryptions.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$,
2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1)) \approx \Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2))$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_1, r_2 are sampled uniformly at random, $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, \boldsymbol{b}_0)$ and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, \boldsymbol{b}_0)$.

Simulation Soundness.

$$\Pr\left[\begin{array}{c}\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \\ \wedge g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0\end{array}\right] \leq \mathsf{negl}(n),$$

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2 is sampled uniformly at random and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, \boldsymbol{b}_0)$.

Hardness of $D_{\boldsymbol{b}}$ relative to Alternate Decoding.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$,
2. *For every Boolean function, represented by a circuit F over m variables, $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,*

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2, r_3 *are sampled uniformly at random,* $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, \boldsymbol{b}_0)$ *and* $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, \boldsymbol{b}_1)$.

Then the construction presented in Figure 12 is a non-malleable code for class $\mathcal{F}$ *against adversaries* $\mathcal{A} \in \mathcal{G}$.

We present the proof of theorem 4 in section 5.1 of the full version [9]

6 Efficient, Multi-Bit NMC for $\mathbf{AC^0}$

Theorem 5. $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *(presented in Figure 12) is an* m*-bit, computational, non-malleable code in the CRS model against tampering by depth-*$(m^{\log^\delta m}/2 - c)$ *circuits with unbounded fan-in and size* $\delta \cdot \frac{\log m}{\log \log m} - p(n)$ *(where* c *is constant and* $p(\cdot)$ *is a fixed polynomial), and* m *is such that* $n = m^{3+5\delta}$*, if the underlying components are instantiated in the following way:*

- $\mathcal{E} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is a public key encryption scheme with perfect correctness and decryption in* AC^0 .
- $\Pi^{\mathsf{NI}} := (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ *is a same-string, weak one-time simulation-sound NIZK with verifier in* AC^0 .
- *For* $b \in \{0,1\}$, D_b *is the distribution that samples bits* $x_1 \ldots x_n$ *uniformly at random, conditioned on* $x_1 \oplus \cdots \oplus x_n = b$.

For as in the one-bit case, given Theorem 1, proof systems Π^{NI} as above exist, under the assumption that same-string, weak one-time simulation-sound NIZK with (arbitrary polynomial-time) deterministic verifier exists. Refer to section 4 of the full version [9] for a discussion of how such NIZK and public key encryption can be instantiated. The proof the of the theorem 5, is presented as proof for Theorem 11 in [9], followed by the analysis for tampering with decision trees in section 6.1.

7 One-Bit NMC Against Streaming Adversaries

In this section, we show that our generic construction yields efficient *unconditional* NMC resilient against the tampering class $\mathcal{F}$ corresponding to streaming adversaries with memory $o(n'')$.

Let n be the parameter for the hard distribution described below, n' be the parameter for the semantically secure parity based encryption scheme against streaming adversaries with $o(n')$ storage (described in section 7.2 of [9]), and n'' be the parameter for the non-interactive simulatable proof system with streaming verifier (described in section 7.4 of [9]). Such that $n \in \omega(n'')$ and $n' \in \omega(n)$.

The Hard Distribution D_b (parameter n) Let $n = (\mu + 1)^2 - 1$. For $b \in \{0,1\}$, a draw from the distribution D_b is defined as follows: Choose a parity χ_S uniformly at random from the set of all (non-zero) parities over μ variables ($\emptyset \neq S \subseteq [\mu]$). Choose $y_1, \ldots, y_\mu \sim \{0,1\}^\mu$ uniformly at random. Choose y

uniformly at random, conditioned on $\chi_S(y) = b$. Output the following n-bit string: $[(y_i, \chi_S(y_i)]_{i\in[\mu]}||y$.

The proof of the hardness of D_b described above, along with the details of the parity-based encryption scheme, and non-interactive simulatable proof system with streaming verifier are described in sections 7.1, 7.2, and 7.4 of [9] respectively.

Theorem 6. *$\Pi = (\mathsf{E}, \mathsf{D})$ (presented in Figure 6) is a one-bit, unconditional non-malleable code against streaming adversaries with space $o(n'')$, if the underlying components are instantiated in the following way:*

- *$\mathcal{E} := (\mathsf{Encrypt}, \mathsf{Decrypt})$ is the parity based encryption scheme (with parameter $n' := n'(n)$).*
- *$\Pi^{\mathsf{NI}} := (\mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ the simulatable proof system with streaming verifier with parameter $n'' := n''(n)$.*
- *For $b \in \{0, 1\}$, D_b is the distribution described above (with paramter n).*

We wish to emphasize that no CRS or computational assumptions are needed for this result. Therefore, we can assume that the adversary $\mathcal{A}$ outputting tampering function f is computationally unbounded. Moreover, the result extends trivially for any number m of bits and all other parameters (n, n', n'') can remain the same and do not need to be increased. To see this, note that the only one additional property that needs to be proved in the multi-bit case (regarding hardness of $D_{\boldsymbol{b}}$ relative to alternate decoding in 4. But in the bounded, it can be achieved without requiring any additional memory beyond what is required in the one-bit case. We refer the interested readers to section 7.5 of [9] for further details.

