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 t^ϵ, where t 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 [DPW10], 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 [CMTV15,CDTV16,GPR16] and theoretical computer science [CZ16]). Indeed, as suggested in the initial work of Dziembowski et al. [DPW10], 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. [DPW10,CG14a,FMVW14]). For efficient and explicit constructions, a large body of works construct NMC against bit-wise tampering (cf. [DPW10,CKM11,CCFP11]), and more generally split-state tampering (cf. [LL12,DKO13,ADL14,CG14a,CG14b,ADKO15a,AAG+16,CGL16,Li16,KOS17a,KOS17b]), where the adversary can tamper each part of the codeword independently of other parts, as well as NMC against permutations, flipping, and setting bits [AGM+15a].

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. [BDKM16] construct NMC against local tampering functions including NC^0, and Chattopadhyay and Li [CL17] 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. [FHMV17] against space-bounded tampering (in the random-oracle model), and by Chandran et al. [CGM+16] 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 $\varepsilon < 1$, (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 [NY90,Sah99,Lin03]. 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 [BL16], 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 [CL17] 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 m^ϵ, where m 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. [FHMV17] (who don't limit the adversary to be streaming), and the block-wise tampering class considered by Chandran et al. [CGM$^+$16] (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 [FHMV17] 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 [FMVW14] 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 [NY90] (later extended to achieve CCA2 [Sah99,Lin03]), using double encryption with simulation-sound NIZK. The main observation is that using the tableaua 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. [DNR04] on top of the scheme of Bogdanov and Lee [BL16]).

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

The CRS will contain a public key $\textsc{pk}$ for an encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as above, and a 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 $c_1 \leftarrow \mathsf{Encrypt}_{\textsc{pk}}(x_1), \ldots, c_n \leftarrow \mathsf{Encrypt}_{\textsc{pk}}(x_n)$ and $c \leftarrow \mathsf{Encrypt}_{\textsc{pk}}(b)$.
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 $\textsc{sk}$ (corresponding to the public key $\textsc{pk}$ 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 $\textsc{sk}$.

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, $\textsc{sk}$, 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 [DW08]).

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 [Tal17]. 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 [Raz16] 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 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 [DPW10]. Split state classes of tampering functions introduced by Liu and Lysyanskaya [LL12], have subsequently received much attention with a sequence of improvements achieving reduced number of states, improved rate, or other desirable features [DKO13,ADL14,CZ14,ADKO15a], [AGM+15c,AAG+16,KLT16,CGL16,Li16,KOS17a,KOS17b]. . Recently [AGM+15b,BDKM16] gave efficient constructions of non-malleable codes for "non-compartmentalized" tampering function classes.

Faust et.al [FMVW14] 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 [FMVW14] 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 [CG14a] 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 [CG14a] is inefficient for negligible error.

Ball et.al [BDKM16] 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 [BDKM16] 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 [CL17] 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 [BDKM16] has randomized decoding). The locality parameters of the NMC in [CL17] are not as good as the one in [BDKM16], but better than the deterministic-decoding construction given in the appendix of the full version of [BDKM16]. Additionally, [CL17] 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 [FHMV17] 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 [CGM+16] 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, [CGM+16] 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 [FMNV14,CG14b,CKO14,ADKO15b,JW15,CGM+15], [DLSZ15,FMNV15,ADKO15a,CKR16,KLT16,DSKS17,DNO17]. We guide the interested reader to [KKS11] and [LL12] 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 $u \in V_2$.

2.1 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 in Section 7 requires a randomized decoding algorithm, while our other instantiations enjoy deterministic decoding. We note that the original definition of non-malleable codes, given in [DPW10], required a deterministic decoding algorithm. Subsequently, in [BDKM16], an alternative definition that allows for randomized decoding was introduced. We follow here the definition of [BDKM16]. Please see [BDKM16] 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.

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, $m \in \Sigma$, 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. 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.

2.2 Public Key Encryption Scheme and PRG

In this section, we present the definitions of well known cryptographic primitives such as public key encryption scheme and pseudorandom generator which are used as building blocks for particular instantiations. A public key encryption scheme $\mathcal{E}$ consists of three algorithms: $(\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

- $\mathsf{Gen}(1^n) \to (\text{PK}, \text{SK})$. The key generation algorithm takes in the security parameter and outputs a public key PK and a secret key SK.
- $\mathsf{Encrypt}(\text{PK}, m) \to c$. The encryption algorithm takes in a public key PK and a message m. It outputs a ciphertext c.
- $\mathsf{Decrypt}(\text{SK}, c) \to m$. The decryption algorithm takes in a ciphertext c and a secret key SK. It outputs a message m.

Correctness. The PKE scheme satisfies correctness if $\mathsf{Decrypt}(\text{SK}, c) = m$ with all but negligible probability whenever PK, SK is produced by Gen and c is produced by $\mathsf{Encrypt}(\text{PK}, m)$.

Security. We define IND-CPA security for PKE schemes in terms of the following game between a challenger and an attacker. We let n denote the security parameter.

Setup Phase. The game begins with a setup phase. The challenger calls $\mathsf{Gen}(1^n)$ to create the initial secret key SK and public key PK.

Challenge Phase. The attacker receives PK from the challenger. The attacker chooses two messages m_0, m_1 which it gives to the challenger. The challenger chooses a random bit $b \in \{0, 1\}$, encrypts m_b, and gives the resulting ciphertext to the attacker. The attacker then outputs a guess b' for b. The attacker wins the game if $b = b'$. We define the advantage of the attacker in this game as $\left|\frac{1}{2} - \Pr[b' = b]\right|$.

Definition 5 (IND-CPA security). *We say a Public Key Encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is IND-CPA secure if any probabilistic polynomial time attacker only has a negligible advantage (negligible in n) in the above game.*

Definition 6 (α-correctness [DNR04]).
For any function $\alpha : \mathbb{N} \to [0, 1]$, a public-key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is α-correct *if* $\Pr[\mathsf{Decrypt}(\text{SK}, (\mathsf{Encrypt}(\text{PK}, m)) \neq m] \leq 1 - \alpha(n)$*, where the probability is taken over the random coins of* Gen *used to generate* $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^n)$*, for uniform random message* $m \in \{0, 1\}^n$*, and for all possible random coins of* $\mathsf{Encrypt}$.

Definition 7 (Almost-all-keys Perfect Decryption [DNR04]).
A public-key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is almost-all-keys perfectly correct *if with all but negligible probability over the random coins of* Gen *used to generate* $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^n)$*, for uniform random message* $m \in \{0, 1\}^n$*, and for all possible random coins of* $\mathsf{Encrypt}$*, it holds that* $\Pr[\mathsf{Decrypt}(\text{SK}, (\mathsf{Encrypt}(\text{PK}, m)) \neq m] = 0$.

Definition 8 (Pseudorandom Generator). *A* pseudorandom generator *is an efficient, deterministic map* $\mathsf{prg} : \{0, 1\}^n \to \{0, 1\}^{\ell(n)}$*, where* $\ell(n) > n$ *such that for all PPT distinguishers D;* $|\Pr[D(G(x)) = 1] - \Pr[D(y) = 1]| \leq \mathsf{negl}(n)$*, when* $x \in \{0, 1\}^n$ *and* $y \in \{0, 1\}^{\ell(n)}$ *are chosen uniform randomly.*

2.3 Non-Interactive Zero Knowledge

Another important cryptographic primitive used in the constructions of non-malleable codes in this paper is non-interactive zero knowledge proof systems.

Definition 9 (Non-Interactive Zero Knowledge [Sah99]). *$\Pi = (\ell, \mathsf{P}, \mathsf{V}, \mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2))$ is an* efficient adaptive single-theorem non-interactive zero knowledge proof system *for language $L \in \mathsf{NP}$ with witness relation W, if ℓ is a polynomial and the following are true:*

- *Completeness: For all $x \in L$, and all w such that $W(x, w) = 1$, for all strings crs of length $\ell(|x|)$, we have $\mathsf{V}(x, \mathsf{P}(x, w, \mathsf{crs}), \mathsf{crs})) = 1$*
- *Soundness: For all adversaries A, if $\mathsf{crs} \in \{0,1\}^{\ell(k)}$ is chosen randomly, then $\Pr[\mathsf{V}(x, \pi, \mathsf{crs}) = 1] \leq \mathsf{negl}(k)$. Where, $(x, \pi) \leftarrow A(\mathsf{crs})$ and $x \notin L$.*
- *Single-Theorem Zero Knowledge: For all non-uniform polynomial-time adversaries $A = (A_1, A_2)$ we have that $|\Pr[\mathsf{Expt}_A(k) = 1] - \Pr[\mathsf{Expt}_A^{\mathsf{Sim}}(k) = 1]| \leq \mathsf{negl}(k)$ for following experiments $\mathsf{Expt}_A(k)$ and $\mathsf{Expt}_A^{\mathsf{Sim}}(k)$*

```
$\mathsf{Expt}_A(k)$:                          $\mathsf{Expt}_A^{\mathsf{Sim}}(k)$:

$\mathsf{crs} \leftarrow \{0,1\}^{\ell(k)}$              $(\mathsf{crs}, \kappa) \leftarrow \mathsf{Sim}_1(1^k)$
$(x, w, \tau) \leftarrow A_1(\mathsf{crs})$              $(x, w, \tau) \leftarrow A_1(\mathsf{crs})$
$\pi \leftarrow \mathsf{P}(x, w, \mathsf{crs})$               $\pi \leftarrow \mathsf{Sim}_2(x, \kappa)$
$\mathsf{return} A_2(\pi, \tau)$                 $\mathsf{return} A_2(\pi, \tau)$
```

Definition 10 (Weak Simulation Soundness [Sah99]). *Let $\Pi = (\ell, \mathsf{P}, \mathsf{V}, \mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2))$ be an efficient adaptive single-theorem non-interactive zero knowledge proof system for language L. We say that Π is* simulation-sound *if for all non-uniform probabilistic polynomial-time adversaries $A = (A_1, A_2)$, $\Pr[\mathsf{Expt}_{A,\Pi}^{\mathsf{Sim}}(k) = 1] \leq \mathsf{negl}(k)$, where $\mathsf{Expt}_{A,\Pi}^{\mathsf{Sim}}(k)$ is the following experiment:*

```
$\mathsf{Expt}_{A,\Pi}^{\mathsf{Sim}}(k)$:

$(\mathsf{crs}, \kappa) \leftarrow \mathsf{Sim}_1(1^k)$
$(x, \tau) \leftarrow A_1(\mathsf{crs})$
$\pi \leftarrow \mathsf{Sim}_2(x, \mathsf{crs}, \kappa)$
$(x^*, \pi^*) \leftarrow A_2(x, \pi, \mathsf{crs}, \tau)$
Output 1 iff $(\pi^* \neq \pi)$ and $(x^* \notin L)$ and $(\mathsf{V}(x^*, \pi^*, \mathsf{crs}) = 1)$
```

We say Π is one-time weak simulation sound *if the above holds for any probabilistic polynomial time A only allowed a single query to Sim.*

Sahai [Sah99] constructed one-time simulation sound NIZK proof system from any given efficient non-interactive single-theorem adaptive zero knowledge proof system, and strong one-time signature schemes (which was built from one-way functions in the same work).

Definition 11 (Same-String NIZK [DDO+01]). *A NIZK argument system is called same-string NIZK if it satisfies the following property for all k:*

- ***(Same-String Zero Knowledge):*** *For all non-uniform probabilistic polynomial-time adversaries A, we have that*
$$|\Pr[X = 1] - \Pr[Y = 1]| \leq \mathsf{negl}(k)$$
where X and Y are as defined in (and all probabilities are taken over) the experiment $\mathsf{Expt}(k)$ below: where $\mathsf{Sim}'(x, w, \mathsf{crs}, \tau) \overset{\text{def}}{=} \mathsf{Sim}_2(x, \mathsf{crs}, \tau)$

```
Expt$(k)$:

  1. $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^k)$
  2. $X \leftarrow A^{\mathsf{P}(\cdot,\cdot,\mathsf{crs})}(\mathsf{crs})$
  3. $Y \leftarrow A^{\mathsf{Sim}'(\cdot,\cdot,\mathsf{crs},\tau)}(\mathsf{crs})$
```

– ***(Same-String Zero Knowledge,cont.):*** *The distribution on* crs *produced by* $\mathsf{Sim}_1(1^k)$ *is the uniform distribution over* $\{0,1\}^{\ell(k)}$.

Definition 12 (Non-Interactive Simulatable Proof System). *A tuple of probabilistic polynomial time algorithms* $\mathit{\Pi}^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ *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*

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

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

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

The security properties that we require of $\mathit{\Pi}^{\mathsf{NI}}$ *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 4 and 10. In subsequent sections, we will show how to construct non-interactive simulatable proof systems satisfying these properties.*

2.4 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 13 (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* $\{\mathcal{L}(C) \mid C \in \mathcal{C}\}$. $\mathit{\Pi} = (\mathsf{P}, \mathsf{V})$ *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 $\mathcal{L}(\mathsf{P/poly})$ with verifier complexity AC^0 in this section. We also construct Circuit SAT proof systems for the class. $\mathcal{L}(\mathsf{P/poly})$ with streaming verifier

Circuit SAT proof system for the class $\mathcal{L}(\mathcal{C})$ with prover complexity $\mathcal{D}$ and verifier complexity AC^0 .

- $\mathsf{P}(C, x, w)$ the prover simply outputs a tableau T of the computation $C(x, w) = 1$.
- $\mathsf{V}(C, x, T)$ the verifier computes an AND of all the local checks.

Completeness clearly holds. To show extractability, note that the inputs to the tableau T correspond to x, w. Thus if tableau T accepts then the extractor can simply output those inputs corresponding to w.

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 .

Circuit SAT proof system for the class $\mathcal{L}(\mathcal{C})$ with prover complexity $\mathcal{D}$ and streaming verifier.

- $\mathsf{P}(C, x, w)$ the prover computes a tableau T of the computation $C(x, w) = 1$. Let d denote the depth of the tableau T. For each level $i \in [d]$, the i-th level, T_i, consisting of ℓ gates is the following ordered tuple:

$$[(G_i^j, in_i^{j,a}, in_i^{j,b}, out_i^j)]_{j \in \ell},$$

  where G_i^j denotes the j-th gate at that level, $(in_i^{j,a}, in_i^{j,b})$ denote the j-th pair of input wires at that level and out_i^j denotes the j-th output wire at that level. For simplicity of notation, we assume that the input wires to the first level, T_1 consist only of x, and that wires corresponding to the input w will occur as outputs of level T_1. The P outputs $(T_1, \dots, T_d)$.
- $\mathsf{V}(C, x, T_1, \dots, T_d)$ the verifier chooses k at random and computes $h_0 = h_k(x)$, where h is a universal hash function. For each level $i \in [d]$, the verifier then does the following:
  - Parse $T_i = [(G_i^j, in_i^{j,a}, in_i^{j,b}, out_i^j)]_{j \in \ell}$.
  - For $j \in [\ell]$, (1) Check consistency of the gate's computation, (2) Add $(in_i^{j,a}, in_i^{j,b})$ to the streaming computation of the hash $h_k([(in_i^{j,a}, in_i^{j,b})]_{j \in \ell})$, (3) Add out_i^j to the streaming computation of the hash $h_k([out_i^j]_{j \in \ell})$.
  - Check that $h_k([(in_i^{j,a}, in_i^{j,b})]_{j \in \ell}) = h_{i-1}$.
  - Set $h_i := h_k([out_i^j]_{j \in \ell})$.
  - If any of the above checks fail, abort and output 0.

