J Cryptol (2024)37:17
https://doi.org/10.1007/s00145-024-09498-2

Journal of CRYPTOLOGY

Research Article

(Continuous) Non-malleable Codes for Partial Functions with Manipulation Detection and Light Updates

Aggelos Kiayias
University of Edinburgh, Edinburgh, UK
IOG, Edinburgh, UK
akiayias@inf.ed.ac.uk

Feng-Hao Liu
Washington State University, Pullman, USA
feng-hao.liu@wsu.edu

Yiannis Tselekounis
Royal Holloway, University of London, Egham, UK
yiannis.tselekounis@rhul.ac.uk

Communicated by Shweta Agrawal.

Received 4 March 2021 / Revised 11 February 2024 / Accepted 23 February 2024
Online publication 3 April 2024

Abstract. Non-malleable codes were introduced by Dziembowski et al. (in: Yao (ed) ICS2010, Tsinghua University Press, 2010), and its main application is the protection of cryptographic devices against tampering attacks on memory. In this work, we initiate a comprehensive study on non-malleable codes for the class of partial functions, that read/write on an arbitrary subset of codeword bits with specific cardinality. We present two constructions: the first one is in the CRS model and allows the adversary to selectively choose the subset of codeword bits, while the latter is in the standard model and adaptively secure. Our constructions are efficient in terms of information rate, while allowing the attacker to access asymptotically almost the entire codeword. In addition, they satisfy a notion which is stronger than non-malleability, that we call non-malleability with manipulation detection, guaranteeing that any modified codeword decodes to either the original message or to $\perp$. We show that our primitive implies All-Or-Nothing Transforms (AONTs), and as a result our constructions yield efficient AONTs under standard assumptions (only one-way functions), which, to the best of our knowledge, was an open question until now. Furthermore, we construct a notion of continuous non-malleable codes (CNMC), namely CNMC with light updates, that avoids the full re-encoding process and only uses shuffling and refreshing operations. Finally, we present a number of additional applications of our primitive in tamper resilience.

Keywords. Non-malleable codes, Manipulation detection, Tamper resilience, All-or-nothing transforms.

© The Author(s) 2024

1. Introduction

Non-malleable codes (NMC) were introduced by Dziembowski, Pietrzak and Wichs [39] as a relaxation of error correction and error detection codes, aiming to provide strong privacy but relaxed correctness. Informally, non-malleability guarantees that any modified codeword decodes either to the original message or to a completely unrelated one, with overwhelming probability. The definition of non-malleability is simulation-based, stating that for any tampering function f, there exists a simulator that simulates the tampering effect by only accessing f, i.e., without making any assumptions on the distribution of the encoded message.

The main application of non-malleable codes that motivated the seminal work by Dziembowski et al. [39] is the protection of cryptographic implementations from *active physical attacks* against memory, known as *tampering attacks*. In this setting, the adversary modifies the memory of the cryptographic device, receives the output of the computation, and tries to extract sensitive information related to the private memory. Security against such types of attacks can be achieved by encoding the private memory of the device using non-malleable codes. Besides that, various applications of non-malleable codes have been proposed in subsequent works, such as CCA secure encryption schemes [29] and non-malleable commitments [4].

Due to their important applications, constructing non-malleable codes has received a lot of attention over recent years. As non-malleability against general functions is impossible [39], various subclasses of tampering functions have been considered, such as split-state functions [1–3,38,39,51,55], bit-wise tampering and permutations [4,5,39], bounded-size function classes [45], bounded depth/fan-in circuits [14], space-bounded tampering [42], and others (cf. Sect. 1.4). One characteristic shared by those function classes is that they allow *full access* to the codeword, while imposing structural or computational restrictions to the way the function computes over the input. In this work, we initiate a comprehensive study on non-malleability for functions that receive *partial access* over the codeword, which is an important yet overlooked class, as we elaborate below.

The class of partial functions. The class of *partial functions* contains all functions that read/write on an arbitrary subset of codeword bits with specific cardinality. The elements of the subset can be chosen selectively or adaptively. Concretely, let c be a codeword with length ν. For $\alpha \in [0, 1)$, the function class $\mathcal{F}^{\alpha\nu}$ (or $\mathcal{F}^{\alpha}$ for brevity) consists of all functions that operate over any subset of bits of c with cardinality at most $\alpha\nu$, while leaving the remaining bits unseen and intact. The work of Cheraghchi and Guruswami [27] explicitly defines this class and uses a subclass (the one containing functions that always touch the first $\alpha\nu$ bits of the codeword) in a negative way, namely as the tool for deriving capacity lower bounds for *information-theoretic* non-malleable codes against split-state functions. Partial functions were also studied implicitly by Faust et al. [45], while aiming for non-malleability against bounded-size circuits.[1]

[1] Specifically, in [45], the authors consider a model where a common reference string (CRS) is available, with length roughly logarithmic in the size of the tampering function class; as a consequence, the tampering function is allowed to read/write the whole codeword while having only partial information over the CRS.

Even though capacity lower bounds for partial functions have been derived (cf. [27]), our understanding about *explicit* constructions is still limited. Existential results can be derived by the probabilistic method, as shown in prior works [27,39],[2] but they do not yield explicit constructions. On the other hand, the capacity bounds do not apply to the computational setting, which could potentially allow more practical solutions. We believe that this is a direction that needs to be explored, as besides the theoretical interest, partial functions is a natural model that complies with existing attacks that require partial access to the registers of the cryptographic implementation [16,19–21,63].[3]

Besides the importance of partial functions in the active setting, i.e., when the function is allowed to partially *read/write* the codeword, the passive analogue of the class, i.e., when the function is only given *read access* over the codeword, matches the model considered by All-Or-Nothing Transforms (AONTs), which is a notion originally introduced by Rivest [60], providing security guarantees similar to those of leakage resilience: reading an arbitrary subset (up to some bounded cardinality) of locations of the codeword does not reveal the underlying message. As non-malleable codes provide privacy, non-malleability for partial functions is the active analogue of (and in fact implies) AONTs, that find numerous applications [22,23,59,60,62].

Plausibility. At a first glance, one might think that partial functions better comply with the framework of error-correction/detection codes (ECC/EDC), as they do not touch the whole codeword. However, if we allow the adversary to access asymptotically almost the entire codeword (which can be more than the minimum distance that an ECC can achieve), it is conceivable it can use this generous *access rate*, i.e., the fraction of the codeword that can be accessed (see below), to create correlated encodings; thus, we believe solving non-malleability in this setting is a natural question. Additionally, ECC/EDC cannot guarantee security against *selective failure attack* for high access rate tampering adversaries and thus provide weaker security compared with the simulation-based non-malleable codes. Below we elaborate.

We illustrate the separation between the notions using the following example. Consider the set of partial functions that operate either on the right or on the left half of the codeword (the function chooses if it is going to be left or right), and the trivial encoding scheme that on input message s outputs (s, s). The decoder, on input (s, s'), checks if $s = s'$, in which case it outputs s, otherwise it outputs $\perp$. This scheme is clearly an EDC against the aforementioned function class,[4] as the output of the decoder is in $\{s, \perp\}$, with probability 1; however, it is considered malleable since the tampering function can create encodings whose validity depends on the message. On the other hand, an ECC would provide a trivial solution in this setting; however, it requires restriction of the adversarial access fraction to 1/2 (of the codeword); by accessing more than this fraction, the attacker can possibly create invalid encodings depending on the message, as general ECCs do not provide privacy. Thus, the ECC/EDC setting is inapt against this type of selective failure tampering in the presence of attackers that access almost the

[2] Informally, prior works [27,39] showed existence of non-malleable codes for classes of certain bounded cardinalities. The results cover the class of partial functions.

[3] The attacks by [16,20,21] require the modification of a single (random) memory bit, while in [19] a single error per each round of the computation suffices. In [63], the attack requires a single faulty byte.

[4] It is not an ECC as the decoder does not know which side has been modified by the tampering function.

entire codeword. Later in this section, we provide an extensive discussion on challenges of non-malleability for partial functions.

Besides the plausibility and the lack of a comprehensive study, partial functions can potentially allow stronger primitives, as constant functions are excluded from the class. This is similar to the path followed by Jafargholi and Wichs [49], aiming to achieve *tamper detection* (cf. Sect. 1.4) against a class of functions that implicitly excludes constant functions and the identity function. In this work, we prove that this intuition holds, by showing that partial functions allow a stronger primitive that we define as *non-malleability* with *manipulation detection* (MD-NMC), which guarantees that any tampered codeword will either decode to the original message or to $\perp$, and additionally, the outcome can be simulated without knowing the underlying message. This implies that the notion can defend against the selective failure tampering attacks, as the decoding of the tampered outcome would not only be either the original message or $\perp$, but also independent of the underlying message. Thus, MD-NMC is not subject to the selective failure attacks, providing stronger security than what ECC/EDC can provide as we argued above.

Given the above, we believe that partial functions is an interesting and well-motivated model. The goal of this work is to answer the following (informally stated) question:

> *Is it possible to construct efficient (high information rate) non-malleable codes for partial functions, while allowing the attacker to access almost the entire codeword?*

We answer the above question in the affirmative. Before presenting our results (cf. Sect. 1.1) and the high level ideas behind our techniques (cf. Sect. 1.2), we identify the several challenges that are involved in tackling the problem.

Challenges. We first define some useful notions used throughout the paper.

- *Information rate*: the ratio of message to codeword length, as the message length goes to infinity.
- *Access rate*: the fraction of the number of bits that the attacker is allowed to access over the total codeword length, as the message length goes to infinity.

The access rate measures the effectiveness of a non-malleable code in the partial function setting and reflects the level of adversarial access to the codeword. In this work, we aim at constructing non-malleable codes for partial functions with high *information rate* and high *access rate*, i.e., both rates should approach 1 simultaneously. Before discussing the challenges posed by this requirement, we first review some known impossibility results. First, non-malleability for partial functions with concrete access rate 1 is impossible, as the function can fully decode the codeword and then re-encode a related message [39]. Second, information-theoretic non-malleable codes with constant information rate (e.g., 0.5) are not possible against partial functions with constant access rate [27],[5] and consequently, solutions in the information-theoretic settings such as ECC and Robust Secret Sharing (RSS) do not solve our problem. Based on these facts, in order to achieve

[5] Informally, in [27] (Theorem 5.3) the authors showed that any information-theoretic non-malleable code with a constant access rate and a constant information rate must have a constant distinguishing probability.

our goal, the only path is to explore the computational setting, aiming for access rate at most $1 - \epsilon$, for some $\epsilon > 0$.

At a first glance, one might think that non-malleability for partial functions is easier to achieve, compared to other function classes, as partial functions cannot touch the whole codeword. Having that in mind, it would be tempting to conclude that existing designs/techniques with minor modifications are sufficient to achieve our goal. However, we will show that this intuition is misleading, by pointing out why prior approaches fail to provide security against partial functions with high access rate.

The current state of the art in the computational setting considers tools such as (Authenticated) Encryption [1,35,36,41,51,55], *non-interactive zero-knowledge* (NIZK) proofs [35,41,43,55], and ℓ-more extractable collision resistant hashes (ECRH) [51], where others use KEM/DEM techniques [1,36]. Those constructions share a common structure, incorporating a short secret key sk (or a short encoding of it), as well as a long ciphertext, e, and a proof π (or a hash value). Now, consider the partial function f that gets full access to the secret key sk and a constant number of bits of the ciphertext e, partially decrypts e and modifies the codeword depending on those bits. Then, it is not hard to see that non-malleability falls apart as the security of the encryption no longer holds. The attack requires access rate only $O((|sk|)/(|sk| + |e| + |\pi|))$, for [35,41,55] and $O(\text{poly}(k)/|s|)$ for [1,36,51]. A similar attack applies to [43], which is in the continual setting.

One possible route to tackle the above challenges is to use an encoding scheme over the ciphertext, such that partial access over it does not reveal the underlying message.[6] The guarantees that we need from such a primitive resemble the properties of AONTs; however, this primitive does not provide security against active, i.e., tampering, attacks. Another approach would be to use Reconstructable Probabilistic Encodings [14] which provide error-correcting guarantees, yet still it is unknown whether we can achieve information rate 1 for such a primitive. In addition, the techniques and tools for protecting the secret key can be used to achieve optimal information rate as they are independent of the underlying message, yet at the same time, they become the weakest point against partial functions with high access rate. Thus, the question is how to overcome the above challenges, allowing access to almost the entire codeword.

In this paper, we solve the challenges presented above based on the following observation: in existing solutions the structure of the codeword is fixed and known to the attacker, and independently of the primitives that we use, the only way to resolve the above issues is by hiding the structure via randomization. This requires a structure recovering mechanism that can either be implemented by an "external" source, or otherwise the structure needs to be reflected in the codeword in some way that the attacker cannot exploit. In the present work, we implement this mechanism in both ways, by first proposing a construction in the *common reference string* (CRS) model, and then we show how to remove the CRS using slightly bigger alphabets. Refer to Sect. 1.2 for a technical overview.

[6] In the presence of NIZKs, we can have attacks with low access rate that read sk, e, and constant number of bits from the proof.

1.1. *Our Results*

We initiate the study of *non-malleable codes* with *manipulation detection* (MD-NMC), and we present the first (to our knowledge) construction for this type of codes. We focus on achieving simultaneously high *information rate* and high *access rate*, in the partial functions setting, which by the results of [27], it can be achieved only in the computational setting.

Our contribution is threefold. First, we construct an information rate 1 non-malleable code in the CRS model, with access rate $1 - 1/\Omega(\log k)$, where k denotes the security parameter. Our construction combines Authenticated Encryption together with an inner code that protects the key of the encryption scheme (cf. Sect. 1.2). The result is informally summarized in the following theorem.

Theorem 1.1. (Informal) *Assuming one-way functions, there exists an explicit computationally secure* MD-NMC *over the binary alphabet in the* CRS *model, against selective selection of codeword locations, achieving information rate* 1 *and access rate* $1 - 1/\Omega(\log k)$*, where k is the security parameter.*

Our scheme, in order to achieve security with error $2^{-\Omega(k)}$, produces codewords of length $|s| + O(k^2 \log k)$, where $|s|$ denotes the length of the message and uses a CRS of length $O(k^2 \log k \log(|s| + k))$. We note that our construction does not require the CRS to be fully tamper-proof, and we refer the reader to Sect. 1.2 for a discussion on the topic.

In our second result, we show how to remove the CRS by slightly increasing the size of the alphabet. Our result is a computationally secure MD-NMC in the standard model, achieving information and access rate $1-1/\Omega(\log k)$. Our construction is proven secure by a reduction to the security of the scheme presented in Theorem 1.1. Below, we informally state our result.

Theorem 1.2. (Informal) *Assuming one-way functions, there exists an explicit, computationally secure* MD-NMC *in the standard model against adaptive selection of codeword locations, with alphabet length* $O(\log k)$*, information rate* $1 - 1/\Omega(\log k)$ *and access rate* $1 - 1/\Omega(\log k)$*, where k is the security parameter.*

Our scheme produces codewords of length $|s|(1 + 1/O(\log k)) + O(k^2 \log^2 k)$.