Acknowledgments

We are grateful to Benjamin Kuykendall for his helpful comments.
The first and fourth authors are supported in part by the Defense Advanced Research Project Agency (DARPA) and Army Research Office (ARO) under Contract #W911NF-15-C-0236, and NSF grants #CNS-1445424 and #CCF-1423306 and the Leona M. & Harry B. Helmsley Charitable Trust. The second and third authors are supported in part by an NSF CAREER Award #CNS-1453045, by a research partnership award from Cisco and by financial assistance award 70NANB15H328 from the U.S. Department of Commerce, National Institute of Standards and Technology. This work was performed, in part, while the first author was visiting IDC Herzliya's FACT center and supported in part by ISF grant no. 1790/13 and the Check Point Institute for Information Security. Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the the Defense Advanced Research Projects Agency, Army Research Office, the National Science Foundation, or the U.S. Government.

References

1. Aggarwal, D., Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Optimal computational split-state non-malleable codes. [44] 393–417
2. Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: Non-malleable reductions and applications. In Servedio, R.A., Rubinfeld, R., eds.: 47th ACM STOC, ACM Press (June 2015) 459–468
3. Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. In Shmoys, D.B., ed.: 46th ACM STOC, ACM Press (May / June 2014) 774–783
4. Aggarwal, D., Dziembowski, S., Kazana, T., Obremski, M.: Leakage-resilient non-malleable codes. [28] 398–426
5. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Explicit non-malleable codes against bit-wise tampering and permutations. In Gennaro, R., Robshaw, M., eds.: Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Volume 9215 of Lecture Notes in Computer Science., Springer (2015) 538–557
6. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Explicit non-malleable codes against bit-wise tampering and permutations. In Gennaro, R., Robshaw, M.J.B., eds.: CRYPTO 2015, Part I. Volume 9215 of LNCS., Springer, Heidelberg (August 2015) 538–557
7. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. [28] 375–397
8. Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes for bounded depth, bounded fan-in circuits. In Fischlin, M., Coron, J.S., eds.: EUROCRYPT 2016, Part II. Volume 9666 of LNCS., Springer, Heidelberg (May 2016) 881–908
9. Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes from average-case hardness: AC0, decision trees, and streaming space-bounded tampering. Cryptology ePrint Archive, Report 2017/1061 (2017) http://eprint.iacr.org/2017/1061.
10. Bogdanov, A., Lee, C.H.: Homomorphic evaluation requires depth. [43] 365–371
11. Chabanne, H., Cohen, G.D., Flori, J., Patey, A.: Non-malleable codes from the wire-tap channel. CoRR **abs/1105.3879** (2011)
12. Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. Cryptology ePrint Archive, Report 2015/129 (2015) http://eprint.iacr.org/2015/129.
13. Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. In Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D., eds.: ICALP 2016. Volume 55 of LIPIcs., Schloss Dagstuhl (July 2016) 31:1–31:14
14. Chandran, N., Kanukurthi, B., Ostrovsky, R.: Locally updatable and locally decodable codes. [47] 489–514
15. Chandran, N., Kanukurthi, B., Raghuraman, S.: Information-theoretic local non-malleable codes and their applications. [44] 367–392
16. Chattopadhyay, E., Goyal, V., Li, X.: Non-malleable extractors and codes, with their many tampered extensions. [53] 285–298