  If all checks succeed, the verifier outputs 1.

Completeness clearly holds. To show extractability, note that the only way the inputs/outputs of level T_1 do not correspond to x, w such that $C(x, w) = 1$ and yet all checks pass is if the proof ouputted by the prover consists of consecutive levels T_i, T_{i+1} such that $h_k([(in_{i+1}^{j,a}, in_{i+1}^{j,b})]_{j \in \ell}) = h_k([out_i^j]_{j \in 2\ell})$ but $[(in_{i+1}^{j,a}, in_{i+1}^{j,b})]_{j \in \ell} \neq [out_i^j]_{j \in 2\ell}$. The probability over choice of k that this occurs for a single pair of consecutive levels is $1/2^{2\ell}$, since h is universal. So the probability it occurs for any pair of consecutive levels is at most $d/2^{2\ell}$, which is negligible.

We also recall some definitions and results related to boolean analysis and present them next.

2.5 Definitons related to Boolean Analysis

Definition 14. *A function $f : \{0,1\}^n \to \{0,1\}$ has* correlation c *with a function $g : \{0,1\}^n \to \{0,1\}$ if*

$$|\Pr_{x \leftarrow U_n}[f(x) = 1 | g(x) = 1|] - \Pr_{x \leftarrow U_n}[f(x) = 1 | g(x) = 0|]| \leq c.$$

Where, U_n is the uniform random distribution over $\{0,1\}^n$.

Note that this is equivalent up to absolute value for a more common definition of correlation in the literature when g is taken to be balanced ($\Pr[g(x) = 1] = 1/2$)

Definition 15. *A function $f : \{0,1\}^n \to \{0,1\}$ has correlation c with a function $g : \{0,1\}^n \to \{0,1\}$ if*

$$\Pr_{x \leftarrow U_n}[f(x) = g(x)] = \frac{1+c}{2}.$$

Where, U_n is the uniform random distribution over $\{0,1\}^n$.

The correlation c can also be expressed as follows: Let $\Pr[f(x) = g(x)] = \frac{1+c}{2}$. Then,

$$\begin{aligned} c &= 2\Pr[f(x) = g(x)]] - 1 \\ &= 2\Pr[f(x) = 1|g(x) = 1]\Pr[g(x) = 1] + 2\Pr[f(x) = 0|g(x) = 0]\Pr[g(x) = 0] - 1 \\ &= \Pr[f(x) = 1|g(x) = 1] + (1 - \Pr[f(x) = 1|g(x) = 0] - 1 \\ &= \Pr[f(x) = 1|g(x) = 1] - \Pr[f(x) = 1|g(x) = 0] \end{aligned}$$

Theorem 2 **([Hås14,IMP12]).** *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a depth-d circuit of size S. Then the correlation of f with parity is bounded by*

$$2^{-c_d n/\log^{d-1}(S)},$$

where c_d is a positive constant dependent only on d.

Definition 16 **(Random Restriction [Hås87]).** *A* random restriction *ρ parameterized by a small positive real number p is mapping which sets the elements x_i of a vector $\boldsymbol{x}$ independently as follows:* $\Pr[x_i = 0] = \frac{1-p}{2}$, $\Pr[x_i = 1] = \frac{1-p}{2}$, *and* $\Pr[x_i = \star] = p$.

Lemma 1 **([Hås87,Vio14]).** *Let $f : \{0,1\}^\ell \to \{0,1\}$ be a function computable by a depth-d AC^0 circuit of size s. Let ρ be a random restriction with $\Pr[\star] = q < 1/9^d$. The probability over ρ that f_ρ cannot be written as a decision tree of depth t is $\leq s(9q^{1/d}t)^t$.*

Theorem 3 **([Raz16]).** *For any $c < \frac{1}{20}$, there exists $\alpha > 0$, such that the the following holds: Let $x \overset{u}{\leftarrow} \{0,1\}^n$. Let $m \leq 2^{\alpha n}$. Let A be an algorithm that is given as input a stream of samples, $(a_1, b_1), \ldots, (a_m, b_m)$, where each a_t is uniformly distributed over $\{0,1\}^n$ and for every t, $b_t = a_t \cdot x$. Assume A uses at most cn^2 memory bits and outputs a string $\tilde{x} \in \{0,1\}^n$. Then, $\Pr[\tilde{x} = x] \leq O(2^{-\alpha n})$.*

Lemma 2 **(Inner Product is a strong extractor [Rao07]).** *Let X, Y be random variables over $\{0,1\}^n$ such that $H_\infty(X) \geq k_X$ and $H_\infty(Y) \geq k_Y$. Let $u \leq k_X$*

$$d(\langle X, Y\rangle | X : U) \leq 2(2^{u-k_X} + 2^{(n+1-u-k_Y)/2}),$$

where $d(X|Y) : \sum_y \Pr[Y = y]\Delta(X|Y = y; U)$ for U the uniform distribution (independent of X).

2.6 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 [Raz16].

We first discuss streaming adversaries in general, and then discuss the specific case of streaming adversaries for learning parity and streaming tampering functions..

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^{n+1} 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 next.

Definition 17 (Streaming Branching Program Adversaries). *A branching program of length m and width w is a directed acyclic graph with vertices arranged in $m+1$ layers containing at most w vertices each. In the first layer, that we call layer 0, there is only one vertex, called the start vertex. A vertex of out-degree 0 is called a leaf. All the vertices in the layer m are leaves. Every non-leaf vertex in the program has exactly 2^{n+1} 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.*

Computation Path: *The stream $S_1, \ldots, S_\ell \in \{0,1\}^B$ that are given as input, define a computation-path in the branching program, by starting form the start vertex and following at step i the edge labeled by S_i, until reaching a leaf.*

Streaming Adversaries for Learning Parity. Recall, that in the Parity Learning setting, the adversary aims to learn a uniform random string $x \in \{0,1\}^n$, from a stream of samples, $(a_1,b_1),(a_2,b_2),\ldots,(a_m,b_m)$, where each a_i is uniformly distributed over $\{0,1\}^n$ and for every i, $b_i = a_i \cdot x$.

Definition 18 (Streaming Branching Program for Parity Learning). *[Raz16] A branching program of length m and width w, for parity learning is a directed acyclic graph with vertices arranged in $m+1$ layers containing at most w vertices each. In the first layer, that we call layer 0, there is only one vertex, called the start vertex. A vertex of out-degree 0 is called a leaf. All the vertices in the layer m are leaves. Every non-leaf vertex in the program has exactly 2^{n+1} outgoing edges, labeled by elements $(a,b) \in \{0,1\}^n \times \{0,1\}$, with exactly one edge labeled by each such (a,b), and all the edges from layer $j-1$ going to vertices in layer j.*

Computation Path: *The samples $(a_1,b_1),(a_2,b_2),\ldots,(a_m,b_m) \in \{0,1\}^n \times \{0,1\}$ that are given as input, define a computation-path in the branching program, by starting form the start vertex and following at step i the edge labeled by (a_i,b_i), until reaching a leaf.*

Streaming Tampering Functions. 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*, which *reads in a block of length B* and *writes out a block of length B* in each step. 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^{n+1} outgoing edges, labeled by pairs of elements $S_{\mathsf{in}}, S_{\mathsf{out}} \in \{0,1\}^B$, with exactly one edge labeled by each such S_{in}, 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, as well as the output in that time step. The stream $S_1, \ldots, S_\ell$, therefore, define a computation-path in the branching program.

Definition 19 (Streaming Tampering Functions). *A branching program of length m and width w is a directed acyclic graph with vertices arranged in $m+1$ layers containing at most w vertices each. In the first layer, that we call layer 0, there is only one vertex, called the start vertex. A vertex of out-degree 0 is called a leaf. All the vertices in the layer m are leaves. Every non-leaf vertex in the program has exactly 2^{n+1} outgoing edges, labeled by pairs of elements $S_{\mathsf{in}}, S_{\mathsf{out}} \in \{0,1\}^B$, with exactly one edge labeled by each such S_{in}, and all the edges from layer $j-1$ going to vertices in layer j.*

Computation Path: *The stream $S_1, \ldots, S_\ell \in \{0,1\}^B$ that are given as input, define a computation-path in the branching program, by starting form the start vertex and following at step i the edge labeled by S_i, until reaching a leaf.*

In this work we consider the *Polynomial-time uniform* family of branching programs which can be informally defined as follows:

A family of branching programs of size s (number of nodes in the branching program), denoted by $\mathsf{BP} = \{\mathsf{BP}_s : s \in \mathbb{N}\}$ is *Polynomial-time uniform* if there exists a deterministic Turing machine M, such that

- M runs in polynomial time (i.e. $\text{poly}(s)$), and
- For all $s \in \mathbb{N}$, M outputs the description (nodes and corresponding labels) of BP_s on input 1^s

3 Generic Construction for One-Bit Messages

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

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a public key encryption scheme with perfect correctness (see Definition 7). 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 12). 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.

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

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_{sim}^i and r to simulate a non-interactive proof T_i' proving $(\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \in \mathcal{L}_i^{x_i}$, relative to $\mathsf{crs}_i^{\mathsf{NI}}$.
6. Use τ_{sim}^0 and r to simulate a non-interactive proof T' proving $(\hat{\boldsymbol{k}}, \boldsymbol{c}, c) \in \mathcal{L}$, relative to $\mathsf{crs}_0^{\mathsf{NI}}$.
7. Output $\mathsf{CW} := (\hat{\boldsymbol{k}}, c_1, \ldots, c_n, c, T', x_1, T_1', .., x_n, T_n')$.

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 4. *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 $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$*,* $f \leftarrow \mathcal{A}(\mathsf{crs})$*,* 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 $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$*,* $f \leftarrow \mathcal{A}(\mathsf{crs})$*,* 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.

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

where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$*,* $f \leftarrow \mathcal{A}(\mathsf{crs})$*,* 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. $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,
where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$*,* $f \leftarrow \mathcal{A}(\mathsf{crs})$*,* 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* $\mathcal{A} \in \mathcal{G}$.

3.1 Proof of Theorem 4

In this subsection we prove Theorem 4.

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
$\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$. 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 4.

Claim. Hybrid 1 is computationally indistinguishable from Hybrid 2.

Proof. The claim follows immediately from the **Simulation of Encryptions** property in Theorem 4.

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 4.

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
$\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma)$ always outputs 0 and so

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

and (2) the **Hardness of D_b relative to Alternate Decoding** property in Theorem 4.

4 One-Bit NMC for AC^0 and beyond

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 5. $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *(presented in Figure 6) is a one-bit, computational, non-malleable code in the CRS model, secure against tampering by* AC^0 *circuits, 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 [Sah99]. 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. [DNR04] to the (reduced decryption error) encryption scheme of Bogdanov and Lee [BL16]. We now provide an instantiation of the public key encryption scheme.

Public key encryption in AC^0 . We now present the result presented by Bogdanov and Lee in [BL16] which showed that the encryption scheme given by Applebaum et al. in [ABW10] can be implemented by circuit with constant depth and size polynomial in the security parameter.

PKE Scheme based on Bipartite Graphs [ABW10]

- $\mathsf{Gen}(1^n)$: The key generation algorithm takes security parameter n as input and outputs a random bipartite graph $G = ((U_1, U_2), E)$ as the public key PK, where $|U_1| = n$ and $|U_2| = r = n^{0.9}$ generated in the following way. First choose the random subsets $S_1 \subseteq U_1$ and $S_2 \subseteq U_2$ of sizes s and $s/3$ respectively for $s = O(\log n)$. Each vertex in S_1 is connected to d (possibly repeated) random vertices in S_2 and each vertex outside S_1 is connected to d random vertices in U_2. The secret key SK is an odd size subset of S_1 such that each vertex in S_2 has an even number of neighbors in SK.
- $\mathsf{Encrypt}(\text{PK}, b)$: To encrypt bit $b \in \{0,1\}$, choose a random subset $S'_2 \subset U_2$ and output $\boldsymbol{c} = \boldsymbol{y} + \boldsymbol{e} + b \cdot \mathbf{1}$, where each coordinate of $\boldsymbol{y} \in \{0,1\}^n$ is the degree of corresponding vertex in S_1 restricted to $S'_2 \bmod 2$, $\boldsymbol{e} \in \{0,1\}^n$ is a vector with each coordinate $(e_i : i \in [n])$ sampled from distribution $\hat{\eta}$ with $\Pr[e_i = 0] = \eta$ independently, and $\mathbf{1} \in \{0,1\}^n$ is the vector of all 1s.
- $\mathsf{Decrypt}(\text{SK}, \boldsymbol{c})$: Output $b = \sum_{i \in \text{SK}} c_i \bmod 2$.

Refer [ABW10] for the security of the scheme presented above. We next present the AC^0 implementation of the PKE presented above as shown in [BL16].

AC^0 *Implementation of [ABW10] PKE Scheme based on Bipartite Graphs* [BL16]

- Gen:
  1. Sample $y_1, y_2, \dots, y_s$ from $[n]$ and $w_1, w_2, \dots, w_{s/3}$ from $[r]$ to represent the subsets $S_1 \subseteq U_1$ and $S_2 \subseteq U_2$ respectively.
  2. Sample $v_{i,1}, v_{i,2}, \dots, v_{i,d}$ from $[r]$ for all $i \in [n]$. These represent the random neighbors of each vertex in $U_1 \setminus S_1$.
  3. Sample $\hat{v}_{i,1}, \hat{v}_{i,2}, \dots \hat{v}_{i,d}$ from $[s/3]$ for all $i \in [s]$. These become the random neighbors of the vertices in S_1 after being mapped to the w_i's by the index function $\iota : [s/3] \to [r]$ such that $\iota(i) = w_i$. This is written as:
$$\iota(i) = \bigvee_{j=1}^{s/3} [(i = j) \wedge w_j]$$
.