In Sect. 1.2, we consider security against continuous attacks. We show how to achieve a weaker notion of continuous security, while avoiding the use of a self-destruct mechanism, which was originally achieved by [41]. Our notion is weaker than full continuous security [43], since the codewords need to be updated with a mechanism that is heavier than self-destruct, still it is deterministic and more efficient than the re-encoding process of [39,55]; it uses only shuffling and refreshing operations, i.e., we avoid cryptographic computations such as group operations and NIZKs. We call such an update mechanism a "light update." Informally, we prove the following result.

Theorem 1.3. (Informal) *One-way functions imply continuous non-malleable codes with deterministic light updates and without self-destruct against adaptive selection of* *codeword locations, in the standard model, with alphabet length* $O(\log k)$*, information rate* $1 - 1/\Omega(\log k)$ *and access rate* $1 - 1/\Omega(\log k)$*, where k is the security parameter.*

As we have already stated, non-malleable codes against partial functions imply AONTs [60]. The first AONT was presented by Boyko [22] in the random oracle model, and then, Canetti et al. [23] consider AONTs with public/private parts as well as a secret-only part, which is the full notion. Canetti et al. [23] provide efficient constructions for both settings, yet the fully secure AONT (called "secret-only" in that paper) is based on non-standard assumptions.[7]

Assuming one-way functions, our results yield efficient, fully secure AONTs, in the standard model. This resolves, the open question left in [23], where the problem of constructing AONT under standard assumptions was posed. Our result is presented in the following theorem.

Theorem 1.4. (Informal) *Assuming one-way functions, there exists an explicit secret-only* AONT *in the standard model, with information rate* 1 *and access rate* $1-1/\Omega(\log k)$*, where k is the security parameter.*

The above theorem is derived by the Informal Theorem 1.1, yielding an AONT whose output consists of both the CRS and the codeword produced by the NMC scheme in the CRS model. A similar theorem can be derived with respect to the Informal Theorem 1.2. Finally, and in connection to AONTs that provide leakage resilience, our results imply leakage-resilient codes [55] for partial functions. In Sect. 2.3, we present the connection between MD-NMC and AONT with a formal description.

We provide concrete instantiations of our constructions, using textbook instantiations [50] for the underlying authenticated encryption scheme. For completeness, we also provide information theoretic variants of our constructions that maintain high access rate and thus necessarily sacrifice information rate.

1.2. *Technical Overview*

On the manipulation detection property. In the present work, we exploit the fact that the class of partial functions does not include constant functions and we achieve a notion that is stronger than non-malleability, which we call *non-malleability* with *manipulation detection*. We formalize this notion as a strengthening of non-malleability, and we show that our constructions achieve this stronger notion. Informally, manipulation detection ensures that any tampered codeword will either decode to the original message or to $\perp$.

A MD-NMC **in the CRS model.** For the exposition of our ideas, we start with a naive scheme (which does not work) and then show how we resolve all the challenges. Let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a (symmetric) authenticated encryption scheme and consider the following encoding scheme: to encode a message s, the encoder computes $(sk||e)$, where $e \leftarrow \mathsf{E}_{sk}(s)$ is the ciphertext and $sk \leftarrow \mathsf{KGen}(1^k)$, is the secret key. We observe that the scheme is secure if the tampering function can only read/write on the ciphertext, e, assuming the authenticity property of the encryption scheme, however, restricting

[7]In [62] the authors present a deterministic AONT construction that provides weaker security.

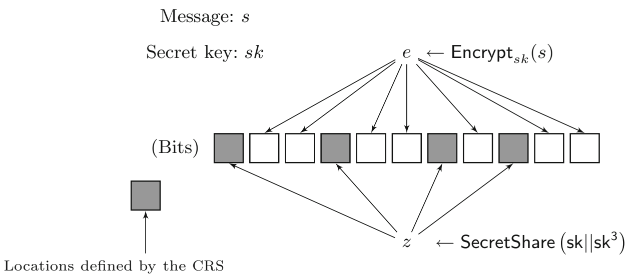
Message: s
Secret key: sk
$e \leftarrow \mathsf{Encrypt}_{sk}(s)$
(Bits)
Locations defined by the CRS
$z \leftarrow \mathsf{SecretShare}(\mathsf{sk}||\mathsf{sk}^3)$

Fig. 1. Description of the scheme in the CRS model.

access to sk, which is short, is unnatural and makes the problem trivial. On the other hand, even partial access to sk, compromises the authenticity property of the scheme, and even if there is no explicit attack against the non-malleability property, there is no hope for proving security based on the properties of (KGen, E, D), in black-box way.

A solution to the above problems would be to protect the secret key using an inner encoding, yet the amount of tampering is now restricted by the capabilities of the inner scheme, as the attacker knows the exact locations of the "*sensitive*" codeword bits, i.e., the non-ciphertext bits. In our construction, we manage to protect the secret key while avoiding the bottleneck on the access rate by designing an inner encoding scheme that provides limited security guarantees when used standalone, still when it is used in conjunction with a *shuffling technique* that permutes the inner encoding and ciphertext bit locations, it guarantees that any attack against the secret key will create an invalid encoding with overwhelming probability, even when allowing access to *almost the entire* codeword (Figs. 1, 2).

Our scheme is depicted in Fig. 3 and works as follows: on input message s, the encoder (i) encrypts the message by computing $sk \leftarrow \mathsf{KGen}(1^k)$ and $e \leftarrow \mathsf{E}_{sk}(s)$, (ii) computes an m-out-of-m secret sharing z of $(sk||sk^3)$ (interpreting both sk and sk^3 as elements in some finite field),[8] and outputs a random shuffling of $(z||e)$, denoted as $P_\Sigma(z||e)$, according to the common reference string Σ. Decoding proceeds as follows: on input c, the decoder (i) inverts the shuffling operation by computing $(z||e) \leftarrow P_\Sigma^{-1}(c)$, (ii) reconstructs $(sk||sk')$, and (iii) if $sk^3 = sk'$, outputs $\mathsf{D}_{sk}(e)$, otherwise, it outputs $\perp$.

Intuitively, the properties that we require from the inner encoding scheme, i.e., the secret sharing and shuffling of $(sk||sk^3)$, are similar to those provided by a robust secret sharing scheme [58], which guarantees tamper detection during the reconstruction phase. In our work, we additionally require simulatability of whether the reconstructed message will be the same or $\perp$. In Sect. 3, we present the intuition behind our construction and a formal security analysis. Our instantiation yields a rate 1 computationally secure MD-NMC in the CRS model, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s| + O(k^2 \log k)$, under mild assumptions (e.g., one-way functions).

[8] In general, any polynomial of small degree, e.g., sk^c, would suffice, depending on the choice of the underlying finite field. Using sk^3 suffices when working over fields of characteristic 2. We could also use sk^2 over fields of characteristic 3 (but not of characteristic 2).

On the CRS. In our work, the tampering function, and consequently the codeword locations that the function is given access to, are fixed before sampling the CRS and this is critical for achieving security. However, proving security in this setting is non-trivial. In addition, the tampering function receives full access to the CRS when tampering with the codeword. This is in contrast to the work by Faust et al. [45] in the information-theoretic setting, where the (internal) tampering function receives partial information over the CRS.

In addition, our results tolerate adaptive selection of the codeword locations, with respect to the CRS, in the following way: each time the attacker requests access to a location, he also learns if it corresponds to a bit of z or e, together with the index of that bit in the original string. In this way, the CRS is gradually disclosed to the adversary while picking codeword locations.

Finally, our CRS sustains a substantial amount of tampering that depends on the codeword locations chosen by the attacker: an attacker that gets access to a sensitive codeword bit is allowed to modify the part of the CRS that defines the location of that bit in the codeword. The attacker is allowed to modify all but $O(k \log(|s| + k))$ bits of the CRS, that is of length $O(k^2 \log k \log(|s| + k))$. To our knowledge, this is the first construction that tolerates, even partial modification of the CRS. In contrast, existing constructions in the CRS model are either using NIZKs [35,41,43,55], or they are based on the *knowledge of exponent assumption* [51], thus tampering access to the CRS might compromise security.

Removing the CRS. A first approach would be to store the CRS inside the codeword together with $P_\Sigma(z||e)$ and give to the attacker read/write access to it. However, the tampering function, besides getting direct (partial) access to the encoding of sk, it also gets indirect access to it by (partially) controlling the CRS. Then, it can modify the CRS in way such that, during decoding, ciphertext locations of its choice will be treated as bits of the inner encoding, z, increasing the tampering rate against z significantly. This makes the task of protecting sk hard, if not impossible (unless we restrict the access rate significantly).

To handle this challenge, we embed the structure recovering mechanism inside the codeword and we emulate the CRS effect by increasing the size of the alphabet, giving rise to a block-wise structure.[9] Notice that, non-malleable codes with large alphabet size (i.e., poly(k) + $|s|$ bits) might be easy to construct, as we can embed in each codeword block the verification key of a signature scheme together with a secret share of the message, as well as a signature over the share. In this way, partial access over the codeword does not compromise the security of the signature scheme while the message remains private, and the simulation is straightforward. This approach, however, comes with a large overhead, decreasing the information rate and access rate of the scheme significantly. In general, and similar to error correcting codes, we prefer smaller alphabet sizes—the larger the size is, the more coarse access structure is required, i.e., in order to access individual bits we need to access the blocks that contain them. In this work, we aim at minimizing this restriction by using small alphabets as below.

[9] Bigger alphabets have been also considered in the context of error-correcting codes, in which the codeword consists of symbols.

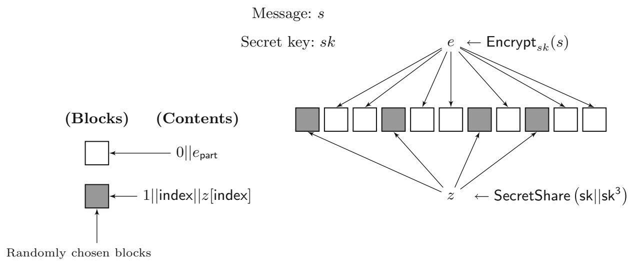
Message: s
Secret key: sk
$e \leftarrow \mathsf{Encrypt}_{sk}(s)$
(Blocks)
(Contents)
$0||e_{\mathsf{part}}$
$1||\mathsf{index}||z[\mathsf{index}]$
Randomly chosen blocks
$z \leftarrow \mathsf{SecretShare}(\mathsf{sk}||\mathsf{sk}^3)$

Fig. 2. Description of the scheme in the standard model.

Our approach on the problem is the following. We increase the alphabet size to $O(\log k)$ bits, and we consider two types of blocks: (i) *sensitive blocks*, in which we store the inner encoding, z, of the secret key, sk, and (ii) *non-sensitive* blocks, in which we store the ciphertext, e, that is fragmented into blocks of size $O(\log k)$. The first bit of each block indicates whether it is a sensitive block, i.e., we set it to 1 for sensitive blocks and to 0, otherwise. Our encoder works as follows: on input message s, it computes z, e, as in the previous scheme and then uses sampling without replacement to generate the indices, $\rho_1, \dots, \rho_{|z|}$, for the sensitive blocks. Then, for every $i \in \{1, \dots, |z|\}$, ρ_i is a sensitive block, with contents $(1||i||z[i])$, while the remaining blocks keep ciphertext pieces of size $O(\log k)$. Decoding proceeds as follows: on input codeword $C = (C_1, \dots, C_{\mathsf{bn}})$, for each $i \in [\mathsf{bn}]$, if C_i is a non-sensitive block, its data will be part of e, otherwise, the last bit of C_i will be part of z, as it is dictated by the index stored in C_i. If the number of sensitive blocks is not the expected, the decoder outputs $\perp$, otherwise, z, e, have been fully recovered and decoding proceeds as in the previous scheme. Our scheme is depicted in Fig. 5.

The security of our construction is based on the fact that, due to our shuffling technique, the position mapping will not be completely overwritten by the attacker, and as we prove in Sect. 4, this suffices for protecting the inner encoding over sk. We prove security of the current scheme (cf. Theorem 4.8) by a reduction to the security of the scheme in the CRS model. Our instantiation yields a rate $1 - 1/\Omega(\log k)$ MD-NMC in the standard model, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s|(1 + 1/O(\log k)) + O(k^2 \log^2 k)$, assuming one-way functions.

It is worth pointing out that the idea of permuting blocks containing sensitive and non-sensitive data was also considered by [61] in the context of list-decodable codes; however, the similarity is only in the fact that a permutation is being used at some point in the encoding process, and our objective, construction and proof are different.

Continuously non-malleable codes with light updates. We observe that the codewords of the block-wise scheme can be updated efficiently, using shuffling and refreshing operations. Based on this observation, we prove that our code is secure against continuous attacks, for a notion of security that is weaker than the original one [43], as we need to update our codeword. However, our update mechanism is using cheap operations, avoiding the full decoding and re-encoding of the message, which is the standard way to achieve continuous security [39,55]. In addition, our solution avoids the usage of a self-destruction mechanism that produces $\perp$ in all subsequent rounds after the first round in which the attacker creates an invalid codeword, which was originally achieved by [41], and makes an important step toward practicality.

The update mechanism works as follows: in each round, it randomly shuffles the blocks and refreshes the randomness of the inner encoding of sk. The idea here is that, due to the continual shuffling and refreshing of the inner encoding scheme, in each round the attacker learns nothing about the secret key, and every attempt to modify the inner encoding, results to an invalid key, with overwhelming probability. Our update mechanism can be made deterministic if we further encode a seed of a PRG together with the secret key, which is similar to the technique presented in [55].

Our results are presented in Sect. 5 (cf. Theorem 5.3), and the rates for the current scheme match those of the one-time secure, block-wise code.

1.3. *Applications*

Security against passive attackers - AONTs. Regarding the passive setting, our model and constructions find useful application in all settings where AONTs are useful (cf. [22,23,59,60]), e.g., for increasing the security of encryption without increasing the key-size, for improving the efficiency of block ciphers and constructing remotely keyed encryption [22,60], and also for constructing computationally secure secret sharing [59]. Other uses of AONTs are related to optimal asymmetric encryption padding [22].

Security against memory tampering - (Binary alphabets, Logarithmic length CRS). As with every NMC, the most notable application of the proposed model and constructions is when aiming for protecting cryptographic devices against memory tampering. Using our CRS based construction, we can protect a large tamperable memory with a small (logarithmic in the message length) tamperproof memory, that holds the CRS.

The construction is as follows. Consider any device performing cryptographic operations, e.g., a smart card, whose memory is initialized when the card is being issued. Each card is initialized with an independent CRS, which is stored in a tamper-proof memory, while the codeword is stored in a tamperable memory. Due to the independency of the CRS values, it is plausible to assume that the adversary is not given access to the CRS prior to tampering with the card; the full CRS is given to the tampering function while it tampers with the codeword during computation. This idea is along the lines of the *only computation leaks information* model [56], where data can only be leaked during computation, i.e., the attacker learns the CRS when the devices perform computations that depend on it. We note that in this work we allow the tampering function to read the full CRS, in contrast to [45], in which the tampering function receives partial information over it (our CRS can also be tampered, cf. the above discussion). In subsequent rounds, the CRS and the codeword are being updated by the device, which is the standard way to achieve security in multiple rounds while using a one-time NMC[39].