17. Chattopadhyay, E., Li, X.: Non-malleable codes and extractors for small-depth circuits, and affine functions. In Hatami, H., McKenzie, P., King, V., eds.: 49th ACM STOC, ACM Press (June 2017) 1171–1184
18. Chattopadhyay, E., Zuckerman, D.: Non-malleable codes against constant split-state tampering. In: 55th FOCS, IEEE Computer Society Press (October 2014) 306–315
19. Chattopadhyay, E., Zuckerman, D.: Explicit two-source extractors and resilient functions. [53] 670–683
20. Cheraghchi, M., Guruswami, V.: Capacity of non-malleable codes. In Naor, M., ed.: ITCS 2014, ACM (January 2014) 155–168
21. Cheraghchi, M., Guruswami, V.: Non-malleable coding against bit-wise and split-state tampering. [47] 440–464
22. Choi, S.G., Kiayias, A., Malkin, T.: BiTR: Built-in tamper resilience. In Lee, D.H., Wang, X., eds.: ASIACRYPT 2011. Volume 7073 of LNCS., Springer, Heidelberg (December 2011) 740–758
23. Coretti, S., Dodis, Y., Tackmann, B., Venturi, D.: Non-malleable encryption: Simpler, shorter, stronger. [43] 306–335
24. Coretti, S., Maurer, U., Tackmann, B., Venturi, D.: From single-bit to multi-bit public-key encryption via non-malleable codes. [28] 532–560
25. Dachman-Soled, D., Kulkarni, M., Shahverdi, A.: Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. In Fehr, S., ed.: PKC 2017, Part I. Volume 10174 of LNCS., Springer, Heidelberg (March 2017) 310–332
26. Dachman-Soled, D., Liu, F.H., Shi, E., Zhou, H.S.: Locally decodable and updatable non-malleable codes and their applications. [28] 427–450
27. De Wolf, R.: A brief introduction to fourier analysis on the boolean cube. Theory of Computing, Graduate Surveys **1** (2008) 1–20
28. Dodis, Y., Nielsen, J.B., eds.: TCC 2015, Part I. In Dodis, Y., Nielsen, J.B., eds.: TCC 2015, Part I. Volume 9014 of LNCS., Springer, Heidelberg (March 2015)
29. Döttling, N., Nielsen, J.B., Obremski, M.: Information theoretic continuously non-malleable codes in the constant split-state model. Cryptology ePrint Archive, Report 2017/357 (2017) http://eprint.iacr.org/2017/357.
30. Dwork, C., Naor, M., Reingold, O.: Immunizing encryption schemes from decryption errors. In Cachin, C., Camenisch, J., eds.: EUROCRYPT 2004. Volume 3027 of LNCS., Springer, Heidelberg (May 2004) 342–360
31. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In Canetti, R., Garay, J.A., eds.: CRYPTO 2013, Part II. Volume 8043 of LNCS., Springer, Heidelberg (August 2013) 239–257
32. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In Yao, A.C.C., ed.: ICS 2010, Tsinghua University Press (January 2010) 434–452
33. Faust, S., Hostáková, K., Mukherjee, P., Venturi, D.: Non-malleable codes for space-bounded tampering. In Katz, J., Shacham, H., eds.: CRYPTO 2017, Part II. Volume 10402 of LNCS., Springer, Heidelberg (August 2017) 95–126
34. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. [47] 465–488
35. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: A tamper and leakage resilient von neumann architecture. In Katz, J., ed.: PKC 2015. Volume 9020 of LNCS., Springer, Heidelberg (March / April 2015) 579–603
36. Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Nguyen, P.Q., Oswald, E.,

eds.: EUROCRYPT 2014. Volume 8441 of LNCS., Springer, Heidelberg (May 2014) 111–128
37. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. [53] 1128–1141
38. Jafargholi, Z., Wichs, D.: Tamper detection and continuous non-malleable codes. [28] 451–480
39. Kalai, Y.T., Kanukurthi, B., Sahai, A.: Cryptography with tamperable and leaky memory. In Rogaway, P., ed.: CRYPTO 2011. Volume 6841 of LNCS., Springer, Heidelberg (August 2011) 373–390
40. Kanukurthi, B., Obbattu, S.L.B., Sekar, S.: Four-state non-malleable codes with explicit constant rate. In Kalai, Y., Reyzin, L., eds.: TCC 2017, Part II. Volume 10678 of LNCS., Springer, Heidelberg (November 2017) 344–375
41. Kanukurthi, B., Obbattu, S.L.B., Sekar, S.: Non-malleable randomness encoders and their applications. Cryptology ePrint Archive, Report 2017/1097 (2017) https://eprint.iacr.org/2017/1097.
42. Kiayias, A., Liu, F.H., Tselekounis, Y.: Practical non-malleable codes from l-more extractable hash functions. In Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S., eds.: ACM CCS 16, ACM Press (October 2016) 1317–1328
43. Kushilevitz, E., Malkin, T., eds.: TCC 2016-A, Part I. In Kushilevitz, E., Malkin, T., eds.: TCC 2016-A, Part I. Volume 9562 of LNCS., Springer, Heidelberg (January 2016)
44. Kushilevitz, E., Malkin, T., eds.: TCC 2016-A, Part II. In Kushilevitz, E., Malkin, T., eds.: TCC 2016-A, Part II. Volume 9563 of LNCS., Springer, Heidelberg (January 2016)
45. Li, X.: Improved two-source extractors, and affine extractors for polylogarithmic entropy. In Dinur, I., ed.: 57th FOCS, IEEE Computer Society Press (October 2016) 168–177
46. Lindell, Y.: A simpler construction of cca2-secure public-key encryption under general assumptions. In Biham, E., ed.: EUROCRYPT 2003. Volume 2656 of LNCS., Springer, Heidelberg (May 2003) 241–254
47. Lindell, Y., ed.: TCC 2014. In Lindell, Y., ed.: TCC 2014. Volume 8349 of LNCS., Springer, Heidelberg (February 2014)
48. Liu, F.H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In Safavi-Naini, R., Canetti, R., eds.: CRYPTO 2012. Volume 7417 of LNCS., Springer, Heidelberg (August 2012) 517–532
49. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC, ACM Press (May 1990) 427–437
50. Raz, R.: Fast learning requires good memory: A time-space lower bound for parity learning. CoRR **abs/1602.05161** (2016)
51. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS, IEEE Computer Society Press (October 1999) 543–553
52. Tal, A.: Tight bounds on the fourier spectrum of AC0. In O'Donnell, R., ed.: 32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia. Volume 79 of LIPIcs., Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017) 15:1–15:31
53. Wichs, D., Mansour, Y., eds.: 48th ACM STOC. In Wichs, D., Mansour, Y., eds.: 48th ACM STOC, ACM Press (June 2016)