The key generation circuit outputs $v_{i,1}, v_{i,2}, \ldots, v_{i,d}$ if the vertex i is not in S_1 and outputs $\iota(\hat{v}_{i,1}), \iota(\hat{v}_{i,2}), \ldots \iota(\hat{v}_{i,d})$ otherwise. Now we can output the j^{th} random neighbor of each vertex $i \in U_1$ as

$$\left[\delta_i \wedge \bigvee_{k=1}^{s} [(i = k) \wedge \iota(\hat{v}_{k,j})]\right] \vee (\bar{\delta_i} \wedge v_{i,j}),$$

where $\delta_i := \bigvee_{k=1}^{s}(i = y_k)$ indicates whether i belongs to S_1.
To come up with secret key SK, we enumerate all the possible subsets of S_1 (this is still efficient since $s = O(\log n)$) and output the first one that satisfies the linear dependency. Given an odd size subset of S_1 indicated by the support of the vector $\boldsymbol{a} \in \{0,1\}^s$, note that the formula

$$f_{\boldsymbol{a}} = \bigvee_{j=1}^{s/3} \bigoplus_{i:a_i=1} \bigoplus_{k=1}^{d} (\hat{v}_{i,k} = j)$$

outputs 0 only if every vertex in S_2 has an even number of neighbors in support of $\boldsymbol{a}$ and outputs 1 otherwise. (Since the XOR involves only $O(d \log n)$ inputs it can be calculated with a circuit of depth 2 and size $n^{O(d)}$.) We can therefore, enumerate all the possible $\boldsymbol{a} \in \{0,1\}^s$ with odd hamming weight and output the first $\boldsymbol{a}$ such that $f_{\boldsymbol{a}} = 0$. The secret key is represented by a vector $\boldsymbol{z}$ containing s entries in $[n]$, where each non-zero entry corresponds to a vertex in SK. More precisely, we output i^{th} entry as

$$z_i = \iota\left(\bigvee_{\boldsymbol{a} \in \{0,1\}^s :\, wt(\boldsymbol{a}) \text{ is odd}} \left[\bar{f_{\boldsymbol{a}}} \wedge \left(\bigwedge_{\boldsymbol{a'} < \boldsymbol{a}} f_{\boldsymbol{a'}}\right) \wedge (a_i \wedge i)\right]\right).$$

– Encrypt: Given a public key represented by the neighbors $v_{i,1}, v_{i,2}, \ldots, v_{i,d}$ of each vertex $i \in U_1$. To encrypt bit $b \in \{0,1\}$, choose a random vector $\boldsymbol{x} \in \{0,1\}^r$ whose support forms the subset S'_2 of U_2, a noise vector $\boldsymbol{e} \in \{0,1\}^n$ by choosing each of its entries independently from $\hat{\eta}$. The i^{th} bit of ciphertext can then be written as

$$\bigvee_{k_i \neq k_j; 1 \leq i \leq j \leq d; k_i \in [r]; a_1, \ldots, a_d : a_1 + \cdots + a_d = 1 \bmod 2} \left[\bigwedge_{j=1}^{d} (v_{i,j} = k_j) \wedge (x_{k_1} = a_1) \wedge \cdots \wedge (x_{k_d} = a_d)\right] \oplus e_i \oplus b$$

– Decrypt: Given the ciphertext $\boldsymbol{c}$ and secret key SK represented by the vector $\boldsymbol{z} \in \{0,1\}^{s \times \log n}$, output

$$\bigoplus_{i=1}^{s} \bigvee_{k=1}^{n} [(z_i = k) \wedge c_k].$$

Reducing the decryption error The [ABW10] encryption scheme suffers from significant encryption error (and thus decryption error) however, this can be minimized arbitrarily by encrypting the message multiple times independently. The decryption algorithm can then take approximate majority to compute the encrypted bit. Approximate majority can be computed with constant depth circuits [Ajt83] (depth 3) and thus the overall decryption algorithm is still in AC^0 .

We now use the following transformation given by [DNR04] to obtain almost-all keys perfect decryption for the above encryption scheme.

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be any public-key encryption scheme. Also let $\ell(n) > n$ be the number of bits used by $\mathsf{Encrypt}$ to encrypt n-bit messages. Let prg be a pseudorandom generator that expands n bits to $\ell(n)$ bits. Then the modified encryption scheme $\mathcal{E}' = (\mathsf{Gen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ is obtained as follows: On input 1^n, Gen' outputs $((\text{PK}, \bar{r}), \text{SK})$ where $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^n)$ and $\bar{r} \in \{0,1\}^{\ell(n)}$ is chosen uniform randomly. To encrypt message m, $\mathsf{Encrypt}'$ samples a random n-bit string r and outputs $\mathsf{Encrypt}(\text{PK}, m)$ using $\mathsf{prg}(r) \oplus \bar{r}$ as randomness for $\mathsf{Encrypt}$. $\mathsf{Decrypt}'$ is same as $\mathsf{Decrypt}$, note that this preserves the computational complexity of decryption.

Theorem 6. *[DNR04] Let* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be any* $(1 - 2^{-4n})$ *correct public key encryption scheme with* $\mathsf{Decrypt}$ *being deterministic. Then* $\mathcal{E}' = (\mathsf{Gen}', \mathsf{Encrypt}', \mathsf{Decrypt})$ *is an almost-all-key perfectly correct public encryption scheme. Furthermore, if* $\mathcal{E}$ *is IND-CPA secure then so is* $\mathcal{E}'$*.*

Note that the above transformation takes a public key encryption scheme $\mathcal{E}$ with sufficiently low decryption error and transforms it into a public key encryption scheme that enjoys perfect correctness, and furthermore, note that the decryption algorithm $\mathsf{Decrypt}$ remains unchanged. Therefore, if we start with the (reduced decryption error) version of the AC^0 Bogdanov and Lee public key encryption scheme, we obtain a perfectly correct public key encryption scheme with decryption in AC^0 , as desired.

Proof (Proof of theorem 5). 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 4 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 $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, 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 $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, 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.**

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

  where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, 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 $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ output 0. This is contradiction to the claim that $\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$. In case (2), the extractor $\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2))$ outputs $k_{n+1}^* := \mathsf{Decrypt}(\textsc{sk}, \hat{k^*}_{n+1})$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ outputs $b^* = c^* \oplus k_{n+1}^*$. Now, if $\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ 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. $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,

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, which says that an AC^0 circuit has at most negligible correlation with parity of its input $\boldsymbol{X}$, denoted $\mathcal{P}(\boldsymbol{X})$. Details follow.

We construct the distribution of circuits $\mathcal{C}^1_{\mathcal{F}}$. A draw $C \sim \mathcal{C}^1_{\mathcal{F}}$ is done as follows:

1. Sample $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.
2. Sample tampering function $\mathcal{A}(\mathsf{crs}) \rightarrow f$.
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, which says that an AC^0 circuit has at most negligible correlation with the parity of its input $\boldsymbol{X}$, denoted $\mathcal{P}(\boldsymbol{X})$. Details follow.

We construct the distribution of circuits $\mathcal{C}^2_{\mathcal{F}}$. A draw $C \sim \mathcal{C}^2_{\mathcal{F}}$ is done as follows:

1. Sample $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.
2. Sample tampering function $\mathcal{A}(\mathsf{crs}) \rightarrow f$.
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 .

We present the analysis for more tampering classes next.

4.1 Tampering classes beyond AC^0 .

Let $\mathcal{F} \subsetneq P$ be a tampering class. Relative to this class $\mathcal{F}$, define the circuit classes $\mathcal{C}^1_{\mathcal{F}}$ and $\mathcal{C}^2_{\mathcal{F}}$ as in the proof above.

Theorem 7. *Let $\{\mathcal{D}_0, \mathcal{D}_1\}$ be (probabilistic polynomial time) samplable distributions with disjoint support. If the following hold:*

- *There exists a ppt distinguishing algorithm D such that for $b \in \{0,1\}$,*
$$\Pr_{\boldsymbol{x} \sim \mathcal{D}_b}[D(\boldsymbol{x}) = b] = 1.$$
- *For all $C \in \mathcal{C}^1_{\mathcal{F}} \cup \mathcal{C}^2_{\mathcal{F}}$*
$$\left| \Pr_{\boldsymbol{x} \sim \mathcal{D}_0}[C(\boldsymbol{x}) = 1] - \Pr_{\boldsymbol{x} \sim \mathcal{D}_1}[C(\boldsymbol{x}) = 1] \right| \leq \mathsf{negl}(n).$$

Then, under the same assumptions as Theorem 5, $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is a computational non-malleable code against tampering by $\mathcal{F}$ that encodes a single bit.

We informally argue that Theorem 8 yields non-malleable codes against new classes: small threshold circuits and time-bounded probabilistic RAM machines. As noted earlier, non-malleable codes (in the CRS model without computational assumptions) from [FMVW14] are resilient against these classes. We provide theorems simply to demonstrate the applicability of our framework to a broad class of correlation bounds.

Theorem 8 ([CSS16]). *For all d there exists $\epsilon_d > 0$ such that the following holds. There exists a probabilistic polynomial time computable f (the Generalized Andreev Function) such that for any depth-d threshold circuit with $n^{1+\epsilon_d}$ wires, C, f has correlation at most $2^{-n^{\Omega(1)}}$ with C.*

Corollary 1. *Let f be as in Theorem 8. Fix x_0, x_1 such that $f(x_b) = b$. Let D_b define a variable, X, which is defined by rejection sampling the uniform distribution over $\{0,1\}^n$ conditioned on $f(X) = b$; if after $O(n)$ tries the rejection sampling has not succeeded, output x_b.*

Then, assuming PKE in AC^0 and same-string weak one-time simulation-simulation sound NIZK, there exists a constant d_0 such that for $d > d_0$, $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is a computational non-malleable code against depth-d threshold circuits with $n^{1+\epsilon'_d}$ wires, where ϵ'_d is any positive constant less than ϵ_d from Theorem 8.

The corollary follows from the fact that given the appropriate choice of security parameters for the encryption scheme and NIZK, any $C \in \mathcal{C}^1_{\mathcal{F}} \cup \mathcal{C}^2_{\mathcal{F}}$ has a representation as depth-d threshold circuit with $1 + \epsilon_d$ wires, so long as d is large enough, and the fact that rejection sampling fails with very low probability as f is balanced.

Using a generalization of a Theorem from [BRSV17] (combined with a result on prime finding from [OS17]):

Theorem 9 ([BRSV17]). *Let k be an integer (constant). Assuming one of the following:*

1. *a randomized variant of the Strong Exponential Time Hypothesis (BPSETH): $\forall \epsilon > 0, \exists q$ such that no randomized algorithm running in time $O(2^{1-\epsilon)n})$ is correct with probability $> 2/3$ on* every *instance of qSAT.*
2. *the randomized k-Orthogonal Vector Conjecture (BPkOVC): the k-Orthogonal Vector problem requires time $\Omega(n^{k-o(1)})$ for randomized algorithms that are correct with probability $> 2/3$ on* every *instance.*
   k-Othogonal Vector is a generalization of the well-studied Orthogonal Vector problem that asks given k sets of vectors $U_1, \ldots, U_k \subset \{0,1\}^{\log^2 n}$ each of size n, does there exist $u^{(1)} \in U_1, \ldots, u^{(k)} \in U_k$ such that $\sum_{i \in [log^2 n]} u_i^{(1)} \cdots u_i^{(k)} = 0$?

Then, there exists a function $\mathcal{FOV}_k$ such that any randomized time $t = \Omega(2^{\log^{\epsilon} n})$ (for $\epsilon > 0$) algorithm whose output (on a random instance x) is correct (the algorithm outputs $\mathcal{FOV}_k(x)$with probability at least δ must obey the following bound:

$$\frac{t}{\delta^2} = \Omega(n^{k-o(1)})$$

Combining the above with simulation sound zk-SNARKS, for example from [GM17], to reduce the proof size and verification time we get the following corollary.

Corollary 2. *Let $k \in \mathbb{N}$. Let $t(n) = \Omega(2^{\log^{\epsilon}(n)})$ and $\delta(n) > 0$ such that $T/\delta^2 = \Omega(n^{k-o(1)})$. Let L_k be as in Theorem 8. Fix x_0, x_1 such that $L_k(x_b) = b$. Let D_b define a variable, X, which is defined by rejection sampling the uniform distribution over $\{0,1\}^n$ conditioned on $L_k(X) = b$; if after $O(n)$ tries the rejection sampling has not succeeded, output x_b.*

Then, assuming BPSETH or BPkOVC, PKE, and simulation sound zk-SNARK, $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is a computational $(\mathit{BPTIME}(t(n)), \delta(n) + \mathsf{negl}(n))$-non-malleable code.

In other words, for all $A \in \mathit{BPP}$,

$$\mathsf{Tamper}_{A,0}^{\Pi,\mathit{BPTIME}(t(n))} \approx_{\delta(n)+\mathsf{negl}(n)} \mathsf{Tamper}_{A,1}^{\Pi,\mathit{BPTIME}(t(n))}$$

Note, however, that the tampering experiments are only inverse-polynomially indistinguishable (not negligible). Stronger bounds on the probability of correctness (δ) in Theorem 9 will yield stronger bounds on the tampering experiments.

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). 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 12). 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 $(\textsc{pk}, \textsc{sk}) \leftarrow \mathsf{Gen}(1^n)$.
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 $\mathsf{crs} := (\textsc{pk}, \overrightarrow{\mathsf{crs}}^{\mathsf{NI}})$.

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 $\textsc{pk}$ (where $\textsc{pk}$ 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 $i \in [m]$, $(x^i_1, \ldots, x^i_n)$ is in the support of D_{b^i} where:
  1. For $i \in [m], j \in [n]$, $x^i_j := c^i_j \oplus k^i_j$
  2. $b^i := c^i \oplus k^i_{n+1}$ (where $\overline{c} := c^1, \ldots, c^m$)
  3. $\hat{\boldsymbol{k}}^i$ is an encryption of $k^i_1, \ldots, k^i_{n+1}$ under $\textsc{pk}$ (where $\textsc{pk}$ 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 $i \in [m]$, $\boldsymbol{x}^i = x^i_1, \ldots, x^i_n$.
2. Choose an $m \cdot (n+1)$-bit key $\overline{\boldsymbol{k}} := [\boldsymbol{k}^i]_{i\in[m]} = [k^i_1, \ldots, k^i_n, k^i]_{i\in[m]}$ uniformly at random. For $i \in [m], j \in [n+1]$, compute $\hat{k}^i_j \leftarrow \mathsf{Encrypt}(\textsc{pk}, k^i_j)$. For $i \in [m]$, let $\hat{\boldsymbol{k}}^i := \hat{k}^i_1, \ldots, \hat{k}^i_{n+1}$.
3. For $i \in [m], j \in [n]$, compute $c^i_j := k^i_j \oplus x^i_j$. Let $\overline{\boldsymbol{c}} := [c^i_j]_{i\in[m], j\in[n]}$.
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^i_j proving $([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}^{x^i_j}_{i,j}$ relative to $\mathsf{crs}^{\mathsf{NI}}_{i,j}$.
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 $\mathsf{crs}^{\mathsf{NI}}_{0,0}$.
7. Output $\mathsf{CW} := ([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}, T, [(x^i_j, T^i_j)]_{i\in[m], j\in[n]})$.