Security against memory tampering - (Logarithmic length alphabets, no CRS). In modern architectures, data are stored and transmitted in chunks; thus, our block-wise encoding scheme can provide tamper resilience in all these settings. For instance, consider the case of arithmetic circuits, having memory consisting of consecutive blocks storing integers. Considering adversaries that access the memory of such circuits in a block-wise manner, is a plausible scenario. In terms of modeling, this is similar to tamper resilience for arithmetic circuits [47], in which the attacker, instead of accessing individual circuit wires carrying bits, accesses wires carrying integers. The case is similar for RAM computation where the CPU operates over 32 or 64 bit registers (securing RAM programs using NMC was also considered by [34–36,44]). We note that the memory segments in which the codeword blocks are stored do not have to be physically separated, as partial functions output values that depend on the whole input in which they receive access to. This is in contrast to the split-state setting in which the tampering function tampers with each state independently, and thus, the states need to be physically separated.

Security against adversarial channels. In Wiretap Channels [17,57,64] the goal is to communicate data privately against eavesdroppers, under the assumption that the channel between the sender and the adversary is "noisier" than the channel between the sender and the receiver. The model that we propose and our block-wise construction can be applied in this setting to provide privacy against a wiretap adversary under the assumption that due to the gap of noise there is a small (of rate $o(1)$) fraction of symbols that are delivered intact to the receiver and dropped from the transmission to the adversary. This enables private, key-less communication between the parties, guaranteeing that the receiver will either receive the original message, or $\perp$. In this way, the communication will be non-malleable in the sense that the receiver cannot be lead to output $\perp$ depending on any property of the plaintext. Our model allows the noise in the receiver side to depend on the transmission to the wiretap adversary, that tampers with a large (of rate $1 - o(1)$) fraction of symbols, leading to an "active" variant of the wiretap model.

1.4. *Related Work*

Manipulation detection has been considered independently of the notion of non-malleability, in the seminal paper by Cramer et. al. [30], who introduced the notion of *algebraic manipulation detection* (AMD) codes, providing security against additive attacks over the codeword. A similar notion was considered by Jafargholi and Wichs [49], called *tamper detection*, aiming to detect malicious modifications over the codeword, independently of how those affect the output of the decoder. Tamper detection ensures that the application of any (admissible) function to the codeword leads to an invalid decoding.

Non-malleable codes for other function classes have been extensively studied, such as constant split-state functions [26,37], block-wise tampering [24,28], while the work of [2] develops beautiful connections among various function classes. There has been even richer classes studied in recent years, such as small depth circuits [12], bounded-degree polynomials over finite fields [11], and bounded polynomial time/depth functions [13, 33,40]. The results [11,12] are information-theoretic, and the others [13,33,40] require stronger complexity/cryptographic assumptions. On the other hand, the constructions of this work only rely on the minimal cryptographic assumption—the existence of one-way functions.

In addition, other variants of non-malleable codes have been proposed, such as continuous non-malleable codes [43], augmented non-malleable codes [1], locally decodable/updatable non-malleable codes [25,34–36,44], and non-malleable codes with split-state refresh [41]. In [15], the authors consider AC0 circuits, bounded-depth decision trees and streaming, space-bounded adversaries. Leakage resilience was also considered as an additional feature, e.g., by [25,36,41,53,55].

A related line of work in tamper resilience aims to protect circuit computation against tampering attacks on circuit wires [31,32,46,48] or gates [54], [9,10] aim at protecting circuits against hardware Trojans, while [18] relies on trusted hardware. In this setting, using non-malleable codes for protecting the circuit's private memory is an option, still in order to achieve security the encoding and decoding procedures should be protected against fault injection attacks using the techniques from [31,32,46,48,54]. The work of [52] is the first that constructs (one-time) NMCs for the class of partial functions that tamper with almost the entire codeword. Whether NMCs could be useful in secure messaging remains an interesting open question [6–8].

2. Preliminaries

In this section, we present basic definitions and notation that will be used throughout the paper.

Definition 2.1. (*Notation*) Let t, i, j, be nonnegative integers. Then, $[t]$ is the set $\{1,\ldots,t\}$. For bit strings x, y, $x||y$, is the concatenation of x, y, $|x|$ denotes the length of x, for $i \in [|x|]$, $x[i]$ is the i-th bit of x, $\|_{j=1}^{t} x_j := x_1||\ldots||x_t$, and for $i \leq j$, $x[i:j] = x[i]||\ldots||x[j]$. For a set I, $|I|$, $\mathcal{P}(I)$, are the cardinality and power set of I, respectively, and for $I \subseteq [|x|]$, $x_{|_I}$ is the projection of the bits of x with respect to I. For a string variable c and value v, $c \leftarrow v$ denotes the assignment of v to c, and $c[I] \leftarrow v$, denotes an assignment such that $c_{|_I}$ equals v. For a distribution D over a set $\mathcal{X}$, $x \leftarrow D$ denotes sampling an element $x \in \mathcal{X}$, according to D, $x \leftarrow \mathcal{X}$ denotes sampling a uniform element x from $\mathcal{X}$, $U_{\mathcal{X}}$ denotes the uniform distribution over $\mathcal{X}$ and $x_1,\ldots,x_t \overset{\mathsf{rs}}{\leftarrow} \mathcal{X}$ denotes sampling a uniform subset of $\mathcal{X}$ with t distinct elements, using rejection sampling. The statistical distance between two random variables X, Y, is denoted by $\Delta(X,Y)$, "$\approx$" and "$\approx_c$", denote statistical and computational indistinguishability, respectively, and $\mathsf{negl}(k)$ denotes an unspecified, negligible function, in k.

2.1. *Non-malleable Codes*

Below, we define coding schemes, based on the definitions of [39,55].

Definition 2.2. (*Coding scheme* [39]) A (κ,ν)-coding scheme, $\kappa,\nu \in \mathbb{N}$, is a pair of algorithms $(\mathsf{Enc},\mathsf{Dec})$ such that: $\mathsf{Enc}: \{0,1\}^\kappa \to \{0,1\}^\nu$ is an encoding algorithm, $\mathsf{Dec}: \{0,1\}^\nu \to \{0,1\}^\kappa \cup \{\perp\}$ is a decoding algorithm, and for every $s \in \{0,1\}^\kappa$, $\Pr[\mathsf{Dec}(\mathsf{Enc}(s)) = s] = 1$, where the probability runs over the randomness used by $(\mathsf{Enc},\mathsf{Dec})$.

We can easily generalize the above definition for larger alphabets, i.e., by considering $\mathsf{Enc}: \{0,1\}^\kappa \to \Gamma^\nu$ and $\mathsf{Dec}: \Gamma^\nu \to \{0,1\}^\kappa \cup \{\perp\}$, for some alphabet Γ.

Definition 2.3. **(Coding scheme in the Common Reference String (CRS) Model** [55]) A (κ, ν)-coding scheme in the CRS model, $\kappa, \nu \in \mathbb{N}$, is a triple of algorithms $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ such that: Init is a randomized algorithm which receives 1^k, where k denotes the security parameter, and produces a common reference string $\Sigma \in \{0,1\}^{\mathrm{poly}(k)}$, and $(\mathsf{Enc}(1^k, \Sigma, \cdot), \mathsf{Dec}(1^k, \Sigma, \cdot))$ is a (κ, ν)-coding scheme, $\kappa, \nu = \mathrm{poly}(k)$.

For brevity, 1^k will be omitted from the inputs of Enc and Dec.

Below we define *non-malleable codes* with *manipulation detection*, which is a stronger notion than the one presented in [39], in the sense that the tampered codeword will always decode to the original message or to $\perp$. Our definition is with respect to alphabets, as in Sect. 4 we consider alphabets of size $O(\log k)$.

Definition 2.4. (*Non-Malleability with Manipulation Detection* (MD-NMC)) Let Γ be an alphabet, let $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ be a (κ, ν)-coding scheme in the common reference string model, and $\mathcal{F}$ be a family of functions $f : \Gamma^\nu \to \Gamma^\nu$. For any $f \in \mathcal{F}$ and $s \in \{0,1\}^\kappa$, define the tampering experiment

$$\mathsf{Tamper}_s^f := \left\{ \begin{array}{c} \Sigma \leftarrow \mathsf{Init}(1^k), c \leftarrow \mathsf{Enc}(\Sigma, s), \tilde{c} \leftarrow f_\Sigma(c), \tilde{s} \leftarrow \mathsf{Dec}(\Sigma, \tilde{c}) \\ \text{Output} : \tilde{s}. \end{array} \right\}$$

which is a random variable over the randomness of Enc, Dec and Init. The coding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ is non-malleable with manipulation detection with respect to the function family $\mathcal{F}$, if for all, sufficiently large k and for all $f \in \mathcal{F}$, there exists a distribution $D_{(\Sigma, f)}$ over $\{0,1\}^\kappa \cup \{\perp, \mathsf{same}^*\}$, such that for all $s \in \{0,1\}^\kappa$, we have:

$$\left\{\mathsf{Tamper}_s^f\right\}_{k\in\mathbb{N}} \approx \left\{ \begin{array}{c} \tilde{s} \leftarrow D_{(\Sigma,f)} \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^*, \text{ and } \perp \text{ otherwise} \end{array} \right\}_{k\in\mathbb{N}}$$

where $\Sigma \leftarrow \mathsf{Init}(1^k)$ and $D_{(\Sigma,f)}$ is efficiently samplable given access to f, Σ. Here, "$\approx$" may refer to statistical, or computational, indistinguishability.

In the above definition, f is parameterized by Σ to differentiate tamper-proof input, i.e., Σ, from tamperable input, i.e., c.

2.2. *Partial Functions*

Below we define the tampering function class that will be used throughout the paper.

Definition 2.5. (*The class of partial functions* $\mathcal{F}_\Gamma^{\alpha\nu}$ (*or* $\mathcal{F}^\alpha$)) Let Γ be an alphabet, $\alpha \in [0,1)$ and $\nu \in \mathbb{N}$. Any $f \in \mathcal{F}_\Gamma^{\alpha\nu}$, $f : \Gamma^\nu \to \Gamma^\nu$, is indexed by a set $I \subseteq [\nu]$, $|I| \leq \alpha\nu$, and a function $f' : \Gamma^{\alpha\nu} \to \Gamma^{\alpha\nu}$, such that for any $x \in \Gamma^\nu$, $(f(x))_{|_I} = f'\left(x_{|_I}\right)$ and $(f(x))_{|_{I^\mathsf{c}}} = x_{|_{I^\mathsf{c}}}$, where $I^\mathsf{c} := [\nu] \backslash I$.

For simplicity, in the rest of the text we will use the notation $f(x)$ and $f(x_{|_I})$ (instead of $f'\left(x_{|_I}\right)$). Also, the length of the codeword, ν, according to Γ, will be omitted from the notation, and whenever Γ is omitted we assume that $\Gamma = \{0,1\}$. In Sect. 3, we consider $\Gamma = \{0,1\}$, while in Sect. 4, $\Gamma = \{0,1\}^{O(\log k)}$, i.e., the tampering function operates over blocks of size $O(\log k)$. When considering the CRS model, the functions are parameterized by the common reference string.

The following lemma is useful for proving security throughout the paper.

Lemma 2.6. *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a* (κ, ν)*-coding scheme and* $\mathcal{F}$ *be a family of functions. For every* $f \in \mathcal{F}$ *and* $s \in \{0,1\}^\kappa$*, define the tampering experiment*

$$\mathsf{Tamper}_s^f := \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \textit{Output } \mathsf{same}^* \textit{ if } \tilde{s} = s, \textit{ and } \tilde{s} \textit{ otherwise.} \end{array} \right\}$$

which is a random variable over the randomness of Enc *and* Dec*.* $(\mathsf{Enc}, \mathsf{Dec})$ *is an* MD-NMC *with respect to* $\mathcal{F}$*, if for any* $f \in \mathcal{F}$ *and all sufficiently large* k*: (i) for any pair of messages* $s_0, s_1 \in \{0,1\}^\kappa$*,* $\left\{\mathsf{Tamper}_{s_0}^f\right\}_{k \in \mathbb{N}} \approx \left\{\mathsf{Tamper}_{s_1}^f\right\}_{k \in \mathbb{N}}$*, and (ii) for any* s*,* $\Pr\left[\mathsf{Tamper}_s^f \notin \{\bot, s\}\right] \leq \mathsf{negl}(k)$*. Here, "*$\approx$*" may refer to statistical, or computational, indistinguishability.*

Proof. By Definition 2.4, we have that $(\mathsf{Enc}, \mathsf{Dec})$ is an MD-NMC against $\mathcal{F}$, if for any $f \in \mathcal{F}$, there exists an efficiently samplable distribution D_f over $\{0,1\}^k \cup \{\bot, \mathsf{same}^*\}$, such that for any message s

$$\left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \text{Output} : \tilde{s} \end{array} \right\} \approx \left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^*, \text{ and } \bot \text{ otherwise} \end{array} \right\} \tag{1}$$

Let $\mathsf{0}$ be the zero message in $\{0,1\}^\kappa$. For any $f \in \mathcal{F}$, we define D_f as follows:

- Sample $c \leftarrow \mathsf{Enc}(\mathsf{0})$ and compute $\tilde{c} \leftarrow f(c)$, $\tilde{s} \leftarrow \mathsf{Dec}(\tilde{c})$.
- **Output**: if $\tilde{s} = \mathsf{0}$, set $\tilde{s} \leftarrow \mathsf{same}^*$, else, $\tilde{s} \leftarrow \bot$. Output $\tilde{s}$.

From the above, we have that for any s,

$$\begin{aligned} &\left\{ \begin{array}{c} \tilde{s} \leftarrow D_f \\ \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^*, \text{ and } \bot \text{ otherwise} \end{array} \right\} \\ &\equiv \left\{ \begin{array}{c} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(\mathsf{0}), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \text{if } \tilde{s} = \mathsf{0}, \tilde{s} \leftarrow \mathsf{same}^*, \text{else}, \tilde{s} \leftarrow \bot. \text{ Output } \tilde{s} \end{array} \right\} \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^*, \text{ and } \bot \text{ otherwise} \end{array} \right\} \\ &\approx \left\{ \begin{array}{c} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \text{if } \tilde{s} = s, \tilde{s} \leftarrow \mathsf{same}^*, \text{else}, \tilde{s} \leftarrow \bot. \text{ Output } \tilde{s} \end{array} \right\} \\ \text{Output } s \text{ if } \tilde{s} = \mathsf{same}^*, \text{ and } \bot \text{ otherwise} \end{array} \right\} \\ &\approx \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{c}) \\ \text{Output} : \tilde{s} \end{array} \right\}, \end{aligned}$$

where the first relation follows by the definition of D_f, the second one follows from the main assumption which states that for any pair of messages s_0, s_1, $\mathsf{Tamper}_{s_0}^f \approx \mathsf{Tamper}_{s_1}^f$, and the third one follows from the assumption that $\Pr\left[\mathsf{Tamper}_s^f \notin \{\bot, s\}\right] \leq$

$\mathsf{negl}(k)$. This concludes our proof since for any $f \in \mathcal{F}$ and any message s, Relation 1 is satisfied. □

For coding schemes in the CRS model the above lemma is similar, and Tamper_s^f internally samples $\Sigma \leftarrow \mathsf{Init}(1^k)$.

2.3. *All-Or-Nothing-Transform*

Here we present the definition of all-or-nothing-transform (AONT), by adopting the notion of [23] and presenting it with respect to coding schemes against partial codeword leakage.

Definition 2.7. Let Γ be an alphabet, and $(\mathsf{Enc}, \mathsf{Dec})$ be a (κ, ν) coding scheme over the alphabet Γ. The encoding is an AONT for parameter $\alpha \in (0, 1)$ if for any pair of messages (s_0, s_1) and every subset $I \subset [\nu]$ with cardinality $\alpha\nu$, we have $\left(s_0, s_1, \mathsf{Enc}(s_0)_{|_I}\right) \approx \left(s_0, s_1, \mathsf{Enc}(s_1)_{|_I}\right)$, where the indistinguishability may be information-theoretic or computational.

Next, we present a simple theorem, showing that MD-NMC against partial functions (with a sufficiently large access rate) implies AONT.

Theorem 2.8. *Let Γ be an alphabet, and* $(\mathsf{Enc}, \mathsf{Dec})$ *be a* (κ, ν) *coding scheme over the alpha bet Γ. Suppose the coding scheme is* MD-NMC *with respect to the partial function class $\mathcal{F}^\alpha$ for some $\alpha > 0.5$, then the coding scheme is an* AONT *with parameter α.*

Proof. We first note that by Lemma 2.6, an equivalent formulation of MD-NMC against an $f \in \mathcal{F}^\alpha$ can be stated as: (1) for any messages s_0, s_1, we have $\left\{\mathsf{Tamper}_{s_0}^f\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{Tamper}_{s_1}^f\right\}_{k\in\mathbb{N}}$, and (2) for any s, $\Pr\left[\mathsf{Tamper}_s^f \notin \{\bot, s\}\right] \leq \mathsf{negl}(k)$.

We prove the theorem via a reduction. Assume that there exist a subset I with $|I| = \alpha\nu$, messages s_0, s_1, and a distinguisher $\mathcal{D}$ that breaks the AONTsecurity; then, we construct a reduction $\mathcal{A}$ that uses a carefully chosen function $f \in \mathcal{F}^\alpha$ to break the MD-NMCsecurity. Without loss of generality, we assume that s_0 and s_1 are both nonzero messages.

First we define the function f: the function has hardcoded (s_0, s_1) and receives as input a partial codeword C^*. f first runs the distinguisher, i.e., computes $b^* = \mathcal{D}(s_0, s_1, C^*)$. If $b^* = 1$, then f outputs $\mathsf{Enc}(0^\kappa)_{|_I}$; otherwise, f acts as the identity function, i.e., just outputting the input C^*. Now, we prove that the coding scheme is not an MD-NMC.

First, we assume that $\Pr\left[\mathsf{Tamper}_s^f \notin \{\bot, s\}\right] = \mathsf{negl}(k)$ for any message s. Otherwise, the manipulation detection property is broken, implying a contradiction. Then, we claim that in either case of $s = s_0$ or $s = s_1$, we have $\mathsf{Tamper}_s^f = \bot$, with an overwhelming probability. We observe that f has changed an $\alpha > 0.5$ fraction of the codeword into $\mathsf{Enc}(0^\kappa)_{|_I}$. This effect is the same as another related tampering function $g \in \mathcal{F}^\alpha$ who changes a $(1 - \alpha) < 0.5$ fraction of $\mathsf{Enc}(0^\kappa)$ into $\mathsf{Enc}(s)$ on the set $[\nu] \setminus I$, meaning that $\mathsf{Tamper}_s^f \approx \mathsf{Tamper}_{0^\kappa}^g$. By assumption, we have

$\Pr\left[\mathsf{Tamper}_s^f \in \{\perp, s\}\right] = \Pr\left[\mathsf{Tamper}_{0^\kappa}^g \in \{\perp, 0^\kappa\}\right] = 1 - \mathsf{negl}(k)$, implying that $\mathsf{Tamper}_s^f = \mathsf{Tamper}_{0^\kappa}^g = \perp$, with overwhelming probability.

Next, we show that $\left\{\mathsf{Tamper}_{s_0}^f\right\}_{k\in\mathbb{N}} \not\approx \left\{\mathsf{Tamper}_{s_1}^f\right\}_{k\in\mathbb{N}}$. We notice that for $b \in \{0, 1\}$, $\mathsf{Tamper}_{s_b}^f$ outputs $\perp$ with probability the same as that of $\mathcal{D}(s_0, s_1, \mathsf{Enc}(s_b)_{|I})$ outputting 1. As the distinguisher $\mathcal{D}$ has a non-negligible gap outputting 1 between $b = 0$ and $b = 1$, the two tampering experiments can be distinguished with non-negligible probability. This breaks the non-malleability property of the coding scheme, reaching a contradiction. □

By the above theorem, we derive that any MD-NMC code against partial functions with sufficiently large access rate, i.e., for $\mathcal{F}^\alpha$ for $\alpha > 0.5$, is also an AONT with the same parameter α. On the other hand, we notice that $\alpha > 0.5$ is necessary for the theorem, as otherwise we can construct a simple counter example—first we observe that the repetition code with majority decoding[10] is an MD-NMC against $\mathcal{F}^\beta$ for any $\beta < 0.5$, as any tampering function in this class cannot change the outcome of the decoding. However, this is certainly not an AONT, as reading one bit is sufficient to recover the underlying message.

2.4. *One-Time Authenticated Encryption*

Below, we define the security notion of *authenticated encryption* required by our construction of non-malleable codes.

Definition 2.9. (*Authenticated encryption*) Let k be the security parameter and $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme. Then, $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is authenticated, semantically secure against (1-query) key recovery attacks, if it satisfies the following properties:

1. **(Correctness)**: For every message s, $\Pr[\mathsf{D}_{sk}(\mathsf{E}_{sk}(s)) = s] = 1$, where $sk \leftarrow \mathsf{KGen}(1^k)$.
2. **(Semantic security)**: for any pair of messages s_0, s_1, $\left(\mathsf{E}_{sk}(s_0)\right) \approx \left(\mathsf{E}_{sk}(s_1)\right)$, where $sk \leftarrow \mathsf{KGen}(1^k)$.
3. **(1-query key recovery security)**: for any message s and any adversary $\mathcal{A}$, we have

$$\Pr\left[sk' = sk \,\middle|\, \begin{array}{c} sk \leftarrow \mathsf{KGen}(1^k); \\ e \leftarrow \mathsf{E}_{sk}(s);\, sk' \leftarrow \mathcal{A}(e) \end{array}\right] \le \mathsf{negl}(k).$$

4. **(Unforgeability)**: For any algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr\left[\tilde{e} \neq e \wedge \mathsf{D}_{sk}(\tilde{e}) \neq \perp \,\middle|\, \begin{array}{c} sk \leftarrow \mathsf{KGen}(1^k);\, (s, st) \leftarrow \mathcal{A}_1(1^k); \\ e \leftarrow \mathsf{E}_{sk}(s);\, \tilde{e} \leftarrow \mathcal{A}_2(e, st) \end{array}\right] \le \mathsf{negl}(k).$$

[10] That is, to encode a bit b, the codeword is simply outputting b^k (concatenation of k b's). To decode, the algorithm just outputs the majority.

When the scheme is computationally secure, we consider computational indistinguishability instead of statistical, and $\mathcal{A}$ is PPT .

We notice that the notions of semantic security and key-recovery are not compatible just from the definitions, i.e., key-recovery security clearly does not imply semantic security, and the other way implication does not hold in general, i.e., if the key space is small (e.g., one-time pad with bit messages and keys), there is always a $1/2$ probability guessing the key correctly. Nevertheless, below we instantiate two simple schemes which satisfy the key-recovery security.

Next we provide the definition of *one-time message authentication code* (MAC) following [50].

Definition 2.10. (*One-time* MAC [50]) Let k be the security parameter. A message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is one-time ϵ-secure, if for all algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr[\mathsf{Mac-forge}_{\mathcal{A},\Pi}(k) = 1] \leq \epsilon,$$

where,

$$\begin{aligned}&\mathsf{Mac-forge}_{\mathcal{A},\Pi}(k) :\\ &sk \leftarrow \mathsf{Gen}(1^k)\\ &(s, st) \leftarrow \mathcal{A}_1(1^k)\\ &t \leftarrow \mathsf{Mac}_{sk}(s)\\ &(\tilde{s}, \tilde{t}) \leftarrow \mathcal{A}_2(t, st)\\ &\text{Output 1 if } \mathsf{Vrfy}_{sk}(\tilde{s}, \tilde{t}) = 1 \text{ and } \tilde{s} \neq s.\end{aligned}$$

Below we describe two instantiations of one-time authenticated encryption; the first is a computationally secure rate 1 scheme, while the latter is information-theoretically secure with a lower rate.

Instantiation 2.11. (Computationally secure authenticated encryption) *Let F_r be a pseudo-random function, $F_r : \{0, 1\}^k \rightarrow \{0, 1\}^k$, let* PRG *be a pseudo-random generator,* $\mathsf{PRG} : \{0, 1\}^k \rightarrow \{0, 1\}^{|s|}$, *and let* $(\mathsf{MKGen}, \mathsf{Mac}, \mathsf{Vrfy})$ *be a message authentication code that outputs tags of length k (cf. [50]). We define a symmetric encryption scheme* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, *as follows:*

- $\mathsf{KGen}(1^k)$*: sample $r \leftarrow \{0, 1\}^k$, $mk \leftarrow$* $\mathsf{MKGen}(1^k)$ *and output $sk = (r, mk)$.*
- $\mathsf{E}_{sk}(\cdot)$*: On input s, sample $\tau \leftarrow \{0, 1\}^k$, set $e = (\mathsf{PRG}(F_r(\tau)) \oplus s, \tau)$, $t =$ $\mathsf{Mac}_{mk}(e)$, and output (e, t).*
- $\mathsf{D}_{sk}(\cdot)$*: On input (e, t), if $\mathsf{Vrfy}_{mk}(e, t) = 1$, parse e as (e', τ) and output $s =$ $\big(\mathsf{PRG}(F_r(\tau)) \oplus e'\big)$, otherwise output $\perp$.*

It is not hard to see that the scheme defined above is a rate 1, computationally secure authenticated encryption scheme [50]. The semantic security follows from the security of PRF/PRG. From the security of the pseudo-random function, the scheme is also secure against 1-query key recovery attack. Particularly, from any adversary who can break the 1-query key recovery security, we can easily derive a reduction who can invert the PRF, i.e., recovering the key r given $(F_r(\tau), \tau)$. The reduction can then be used to distinguish $F_r(\cdot)$ from the truly random function given oracle access, as the truly random function cannot be compressed into a short key r, following the compression argument.

Next we describe a simple one-time information theoretic construction.

Instantiation 2.12. (Information-theoretically secure authenticated encryption) *Let $\mathcal{H}$, $\bar{\mathcal{H}}$, be pair-wise independent hash function families, such that for any $h \in \mathcal{H}$, $h : \{0,1\}^{O(|s|)} \rightarrow \{0,1\}^{|s|}$ and for any $\bar{h} \in \bar{\mathcal{H}}$, $\bar{h} : \{0,1\}^{O(|s|)} \rightarrow \{0,1\}^{|s|}$. We define a symmetric encryption scheme* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$*, as follows:*

- $\mathsf{KGen}(1^k)$*: sample $h \leftarrow \mathcal{H}$, $\bar{h} \leftarrow \bar{\mathcal{H}}$ and set $sk = (h, \bar{h})$.*
- $\mathsf{E}_{sk}(\cdot)$*: On input s, sample $r \leftarrow \{0,1\}^{|s|}$, set $e = (r||(h(r) + s))$ and output $(e, \bar{h}(e))$.*
- $\mathsf{D}_{sk}(\cdot)$*: On input (e, t), if $\bar{h}(e) = t$, parse e as $(r||e')$ and output $s = h(r) + e'$, otherwise output $\perp$.*

It is easy to verify that the security (i.e., semantic, unforgeability) of the above scheme comes from the pair-wise independence of $\mathcal{H}$, $\bar{\mathcal{H}}$. As long as $|h| > |s| + k$, then the conditional entropy of sk given e is still greater to k, meaning that the information-theoretic adversary has at most 2^{-k} probability to predict sk successfully.

2.5. *Secret Sharing*

In this section, we present the definition and a concrete instantiation of the m-out-of-m secret sharing scheme later used in this work.

Definition 2.13. An m-out-of-m secret sharing scheme has the following two algorithms $(\mathsf{SS}_m, \mathsf{Rec}_m)$ that works as follow:

- SS_m: on input x outputs shares $(z_1, \ldots, z_m)$. Denote $z = ||_{i=1}^{m} z_i$, where $||$ denotes concatenation.
- Rec_m: on input shares $(z_1, \ldots, z_m)$ (denoted as z as above) outputs the message x.

The correctness requires that $\mathsf{Rec}_m(\mathsf{SS}_m(x)) = x$ holds with probability 1. The security requires that the message x is information-theoretically hidden given any proper subset of shares $(z_1, \ldots, z_m)$.

Instantiation. Given any finite field $\mathbf{GF}(p^e)$ where p is some prime and e is some non-negative integer, there is a simple additive m-out-of-m secret sharing scheme that works as follow. SS_m takes input $x \in \mathbf{GF}(p^e)$ and samples uniformly random shares $(z_1, \ldots, z_m)$ where each $z_i \in \mathbf{GF}(p^e)$ and $x = \sum_{i \in [m]} z_i$. Similarly, Rec_m takes input $(z_1, \ldots, z_m)$ and just outputs $\sum_{i \in [m]} z_i$. It is easy to verify that both the correctness and security hold.

Remark. In this work, we always refer the particular instantiation above as $(\mathsf{SS}_m, \mathsf{Rec}_m)$. Moreover, to simplify the presentation, we use $\mathsf{SS}_m(x \| y)$ to denote $\mathsf{SS}_m(x) \| \mathsf{SS}_m(y)$ for sharing multiple inputs. For this case, the reconstruction works analogously.

3. An MD-NMC for Partial Functions, in the CRS Model

In this section, we consider $\Gamma = \{0, 1\}$ and we construct a rate 1 MD-NMC for $\mathcal{F}^{\alpha}$, with access rate $\alpha = 1 - 1/\Omega(\log k)$.

Before presenting the encoding scheme for $\mathcal{F}^{\alpha}$, we provide the intuition behind the construction. As a staring point, we consider a naive scheme (which does not work) and then show how we resolve all the challenges. Let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a (symmetric) authenticated encryption scheme and consider the following encoding scheme: to encode a message s, the encoder computes $(sk||e)$, where $e \leftarrow \mathsf{E}_{sk}(s)$ is the ciphertext and $sk \leftarrow \mathsf{KGen}(1^k)$, is the secret key. We observe that the scheme is secure if the tampering function can only read/write on the ciphertext, e, assuming the authenticity property of the encryption scheme, however, restricting access to sk, which is short, is unnatural and makes the problem trivial. On the other hand, even partial access to sk, compromises the authenticity property of the scheme, and even if there is no explicit attack against the non-malleability property of the code, there is no hope for proving security based on the properties of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, in a black-box way.

A solution to the above problems would be to protect the secret key using an inner encoding, yet the amount of tampering is now restricted by the capabilities of the inner scheme, as the attacker knows the exact locations of the "*sensitive*" codeword bits, i.e., the non-ciphertext bits. In the proposed construction, we manage to protect the secret key while avoiding the bottleneck on the access rate, by designing an inner encoding scheme that provides limited security guarantees when used standalone, still when it is used in conjunction with a *shuffling technique* that permutes the inner encoding and ciphertext bit locations, it guarantees that any attack against the secret key will create an invalid encoding with overwhelming probability, even when allowing access to *almost the entire* codeword.