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

1. Parse $\mathsf{CW} := ([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}, T, [(x^i_j, T^i_j)]_{i\in[m], j\in[n]})$
2. Check that V^{NI} outputs 1 on all proofs $[T^i_j]_{i\in[m], j\in[n]}, T$, relative to the corresponding CRS.
3. If yes, output $[b^i]_{i\in[m]}$ such that $x^i_1 \ldots x^i_n$ 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 $i \in [m]$, $\boldsymbol{x}^i = x^i_1, \ldots, x^i_n$.
2. Choose an $m \cdot (n+1)$-bit key $\overline{\boldsymbol{k}} := [\boldsymbol{k}^i]_{i \in [m]} = [k^i_1, \ldots, k^i_n, k^i]_{i \in [m]}$ uniformly at random. For $i \in [m], j \in [n+1]$, compute $\hat{k}^i_j \leftarrow \mathsf{Encrypt}(\textsc{pk}, k^i_j)$. For $i \in [m]$, let $\hat{\boldsymbol{k}}^i := \hat{k}^i_1, \ldots, \hat{k}^i_{n+1}$.
3. For $i \in [m], j \in [n]$, compute $c^i_j := k^i_j \oplus x^i_j$. Let $\overline{\boldsymbol{c}} := [c^i_j]_{i \in [m], j \in [n]}$.
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 $\tau^{i,j}_{\mathsf{sim}}$ and $r$, a non-interactive proof $T'^i_j$ proving $s := ([\hat{\boldsymbol{k}}^i]_{i \in [m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}^{x^i_j}_{i,j}$, relative to $\mathsf{crs}^{\mathsf{NI}}_{i,j}$.
6. Simulate, using $\tau^{0,0}_{\mathsf{sim}}$ 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 $\mathsf{crs}^{\mathsf{NI}}_{0,0}$.
7. Output $\mathsf{CW} := ([\hat{\boldsymbol{k}}^i]_{i \in [m]}, \overline{\boldsymbol{c}}, \overline{c}, T', [(x^i_j, T'^i_j)]_{i \in [m], j \in [n]})$.
```

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 $i \in [m]$, $\boldsymbol{x}^i = x^i_1, \ldots, x^i_n$.
2. Choose $[c'^i_j]_{i \in [m], j \in [n]}$ uniformly at random. Let $\overline{\boldsymbol{c}}' := [c'^i_j]_{i \in [m], j \in [n]}$.
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 $\overline{\boldsymbol{k}}' := [\boldsymbol{k}'^i]_{i \in [m]} = [c'^i_1, \ldots, c'^i_n, c'^i]_{i \in [m]}$. For $i \in [m], j \in [n+1]$, compute $\hat{k}'^i_j \leftarrow \mathsf{Encrypt}(\textsc{pk}, k'^i_j)$. For $i \in [m]$, let $\hat{\boldsymbol{k}}'^i := \hat{k}'^i_1, \ldots, \hat{k}'^i_{n+1}$.
5. For $i \in [m], j \in [n]$, simulate, using $\tau^{i,j}_{\mathsf{sim}}$ and $r$, a non-interactive proof $T'^i_j$ proving $s := ([\hat{\boldsymbol{k}}^i]_{i \in [m]}, \overline{\boldsymbol{c}}, \overline{c}) \in \mathcal{L}^{x^i_j}_{i,j}$, relative to $\mathsf{crs}^{\mathsf{NI}}_{i,j}$.
6. Simulate, using $\tau^{0,0}_{\mathsf{sim}}$ 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 $\mathsf{crs}^{\mathsf{NI}}_{0,0}$.
7. Output $\mathsf{CW} := ([\hat{\boldsymbol{k}}'^i]_{i \in [m]}, \overline{\boldsymbol{c}}', \overline{c}', T', [(x^i_j, T'^i_j)]_{i \in [m], j \in [n]})$.
```

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

```
$\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := ([\hat{\boldsymbol{k}}^i]_{i \in [m]}, \overline{\boldsymbol{c}}, \overline{c}, T, [(x^i_j, T^i_j)]_{i \in [m], j \in [n]})$,
2. Output $[\mathsf{Decrypt}(\textsc{sk}, \hat{k}^i_{n+1})]_{i \in [m]}$.
```

Fig. 15. Extracting procedure Ext.

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

1. Parse $\mathsf{CW} := ([\hat{\boldsymbol{k}}^i]_{i \in [m]}, , \overline{\boldsymbol{c}}, \overline{c}, T, [(x^i_j, T^i_j)]_{i \in [m], j \in [n]})$,
2. Check that $\mathsf{V}^{\mathsf{NI}}$ outputs 1 on all proofs $[T^i_j]_{i \in [m], j \in [n]}, T$, 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 10. *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 $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\mathcal{T}}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_0, r_1 *are sampled uniformly at random,* $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, \boldsymbol{b}_0)$ *and* $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\mathcal{T}}_{\mathsf{sim}}, r_1, \boldsymbol{b}_0)$.

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

– ***Simulation Soundness.***

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

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

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

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 10 next.

5.1 Generic Analysis

Similarly to the one-bit case, 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 $m \in \Sigma$ 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 $\mathsf{CW} = ([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}, T, [(x^i_j, T^i_j)]_{i\in[m],j\in[n]})$, $\mathsf{CW}^* = ([\hat{\boldsymbol{k}}^{*i}]_{i\in[m]}, \overline{\boldsymbol{c}}^*, \overline{c}^*, T^*, [(x^i_j, T^{*i}_j]_{i\in[m],j\in[n]})$.

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^*, $[T^{*i}_j)]_{i\in[m],j\in[n]}$ and $([\hat{\boldsymbol{k}}^i]_{i\in[m]}, \overline{\boldsymbol{c}}, \overline{c}) = ([\hat{\boldsymbol{k}}^{*i}]_{i\in[m]}, \overline{\boldsymbol{c}}^*, \overline{c}^*)$. This, in turn, means that there must be some bit x^i_j, x^{*i}_j that CW and CW^* differ on. But note that by assumption $c^i_j = c^{*i}_j$. Due to the fact that CW is well-formed and perfect correctness of the encryption scheme, it must mean that $c^{*i}_j \notin \mathcal{L}^{x^{*i}_j}_{i,j}$. But recall that by assumption, proof T^{*i}_j 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 $\boldsymbol{b}_0, \boldsymbol{b}_1$ and any adversary $A \in \mathcal{G}$ outputting tampering function $f \in \mathcal{F}$, $\{\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,\boldsymbol{b}_0,g}\}_{k\in\mathbb{N}} \approx \{\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,\boldsymbol{b}_1,g}\}_{k\in\mathbb{N}}$

To do so we consider the following hybrid argument, which proceeds almost identically to the hybrid argument for the one-bit case:

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

The proofs of indistinguishability of consecutive hybrid distributions follow identically to the one bit case, except for the final hybrid.

Claim. Hybrid 3 is computationally indistinguishable from Hybrid 4.

Proof. First note that for $\gamma \in \{2, 3\}$ if $g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1$ then $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma)$ always outputs $\mathbf{0}$ and so

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

Now, assume towards contradiction that the two distributions
$\Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2))$ and
$\Psi(g, \mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3, \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3))$ are distinguishable. By the above, this implies that $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$ are distinguishable. Note that since D' outputs m bits, this implies that there exists a distinguishing circuit F over m-bit inputs such that

$$\begin{aligned}\big| &\Pr[F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) = 1]\\ &\quad - \Pr[F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)] = 1\big| \geq \mathsf{negl}(n).\end{aligned}$$

But this yields a contradiction to the **Hardness of D_b relative to Alternate Decoding** property in Theorem 10.

6 Efficient, Multi-Bit NMC for AC^0

Theorem 11. *$\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ (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 $x_1 \dots x_n$ uniformly at random, conditioned on $x_1 \oplus \dots \oplus x_n = b$.*

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. See the beginning of Section 4 for a discussion of how such NIZK and public key encryption can be instantiated.

Before proving the theorem, we state some claims on Fourier concentration of AC^0 circuits and then prove Claim 6, which will be used in the proof of Theorem 11.

Claim ([Tal17]). AC^0 circuits of depth d and size k have at most $2^{-\Omega(n/(\log k)^{d-1}}$ of their Fourier mass at level n or above.

Setting $d = (2+\delta) \cdot \frac{\log m}{\log\log m}$, $k = m^{\log^\delta m}$, $n = m^{3+5\delta}$, for constant $0 \le \delta < 1$, and noting that

$$\frac{n}{(\log k)^{d-1}} \ge \frac{n}{(\log k)^{d}} = \frac{m^{3+5\delta}}{(\log m)^{(1+\delta)d}} = \frac{m^{3+5\delta}}{2^{(1+\delta)d \cdot \log\log m}} = \frac{m^{3+5\delta}}{2^{(1+\delta)(2+\delta)\log m}}$$
$$= \frac{m^{3+5\delta}}{m^{2+3\delta+\delta^2}} = m^{1+2\delta-\delta^2} \in \Omega(m^{1+\delta}),$$

We have the following corollary:

Corollary 3. *An AC^0 circuit of depth $d = (2+\delta) \cdot \frac{\log m}{\log\log m}$ and size $k = m^{\log^\delta m}$ has at most $\epsilon \in 2^{-\Omega(m^{1+\delta})}$ of its Fourier mass at level $n := m^{3+5\delta}$ or above.*

We now prove the main technical claim of this section:

Claim. Let n be security parameter. Let $C \in \mathsf{AC}^0$ be a circuit of depth $d \le (2+\delta) \cdot \frac{\log m}{\log\log m}$ and size $k \le m^{\log^\delta m}$ that takes inputs $\boldsymbol{x}$ of length n bits. Let m be such that $n = m^{3+5\delta}$, where $0 < \delta \le 1$. For $\gamma \in \{0,1\}$ let $\boldsymbol{X}^\gamma$ be a random variable distributed as D_γ. Then for every Boolean function F over m variables,

$$|\Pr[F(C(\boldsymbol{X}^0)) = 1] - \Pr[F(C(\boldsymbol{X}^1)) = 1]| \in 2^{-\Omega(m^\delta)}.$$

Note, the above claim implies that

$$F(C(\boldsymbol{X}^0)) \overset{s}{\approx} F(C(\boldsymbol{X}^1)).$$

Proof (of Claim 6). The conclusion of the claim is implied by showing that $|\Pr[F(C(\boldsymbol{x})) = 1 \mid \mathsf{PAR}(\boldsymbol{x}) = 1] - \Pr[F(C(\boldsymbol{x})) = 1 \mid \mathsf{PAR}(\boldsymbol{x}) = -1]| \in 2^{-\Omega(m^\delta)}$, where the probability is taken over choice of $\boldsymbol{x}$ from the distribution which sets $\boldsymbol{x} \leftarrow D_0$ with probability $1/2$ and $\boldsymbol{x} \leftarrow D_1$ with probability $1/2$. Thus, in order to prove the claim, it is sufficient to show that for every (inefficient) distinguisher F,

$$|E[F \circ C(\boldsymbol{x}) \cdot \mathsf{PAR}(\boldsymbol{x})]| \in 2^{-\Omega(m^\delta)}.$$

Recall that the correlation of $F \circ C$ with $\mathsf{PAR}(\boldsymbol{x})$ is defined as $|E[F \circ C(\boldsymbol{x}) \cdot \mathsf{PAR}(\boldsymbol{x})]|$. Thus, to complete the proof, we must show that for every (inefficient) F, the correlation of $F \circ C$ with $\mathsf{PAR}(\boldsymbol{x})$ is negligible.

Analyzing the correlation of $\chi_S \circ C$ with $\mathsf{PAR}(\boldsymbol{x})$. First, note that since each output bit of C, computed by C_i, $i \in [m]$ is in AC^0 it has depth at most $\delta \cdot \frac{\log n}{\log \log n}$.

We next claim that for $S \subseteq [m]$, there is a circuit computing $\chi_S \circ C(\boldsymbol{x}) = \chi_S(C_1(\boldsymbol{x}), \ldots, C_n(\boldsymbol{x}))$ of depth at most $d = (2+\delta) \cdot \frac{\log m}{\log \log m}$ and size at most $k = m^{\log^\delta m}$.

This follows since the circuit for $\chi_S(C_1(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x}))$ can be constructed by computing $C(\boldsymbol{x}) := C_1(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x})$ in size $m^{\log^\delta m}/2$ and depth $\delta \cdot \frac{\log m}{\log \log m}$ and then feeding this into a circuit that computes parity over (at most) m bits, which (by recursively computing parity over $\log m$ bits in depth 2 and polynomial size), has size $m^{\log^\delta m}/2$ and depth $2\frac{\log m}{\log \log m}$.

By plugging in Claim 3, we have that $\left(\widehat{\chi_S \circ C}([n])\right)^2 = \epsilon$. Since $|E[\chi_S \circ C(\boldsymbol{x}) \cdot \mathsf{PAR}(\boldsymbol{x})]| = \widehat{\chi_S \circ C}([n])$, we have that for $S \subseteq [m]$, the correlation of $\chi_S(C_1(\boldsymbol{x}), \ldots, C_n(\boldsymbol{x}))$ with $\mathsf{PAR}(\boldsymbol{x})$ is at most $\sqrt{\epsilon} \in 2^{-\Omega(m^{1+\delta})}$:

$$|E[\chi_S \circ C(\boldsymbol{x}) \cdot \mathsf{PAR}(\boldsymbol{x})]| \leq \sqrt{\epsilon}. \tag{6.1}$$

Analyzing the correlation of $F \circ C$ with $\mathsf{PAR}(\boldsymbol{x})$. Since $F \circ C(\boldsymbol{x}) = \sum_{S \subseteq [m]} \hat{F}(S) \cdot \chi_S(C_1(\boldsymbol{x}), \ldots, C_n(\boldsymbol{x}))$, we have that

$$\begin{aligned}
|E[F \circ C(\boldsymbol{x}) \cdot \mathsf{PAR}(\boldsymbol{x})]| &= |\sum_{S \subseteq [m]} \hat{F}(S) E[\chi_S(C_1(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x})) \cdot \mathsf{PAR}(\boldsymbol{x})]| \\
&\leq \sum_{S \subseteq [m]} |\hat{F}(S)||E[\chi_S(C_1(\boldsymbol{x}), \ldots, C_m(\boldsymbol{x})) \cdot \mathsf{PAR}(\boldsymbol{x})]| \qquad (6.2) \\
&\leq 2^m \cdot \sqrt{\epsilon} \in 2^{-\Omega(m^\delta)}, \qquad (6.3)
\end{aligned}$$

where (5.1) follows by the triangle inequality and (5.2) follows from (6.1) and the fact that for all $S \subseteq [n]$, $|\hat{F}(S)| \leq 1$.

So we have shown that $|E[F \circ C(\boldsymbol{x}) \cdot \mathsf{PAR}(\boldsymbol{x})]|$ is negligible (in m and therefore also in n, since m and n are polynomially related), thus completing the proof.

We are now ready to complete the proof of the theorem.

Proof (of Theorem 11). The proof proceeds identically to the one-bit proof, until we reach the final property:
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. For every Boolean function, represented by a circuit F over m variables,

$$F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx F \circ \mathsf{D}'(\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3),$$

where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, 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 $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, \boldsymbol{b}_1)$.

We consider a sequence of distributions where we switch the internal random variables of E_2 from from $\boldsymbol{x}^i \leftarrow D_{b_0^i}$, for all $i \in [m]$ to $\boldsymbol{x}^i \leftarrow D_{b_1^i}$, for all $i \in [m]$. Namely, for each $i \in \{0, \dots, m\}$ we consider a distribution where for $j \leq i$, $\boldsymbol{x}^j \leftarrow D_{b_1^i}$ and for $j > i$, $\boldsymbol{x}^j \leftarrow D_{b_0^i}$.

We must show that (1) and (2) hold for each consecutive pair of distributions. When considering the i-th consecutive pair, fix all random variables except the i-th variable $\boldsymbol{X}^i$ to values $\boldsymbol{x}^1, \dots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \dots, \boldsymbol{x}^m$. Let $\boldsymbol{X}^i$ be a random variable such that with probability $1/2$, $\boldsymbol{X}^i \leftarrow D_{b_0^i}$ and with probability $1/2$, $\boldsymbol{X}^i \leftarrow D_{b_1^i}$. $\boldsymbol{X}^i = \boldsymbol{X}^{i,\gamma}$ where $\gamma \leftarrow \{0,1\}$, and let random variable CW^i denote the output of E_2 when using random variables $\boldsymbol{x}^1, \dots, \boldsymbol{x}^{i-1}, \boldsymbol{X}^i, \boldsymbol{x}^{i+1}, \dots, \boldsymbol{x}^m$.