The proposed scheme is depicted in Fig. 3 and works as follows: on input message s, the encoder (i) encrypts the message by computing $sk \leftarrow \mathsf{KGen}(1^k)$ and $e \leftarrow \mathsf{E}_{sk}(s)$, (ii) computes an m-out-of-m secret sharing, z, of $(sk||sk^3)$ (interpreting both sk and sk^3 as elements in some finite field),[11] and outputs a random shuffling of $(z||e)$, denoted as $P_{\Sigma}(z||e)$, according to the common reference string, Σ. Decoding proceeds as follows: on input c, the decoder (i) inverts the shuffling operation by computing $(z||e) \leftarrow P_{\Sigma}^{-1}(c)$, (ii) reconstructs $(sk||sk')$, and (iii) if $sk^3 = sk'$, it outputs $\mathsf{D}_{sk}(e)$, otherwise, it outputs $\perp$. The proposed instantiation yields a rate 1 computationally secure MD-NMC in the CRS model, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s| + O(k^2 \log k)$, under mild assumptions, e.g., one-way functions.

Below, we formally define our construction.

Construction 3.1. *Let $k, m \in \mathbb{N}$, let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme, $(\mathsf{SS}_m, \mathsf{Rec}_m)$ be an m-out-of-m secret sharing scheme as Sect. 2.5, and let $l \leftarrow 2m|sk|$, where sk follows $\mathsf{KGen}(1^k)$. We define an encoding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, that outputs $\nu = l + |e|$ bits, $e \leftarrow \mathsf{E}_{sk}(s)$, as follows:*

[11] In general, any polynomial of small degree, e.g., sk^c, would suffice, depending on the choice of the underlying finite field. Using sk^3 suffices when working over fields of characteristic 2. We could also use sk^2 over fields of characteristic 3.

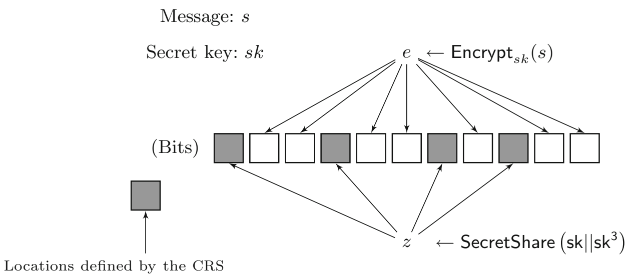
Message: s
Secret key: sk
$e \leftarrow \mathsf{Encrypt}_{sk}(s)$
(Bits)
$z \leftarrow \mathsf{SecretShare}(\mathsf{sk}||\mathsf{sk}^3)$
Locations defined by the CRS

Fig. 3. Description of the MD-NMC scheme in the CRS model .

– $\mathsf{Init}(1^k)$: *Sample* $r_1, \ldots, r_l \overset{\mathsf{rs}}{\leftarrow} \{0,1\}^{\log(\nu)}$, *and output* $\Sigma := (r_1, \ldots, r_l)$.
– $\mathsf{Enc}(\Sigma, \cdot)$: *for input message* s, *sample* $sk \leftarrow \mathsf{KGen}(1^k)$, $e \leftarrow \mathsf{E}_{sk}(s)$.

  • **(Secret share)** *Sample* $z \leftarrow \mathsf{SS}_m(sk||sk^3)$, *where* $z = ||_{i=1}^{2|sk|} z_i$, $z \in \{0,1\}^{2m|sk|}$, *and for* $i \in [|sk|]$, z_i *(resp.* $z_{|sk|+i}$*) is an* m*-out-of-*m *secret sharing of* $sk[i]$ *(resp.* $sk^3[i]$*).*
  • **(Shuffle)** *Compute* $c \leftarrow P_\Sigma(z||e)$ *as follows:*

    1. **(Sensitive bits)**: *Set* $c \leftarrow 0^\nu$. *For* $i \in [l]$, $c[r_i] \leftarrow z[i]$.
    2. **(Ciphertext bits)**: *Set* $i \leftarrow 1$. *For* $j \in [l + |e|]$, *if* $j \notin \{r_p \mid p \in [l]\}$: $c[j] \leftarrow e[i]$, $i{+}{+}$.

  Output c.
– $\mathsf{Dec}(\Sigma, \cdot)$: *on input* c, *compute* $(z||e) \leftarrow P_\Sigma^{-1}(c)$, $(sk||sk') \leftarrow \mathsf{Rec}_m(z)$, *and if* $sk^3 = sk'$, *output* $\mathsf{D}_{sk}(e)$, *otherwise output* $\perp$.

The set of indices of z_i *in the codeword will be denoted by* Z_i.

In the above, we consider sk, sk^3, as elements over $\mathbf{GF}(2^{\text{poly}(k)})$.

In a high level, the construction presented above combines authenticated encryption with an inner encoding that works as follows. It interprets sk as an element in the finite field $\mathbf{GF}(2^{|sk|})$ and computes sk^3 as a field element. Then, for each bit of $(sk||sk^3)$, it computes an m-out-of-m secret sharing of the bit, for some parameter m (we note that elements in $\mathbf{GF}(2^{|sk|})$ can be interpreted as bit strings). Then, by combining the inner encoding with the shuffling technique, we get an encoding scheme whose security follows from the observations that we briefly present below:

– For any tampering function which does not have access to all m shares of a single bit of $(sk||sk^3)$, the tampering effect on the secret key can be expressed essentially as a linear shift, i.e., as $((sk+\delta)||(sk^3+\eta))$ for some $(\delta, \eta) \in \mathbf{GF}(2^{|sk|}) \times \mathbf{GF}(2^{|sk|})$, independent of sk.
– By permuting the locations of the inner encoding and the ciphertext bits, we have that with overwhelming probability any tampering function who reads/writes on a $(1 - o(1))$ fraction of codeword bits, will not learn any single bit of $(sk||sk^3)$.
– With overwhelming probability over the randomness of sk and the CRS, for non-zero η and δ, $(sk+\delta)^3 \neq sk^3+\eta$, and this property enables us to design a consistency check mechanism whose output is simulatable, without accessing sk.

– The security of the final encoding scheme follows by composing the security of the inner encoding scheme with the authenticity property of the encryption scheme.

Intuitively, the properties that we require from the inner encoding scheme (after the shuffling operation) employed by our construction are similar to those provided by a robust secret sharing scheme [58], which guarantees tamper detection during the reconstruction phase. In our work, we additionally require simulatability of whether the reconstructed message will be the same or $\perp$.

Below we present the formal security proof of the above ideas.

Theorem 3.2. *Let $k, m \in \mathbb{N}$ and $\alpha \in [0, 1)$. Assuming $(\mathsf{SS}_m, \mathsf{Rec}_m)$ is an m-out-of-m secret sharing scheme and $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is* 1-IND-CPA *secure (cf. Definition 2.9),*[12] *authenticated encryption scheme, the code of Construction 3.1 is a* MD-NMC *against $\mathcal{F}^{\alpha}$ (cf. Definition 2.5), for any α, m, such that $(1-\alpha)m = \omega(\log(k))$.*

Proof. Let I be the set of indices chosen by the attacker and $I^{\mathsf{c}} = [\nu]\setminus I$, where $\nu = 2m|sk| + |e|$. The tampered components of the codeword will be denoted using the symbol "~" on top of the original symbol, i.e., we have $\tilde{c} \leftarrow f(c)$, the tampered secret key sk (resp. sk^3) that we get after executing $\mathsf{Rec}_m(\tilde{z})$ will be denoted by $\tilde{sk}$ (resp. $\tilde{sk}'$). Also the tampered ciphertext will be $\tilde{e}$. We prove the needed using a series of hybrid experiments that are depicted in Fig. 4. Below, we describe the hybrids.

- $\mathsf{Exp}_0^{\Sigma,f,s}$: We prove security of our code using Lemma 2.6, i.e., by showing that (i) for any s_0, s_1, $\mathsf{Tamper}_{s_0}^f \approx \mathsf{Tamper}_{s_1}^f$, and (ii) for any s, $\Pr\left[\mathsf{Tamper}_s^f \notin \{\perp, s\}\right] \leq \mathsf{negl}(k)$, where Tamper_s^f is defined in Lemma 2.6. For any f, s, the first experiment, $\mathsf{Exp}_0^{\Sigma,f,s}$, matches the experiment Tamper_s^f in the CRS model, where Σ is sampled by Tamper_s^f.
- $\mathsf{Exp}_1^{\Sigma,f,s}$: In the second experiment we define Z_i, $i \in [2|sk|]$, to be the set of codeword indices in which the secret sharing z_i is stored, $|Z_i| = m$. The main difference from the previous experiment is that the current one outputs $\perp$, if there exists a bit of sk or sk^3 for which the tampering function reads all the shares of it, while accessing at most $\alpha\nu$ bits of the codeword. Intuitively, and as we prove in Claim 3.3, by permuting the location indices of $z||e$, this event happens with probability negligible in k, and the attacker does not learn any bit of sk and sk^3, even if it is given access to $(1-o(1))\nu$ bits of the codeword.
- $\mathsf{Exp}_2^{\Sigma,f,s}$: By the previous hybrid, we have that for all $i \in [2|sk|]$, the tampering function will not access all bits of z_i, with overwhelming probability. In the third experiment, we unfold the encoding procedure, and in addition, we substitute the secret sharing procedure SS_m with $\bar{\mathsf{SS}}_m^f$ that computes shares z_i^* that reveal no information about $sk||sk^3$; for each i, $\bar{\mathsf{SS}}_m^f$ simply "drops" the bit of z_i with the largest index that is not being accessed by f. We formally define $\bar{\mathsf{SS}}_m^f$ below.

[12]This is an abbreviations for indistinguishability under chosen plaintext attack, for a single pre-challenge query to the encryption oracle.

```
$\mathsf{Exp}_0^{\Sigma,f,s}$ :
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$c \leftarrow \mathsf{Enc}(\Sigma, s), \tilde{c} \leftarrow 0^\nu$
$\tilde{c}[I] \leftarrow f_\Sigma(c_{|I}), \tilde{c}[I^c] \leftarrow c_{|I^c}$
$\tilde{s} \leftarrow \mathsf{Dec}(\tilde{c})$

Output $\mathsf{same}^*$ if $\tilde{s} = s$ and $\tilde{s}$ otherwise.
```

```
$\mathsf{Exp}_1^{\Sigma,f,s}$ :
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$c \leftarrow \mathsf{Enc}(\Sigma, s), \tilde{c} \leftarrow 0^\nu$
$\tilde{c}[I] \leftarrow f_\Sigma(c_{|I}), \tilde{c}[I^c] \leftarrow c_{|I^c}$
If $\exists i : |(I \cap Z_i)| = m$:
    $\tilde{s} \leftarrow \perp$
Else:
    $\tilde{s} \leftarrow \mathsf{Dec}(\tilde{c})$

Output $\mathsf{same}^*$ if $\tilde{s} = s$ and $\tilde{s}$ otherwise.
```

```
$\mathsf{Exp}_2^{\Sigma,f,s}$ :
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$sk \leftarrow \mathsf{KGen}(1^k), e \leftarrow \mathsf{E}_{sk}(s)$
$z^* \leftarrow \bar{\mathsf{SS}}_m^f(\Sigma, sk), c \leftarrow P_\Sigma(z^* || e)$
$\tilde{c} \leftarrow 0^\nu, \tilde{c}[I] \leftarrow f_\Sigma(c_{|I}), \tilde{c}[I^c] \leftarrow c_{|I^c}$

If $\exists i : |(I \cap Z_i)| = m$:
    $\tilde{s} \leftarrow \perp$
Else:
    If $\exists i : \bigoplus_{j \in (I \cap Z_i)} c[j] \neq \bigoplus_{j \in (I \cap Z_i)} \tilde{c}[j]$:
        $\tilde{s} \leftarrow \perp$
    Else:
        $\tilde{s} \leftarrow \mathsf{D}_{sk}(\tilde{e})$

Output $\mathsf{same}^*$ if $\tilde{s} = s$ and $\tilde{s}$ otherwise.
```

```
$\mathsf{Exp}_3^{\Sigma,f,s}$ :
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$sk \leftarrow \mathsf{KGen}(1^k), e \leftarrow \mathsf{E}_{sk}(s)$
$z^* \leftarrow \bar{\mathsf{SS}}_m^f(\Sigma, sk), c \leftarrow P_\Sigma(z^* || e)$

$\tilde{c} \leftarrow 0^\nu, \tilde{c}[I] \leftarrow f_\Sigma(c_{|I})$

If $\exists i : |(I \cap Z_i)| = m$:
    $\tilde{s} \leftarrow \perp$
Else:
    If $\exists i : \bigoplus_{j \in (I \cap Z_i)} c[j] \neq \bigoplus_{i \in (I \cap Z_i)} \tilde{c}[j]$:
        $\tilde{s} \leftarrow \perp$
    Else: $\tilde{s} \leftarrow \perp$
        If $\tilde{e} = e$:
            $\tilde{s} \leftarrow \mathsf{same}^*$

Output $\tilde{s}$.
```

Fig. 4. The hybrid experiments for the proof of Theorem 3.2. The gray part signifies the portion of the code of an experiment that differs from the previous one.

$\bar{\mathsf{SS}}_m^f(\Sigma, sk)$:

1. Sample $(z_1, \ldots, z_{2|sk|}) \leftarrow \mathsf{SS}_m\left(sk||sk^3\right)$ and set $z_i^* \leftarrow z_i, i \in [2|sk|]$.
2. For $i \in [2|sk|]$, let $l_i := \max_d \{d \in [m] \wedge \mathsf{Ind}(z_i[d]) \notin I)\}$, where Ind returns the index of $z_i[d]$ in c, i.e., l_i is the largest index in $[m]$ such that $z_i[l_i]$ is not accessed by f.
3. (**Output**): For all i set $z_i^*[l_i] = *$, and output $z^* :=||_{i=1}^{2|sk|} z_i^*$.

In $\mathsf{Exp}_1^{\Sigma,f,s}$, $z = ||_{i=1}^{2|sk|} z_i$, and each z_i is an m-out-of-m secret sharing for a bit of sk or sk^3. From Claim 3.3, we have that for all i, $|I \cap Z_i| < m$ with overwhelming probability,

and we can observe that the current experiment is identical to the previous one up to the point of computing $f(c_{|_I})$, as $c_{|_I}$ and $f(c_{|_I})$ depend only on z^*, that carries no information about sk and sk^3.
Another difference between the two experiments is in the external "Else" branch: $\mathsf{Exp}_1^{\Sigma,f,s}$ makes a call to the decoder while $\mathsf{Exp}_2^{\Sigma,f,s}$, before calling $\mathsf{D}_{sk}(\tilde{e})$, checks if the tampering function has modified the shares in a way such that the reconstruction procedure $((\tilde{sk}, \tilde{sk}') \leftarrow \mathsf{Rec}_m(\tilde{z}))$ will give $\tilde{sk} \neq sk$ or $\tilde{sk}' \neq sk'$. This check is done by the statement "If $\exists i : \bigoplus_{j\in(I\cap Z_i)} c[j] \neq \bigoplus_{j\in(I\cap Z_i)} \tilde{c}[j]$", without touching sk or sk^3.[13] In case modification is detected the current experiments outputs $\perp$. The intuition is that an attacker that partially modifies the shares of sk and sk^3, creates shares of $\tilde{sk}$ and $\tilde{sk}'$, such that $\tilde{sk}^3 = \tilde{sk}'$, with negligible probability in k. We prove this by a reduction to the 1-IND-CPA security of the encryption scheme: any valid modification over the inner encoding of the secret key gives us method to compute the original secret key sk, with non-negligible probability. The ideas are presented formally in Claim 3.4.
- $\mathsf{Exp}_3^{\Sigma,f,s}$: The difference between the current experiment and the previous one is that instead of executing the decryption, $\mathsf{D}_{sk}(\tilde{e})$, we first check if the attacker has modified the ciphertext, in which case the current experiment outputs $\perp$, otherwise it outputs same^*. By the previous hybrid, we reach this newly introduced "Else" branch of $\mathsf{Exp}_3^{\Sigma,f,s}$, only if the tampering function didn't modify the secret key. Thus, the indistinguishability between the two experiments follows from the authenticity property of the encryption scheme in the presence of z^*: given that $\tilde{sk} = sk$ and $\tilde{sk}' = sk'$, we have that if the attacker modifies the ciphertext, then with overwhelming probability $\mathsf{D}_{sk}(\tilde{e}) = \perp$, otherwise, $\mathsf{D}_{sk}(\tilde{e}) = s$, and the current experiment correctly outputs $\perp$ or same^* (cf. Claim 3.5).
- Finally, we prove that for any $f \in \mathcal{F}^\alpha$, and message s, $\mathsf{Exp}_3^{f,s}$ is indistinguishable from $\mathsf{Exp}_3^{f,\mathbf{0}}$, where $\mathbf{0}$ denotes the zero message. This follows by the semantic security of the encryption scheme, and gives us the indistinguishability property required by Lemma 2.6. The manipulation detection property is derived by the indistinguishability between the hybrids and the fact that the output of $\mathsf{Exp}_3^{\Sigma,f,s}$ is in the set $\{\mathsf{same}^*, \perp\}$.

In what follows, we prove indistinguishability between the hybrids using a series of claims.

Claim 3.3. For $k, m \in \mathbb{N}$, assume $(1-\alpha)m = \omega(\log(k))$. Then, for any $f \in \mathcal{F}^\alpha$ and any message s, we have $\mathsf{Exp}_0^{\Sigma,f,s} \approx \mathsf{Exp}_1^{\Sigma,f,s}$, where the probability runs over the randomness used by Init, Enc.

Proof. The difference between the two experiments is that $\mathsf{Exp}_1^{\Sigma,f,s}$ outputs $\perp$ when the attacker learns all shares of some bit of sk or sk^3, otherwise it produces output as $\mathsf{Exp}_0^{\Sigma,f,s}$ does. Let E be the event "$\exists i : |(I \cap Z_i)| = m$". Clearly, $\mathsf{Exp}_0^{\Sigma,f,s} = \mathsf{Exp}_1^{\Sigma,f,s}$ conditioned on $\neg E$, thus the statistical distance between the two experiments is bounded by $\Pr[E]$. In the following, we show that $\Pr[E] \leq \mathsf{negl}(k)$. We define by E_i the event in which f learns the entire z_i. Assuming the attacker reads n bits of the codeword, we

[13] Recall that our operations are over $\mathbf{GF}(2^{\mathrm{poly}(k)})$.

have that for all $i \in [2|sk|]$,

$$\Pr_{\Sigma}[E_i] = \Pr_{\Sigma}[\ |I \cap Z_i| = m\ ] = \prod_{j=0}^{m-1} \frac{n-j}{\nu - j} \leq \left(\frac{n}{\nu}\right)^m .$$

We have $n = \alpha\nu$ and assuming $\alpha = 1 - \epsilon$ for $\epsilon \in (0, 1]$, we have

$$\Pr[E_i] \leq (1-\epsilon)^m \leq 1/e^{m\epsilon},$$

and

$$\Pr[E] = \Pr_{\Sigma}\left[\bigcup_{i=1}^{2|sk|} E_i\right] \leq \frac{2|sk|}{e^{m\epsilon}},$$

which is negligible when $(1-\alpha)m = \omega(\log(k))$, and the proof of the claim is complete. □

Claim 3.4. Assuming $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is secure against the 1-query key recovery attack as Definition 2.9, then for any $f \in \mathcal{F}^{\alpha}$ and any message s, $\mathsf{Exp}_1^{\Sigma,f,s} \approx \mathsf{Exp}_2^{\Sigma,f,s}$, where the probability runs over the randomness used by Init, Enc.

Proof. In $\mathsf{Exp}_2^{\Sigma,f,s}$, we unfold the encoding procedure; however, instead of calling SS_m, we make a call to $\bar{\mathsf{SS}}_m^f$. As we have already stated above, this modification does not induce any difference between the output of $\mathsf{Exp}_2^{\Sigma,f,s}$ and $\mathsf{Exp}_1^{\Sigma,f,s}$, with overwhelming probability, as z^* is indistinguishable from z in the eyes of f. Another difference between the two experiments is in the external "Else" branch: $\mathsf{Exp}_1^{\Sigma,f,s}$ makes a call on the decoder while $\mathsf{Exp}_2^{\Sigma,f,s}$, before calling $\mathsf{D}_{sk}(\tilde{e})$, checks if the tampering function has modified the shares in a way such that the reconstruction procedure will give $\tilde{sk} \neq sk$ or $\tilde{sk}' \neq sk'$. This check is done by the statement "If $\exists i : \bigoplus_{j\in(I\cap Z_i)} c[j] \neq \bigoplus_{j\in(I\cap Z_i)} \tilde{c}[j]$", without touching sk or sk^3 (cf. Claim 3.3).[14] We define the events E, E' as follows

$$E : \mathsf{Dec}(\tilde{c}) \neq \perp,\ E' : \exists i : \bigoplus_{j\in(I\cap Z_i)} c[j] \neq \bigoplus_{j\in(I\cap Z_i)} \tilde{c}[j].$$

Clearly, conditioned on $\neg E'$ the two experiments are identical, since we have $\tilde{sk} = sk$ and $\tilde{sk}' = sk'$, and the decoding process will output $\mathsf{D}_{sk}(\tilde{e})$ in both experiments. Thus, the statistical distance is bounded by $\Pr[E']$. Now, conditioned on $E' \wedge \neg E$, both experiments output $\perp$. Thus, we need to bound $\Pr[E \wedge E']$. Assuming $\Pr[E \wedge E'] > p$, for $p = 1/\text{poly}(k)$, we define an attacker $\mathcal{A}$ that simulates $\mathsf{Exp}_2^{\Sigma,f,s}$, and uses f, s to break the 1-query key recovery security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ in the presence of z^*, with probability at least $p/2$.

[14]Recall that our operations are over $\mathbf{GF}(2^{\text{poly}(k)})$.

First we prove that any secure (against 1-query key recovery attacks) encryption scheme, remains secure even if the attacker receives $z^* \leftarrow \bar{\mathsf{SS}}_m^f(\Sigma, sk)$. This is because that z^* can be simulated without knowing sk, by using random shares on positions that the tampering function can see and $*$'s otherwise. Then this fact follows by a simple reduction argument.

Now we prove our claim. Assuming $\Pr[E \wedge E'] > p$, for $p = 1/\text{poly}(k)$, we define an attacker $\mathcal{A}$ that breaks the 1-query key recovery security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ in the presence of z^*, with non-negligible probability. $\mathcal{A}$ receives the encryption of s, which corresponds to the oracle query right before receiving the challenge ciphertext, the challenge ciphertext $e \leftarrow \mathsf{E}_{sk}(s_b)$, for uniform $b \in \{0, 1\}$ and uniform messages s_0, s_1, as well as z^*. $\mathcal{A}$ is defined below.

$\mathcal{A}\left(z^* \leftarrow \bar{\mathsf{SS}}_m^f(\Sigma, sk), e' \leftarrow \mathsf{E}_{sk}(s), e \leftarrow \mathsf{E}_{sk}(s_b)\right)$:

1. **(Define the shares that will be accessed by** f**)**: For $i \in [2|sk|]$, define $w_i := (z_i^*)_{|[m]\setminus\{l_i\}}$ and for $i \in [m-1]$ define $C_i = ||_{j=1}^{|sk|} w_j[i]$, $D_i = ||_{j=|sk|+1}^{2|sk|} w_j\,[i]$.
2. **(Apply** f**)** Set $c \leftarrow P_\Sigma(z^*||e)$, compute $\tilde{c}[I] \leftarrow f_\Sigma(c_{|_I})$ and let $\tilde{C}_i, \tilde{D}_i, i \in [m]$, be the tampered shares resulting after the application of f to $c_{|_I}$.
3. **(Searching the secret key)** Let $U = \sum_{i=1}^{m-1} C_i$, $V = \sum_{i=1}^{m-1} D_i$, i.e., U, V denote the sum of the shares that are being accessed by the attacker (maybe partially), and $\tilde{U} = \sum_{i=1}^{m-1} \tilde{C}_i$, $\tilde{V} = \sum_{i=1}^{m-1} \tilde{D}_i$, are the corresponding tampered values after applying f on U, V. Define

$$p(X) := (U - \tilde{U})X^2 + (U^2 - \tilde{U}^2)X + (U^3 - \tilde{U}^3 - V + \tilde{V}),$$

and compute the set of roots of $p(X)$, denoted as $\mathcal{X}$, which are at most two. Then set

$$\hat{\mathcal{SK}} := \{x + U | x \in \mathcal{X}\}\,. \tag{2}$$

4. **(Output)** Just output a random element in $\hat{\mathcal{SK}}$.

In the first step, $\mathcal{A}$ removes the dummy symbol "$*$" and computes the shares that will be partially accessed by f, denoted as C_i for sk and as D_i for sk^3. In the second step, it simulates the codeword partially, applies the tampering function on it, and defines the tampered shares, $\tilde{C}_i, \tilde{D}_i$. Conditioned on E', it is not hard to see that $\mathcal{A}$ simulates perfectly $\mathsf{Exp}_2^{\Sigma,f,s}$. In particular, it simulates perfectly the input to f as it receives $e \leftarrow \mathsf{E}_{sk}(s)$ and all but $2|sk|$ of the actual bit-shares of sk, sk^3. Part of those shares will be accessed by f. Since for all i, $|I \cap Z_i| < m$, the attacker is not accessing any single bit of sk, sk^3. Let C_m, D_m, be the shares (not provided by the encryption oracle) that completely define sk and sk^3, respectively. By the definition of the encoding scheme and the fact that $sk, sk^3 \in \mathbf{GF}(2^{\text{poly}(k)})$, we have $\sum_{i=1}^m C_i = sk$, $\sum_{i=1}^m D_i = sk^3$, and

$$(U + C_m)^3 = V + D_m. \tag{3}$$

In order for the decoder to output a non-bottom value, the shares created by the attacker must decode to $\tilde{sk}, \tilde{sk}'$, such that $\tilde{sk}^3 = \tilde{sk}'$, or in other words, if

$$\left(\tilde{U} + C_m\right)^3 = \tilde{V} + D_m. \tag{4}$$

From 3 and 4 we receive

$$(U - \tilde{U})C_m^2 + (U^2 - \tilde{U}^2)C_m + (U^3 - \tilde{U}^3) = V - \tilde{V}. \tag{5}$$

Clearly, $\Pr[E \wedge E' \wedge (U = \tilde{U})] = 0$. Thus, assuming $\Pr[E \wedge E'] > p$, for $p > 1/\text{poly}(k)$, we receive

$$\begin{aligned} p = \Pr\left[E \wedge E' \wedge (U \neq \tilde{U})\right] &\leq \Pr\left[\mathsf{Dec}(\tilde{c}) \neq \bot \wedge E' \wedge U \neq \tilde{U}\right] \\ &\leq \Pr\left[\tilde{sk}^3 = \tilde{sk}' \wedge E' \wedge (U \neq \tilde{U})\right] \\ &\overset{(5,2)}{=} \Pr\left[C_m \in \mathcal{X}\right] \overset{(2)}{\leq} \Pr\left[sk \in \hat{\mathcal{SK}}\right], \end{aligned} \tag{6}$$

and $\mathcal{A}$ manages to recover C_m, and thus the set $\hat{\mathcal{SK}}$ that contains sk, with non-negligible probability at least p. As $\hat{\mathcal{SK}}$ is derived from solving a quadratic equation, the cardinality, i.e., $|\hat{\mathcal{SK}}|$ is at most 2. Thus, a random guess would hit the sk with probability at least $p/2$. This is a contradiction to the 1-query key recovery security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$.

Thus, we have $\Pr[E \wedge E'] \leq \mathsf{negl}(k)$, and both experiments output $\bot$ with overwhelming probability. □

Claim 3.5. Assuming the authenticity property of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, for any $f \in \mathcal{F}^\alpha$ and any message s, $\mathsf{Exp}_2^{\Sigma,f,s} \approx \mathsf{Exp}_3^{\Sigma,f,s}$, where the probability runs over the randomness used by Init, KGen and E.

Proof. Before proving the claim, recall that the authenticity property of the encryption scheme is preserved under the presence of z^* (cf. Claim 3.4). Let E be the event $\tilde{sk} = sk \wedge \tilde{sk}' = sk^3$ and E' be the event $\tilde{e} \neq e$. Conditioned on $\neg E$, the two experiments are identical, as they both output $\bot$. Also, conditioned on $E \wedge \neg E'$, both experiments output same^*. Thus, the statistical distance between the two experiments is bounded by $\Pr[E \wedge E']$. Let B be the event $\mathsf{D}_{sk}(\tilde{e}) \neq \bot$. Conditioned on $E \wedge E' \wedge \neg B$ both experiments output $\bot$. Thus, we need to bound $\Pr[E \wedge E' \wedge B]$.

Assuming there exist s, f, for which $\Pr[E \wedge E' \wedge B] > p$, where $p = 1/\text{poly}(k)$, we define an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that simulates $\mathsf{Exp}_3^{\Sigma,f,s}$ and breaks the authenticity property of the encryption scheme in the presence of z^*, with non-negligible probability. $\mathcal{A}$ is defined as follows: sample $(s, st) \leftarrow \mathcal{A}_1(1^k)$, and then, on input (z^*, e, st), where $e \leftarrow \mathsf{E}_{sk}(s)$, $\mathcal{A}_2$, samples $\Sigma \leftarrow \mathsf{Init}(1^k)$, sets $\tilde{c} \leftarrow 0^\nu$, $c \leftarrow P_\Sigma(z^* \| e)$, computes $\tilde{c}[I] \leftarrow f(c_{|_I})$, $\tilde{c}[I^\mathsf{c}] \leftarrow c_{|_{I^\mathsf{c}}}$, $(\tilde{z}^* \| \tilde{e}) \leftarrow P_\Sigma^{-1}(\tilde{c})$, and outputs $\tilde{e}$. Assuming $\Pr[E \wedge E' \wedge B] > p$, we have that $\mathsf{D}_{sk}(\tilde{e}) \neq \bot$ and $\tilde{e} \neq e$, with non-negligible probability and the authenticity property of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ breaks. □

Claim 3.6. Assuming $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is semantically secure, for any $f \in \mathcal{F}^{\alpha}$ and any message s, $\mathsf{Exp}_3^{\Sigma,f,s} \approx \mathsf{Exp}_3^{\Sigma,f,\mathbf{0}}$, where the probability runs over the randomness used by Init, KGen, E. "$\approx$" may refer to statistical or computational indistinguishability, and $\mathbf{0}$ denotes the zero message.