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, for some $i \in [m]$, there is a circuit that takes as input $\boldsymbol{X}^i$, hardwires all other random variables, and outputs 1 in the case that $g(\mathsf{crs}, \mathsf{CW}^i, f(\mathsf{CW}^i), r) = 1$ and 0 otherwise, implying that it has non-negligible correlation to the parity of its input $\boldsymbol{X}^i$. We will show that the above can be computed by an AC^0 circuit with input $\boldsymbol{X}^i$, thus contradicting Theorem 2, which says that an AC^0 circuit has at most negligible correlation with parity of its input $\boldsymbol{X}^i$, denoted $\mathcal{P}(\boldsymbol{X}^i)$. Details follow.

We construct the distribution of circuits $\mathcal{C}^1_{\mathcal{F}}$. A draw $C \sim \mathcal{C}^1_{\mathcal{F}}$ is done as follows:

1. Sample $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.
2. Sample tampering function $\mathcal{A}(\mathsf{crs}) \rightarrow f$.
3. Sample $\overline{\boldsymbol{c}}', \overline{c}'$ uniformly at random,
4. Set the $m \cdot (n+1)$-bit key $\overline{\boldsymbol{k}}' := [\boldsymbol{k}'^{i}]_{i \in [m]} = [c_1'^{i}, \dots, c_n'^{i}, c'^{i}]_{i \in [m]}$. For $i \in [m], j \in [n+1]$, compute $\hat{k}_j'^{i} \leftarrow \mathsf{Encrypt}(\textsc{pk}, k_j'^{i})$. For $i \in [m]$, let $\hat{\boldsymbol{k}}'^{i} := \hat{k}_1'^{i}, \dots, \hat{k}_{n+1}'^{i}$.
5. Sample r uniformly at random.
6. Sample simulated proofs $[T_j'^{\beta,i}]_{\beta \in \{0,1\}, i \in [m], j \in [n]}$ and T' (as described in Figure 13).
7. Sample $\boldsymbol{x}^1, \dots, \boldsymbol{x}^{i-1}$ from $D_{b_0^i}$, and $\boldsymbol{x}^{i+1}, \dots, \boldsymbol{x}^m$ from $D_{b_1^i}$.
8. Output the following AC^0 circuit C that has the following structure:
   - **hardcoded variables:** $\mathsf{crs}, \textsc{sk}, f, [\hat{\boldsymbol{k}}'^{i}]_{i \in [m]}, \overline{\boldsymbol{c}}', \overline{c}', r, [T_j'^{\beta,i}]_{\beta \in \{0,1\}, i \in [m], j \in [n]}, \boldsymbol{x}^1, \dots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \dots, \boldsymbol{x}^m$.
   - **input:** $\boldsymbol{X}^i$.
   - **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 $j \in [n]$, select the correct simulated proof $T_j'^{X_j^i, i}$ based on the corresponding input bit X_j^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 $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ and $\mathsf{D}'(\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$ have non-negligible statistical distance. This implies that there exists a distinguisher F (represented by an m-bit Boolean function) such that $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ is far from $F \circ \mathsf{D}'(\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$. This implies that, for some $i \in [m]$, the output of $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}^i)), f(\mathsf{CW}^i); r_i)$ is correlated with the parity of its input $\boldsymbol{X}^i$. We will show that $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}^i)), f(\mathsf{CW}^i); r_i)$ can be computed by an AC^0 circuit C (drawn from some distribution $\mathcal{C}$) with input $\boldsymbol{X}^i$. We then use Claim 6, which says that if C is an AC^0 circuit taking inputs of length n bits and F is any m-bit function then the output $F(C(\boldsymbol{X}^i))$, conditioned on the parity of $\boldsymbol{X}^i$ being 0 is statistically close to the output $F(C(\boldsymbol{X}^i))$, conditioned on the parity of $\boldsymbol{X}^i$ being 1. This yields a contradiction, since it means that $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}^i)), f(\mathsf{CW}^i); r_i)$ cannot be correlated with the parity of its input $\boldsymbol{X}^i$. Details follow.

We construct the distribution of circuits $\mathcal{C}^2_{\mathcal{F}}$. A draw $C \sim \mathcal{C}^2_{\mathcal{F}}$ is done as follows:

1. Sample $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.
2. Sample tampering function $\mathcal{A}(\mathsf{crs}) \rightarrow f$.
3. Sample $\overline{\boldsymbol{c}}', \overline{c}'$ uniformly at random,
4. Set the $m \cdot (n+1)$-bit key $\overline{\boldsymbol{k}}' := [\boldsymbol{k}'^{i}]_{i \in [m]} = [c'^{i}_1, \ldots, c'^{i}_n, c'^{i}]_{i \in [m]}$. For $i \in [m], j \in [n+1]$, compute $\hat{k}'^{i}_j \leftarrow \mathsf{Encrypt}(\textsc{pk}, k'^{i}_j)$. For $i \in [m]$, let $\hat{\boldsymbol{k}}'^{i} := \hat{k}'^{i}_1, \ldots, \hat{k}'^{i}_{n+1}$.
5. Sample r uniformly at random.
6. Sample simulated proofs $[T'^{\beta,i}_j]_{\beta \in \{0,1\}, i \in [m], j \in [n]}$ and T' (as described in Figure 13).
7. Sample $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{i-1}$ from $D_{b^i_0}$, and $\boldsymbol{x}^{i+1}, \ldots, \boldsymbol{x}^m$ from $D_{b^i_1}$.
8. Output the following AC^0 circuit C that has the following structure:
   - **hardcoded variables:** $\mathsf{crs}, \textsc{sk}, f, [\hat{\boldsymbol{k}}'^{i}]_{i \in [m]}, \overline{\boldsymbol{c}}', \overline{c}', r, [T'^{\beta,i}_j]_{\beta \in \{0,1\}, i \in [m], j \in [n]}, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \ldots, \boldsymbol{x}^m$.
   - **input:** $\boldsymbol{X}^i$.
   - **computes and outputs:**

$$\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}^i)), f(\mathsf{CW}^i); r_i)$$

.

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^i is in AC^0 since all it does is, for $j \in [n]$, select the correct simulated proof $T'^{X^i_j, i}_j$ based on the corresponding input bit X^i_j. Additionally, f in AC^0 and D' is in AC^0 , since xor of two streams of bits is in AC^0 and V^{SAT} is in AC^0 . Thus, the entire circuit is in AC^0 .

6.1 Tampering with decision trees

Theorem 12. *$\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ (presented in Figure 12) is an m-bit, computational, non-malleable code in the CRS model against tampering by depth-d circuits with unbounded fan-in and size $\leq 2^{m^{\varepsilon}}$ (where d, ε are constants), and m is such that $n = m^{1+\varepsilon}$, 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 $x_1 \oplus \cdots \oplus x_n = b$.*

The proof of this theorem follows exactly as the proof of Theorem 11, except we replace Claim 6 with Claim 6.1 below. But first, we present a simple corollary of the theorem.

Corollary 4. *Under the assumptions of Theorem 12, $(\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is an m-bit, computational, non-malleable code against tampering by decision trees of complexity $\leq m^{\varepsilon}$, where $0 < \varepsilon \leq 1$ is a constant, and $n = m^{1+\varepsilon}$.*

This follows from the above theorem when put together with the fact that decision trees of depth t can be represented as a disjunction of 2^t terms (each term is a path to some 1).

Claim. Let n be security parameter. Fix some $d \in \mathbb{Z}$. Let $\varepsilon \geq 1/d$. Let m be such that $n = m^{1+\varepsilon}$ Let $C : \{0,1\}^n \rightarrow \{0,1\}^m$ be composed of depth-d circuit with unbounded fan-in and size $s = 2^{m^{\varepsilon}}$. For $\gamma \in \{0,1\}$ let $\boldsymbol{X}^{\gamma}$ be a random variable distributed as D_{γ}. Then for every Boolean function $F : \{0,1\}^m \rightarrow \{0,1\}$ over m variables,

$$\left|\Pr[F \circ C(\boldsymbol{X}^0) = 1] - \Pr[F \circ C(\boldsymbol{X}^1) = 1]\right| \leq \frac{1}{2} + \frac{(k+1)}{2\exp(m^{\varepsilon})}.$$

Proof. Let $n = m^{1+\varepsilon}/q$ for $1 > \varepsilon \geq 1/d$. Let ρ be a random restriction over $\{0,1\}^\ell$ such that $\Pr[\star] = q = \frac{1}{18^d e^d m^{d\varepsilon}} \leq \frac{1}{18^d e^d m}$. Let $t = m^\varepsilon$. Then, by lemma 1 and a union bound the probability that some output bit of C_ρ cannot be represented by a decision tree of depth $t-2$ is at most $ms(9q^{1/d}t)^t$.

$$ms(9q^{1/d}t)^t \leq m2^{m^\varepsilon}(\frac{9m^\varepsilon}{18em^\varepsilon}m^\varepsilon)^{m^\varepsilon} \tag{6.4}$$
$$= m\exp(-m^\varepsilon). \tag{6.5}$$

If C_ρ can be represented by a decision tree of depth $t-1$ call C_ρ "simple."

Any $F : \{0,1\}^m \to \{0,1\}$ has decision tree complexity at most than m. If we compose this with a simple C_ρ, the resulting decision tree has complexity $< m^{1+\varepsilon} - 1$ [Tal13].

Additionally, by standard Chernoff bounds, the probability that ρ contains more than $4\ell q$ $\star$'s is at most $\exp(-m^\varepsilon)$. Call such a ρ "bad." Note that if this is the case, then C_ρ is a function over at least $m^{1+\varepsilon}$ variables.

If neither event happens, ρ is not bad and C_ρ is simple, then $F \circ C_\rho$ is completely uncorrelated with parity. Otherwise, the correlation is bounded by 1. Therefore, we can simply bound correlation with the probability that either ρ is bad or C_ρ is not simple: $(m+1)\exp(-m^\varepsilon)$.

7 One-Bit NMC Against Streaming Adversaries

We begin by describing constructions of the underlying components required to instantiate the generic constructions in the streaming adversaries setting.

In the following, we assume that the tampering class $\mathcal{F}$ corresponds to streaming adversaries with memory $o(n'')$. We then choose parameter $n \in \omega(n'')$ and parameter $n' \in \omega(n)$. n is the parameter for the hard distribution described in Section 7.1, n' is the parameter for the encryption scheme described in Section 7.2, n'' is the parameter for the weak encryption scheme $(\mathsf{Hide}, \mathsf{Rec})$ described in Section 7.3.

7.1 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 hardness of the distribution follows from Theorems 3 and lemma 2.

Claim. Let A be a streaming algorithm with $o(n)$ space, and $\alpha > 0$. Then,

$$\|\Pr_{x\sim D_0}[A(x) = 0] - \Pr_{x\sim D_1}[A(x) = 0]\| \leq 2^{\alpha n/3}.$$

7.2 Encryption scheme $\mathcal{E} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ (parameter $n' \in \omega(n)$)

The Learning Parity problem yields an encryption scheme with semantic security against streaming adversaries with $o(n')$ storage. We can use this encryption scheme to encrypt the key k, bit-by-bit, thus yielding an encryption scheme with the necessary properties.

To encrypt a bit b, $\mathsf{Encrypt}(b)$ outputs z, where $z \sim D_b$ and D_b is the same as above, except with parameter n'.

To decrypt a ciphertext z, with $\Theta(n')$ storage, $\mathsf{Decrypt}(z)$ runs the parity learning algorithm to recover b.

Renaming variables and plugging in Claim 7.1 from above, we have

Claim. Let A be a streaming algorithm with $o(n)$ space, and $\alpha > 0$. Then,

$$\|\Pr_{z\sim\mathsf{Encrypt}(0)}[A(z) = 0] - \Pr_{z\sim\mathsf{Encrypt}(1)}[A(z) = 0]\| \leq 2^{\alpha n/3}.$$

7.3 Weak Encryption Scheme (parameter $n'' \in o(n)$)

Let $n'' = (\mu''+1)^2 - 1$ Given a bit string $k = k_1, \dots, k_{\mu''}$ of length μ'' bits, and a vector $y := y_1, \dots, y_{\mu''}$ of length μ'' bits, let $S_k \subseteq [\mu'']$ denote the set of positions in k that are set to 1. Let $m = m_1, \dots, m_\ell$ be a bit string of length ℓ bits (where ℓ is polynomial in μ''). let $\chi_{S_k}(y) := \bigoplus_{i \in S_k} y_i$.

On input $m \in \{0,1\}^\ell$ and k as above, $\mathsf{Hide}(k, m)$ chooses random strings $y_1^0, \dots, y_{\mu''}^0$, $y_1, \dots, y_\ell \leftarrow U_{\mu''}$ and outputs $([y_i^0, \chi_{S_k}(y_i^0)]_{i\in[\mu'']}, [(y_i, \chi_{S_k}(y_i) \oplus m_i)]_{i\in[\ell]})$.

On input $([y_i^0, \chi_{S_k}(y_i^0)]_{i\in[\mu'']}, [(y_i, \chi_{S_k}(y_i) \oplus m_i)]_{i\in[\ell]})$, Rec uses the first μ'' examples to learn χ_{S_k} and then returns $[m^i]_{i\in[\ell]} := \chi_{S_k}(y_i) \oplus m_i'$.

7.4 Non-Interactive Simulatable Proof System (parameter $n'' \in o(n)$)

In the following construction, inputs and proofs have λ parallel components, corresponding to λ parallel invocations of the MPC-in-the-head paradigm. To simplify the exposition, we assume that the bounded, streaming computations read in λ symbols in parallel from each of the λ parallel components and output λ symbols in parallel for each of the λ parallel components. Note that this increases the required storage by a factor of λ, but since we set $\lambda \ll n''$, the overall storage bound remains below n''.

We begin by introducing a simplified proof system and proving its soundness. We then present the actual proof system used in our construction. Looking ahead, proving that the **Simulation Soundness** property required by Theorem 4 holds, will reduce to the soundness of the *simplified* proof system.

Simplified Proof System Π' Let λ' be security parameter and ℓ is a constant (e.g. $\ell = 5$).

P: On input statement s, encoding $[s_u^1, \dots, s_u^{\lambda'}]_{u\in[\ell]}$ and witness w:

1. **Check that for $q \in [\lambda']$, $s_1^q \oplus \cdots \oplus s_\ell^q = s$:** Compute streaming hash $h^* := H^h(s)$ and λ' streaming hashes in parallel, $h_q := [H^h(s_1^q \oplus \cdots \oplus s_\ell^q)]_{q\in\lambda'}$, where H is Merkle Damgard and $h \leftarrow \mathcal{H}$, where $\mathcal{H}$ is a universal family of hash function. Check that for all $q \in [\lambda']$, $h_q = h^*$. If not, output $\perp$.
2. **Run MPC-in-the-head:** For $q \in [\lambda]$, secret share w into ℓ additive shares $(w_1^q, \dots, w_\ell^q)$ and run $\mathsf{MPC}(P_1(s_1^q, w_1^q) \dots, P_\ell(s_\ell^q, w_\ell^q)$, producing views $[\mathsf{View}_u^q]_{q\in[\lambda'],u\in[\ell]}$ (here, each view is a tableau of the parties' computation, as described in the construction of circuit SAT proof system for streaming verifiers in Section 2.4).
   Note that of the input wires to the views, some will be public (corresponding to the shares of s) and some will be private (corresponding to the shares of w).
3. **Encrypt the Views.** For $q \in [\lambda'], u \in [\ell]$, choose k_u^q uniformly at random from $\{0,1\}^{\mu''}$. Compute $S_u^q \leftarrow \mathsf{Hide}(k_u^q, \mathsf{View}_u^q)$, where Hide is run with parameter n''. Output proof $T = ([\hat{k}_u^q, S_u^q]_{q\in[\lambda'],u\in[\ell]})$.