Proof. Recall that $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is semantically secure even in the presence of $z^* \leftarrow \bar{\mathsf{SS}}_m^f(\Sigma, sk)$ (cf. 3.4), and toward contradiction, assume there exist $f \in \mathcal{F}^{\alpha}$, message s, and PPT distinguisher $\mathcal{D}$ such that

$$\left|\Pr\left[\mathcal{D}\left(\Sigma, \mathsf{Exp}_3^{\Sigma,f,s}\right) = 1\right] - \Pr\left[\mathcal{D}\left(\Sigma, \mathsf{Exp}_3^{\Sigma,f,\mathbf{0}}\right)\right] = 1\right| > p,$$

for $p = 1/\mathrm{poly}(k)$. We are going to define an attacker $\mathcal{A}$ that breaks the semantic security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ in the presence of z^*, using $s_0 := s$, $s_1 := \mathbf{0}$. $\mathcal{A}$, given z^*, e, executes $\mathsf{Program}$.

```
$\mathsf{Program}(z^*, e)$ :
$c \leftarrow P_{\Sigma}(z^* || e)$, $\tilde{c} \leftarrow 0^{\nu}$, $\tilde{c}[I] \leftarrow f(c_{|_I})$
If $\exists i : |(I \cap Z_i)| = m$ : $\tilde{s} \leftarrow \perp$
Else:
  If $\exists i : \bigoplus_{j \in (I \cap Z_i)} c[j] \neq \bigoplus_{j \in (I \cap Z_i)} \tilde{c}[j]$ :
    $\tilde{s} \leftarrow \perp$
  Else: $\tilde{s} \leftarrow \perp$
    If $\tilde{e} = e$ :
      $\tilde{s} \leftarrow \mathsf{same}^*$
Output $\tilde{s}$.
```

It is not hard to see that $\mathcal{A}$ simulates $\mathsf{Exp}_3^{\Sigma,f,s_b}$; thus, the advantage of $\mathcal{A}$ against the semantic security of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is the same with the advantage of $\mathcal{D}$ in distinguishing between $\mathsf{Exp}_3^{\Sigma,f,s_0}$, $\mathsf{Exp}_3^{\Sigma,f,s_1}$, which by assumption is non-negligible. We have reached a contradiction, and the proof of the claim is complete. □

From the above claims, we have that for any $f \in \mathcal{F}^{\alpha}$ and any s, $\mathsf{Exp}_0^{\Sigma,f,s} \approx \mathsf{Exp}_3^{\Sigma,f,\mathbf{0}}$, thus for any $f \in \mathcal{F}^{\alpha}$ and any s_0, s_1, $\mathsf{Exp}_0^{\Sigma,f,s_0} \approx \mathsf{Exp}_0^{\Sigma,f,s_1}$. Also, by the indistinguishability between $\mathsf{Exp}_0^{\Sigma,f,s}$ and $\mathsf{Exp}_3^{\Sigma,f,\mathbf{0}}$, the second property of Lemma 2.6 has been proven as the output of $\mathsf{Exp}_3^{\Sigma,f,\mathbf{0}}$ is in $\{s, \perp\}$, with overwhelming probability, and non-malleability with manipulation detection of our code follows by Lemma 2.6, since $\mathsf{Exp}_0^{\Sigma,f,s}$ is identical to Tamper_s^f of Lemma 2.6. □

Instantiations and rates. By instantiating Construction 3.1 with the authenticated encryption scheme 2.11, Theorem 3.2, for $m = k \log k$, $\alpha = 1 - 1/\Omega(\log k)$, yields a rate 1 MD-NMC, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s| + O(k^2 \log k)$, assuming one-way functions. Furthermore, by instantiating Construction 3.1 with 2.12, Theorem 3.2, for $m = |s| \log |s|$, $\alpha = 1 - 1/O(\log(|s|))$, yields an unconditionally secure MD-NMC in the CRS model, with concrete information rate $1/O(|s| \log(|s|))$, access rate $1 - 1/\Omega(\log(|s|))$ and codewords of length $O(|s|^2 \log |s|)$.

On the CRS. In the above, the tampering function, and consequently the codeword locations that the function is given access to, are fixed before sampling the CRS and this is critical for achieving security. However, by the proof of Theorem 3.2, we observe that proving security in this setting is highly non-trivial. In addition, the tampering function receives full access to the CRS when tampering with the codeword, which is in contrast to the work by Faust et. al. [45] in the information-theoretic setting, where the (internal) tampering function receives partial information over the CRS.

In addition, the proposed scheme tolerates adaptive selection of the codeword locations, with respect to the CRS, in the following way: each time the attacker requests access to a location, he also learns if it corresponds to a bit of z or e, together with the index of that bit in the original string. In this way, the CRS is gradually disclosed to the adversary while picking codeword locations.

Finally, our CRS sustains a substantial amount of tampering that depends on the codeword locations chosen by the attacker: an attacker that gets access to a sensitive codeword bit is allowed to modify the part of the CRS that defines the location of that bit in the codeword. The attacker is allowed to modify all but $O(k \log(|s| + k))$ bits of the CRS, that is of length $O(k^2 \log k \log(|s| + k))$. To our knowledge, this is the first construction that tolerates, even partial modification of the CRS. In contrast, existing constructions in the CRS model are either using NIZKs [35,41,43,55], or they are based on the *knowledge of exponent assumption* [51], thus tampering access to the CRS would compromise security.

4. Removing the CRS

In the present section, we show how to construct an MD-NMC for partial functions, in the standard model.

A first approach would be to store the CRS of Construction 3.1, inside the codeword together with $P_{\Sigma}(z||e)$, and give to the attacker read/write access to it. However, the tampering function, besides getting direct (partial) access to the encoding of sk, it also gets indirect access to it by (partially) controlling the CRS. Then, it can modify the CRS in a way such that, during decoding, ciphertext locations of its choice will be treated as bits of the inner encoding, z, increasing the tampering rate against z significantly. This makes the task of protecting sk hard, if not impossible (unless we restrict the access rate significantly).

To handle this challenge, we embed a structure recovering mechanism inside the codeword and we emulate the CRS effect by increasing the size of the alphabet, giving rise to a block-wise structure.[15] Notice that, non-malleable codes with large alphabet size (i.e., $\text{poly}(k) + |s|$ bits) might be easy to construct, as we can embed in each codeword block the verification key of a signature scheme together with a secret share of the message, as well as a signature over the share. In this way, partial access over the codeword does not compromise the security of the signature scheme while the message remains private, and the simulation is straightforward. This approach, however, comes

[15] Bigger alphabets have been also considered in the context of error-correcting codes, in which the codeword consists of symbols.

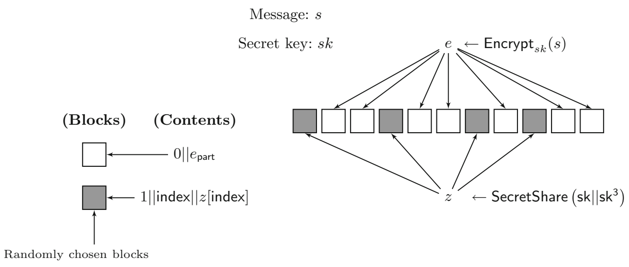
Message: s
Secret key: sk
$e \leftarrow \mathsf{Encrypt}_{sk}(s)$
(Blocks)
(Contents)
$0||e_{\mathsf{part}}$
$1||\mathsf{index}||z[\mathsf{index}]$
Randomly chosen blocks
$z \leftarrow \mathsf{SecretShare}(\mathsf{sk}||\mathsf{sk}^3)$

Fig. 5. Description of the scheme in the standard model.

with a large overhead, decreasing the information rate and access rate of the scheme significantly. In general, and similar to error correcting codes, we prefer smaller alphabet sizes—the larger the size is, the more coarse access structure is required, i.e., in order to access individual bits we need to access the blocks that contain them. The present work aims at minimizing this restriction by using small alphabets, as described below.

Our approach on the problem is the following. We increase the alphabet size to $O(\log k)$ bits, and we consider two types of blocks: (i) *sensitive blocks*, in which we store the inner encoding, z, of the secret key, sk, and (ii) *non-sensitive* blocks, in which we store the ciphertext, e, that is fragmented into blocks of size $O(\log k)$. The first bit of each block indicates whether it is a sensitive block, i.e., we set it to 1 for sensitive blocks and to 0, otherwise. Our encoder works as follows: on input message s, it computes z, e, as in the previous scheme and then uses rejection sampling to sample the indices, $\rho_1, \ldots, \rho_{|z|}$, for the sensitive blocks. Then, for every $i \in \{1, \ldots, |z|\}$, C_{ρ_i} is a sensitive block, with contents $(1||i||z[i])$, while the remaining blocks keep ciphertext pieces of size $O(\log k)$. Decoding proceeds as follows: on input codeword $C = (C_1, \ldots, C_{\mathsf{bn}})$, for each $i \in [\mathsf{bn}]$, if C_i is a non-sensitive block, its data will be part of e, otherwise, the last bit of C_i will be part of z, as it is dictated by the index stored in C_i. If the number of sensitive blocks is not the expected, the decoder outputs $\perp$, otherwise, z, e, have been fully recovered and decoding proceeds as in the previous scheme. The proposed scheme is depicted in Fig. 5.

The security of our construction is based on the fact that, due to our shuffling technique, the position mapping will not be completely overwritten by the attacker, and we prove later in this section, this suffices for protecting the inner encoding over sk. We prove security of the current scheme (cf. Theorem 4.8) by a reduction to the security of the scheme in the CRS model. Our instantiation yields a rate $1 - 1/\Omega(\log k)$ MD-NMC in the standard model, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s|(1 + 1/O(\log k)) + O(k^2 \log^2 k)$, assuming one-way functions.

It is worth pointing out that the idea of permuting blocks containing sensitive and non-sensitive data was also considered by [61] in the context of list-decodable codes; however, the similarity is only in the fact that a permutation is being used at some point in the encoding process, and our objective, construction and proof are different.

In what follows, we consider alphabets of size $O(\log(k))$ and we provide a computationally secure, rate $1 - 1/\Omega(\log k)$ encoding scheme in the standard model, tolerating modification of $(1-o(1))\nu$ blocks, where ν is the total number of blocks in the codeword.

The projection operation will be also used with respect to bigger alphabets, enabling the projection of blocks.

Our construction is defined below.

Construction 4.1. *Let $k, m \in \mathbb{N}$, let* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *be a symmetric encryption scheme and* $(\mathsf{SS}_m, \mathsf{Rec}_m)$ *be an m-out-of-m secret sharing scheme. We define an encoding scheme* $(\mathsf{Enc}^*, \mathsf{Dec}^*)$*, as follows:*

– $\mathsf{Enc}^*(1^k, \cdot)$*: for input message s, sample* $sk \leftarrow \mathsf{KGen}\left(1^k\right)$, $e \leftarrow \mathsf{E}_{sk}(s)$.

- **(Secret share)** *Sample* $z \leftarrow \mathsf{SS}_m(sk\|sk^3)$*, where* $z = \|_{i=1}^{2|sk|} z_i$, $z \in \{0,1\}^{2m|sk|}$, *and for* $i \in [|sk|]$, z_i *(resp.* $z_{|sk|+i}$*) is an m-out-of-m secret sharing of* $sk[i]$ *(resp.* $sk^3[i]$*).*
- **(Construct blocks & permute)** *Set* $l \leftarrow 2m|sk|$, $\mathsf{bs} \leftarrow \log l + 2$, $d \leftarrow |e|/\mathsf{bs}$, $\mathsf{bn} \leftarrow l + d$, *sample* $\rho := (\rho_1, \ldots, \rho_l) \overset{\mathsf{rs}}{\leftarrow} \{0,1\}^{\log(\mathsf{bn})}$ *and compute* $C \leftarrow \Pi_\rho(z\|e)$ *as follows:*

  1. *Set* $t \leftarrow 1$, $C_i \leftarrow 0^{\mathsf{bs}}$, $i \in [\mathsf{bn}]$.
  2. **(Sensitive blocks)** *For* $i \in [l]$, *set* $C_{\rho_i} \leftarrow (1\|i\|z[i])$.
  3. **(Ciphertext blocks)** *For* $i \in [\mathsf{bn}]$, *if* $i \neq \rho_j$, $j \in [l]$, $C_i \leftarrow (0\|e[t : t + (\mathsf{bs}-1)])$, $t \leftarrow t + (\mathsf{bs}-1)$.[16]

Output $C := (C_1\|\ldots\|C_{\mathsf{bn}})$.

– $\mathsf{Dec}^*(1^k, \cdot)$*: on input C, parse it as* $(C_1\|\ldots\|C_{\mathsf{bn}})$*, set* $t \leftarrow 1$, $l \leftarrow 2m|sk|$, $z \leftarrow 0^l$, $e \leftarrow \mathsf{0}$, $\mathcal{L} = \emptyset$ *and compute* $(z\|e) \leftarrow \Pi^{-1}(C)$ *as follows:*

- *For* $i \in [\mathsf{bn}]$,
  - ∗ **(Sensitive block)** *If* $C_i[1] = 1$*, set* $j \leftarrow C_i[2 : \mathsf{bs}-1]$, $z[j] \leftarrow C_i[\mathsf{bs}]$, $\mathcal{L} \leftarrow \mathcal{L} \cup \{j\}$.
  - ∗ **(Ciphertext block)** *Otherwise, set* $e[t : t + \mathsf{bs} - 1] = C_i[2 : \mathsf{bs}]$, $t \leftarrow t + \mathsf{bs} - 1$.
- *If* $|\mathcal{L}| \neq l$*, output* $\perp$*, otherwise output* $(z\|e)$.

If $\Pi^{-1}(C) = \perp$*, output* $\perp$*, otherwise, compute* $(sk\|sk') \leftarrow \mathsf{Rec}_m(z)$*, and if* $sk^3 = sk'$*, output* $\mathsf{D}_{sk}(e)$*, otherwise output* $\perp$.

The set of indices of the blocks in which z_i *is stored will be denoted by* Z_i.

We prove security for the above construction by a reduction to the security of Construction 3.1. We note that our reduction is non-black box with respect to the coding scheme in which security is reduced to; a generic reduction, i.e., non-malleable reduction [2], from the standard model to the CRS model is an interesting open problem and thus out of the scope of the present work.

In the following, we consider $\Gamma = \{0,1\}^{O(\log(k))}$.[17] The straightforward way to prove that $(\mathsf{Enc}^*, \mathsf{Dec}^*)$ is secure against $\mathcal{F}_\Gamma^\alpha$ by a reduction to the security of the bit-wise code of Sect. 3, would be as follows: for any $\alpha \in [0,1)$, $f \in \mathcal{F}_\Gamma^\alpha$ and any message s, we

[16] Here we assume that $\mathsf{bs} - 1$, divides the length of the ciphertext e. We can always achieve this property by padding the message s with zeros, if necessary.