V: On input statement s, encoding $[s_u^1, \dots, s_u^{\lambda'}]_{u\in[\ell]}$ and proof T, parse $T = ([\hat{k}_u^q, S_u^q]_{q\in[\lambda'],u\in[\ell]})$.

1. **Generate randomness.** Choose randomness $r_1, \dots, r_{\lambda'}$ and hash function $h \leftarrow \mathcal{H}$. For each $q \in [\lambda]$, choose a subset $\mathcal{S}_q \subseteq [\ell]$ using random coins r_q.
2. **Check that for $q \in [\lambda]$, $s_1^q \oplus \cdots \oplus s_\ell^q = s$:** Repeat the same steps as P to check that for $q \in [\lambda]$, $s_1^q \oplus \cdots \oplus s_\ell^q = s$ if not, output $\perp$.
3. **Prepare hashes of input for later equality checks.** *This is done in parallel to the previous item.* For $q \in [\lambda']$, $u \in \mathcal{S}_q$, compute $h_{q,u} = H^h(s_u^q)$.
4. **Open selected views.** For $q \in [\lambda]$, $u \in \mathcal{S}_q^1$, recover $k_u^q = \mathsf{Decap}(\hat{k}_u^q)$, recover View_u^q, where $\mathsf{View}_u^q := \mathsf{Rec}(S_u^q)$ (where Rec is run with parameter n'') and corresponding inputs $\widetilde{s}_u^q, \widetilde{w}_u^q$.
5. **Check consistency of views.** *This is done in parallel to the previous item.* (1) Check that the opened views are internally consistent (using the verifier described in the construction of circuit SAT proof system for streaming verifiers in Section 2.4). (2) Check that the opened views are consistent with each other (i.e. same transcript) using similar hashing techniques as above. (3) Check that $h_{q,u} = H^h(\widetilde{s}_u^q)$.

6. **Output.** If all checks succeed, output 1. Otherwise, output 0.

Claim. Soundness of proof system follows from perfect correctness of the MPC and security of the universal hash function family $\mathcal{H}$.

The Actual Proof System Π As above, in the following construction, inputs and proofs have λ parallel components, corresponding to λ parallel invocations of the MPC-in-the-head paradigm. To simplify the exposition, we assume that the bounded, streaming computations read in λ symbols in parallel from each of the λ parallel components and output λ symbols in parallel for each of the λ parallel components. Note that this increases the required storage by a factor of λ, but since we set $\lambda \ll n''$, the overall storage bound remains below n''.

Let λ be security parameter and ℓ is a constant.

<u>P:</u> On input statement $s := s_1, \ldots, s_t$, encoding $[s_1^q, \ldots, s_\ell^q]_{q\in[\lambda]}$ and witness w:

1. **Check that for $q \in [\lambda]$, $s_1^q \oplus \cdots \oplus s_\ell^q = s$:** Compute streaming hash $h^* := H^h(s)$ (with block length λ) and λ streaming hashes (all with block length λ) in parallel, $h_q := [H^h(s_1^q \oplus \cdots \oplus s_\ell^q)]_{q\in\lambda}$, where H is Merkle Damgard and $h \leftarrow \mathcal{H}$, where $\mathcal{H}$ is a universal family of hash function. Check that for all $q \in [\lambda]$, $h_q = h^*$. If not, output $\perp$.
2. **Run MPC-in-the-head:** For $q \in [\lambda]$, secret share w into ℓ additive shares $(w_1^q, \ldots, w_\ell^q)$ and run $\mathsf{MPC}(P_1(s_1^q, w_1^q) \ldots, P_\ell(s_\ell^q, w_\ell^q))$, producing views $[\mathsf{View}_u^q]_{q\in[\lambda],u\in[\ell]}$.
Note that of the input wires to the views, some will be public (corresponding to the shares of s) and some will be private (corresponding to the shares of w).
3. **Select the Slots.** For each position q, u there are $\ell \cdot 2t$ slots $[S_{q,u}^{z,p}]_{z\in[\ell],p\in[2t]}$, where $t = |s|$. Let $s^q[z,p]$ denote the p-th bit position of the string s_z^q. Let $\mathcal{S}'_{q,z}$ be the set of positions in the string $[s^q[z,p]||\overline{s}^q[z,p]]_{p\in[t]}$ that are set to 1. Note that $|\mathcal{S}'_{q,z}| = t$.
4. **Encrypt the Views.** For $q \in [\lambda], u \in [\ell], z \in [\ell], p \in [2t]$, choose $k_{q,u}^{z,p}$ uniformly at random from $\{0,1\}^{\mu''}$. For $q \in [\lambda], u \in [\ell], z \in [\ell]$ and $p \in \mathcal{S}'_{q,z}$, compute $S_{q,u}^{z,p} \leftarrow \mathsf{Hide}(k_{q,u}^{z,p}, \mathsf{View}_u^q)$, where Hide is run with parameter n''. For $q \in [\lambda], u \in [\ell], z \in [\ell]$ and $p \notin \mathcal{S}'_{q,z}$, set $S_{q,u}^{z,p} \leftarrow \mathsf{Hide}(k_{q,u}^{z,p}, \mathbf{0})$. Output proof $T = ([\hat{k}_{q,u}^{z,p}, S_{q,u}^{z,p}]_{q\in[\lambda],u\in[\ell],z\in[\ell],p\in[2t]})$.

<u>V:</u> On input statement $s := s_1, \ldots, s_t$, encoding $[s_1^q, \ldots, s_\ell^q]_{q\in[\lambda]}$, and proof T, parse $T = ([\hat{k}_{q,u}^p, S_{q,u}^p]_{q\in[\lambda],u\in[\ell],p\in[2t]})$.

1. **Generate Randomness.** Choose randomness $(r_1^1, r_1^2) \ldots, (r_\lambda^1, r_\lambda^2)$ and hash function $h \leftarrow \mathcal{H}$. Choose subsets $\mathcal{S}_q^1, \mathcal{S}_q^2 \subseteq [\ell]$, each of size 2, using random coins (r_q^1, r_q^2).
2. **Check that for $q \in [\lambda]$, $s_1^q \oplus \cdots \oplus s_\ell^q = s$:** Repeat the same steps as P to check that for $q \in [\lambda]$, $s_1^q \oplus \cdots \oplus s_\ell^q = s$ if not, output $\perp$.
3. **Prepare hashes of input for later equality checks.** *This is done in parallel to the previous item.* Do the following in parallel: (1) For $q \in [\lambda]$, $u \in \mathcal{S}_q^1$, compute $h_{q,u}^1 = H^h(s_u^q)$ in a streaming fashion, using block size λ and space $O(\lambda^2)$. (2) For $q \in [\ell]$, $u \in \mathcal{S}_q^2$, compute $h_{q,u}^2 = H^h(s_u^q)$ in a streaming fashion, using block size λ and space $O(\lambda^2)$.
4. **Open selected views and check consistency across slots.** For $q \in [\lambda]$, $u \in \mathcal{S}_q^1$, do the following: (1) For each $z \in [\mathcal{S}_q^2]$, $p \in [2t]$, recover $k_{q,u}^{z,p} = \mathsf{Decap}(\hat{k}_{q,u}^p)$. (2) For each $z \in [\mathcal{S}_q^2]$, recover $\mathsf{View}_{q,u}^{z,p}$, where $\mathsf{View}_{q,u}^{z,p} := \mathsf{Rec}(S_{q,u}^{z,p})$, and Rec is run with parameter n''. Let $[\mathsf{View}_{q,u}^{z,p}]_{p\in\mathcal{S'}_{q,u}^z}$ be the views (out of $[2t]$) that do not decrypt to $\mathbf{0}$ (i.e. $\mathcal{S'}_{q,u}^z$ is the set of slots that are filled). Let $s'^z_{q,u}$ denote the vector corresponding to $\mathcal{S'}_{q,u}^z$. (3) Use hashing as above to check that for each q, r all the recovered views $\mathsf{View}_{q,u}^{z,p}$ are identical. (4) Let View_u^q denote the contents of these identical views and let $(\widetilde{s}_u^q, \widetilde{w}_u^q)$ be the corresponding inputs.
5. **Check consistency of views.** *This is done in parallel to the previous item.* (1) For $q \in [\lambda]$, $u \in \mathcal{S}_q^1$ check that the view View_u^q is internally consistent (using the verifier described in the construction of circuit SAT proof system for streaming verifiers in Section 2.4). (2) For $q \in [\lambda]$, Check that the views $[\mathsf{View}_u^q]_{u\in\mathcal{S}'_q}$ are consistent with each other (i.e. same transcript) using similar hashing techniques as above. (3) For each $u \in \mathcal{S}_q^1$, check that $h_{q,u}^1 = H^h(\widetilde{s}_u^q)$. (4) For each $u \in \mathcal{S}_q^2$, check that $h_{q,u}^2 = H^h(\widetilde{s}_u^q)$.

6. **Output.** If all checks succeed, output 1. Otherwise, output 0.

Sim: On input statement $s := s_1, \ldots, s_t$, encoding $[s^q_1, \ldots, s^q_\ell]_{q \in [\lambda]}, \ldots, [s^q_1, \ldots, s^q_\ell]_{q \in [\lambda]}$:

1. **Check that for $q \in [\lambda]$, $s^q_1 \oplus \cdots \oplus s^q_\ell = s$:** Compute streaming hash $h^* := H^h(s)$ and λ streaming hashes in parallel, $h_q := [H^h(s^q_1 \oplus \cdots \oplus s^q_\ell)]_{q \in \lambda}$, where H is Merkle Damgard and $h \leftarrow \mathcal{H}$, where $\mathcal{H}$ is a universal family of hash function. Check that for all $q \in [\lambda]$, $h_q = h^*$. If not, output $\perp$.
2. **Run MPC-in-the-head Simulation:** For $q \in [\lambda]$, choose subset $\mathcal{S}^1_q \subseteq [\ell]$ using random coins r^1_q. run $\mathsf{Sim}^{\mathsf{MPC}}$ to produce the views of parties P_u, $u \in \mathcal{S}^1_q$ (note that each of these parties has public input s^q_u) producing views $[\mathsf{View}^q_u]_{q \in [\lambda], u \in \mathcal{S}_q}$.
3. **Select the Slots.** For each position q, u there are $\ell \cdot 2t$ slots $[q, u, z, p]_{z \in [\ell], p \in [2t]}$, where $t = |s|$. Let $s^q[z, p]$ denote the p-th bit position of the string s^q_z. Let $\mathcal{S}'_{q,z}$ be the set of positions in the string $[s^q[z, i] || \overline{s}^q[z, i]]_{i \in [t]}$ that are set to 1. Note that $|\mathcal{S}'_{q,z}| = t$.
4. **Encrypt the Views.** Choose subset $\mathcal{S}^2_q \subseteq [\ell]$ using random coins r^2_q. For $q \in [\lambda], u \in [\ell], z \in [\ell], p \in [2t]$, choose $k^{z,p}_{q,u}$ uniformly at random from $\{0,1\}^{\mu''}$. For $q \in [\lambda], u \in [\mathcal{S}^1_q], z \in [\mathcal{S}^2_q]$ and $p \in \mathcal{S}'_{q,z}$, compute $S^{z,p}_{q,u} \leftarrow \mathsf{Hide}(k^{z,p}_{q,u}, \mathsf{View}^q_u)$, where Hide is run with parameter n''. For $q \in [\lambda]$ and u, z, p such that $u \notin [\mathcal{S}^1_q]$ OR $z \notin [\mathcal{S}^2_q]$ OR $p \notin \mathcal{S}'_{q,z}$, set $S^{z,p}_{q,u} \leftarrow \mathsf{Hide}(k^{z,p}_{q,u}, \mathbf{0})$. Output proof $\pi = ([\hat{k}^{z,p}_{q,u}, S^{z,p}_{q,u}]_{q \in [\lambda], u \in [\ell], z \in [\ell], p \in [2t]})$.

Remark 1. Note that if the simulated proof and encoding $[s^q_1, \ldots, s^q_\ell]_{q \in [\lambda]}$ are generated at the same time, then we can *first* produce the simulated proof π *independently* of s. This can be done because the simulated proof depends only on $[s^q_u]_{q \in [\lambda], u \in [\mathcal{S}^1_q \cup \mathcal{S}^2_q]}$, which can be chosen uniformly at random (since our parameter settings ensure that $|\mathcal{S}^1_q| + |\mathcal{S}^2_q| = \ell - 1$). Given the simulated proof π and the choice of $[s^q_y]_{q \in [\lambda], y \in [\mathcal{S}^1_q \cup \mathcal{S}^2_q]}$, we can then output the entire encoding $[s^q_1[1], \ldots, s^q_\ell[1]]_{q \in [\lambda]}, \ldots, [s^q_1[t'], \ldots, s^q_\ell[t']]_{q \in [\lambda]}$ and proof π in a *streaming fashion*, given input s in a streaming fashion, requiring only $O(\lambda^2)$ memory. This is done by hardwiring $[s^q_y]_{q \in [\lambda], y \in [\mathcal{S}^1_q \cup \mathcal{S}^2_q]}$, and, in a block-by-block streaming fashion (with block length λ), outputting, in parallel, the i-th block of each share for each $q \in [\lambda]$, along with the missing share: $[s \oplus \left(\bigoplus_{y \in [\mathcal{S}^1_q \cup \mathcal{S}^2_q]} s^q_y\right)]_{q \in \lambda}$.

Remark 2. Note that for two statements $s_1 \neq s_2$ and their proofs π_{s_1}, π_{s_2}, for each q, there exists a pair (z^*_q, p^*_q) such that for each $u \in [\ell]$, slot $[q, u, z^*_q, p^*_q]$ contains encryptions of $\mathbf{0}$ in π_{s_1} and encryptions of View^q_u in π_{s_2}. Moreover, for any two statements $s_1 \neq s_2$ and every $q \in [\lambda]$, the probability over choice of r^2 that $z^*_q \in [\mathcal{S}^2_q]$, (which means that slots $[q, u, z^*, p^*]_{u \in \mathcal{S}^1_q}$ will be checked by V) is at least $1/\ell$.

Claim. Let $\mathcal{A}$ be an *unbounded* adversary that takes as input random variable $S^1_{\mathsf{tamp}} || S^2_{\mathsf{tamp}}$. Let S^1, S^2 denote the random variables corresponding to the initial contents of $\mathcal{A}$'s input (before tampering).

Let f be a streaming tampering function with memory $o(n'')$ that reads in random variable $S^1 || I || S^2$ (chunk-by-chunk), where $I = \mathsf{Hide}(k, m)$ is an encoding of m with random key k and parameter n'', and (in a streaming fashion) outputs the random variable $S^1_{\mathsf{tamp}} || I_{\mathsf{tamp}} || S^2_{\mathsf{tamp}}$ (chunk-by-chunk). For $i \in [3]$, let $f(S^1 || I || S^2)[i]$ denote the i-th chunk outputted by f.

Then for any m_0, m_1, when I encodes m_0 vs. I encodes m_1 the resulting output distributions of $\mathcal{A}(f(S^1 || I || S^2)[1], f(S^1 || I || S^2)[3])$ are statistically close.

Proof. If the claim is false, then there exists a distinguisher D. Using D, $\mathcal{A}$, f, we can now construct a streaming branching program with space $o(n'')$ that distinguishes whether I encodes 0 or 1. We do so in the following way:

1. Fix the random variables $S^1 = s^1$ and $S^2 = s^2$
2. Construct a branching program $BP_{s^1, s^2, D, \mathcal{A}, f}$ that hardcodes s^1, s^2 and emulates $f(s^1 || I || s^2)$. Note the following about the emulation:
   - $S^1_{\mathsf{tamp}} = s^1_{\mathsf{tamp}} = f(s^1 || I || s^2)[1]$ and the entire inner state of f up to the moment right before it starts reading I can be hardcoded into the transition function for BP.

  - from this point on, we can emulate $f(s^1||I||s^2)$ using space $o(n'')$ until the moment that f finishes reading I.
  - from this point on, we can determine the output of $\mathcal{A}(s^1_{\mathsf{tamp}}, f(S^1||I||S^2)[3])$ without requiring any more memory. To do this, we use the fact that $s^1, s^1_{\mathsf{tamp}}, s^2$ are hardcoded and simply precompute the output of $\mathcal{A}(s^1_{\mathsf{tamp}}, f(s^1||I||s^2)[3])$ for each of the possible $2^{o(n'')}$ internal states of f (by using the internal state of f at the moment that f finishes reading I to compute $S^2_{\mathsf{tamp}} = s^2_{\mathsf{tamp}}$ and then running $\mathcal{A}$ on $(s^1_{\mathsf{tamp}}, s^2_{\mathsf{tamp}})$) and then output whatever D outputs. This implies that given the internal state of f at the moment f finishes reading I, we can immediately transition to the output level of the branching program, without requiring additional state.
3. Note that BP succeeds with the same probability as D.

Theorem 13. *$\Pi = (\mathsf{E}, \mathsf{D})$ (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 encryption scheme described in Section 7.2 (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 described in Section 7.4 with parameter $n'' := n''(n)$.*
- *For $b \in \{0, 1\}$, D_b is the distribution described in Section 7.1 (with paramter n).*

Note that no CRS or computational assumptions are needed for this result. Indeed we can assume that the adversary $\mathcal{A}$ outputting tampering function f is computationally unbounded. To maintain consistency, we continue to use the variables crs, SK, $\overrightarrow{\tau}_{\mathsf{sim}}$ but we simply assume that all of them are set to $\perp$.

Proof. To prove the theorem, we need to show that the necessary properties from Theorem 4 hold. We next go through these one by one.

- **Simulation of proofs.**
  1. $$\Pr[g(\mathsf{CW}_0, f(\mathsf{CW}_0), r_0) = 1] \approx \Pr[g(\mathsf{CW}_1, f(\mathsf{CW}_1), r_1), = 1],$$
  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_1, r_2 are sampled uniformly at random, $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, \boldsymbol{b}_0)$ and $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, \boldsymbol{b}_0)$.
  To prove this, we must switch from all real proofs (as outputted by E) to all simulated proofs (as outputted by E). Looking closer at the construction from Section 7.4, to switch from a real to a simulated proof, we must go through a sequence of hybrids starting from honestly generated proofs from E and ending with simulated proofs from E_1. In hybrid $H_{[q,u,p]}$ we switch to using encoding algorithm $\mathsf{E}^{[q,u,p]}$, which works the same way as the encoding in the previous hybrid, except when generating the proofs, if $r \notin \mathcal{S}_q$, it sets random variable $S^p_{q,u}$ to $S^p_{q,u} \leftarrow \mathsf{Hide}(k^p_{q,u}, \mathbf{0})$. Note that for $u \in \mathcal{S}_q$, $H_{[q,u,p]}$ is identical to the previous hybrid. Let $\mathsf{CW}^{[q,u,p]}$ denote the random variable representing the codeword in each hybrid distribution. We use Claim 7.4 to show that for every fixed random string r and $u \notin \mathcal{S}_q$, the output of g in consecutive hybrids is indistinguishable and the output of Ψ in consecuitve hybrids is indistinguishable. To see this, note that we set $\mathcal{A}$ from Claim 7.4 to be equal to D, $f = f$, S^1 denotes the codeword up to the $[q, u, p]$ position, S^2 denotes the codeword after the $[q, u, p]$ position, and $I := S^p_{q,u}$. The key is that V^{NI} (which checks the proofs during computation of g and Ψ) with random coins $r := r_1, \ldots r_q$ will not check slot $[q, u, p]$ when determining its output and so the conditions of Claim 7.4 are satisfied.
- **Simulation of Encryption.**
  1. $$\Pr[g(\mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1] \approx \Pr[g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2), = 1],$$

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 $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, 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 $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, \boldsymbol{b}_0)$.
To see this, we will show that $g(\mathsf{CW}_1, f(\mathsf{CW}_1), r_1)$, $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1))$ can be computed in a streaming fashion with memory $o(n')$, while distinguishing encryptions of k_i from encryptions of k'_i in a streaming fashion requires memory $\Omega(n')$ (see Claim 7.2). We will show that each of $\mathsf{E}_1/\mathsf{E}_2, f, \mathsf{D}, g$ can be computed in a streaming fashion with memory $o(n')$. This implies that their (parallel) composition can also be computed in a streaming fashion with memory $o(n')$.
To see that this is true for $\mathsf{E}_1/\mathsf{E}_2$, we use the observation from Remark 1. It is true for f by definition of the tampering class $\mathcal{F}$. D consists of (1) determining b such that $\boldsymbol{x}$ is in the support of D_b and (2) running the verifier for Π. Note that (1) can be done in a streaming fashion using $\Theta(n)$ bits of memory. Since we choose $n' = \omega(n)$, the required memory is $o(n')$. (2) can be done in a streaming fashion with space $o(n')$, since the only memory intensive part of the verification is running Rec. Similar to the above, we set parameters of $\mathsf{Hide}/\mathsf{Rec}$ such that this can be done using $\Theta(n'')$ bits of memory, where $n'' = o(n)$. Finally, g consists of a bit-wise comparison of two strings obtained in a streaming fashion and running the verifier for Π, both of which can be done in a streaming fashion with memory $o(n')$. Thus, we have shown that each of each of $\mathsf{E}_1/\mathsf{E}_2, f, \mathsf{D}, g$ can be computed in a streaming fashion with memory $o(n')$.

– **Simulation Soundness.**

$$\Pr_r[\mathsf{D}(f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \wedge g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0] \leq \mathsf{negl}(n),$$

where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2 is sampled uniformly at random and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, 0)$.
We give a reduction from the above property with Π instantiated with security parameter λ to the soundness of Π' with security parameter $\lambda' = \lambda/2\ell$.
First, if $g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0$ then it must be the case that either the verification of the proofs rejects (in both D and D', since they are the same) or $s_1 \neq s_2$, where s_1 is the statement for the proofs in CW and s_2 is the statement for the proofs in $f(\mathsf{CW})$, and $s_2 \notin \mathcal{L}$. Therefore, by Remark 2, for each q, there exists a pair (z^*_q, p^*_q) such that for each $u \in [\ell]$, slot $[q, u, z^*_q, p^*_q]$ contains encryptions of $\mathbf{0}$ in π_{s_1} and (is supposed to contain) encryptions of View^q_u in π_{s_2}.
We now consider the distribution over slots $([q, u, z^*_q, p^*_q]_{q \in [\lambda], u \in [\ell]})$. Note that by Claim 7.4 the distribution over these slots only is statistically close in the case that f gets as input a codeword with a simulated proof, versus a proof where all $S^{z,p}_{q,u}$ encrypt $\mathbf{0}$.
Therefore, our reduction R will construct a simulated proof $\pi' = ([\hat{k}^{z,p}_{q,u}, S^{z,p}_{q,u}]_{q \in [\lambda], u \in [\ell], z \in [\ell], p \in [2t]})$, for s_1 where all $S^{z,p}_{q,u}$ encrypt $\mathbf{0}$. Note that this simulated proof π' has *no dependence* on $(r^1_1, r^2_1) \ldots, (r^1_\lambda, r^2_\lambda)$, since all slots encrypt $\mathbf{0}$ so there is no information at all in the proof. R will then extract a proof $\pi'' = ([\hat{k}^{z,p}_{q,u}, S^{z,p}_{q,u}]_{q \in [\lambda], r \in [\ell], z \in [\ell], p \in [2t]})$ for some statement $s_2 \neq s_1$ from the tampered codeword. It will now choose random coins r^2_q and sets $\mathcal{S}^2_q \subseteq [\ell]$, for $q \in [\lambda]$ (using random coins r^2_q). Using Remark 2, we know that the probability over choice of random coins r^2_q that $z^*_q \in \mathcal{S}^2_q$ is at least $1/\ell$. Therefore, with all but negligible probability, there is a set $\mathcal{Q} \subseteq [\lambda]$ of size at least $1/2\ell \cdot \lambda$ such that $z^*_q \in \mathcal{S}^2_q$ for all $q \in \mathcal{Q}$. Moreover, note that if w.h.p. over choice of $r^1_1, \ldots, r^2_\lambda$, all checks for $[\hat{k}^{z,p}_{q,u}, S^{z,p}_{q,u}]_{q \in Q, u \in [\ell], z \in [\ell], p \in [2t]}$ pass then it must be the case that $[\hat{k}^{z^*_q, p^*_q}_{q,u}, S^{z^*_q, p^*_q}_{q,u}]_{q \in [Q], u \in [\ell]}$ is a proof for statement s_2 for proof system Π' for which $\mathsf{V}^{\Pi'}$ accepts w.h.p. But this breaks the soundness of proof system Π' with security parameter $\lambda' = |Q| \geq \lambda/2\ell$.

– **Hardness of D_b relative to Alternate Decoding.**

1.

$$\Pr[g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1],$$

2.

$$\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx \mathsf{D}'(\mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3),$$

where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, 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 $\boldsymbol{X} \leftarrow D_0$ with probability $1/2$ and $\boldsymbol{X} \leftarrow D_1$ 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 $\Pr[g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$ and $\Pr[g(\mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$ differ by a non-negligible amount. This implies that a circuit that takes as input $\boldsymbol{X}$, hardwires all other random variables, and outputs 1 in the case that $g(\mathsf{CW}, f(\mathsf{CW}), r) = 1$ and 0 otherwise, implying that it has non-negligible correlation to the function that outputs b such that $\boldsymbol{X}$ is in the support of D_b. We will show that the above can be computed by a streaming adversary with storage $o(n)$ and input $\boldsymbol{X}$, thus contradicting Claim 7.1. Indeed, this follows since the output of E_2 can be computed in a streaming fashion using a similar trick to the AC^0 case, f can be computed by streaming adversaries with storage $o(n)$ by definition of tampering class $\mathcal{F}$, and verification for $\mathit{\Pi}$ can also be computed in a streaming fashion with memory $\Theta(n'')$, where $n'' = o(n)$. So the composition of the three can also be computed by streaming adversaries with storage $o(n)$.

To show (2), assume $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$ have non-negligible statistical distance. This implies that a circuit that takes as input $\boldsymbol{X}$, hardwires all other random variables, and outputs $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$ has non-negligible correlation to the function that outputs b such that $\boldsymbol{X}$ is in the support of D_b. We will show that $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$ can be computed by a streaming adversary with storage $o(n)$ and input $\boldsymbol{X}$, thus contradicting Claim 7.1. To show this, note first that the output k_{n+1} of $\mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}))$ can be given as non-uniform advice since it does not depend on $\boldsymbol{X}$. This is the case because the key is extracted by looking at the first part of the tampered codeword, which is independent of $\boldsymbol{X}$. Since the tampering function is streaming as well, it means that the output of the tampering function was determined independently of $\boldsymbol{X}$.
Now, we must show that E_2, f, and $\mathsf{D}'(\mathsf{crs}, k_{n+1}, \cdot; r_2)$ can all be computed in a streaming fashion. We have already argued that E_2, f can be computed in a streaming fashion. Note that $\mathsf{D}'(\mathsf{crs}, k_{n+1}, \cdot; r_2)$ simply decrypts (by xor'ing k_{n+1} with c) and checks all proofs using the verifier of $\mathit{\Pi}$, which can be done in a streaming fashion, with space $\Theta(n'') = o(n)$.

7.5 Multi-Bit NMC Against Streaming Adversaries

The result from the previous section extends trivially for any number m of bits. Moreover, when we increase the number of bits m, all other parameters (n, n', n'') can remain the same and do not need to be increased as in our previous multi-bit constructions. To see this, note that the only additional property that needs to be proved in the multi-bit case is that for every Boolean function, represented by a circuit F over m variables,

$$F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3).$$

But in the bounded, streaming model, F as above can be computed without requiring any additional memory beyond what is required in the one-bit case. To see this, recall that the streaming adversary can receive the *decryptions* of the m ciphertexts in the tampered codeword as non-uniform advice, since tampering on this part of the codeword does not depend on the values of $[\boldsymbol{x}^i]_{i \in [m]}$. Thus, the streaming adversary needs only to check the $m \cdot n + 1$ proofs, in a streaming fashion, in order to determine the output of D': If all proofs verify correctly, the output of D' will consist of the hardcoded, "candidate" bits; otherwise, D' will output $\mathbf{0}$. Thus, the streaming adversary can compute the output of D' using the same amount of space as in the one-bit case. Now, F needs to be applied to the output of D'. But note that computing F does not require any additional space. Indeed, given the state of the streaming adversary at the moment the output of D' is determined, we can simply hardcode the output of F in the transition function. Thus, no additional memory is required.

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

AAG$^+$16. Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Kushilevitz and Malkin [KM16b], pages 393–417. 2, 7

ABW10. Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *42nd ACM STOC*, pages 171–180. ACM Press, June 2010. 22, 23

ADKO15a. Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 459–468. ACM Press, June 2015. 2, 7, 8

ADKO15b. Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In Dodis and Nielsen [DN15], pages 398–426. 8

ADL14. Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th ACM STOC*, pages 774–783. ACM Press, May / June 2014. 2, 7

AGM$^+$15a. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In Rosario Gennaro and Matthew Robshaw, editors, *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*, pages 538–557. Springer, 2015. 2

AGM$^+$15b. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 538–557. Springer, Heidelberg, August 2015. 7

AGM$^+$15c. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In Dodis and Nielsen [DN15], pages 375–397. 7

Ajt83. M Ajtai. σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:607–620, 1983. 23

BDKM16. Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 881–908. Springer, Heidelberg, May 2016. 2, 7, 8

BL16. Andrej Bogdanov and Chin Ho Lee. Homomorphic evaluation requires depth. In Kushilevitz and Malkin [KM16a], pages 365–371. 3, 4, 22

BRSV17. Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 483–496. ACM, 2017. 26

CCFP11. Hervé Chabanne, Gérard D. Cohen, Jean-Pierre Flori, and Alain Patey. Non-malleable codes from the wire-tap channel. *CoRR*, abs/1105.3879, 2011. 2

CDTV16. Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. In Kushilevitz and Malkin [KM16a], pages 306–335. 1

CG14a. Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In Moni Naor, editor, *ITCS 2014*, pages 155–168. ACM, January 2014. 2, 7

CG14b. Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In Lindell [Lin14], pages 440–464. 2, 8

CGL16. Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In Wichs and Mansour [WM16], pages 285–298. 2, 7

CGM$^+$15. Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. Cryptology ePrint Archive, Report 2015/129, 2015. http://eprint.iacr.org/2015/129. 8

CGM$^+$16. Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl, July 2016. 2, 3, 8

CKM11. Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. BiTR: Built-in tamper resilience. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 740–758. Springer, Heidelberg, December 2011. 2

CKO14. Nishanth Chandran, Bhavana Kanukurthi, and Rafail Ostrovsky. Locally updatable and locally decodable codes. In Lindell [Lin14], pages 489–514. 8

CKR16. Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In Kushilevitz and Malkin [KM16b], pages 367–392. 8

CL17. Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 1171–1184. ACM Press, June 2017. 2, 3, 7, 8

CMTV15. Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. In Dodis and Nielsen [DN15], pages 532–560. 1

CSS16. Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 1:1–1:35. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. 26

CZ14. Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *55th FOCS*, pages 306–315. IEEE Computer Society Press, October 2014. 7

CZ16. Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In Wichs and Mansour [WM16], pages 670–683. 1

DDO$^+$01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001. 12

DKO13. Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 239–257. Springer, Heidelberg, August 2013. 2, 7

DLSZ15. Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. In Dodis and Nielsen [DN15], pages 427–450. 8

DN15. Yevgeniy Dodis and Jesper Buus Nielsen, editors. *TCC 2015, Part I*, volume 9014 of *LNCS*. Springer, Heidelberg, March 2015. 43, 44, 45

DNO17. Nico Döttling, Jesper Buus Nielsen, and Maciej Obremski. 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. 8

DNR04. Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 342–360. Springer, Heidelberg, May 2004. 4, 11, 22, 23, 24

DPW10. Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 434–452. Tsinghua University Press, January 2010. 1, 2, 7, 8, 46

DSKS17. Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 310–332. Springer, Heidelberg, March 2017. 8

DW08. Ronald De Wolf. A brief introduction to fourier analysis on the boolean cube. *Theory of Computing, Graduate Surveys*, 1:1–20, 2008. 5

FHMV17. Sebastian Faust, Kristina Hostáková, Pratyay Mukherjee, and Daniele Venturi. Non-malleable codes for space-bounded tampering. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 95–126. Springer, Heidelberg, August 2017. 2, 3, 8

FMNV14. Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In Lindell [Lin14], pages 465–488. 8

FMNV15. Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A tamper and leakage resilient von neumann architecture. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 579–603. Springer, Heidelberg, March / April 2015. 8

FMVW14. Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 111–128. Springer, Heidelberg, May 2014. 2, 3, 7, 26

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 581–612. Springer, 2017. 27

GPR16. Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Wichs and Mansour [WM16], pages 1128–1141. 1

Hås87. Johan Håstad. Computational limitations of small-depth circuits. 1987. 15

Hås14. Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM Journal on Computing*, 43(5):1699–1708, 2014. 15

IMP12. Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for ac 0. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 961–972. Society for Industrial and Applied Mathematics, 2012. 15

JW15. Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In Dodis and Nielsen [DN15], pages 451–480. 8

KKS11. Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 373–390. Springer, Heidelberg, August 2011. 8

KLT16. Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Practical non-malleable codes from l-more extractable hash functions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1317–1328. ACM Press, October 2016. 7, 8

KM16a. Eyal Kushilevitz and Tal Malkin, editors. *TCC 2016-A, Part I*, volume 9562 of *LNCS*. Springer, Heidelberg, January 2016. 43

KM16b. Eyal Kushilevitz and Tal Malkin, editors. *TCC 2016-A, Part II*, volume 9563 of *LNCS*. Springer, Heidelberg, January 2016. 43, 44

KOS17a. Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 344–375. Springer, Heidelberg, November 2017. 2, 7

KOS17b. Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable randomness encoders and their applications. Cryptology ePrint Archive, Report 2017/1097, 2017. https://eprint.iacr.org/2017/1097. 2, 7

Li16. Xin Li. Improved two-source extractors, and affine extractors for polylogarithmic entropy. In Irit Dinur, editor, *57th FOCS*, pages 168–177. IEEE Computer Society Press, October 2016. 2, 7

Lin03. Yehuda Lindell. A simpler construction of cca2-secure public-key encryption under general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 241–254. Springer, Heidelberg, May 2003. 2, 3

Lin14. Yehuda Lindell, editor. *TCC 2014*, volume 8349 of *LNCS*. Springer, Heidelberg, February 2014. 44, 45

LL12. Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 517–532. Springer, Heidelberg, August 2012. 2, 7, 8

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990. 2, 3

OS17. Igor Carboni Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 665–677. ACM, 2017. 26

Rao07. Anup Rao. An exposition of bourgains 2-source extractor. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 14, 2007. 15

Raz16. Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *CoRR*, abs/1602.05161, 2016. 6, 15, 16

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. 2, 3, 12, 22

Tal13. Avishay Tal. Properties and applications of boolean function composition. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 441–454. ACM, January 2013. 36

Tal17. Avishay Tal. Tight bounds on the fourier spectrum of AC0. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 15:1–15:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. 6, 32

Vio14. Emanuele Viola. Extractors for circuit sources. *SIAM J. Comput.*, 43(2):655–672, 2014. 15

WM16. Daniel Wichs and Yishay Mansour, editors. *48th ACM STOC*. ACM Press, June 2016. 44, 45

A Standard Definitions and Preliminaries

A.1 Non-Malleability

The following Definitions 20, 21 are the standard definitions of non-malleability and strong non-mallebility, appropriate for the information theroetic setting (without CRS). These definitions are special cases of the corresponding Definitions 2, 3, when taking crs to be $\perp$ and $\mathcal{G}$ to be the set of all functions (namely the adversary is not restricted, and there's no CRS). Similarly, Definition 22 corresponds to a special case of Definition 4 of medium non-malleability that we introduced.

Definition 20 (Non-malleability [DPW10]). *Let k be the security parameter, $\mathcal{F}$ be some family of functions. For each function $f \in \mathcal{F}$, and $m \in \Sigma$, define the tampering experiment:*

$$\mathbf{Tamper}_m^f \stackrel{\text{def}}{=} \left\{ \begin{matrix} c \leftarrow \mathsf{E}(m), \tilde{c} := f(c), \tilde{m} := \mathsf{D}(\tilde{c}). \\ Output:\ \tilde{m}. \end{matrix} \right\},$$

where the randomness of the experiment comes from the encoding algorithm. We say a coding scheme (E, D) is non-malleable with respect to $\mathcal{F}$ if for each $f \in \mathcal{F}$, there exists a PPT simulator Sim such that for any message $m \in \Sigma$, we have

$$\mathbf{Tamper}_m^f \approx \mathbf{Ideal}_{\mathsf{Sim},m} \stackrel{\text{def}}{=} \left\{ \begin{matrix} \tilde{m} \cup \{\mathsf{same}^*\} \leftarrow \mathsf{Sim}^{f(\cdot)}. \\ Output: m \text{ if output of } \mathsf{Sim} \text{ is } \mathsf{same}^*;\ \text{otherwise } \tilde{m}. \end{matrix} \right\}$$

Here the indistinguishability can be either computational or statistical.

Definition 21 (Strong Non-malleability [DPW10]). *Let k be the security parameter, $\mathcal{F}$ be some family of functions. For each function $f \in \mathcal{F}$, and $m \in \Sigma$, define the tampering experiment*

$$\mathbf{StrongNM}_m^f \stackrel{\text{def}}{=} \left\{ \begin{matrix} c \leftarrow \mathsf{E}(m), \tilde{c} := f(c), \tilde{m} := \mathsf{D}(\tilde{c}) \\ Output:\ \mathsf{same}^* \text{ if } \tilde{c} = c, \text{ and } \tilde{m} \text{ otherwise.} \end{matrix} \right\}$$

The randomness of this experiment comes from the randomness of the encoding algorithm. We say that a coding scheme (E, D) is strong non-malleable with respect to the function family $\mathcal{F}$ if for any $m, m' \in \Sigma$ and for each $f \in \mathcal{F}$, we have:

$$\{\mathbf{StrongNM}_m^f\}_{k \in \mathbb{N}} \approx \{\mathbf{StrongNM}_{m'}^f\}_{k \in \mathbb{N}}$$

where $\approx$ can refer to statistical or computational indistinguishability.

Definition 22 (Medium Non-malleability). *Let k be the security parameter, $\mathcal{F}$ be some family of functions. Let $c \leftarrow \mathsf{E}(m)$ and let $g(c, \tilde{c}, r)$ be a predicate such that, for every c in the support of $\mathsf{E}(m)$ and every $\tilde{c}$,*

$$\Pr[g(c, \tilde{c}, r) = 1] \wedge \mathsf{D}(\tilde{c}; r) \neq m] \leq \mathsf{negl}(n).$$

For g as above, each function $f \in \mathcal{F}$, and $m \in \Sigma$, define the tampering experiment

$$\mathbf{MediumNM}^{f}_{m,g} \stackrel{\text{def}}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{E}(m), \tilde{c} := f(c), r \leftarrow U_\ell, \tilde{m} := \mathsf{D}(\tilde{c}; r) \\ \textit{Output}: \ \mathsf{same}^* \ \textit{if } g(c, \tilde{c}, r) = 1, \ \textit{and } \tilde{m} \ \textit{otherwise.} \end{array} \right\}$$

The randomness of this experiment comes from the randomness of the encoding algorithm and r (the random coins od decoding). We say that a coding scheme (E, D) is medium non-malleable with respect to the function family $\mathcal{F}$ if there exists a g as above and for any $m, m' \in \Sigma$ and for each $f \in \mathcal{F}$, we have:

$$\{\mathbf{MediumNM}^{f}_{m,g}\}_{k \in \mathbb{N}} \approx \{\mathbf{MediumNM}^{f}_{m',g}\}_{k \in \mathbb{N}}$$

where $\approx$ can refer to statistical or computational indistinguishability.

It is straightforward to check that Medium Non-Malleability implies non-malleability.

B Figure to explain MPC in head from section 7.4

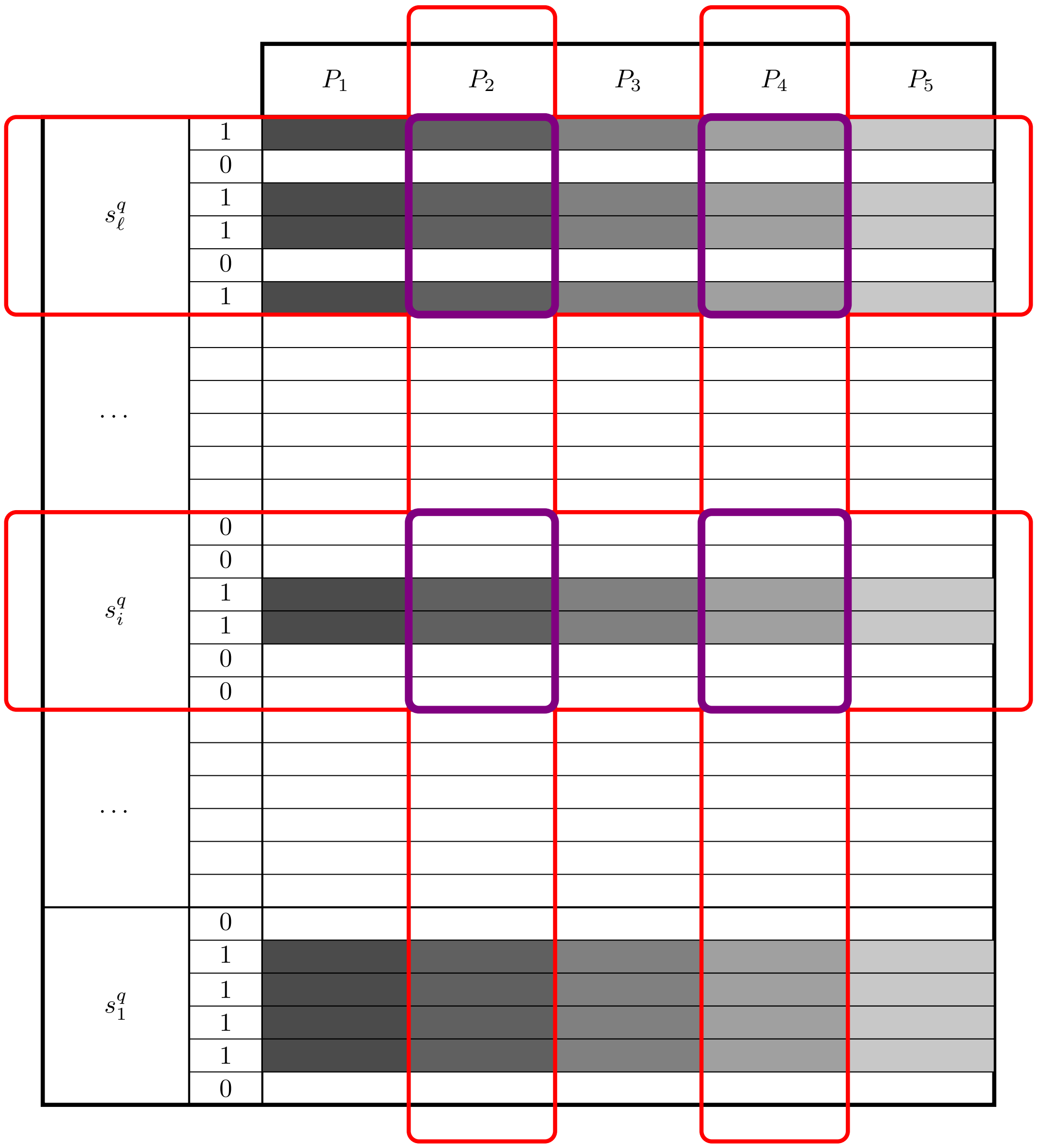

Fig. 18. A pictorial representation of the Prover's output in the NI Simulatable Proof System Π. Let $\ell = 5$ be the number of parties, and λ be the security parameter. In the q-th iteration, each party P_i for $i \in [\ell]$ has inputs (w_i^q, s_i^q). We encode each s_i^q as $s_i^q||\overline{s}_i^q$, where $\overline{s}_i^q$ is the bit-wise complement of s_i^q. For example 001 is encoded as 001100. For each bit of the encoding of s_i^q, if the bit is 1 then each party P_i places a weak encryption of its view, view_i, in the corresponding slot (represented by filled-in rectangles of various shades of gray in the figure). Otherwise, if the bit is 0 then each party places a weak encryption of all 0's in the corresponding slot, (represented by blank rectangles in the figure). During verification, the verifier checks in the first step that input $s = s_1^q \oplus s_2^q \oplus \ldots \oplus s_\ell^q$. To check the consistency of the views, the verifier selects 2 columns (P_2 and P_4) and 2 rows (s_i^q and s_ℓ^q) at random and does the following: (1) checks that s_i^q and s_ℓ^q are consistent with the values read in the first step (2) runs Rec on the weakly encrypted views and checks that the resulting views are consistent with each other and internally.