[17] Recall that, whenever Γ is omitted from the notation, we assume that $\Gamma = \{0,1\}$.

have to define α', $g \in \mathcal{F}^{\alpha'}$, such that the output of the tampered execution with respect to $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, f, s, is indistinguishable from the tampered execution with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, g, s, and g is an admissible function for $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$. However, this approach might be tricky as it requires the establishment of a relation between α and α' such that the sensitive blocks that f will receive access to, will be simulated using the sensitive bits accessed by g. Our approach is cleaner: for the needs of the current proof we leverage the power of Construction 3.1, by allowing the attacker to choose adaptively the codeword locations, as long as it does not request to read all shares of the secret key. Then, for every block that is accessed by the block-wise attacker f, the bit-wise attacker g requests access to the locations of the bit-wise code that enable him to fully simulate the input to f. We formally present our ideas in the following sections. In Sect. 4.1 we introduce the function class $\mathcal{F}_{\mathsf{ad}}$ that considers adaptive adversaries with respect to the CRS and we prove security of Construction 3.1 in Corollary 4.3 against a subclass of $\mathcal{F}_{\mathsf{ad}}$, and then, we reduce the security of the block-wise code $(\mathsf{Enc}^*, \mathsf{Dec}^*)$ against $\mathcal{F}_{\Gamma}^{\alpha}$ to the security of Construction 3.1 against $\mathcal{F}_{\mathsf{ad}}$ (cf. Sect. 4.2).

4.1. *Security Against Adaptive Adversaries*

In the current section, we prove that Construction 4.1 is secure against the class of functions that request access to the codeword adaptively, i.e., depending on the CRS, as long as they access a bounded number of sensitive bits. Below, we formally define the function class $\mathcal{F}_{\mathsf{ad}}$, in which the tampering function picks up the codeword locations depending on the CRS, and we consider $\Gamma = \{0, 1\}$.

Definition 4.2. (*The function class* $\mathcal{F}_{\mathsf{ad}}^{\nu}$ *(or* $\mathcal{F}_{\mathsf{ad}}$*)*) Let $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ be an (κ, ν)-coding scheme and let Σ be the range of $\mathsf{Init}(1^k)$. For any $g = (g_1, g_2) \in \mathcal{F}_{\mathsf{ad}}^{\nu}$, we have $g_1 : \Sigma \rightarrow \mathcal{P}([\nu])$, $g_2^{\Sigma} : \{0,1\}^{|\mathsf{range}(g_1)|} \rightarrow \{0,1\}^{|\mathsf{range}(g_1)|} \cup \{\perp\}$, and for any $c \in \{0,1\}^{\nu}$, $g^{\Sigma}(c) = g_2\left(c_{|g_1(\Sigma)}\right)$. For brevity, the function class will be denoted as $\mathcal{F}_{\mathsf{ad}}$.

Construction 3.1 remains secure against functions that receive full access to the ciphertext, as long as they request to read all but one shares for each bit of sk and sk^3. The result is formally presented in the following corollary.

Corollary 4.3. *Let $k, m \in \mathbb{N}$. Assuming* $(\mathsf{SS}_m, \mathsf{Rec}_m)$ *is an m-out-of-m secret sharing scheme and* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is* 1-IND-CPA *secure authenticated encryption scheme, the code of Construction 4.1 is an* MD-NMC *against any $g = (g_1, g_2) \in \mathcal{F}_{\mathsf{ad}}$, assuming that for all $i \in [2|sk|]$, $(Z_i \cap g_1(\Sigma)) < m$, where $sk \leftarrow \mathsf{KGen}(1^k)$ and $\Sigma \leftarrow \mathsf{Init}(1^k)$.*

Proof. Let $g = (g_1, g_2)$ be as stated above. For any message s, the tampered execution with respect to g and $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, is defined as follows.

$$\mathsf{Tamper}_s^g := \left\{ \begin{array}{l} \Sigma \leftarrow \mathsf{Init}(1^k), c \leftarrow \mathsf{Enc}(\Sigma, s), I \leftarrow g_1(\Sigma) \\ \tilde{c} \leftarrow g_2^{\Sigma}(c_{|I}), \tilde{s} \leftarrow \mathsf{Dec}(\Sigma, \tilde{c}) \\ \\ \text{If } \tilde{s} = s, \text{ output } \mathsf{same}^*, \text{ otherwise, output } \tilde{s}. \end{array} \right\}$$

The proof is along the lines of the proof of Theorem 3.2, i.e., we prove that for any g having the properties stated above, and any pair of messages s_0, s_1, $\mathsf{Tamper}^g_{s_0} \approx \mathsf{Tamper}^g_{s_1}$, and the output of the tampered execution is either the original message, or $\perp$, with overwhelming probability. Below, we revisit the hybrids of Theorem 3.2 and we prove that the indistinguishability between adjacent hybrids, holds with respect to g.

- $\mathsf{Exp}_0^{\Sigma,g,s}$: For any f, s, the first experiment, $\mathsf{Exp}_0^{\Sigma,g,s}$, is identical to the experiment Tamper^g_s.
- $\mathsf{Exp}_1^{\Sigma,g,s}$: In the second experiment, we have $Z_i, i \in [2|sk|]$, to be the set of indices in which z_i is stored, $|Z_i| = m$. The main difference from the previous experiment is that the current one outputs $\perp$, if there exists a bit of sk or sk^3 for which the tampering function reads all shares of it. However, by the definition of g we know that this happens with zero probability; thus, we have that the following claim holds,

Claim 4.4. Let $k, m \in \mathbb{N}$. For any $g = (g_1, g_2) \in \mathcal{F}_{\mathsf{ad}}$, assuming that for all $i \in [2|sk|]$, $(Z_i \cap g_1(\Sigma)) < m$ and any message s, we have $\mathsf{Exp}_0^{\Sigma,g,s} = \mathsf{Exp}_1^{\Sigma,g,s}$, where $sk \leftarrow \mathsf{KGen}(1^k)$, $\Sigma \leftarrow \mathsf{Init}(1^k)$.

- $\mathsf{Exp}_2^{\Sigma,g,s}$: In the current experiment, we unfold the encoding procedure, and in addition, we substitute the secret sharing procedure SS_m with $\bar{\mathsf{SS}}^g_m$, where $\bar{\mathsf{SS}}^g_m$ is defined as $\bar{\mathsf{SS}}^f_m$ does with respect to f, in Claim 3.4 of Theorem 3.2. From the above claim, we have that for all i, $|I \cap Z_i| < m$, and we observe that the current experiment is identical to the previous one up to the point of computing $g(c_{|_I})$, as $c_{|_I}$ carries no information about sk and sk^3. Thus, the transition between the current experiment and the previous one is identical to that of Theorem 3.2: an attacker that partially modifies the shares of sk and sk^3, creates shares of $\tilde{sk}$ and $\tilde{sk}'$, such that $\tilde{sk}^3 = \tilde{sk}'$, with negligible probability in k, which is proved by a reduction to the 1-IND-CPA security of the encryption scheme in the presence of z^*. Thus, we have the following claim.

Claim 4.5. Assuming $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is 1-IND-CPA secure (cf. Definition 2.9), for any $g \in \mathcal{F}_{\mathsf{ad}}$ and any message s, $\mathsf{Exp}_1^{\Sigma,g,s} \approx \mathsf{Exp}_2^{\Sigma,g,s}$, where the probability runs over the randomness used by Init, Enc.

- $\mathsf{Exp}_3^{\Sigma,g,s}$: As in Theorem 3.2, the indistinguishability between the two experiments follows from the authenticity property of the encryption scheme in the presence of z^*. Thus, the following holds.

Claim 4.6. Assuming the authenticity property of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, for any $g \in \mathcal{F}_{\mathsf{ad}}$ and any message s, $\mathsf{Exp}_2^{\Sigma,f,s} \approx \mathsf{Exp}_3^{\Sigma,f,s}$, where the probability runs over the randomness used by Init, KGen and E.

- Finally, since g learns nothing about sk, we have that for any $g \in \mathcal{F}_{\mathsf{ad}}$, and message s, $\mathsf{Exp}_3^{g,s}$ is indistinguishable from $\mathsf{Exp}_3^{g,\mathsf{0}}$, where $\mathsf{0}$ denotes the zero message.

This follows by the semantic security of the encryption scheme (Definition 2.9). Formally, we prove the following claim.

Claim 4.7. Assuming $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is semantically secure, for any $g \in \mathcal{F}_{\mathsf{ad}}$ and any message s, $\mathsf{Exp}_3^{g,s} \approx \mathsf{Exp}_3^{g,\mathbf{0}}$, where the probability runs over the randomness used by Init, KGen, E, "$\approx$" may refer to statistical or computational indistinguishability, and $\mathbf{0}$ is the zero message.

From the above claims we have that for any $g \in \mathcal{F}_{\mathsf{ad}}$ and any s_0, s_1, assuming that for all $i \in [2|sk|]$, $(Z_i \cap g_1(\Sigma)) < m$, $\mathsf{Exp}_0^{g,s_0} \approx \mathsf{Exp}_0^{g,s_1}$, and non-malleability with manipulation detection follows by Lemma 2.6, since $\mathsf{Exp}_0^{g,s}$ is identical to Tamper_s^g of Lemma 2.6, and by the indistinguishability between $\mathsf{Exp}_0^{\Sigma,g,s}$ and $\mathsf{Exp}_3^{\Sigma,g,s}$, the second property of Lemma 2.6 has been proven as the output of $\mathsf{Exp}_3^{\Sigma,g,s}$ is in $\{s, \perp\}$, with overwhelming probability. □

4.2. MD-NMC *Security of the Block-Wise Code*

In the current section, we prove security of Construction 4.1 against $\mathcal{F}_{\Gamma}^{\alpha}$, for $\Gamma = \{0,1\}^{O(\log(k))}$.

Theorem 4.8. *Let $k, m \in \mathbb{N}$, $\Gamma = \{0,1\}^{O(\log(k))}$ and $\alpha \in [0,1)$. Assuming $(\mathsf{SS}_m, \mathsf{Rec}_m)$ is an m-out-of-m secret sharing scheme and $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is a* 1-IND-CPA *secure authenticated encryption scheme, the code of Construction 4.1 is an* MD-NMC *against $\mathcal{F}_{\Gamma}^{\alpha}$, for any α, m, such that $(1-\alpha)m = \omega(\log(k))$.*

Proof. Following Lemma 2.6, we prove that for any $f \in \mathcal{F}_{\Gamma}^{\alpha}$, and any pair of messages s_0, s_1, $\mathsf{Tamper}_{s_0}^f \approx \mathsf{Tamper}_{s_1}^f$, and for any s, $\Pr\left[\mathsf{Tamper}_s^f \notin \{\perp, s\}\right] \leq \mathsf{negl}(k)$, where Tamper denotes the experiment defined in Lemma 2.6 with respect to the encoding scheme of Construction 4.1, $(\mathsf{Enc}^*, \mathsf{Dec}^*)$. Our proof is given by a series of hybrids depicted in Fig. 9. We reduce the security $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, to the security of Construction 3.1, $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, against $\mathcal{F}_{\mathsf{ad}}$ (cf. Corollary 4.3). The idea is to move from the tampered execution with respect to $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, f, to a tampered execution with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, g, such that the two executions are indistinguishable and $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ is secure against g.

Let I_{b} be the set of indices of the blocks that f chooses to tamper with, where $|I_{\mathsf{b}}| \leq \alpha\nu$, and let $l \leftarrow 2m|sk|$, $\mathsf{bs} \leftarrow \log l + 2$, $\mathsf{bn} \leftarrow l + |e|/\mathsf{bs}$. Below we describe the hybrids of Fig. 9.

- $\mathsf{Exp}_0^{f,s}$: The current experiment is the experiment Tamper_s^f, of Lemma 2.6, with respect to $(\mathsf{Enc}^*, \mathsf{Dec}^*)$, f, s.
- $\mathsf{Exp}_1^{(g_1,g_2),s}$: The main difference between $\mathsf{Exp}_0^{f,s}$ and $\mathsf{Exp}_1^{(g_1,g_2),s}$ is that in the latter one, we introduce the tampering function (g_1, g_2), that operates over codewords of $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ and we modify the encoding steps so that the experiment creates codewords of the bit-wise code $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$. (g_1, g_2) simulates partially the block-wise codeword C, while given partial access to the bit-wise codeword $c \leftarrow$

```
$g_1(\Sigma = (r_1, \ldots, r_l))$ :
  • (Simulate block shuffling):
    Sample $\rho := (\rho_1, \ldots, \rho_l) \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$
  • (Construct $I$): Set $I = \emptyset$,
      ∗ (Add ciphertext locations to $I$):
        For $j \in [|e| + l]$, if $j \notin \{r_i | i \in [l]\}$, $I \leftarrow (I \cup j)$.
      ∗ (Add sensitive bit locations to $I$ according to $I_{\mathsf{b}}$):
        For $j \in [\mathsf{bn}]$, if $j \in I_{\mathsf{b}}$ and $\exists i \in [l]$ such that $j = \rho_i$, $I \leftarrow (I \cup r_i)$.
  • Output: Output $I$.
```

Fig. 6. The function g_1 that appears in the hybrid experiments of Fig. 9 .

$\mathsf{Enc}(s)$. As we prove in Claim 4.9, it simulates perfectly the tampering effect of f against $C \leftarrow \mathsf{Enc}^*(s)$.

g_1 operates as follows (cf. Figure 6): it simulates perfectly the randomness for the permutation of the block-wise code, denoted as ρ, and constructs a set of indices I, such that g_2 will receive access to, and tamper with, $c_{|_I}$.
The set I is constructed with respect to the set of blocks I_{b}, that f chooses to access, as well as Σ, that reveals the original bit positions, i.e., the ones before permuting $(z||e)$. g_2 receives $c_{|_I}$, reconstructs I, simulates partially the blocks of the block-wise codeword, C, and applies f on the simulated codeword. The program of g_2 is given in Fig. 7.
In Claim 4.9, we show that g_2, given $c_{|_I}$, simulates perfectly $C_{|_{I_{\mathsf{b}}}}$, which implies that $g_2^{\Sigma}(c_{|_I}) = f(C_{|_{I_{\mathsf{b}}}})$, and the two executions are identical.

– $\mathsf{Exp}_2^{(g_1,g_3),s}$: In the current experiment, we substitute the function g_2 with g_3, and Dec^* with Dec, respectively. By inspecting the code of g_2 and g_3 (cf. Figures 7,8, respectively), we observe that latter function executes the code of the former, plus the "Check labels and simulate $\tilde{c}[I]$" step. Thus, the two experiments are identical up to the point of computing $f(C^*_{|_{I_{\mathsf{b}}}})$.
The main idea here is that we want the current execution to be with respect to (Init, Enc, Dec) against (g_1, g_3). Thus, we substitute Dec^* with Dec, and we expand the function g_2 with some extra instructions/checks that are missing from Dec. We name the resulting function as g_3 and we prove that the two executions are identical.
– Finally, we prove that for any f and any s,

$$\Pr\left[\mathsf{Exp}_2^{(g_1,g_3),s} \notin \{\perp, s\}\right] \leq \mathsf{negl}(k).$$

We do so by proving that (g_1, g_3) is admissible for (Init, Enc, Dec,), i.e., $(g_1, g_3) \in \mathcal{F}_{\mathsf{ad}}$, and g_3 will not request to access more that $m - 1$ shares for each bit of sk, sk^3 (cf. Corollary 4.3). This implies security according to Lemma 2.6.

In what follows, we prove indistinguishability between the hybrids.

Claim 4.9. For any $f \in \mathcal{F}_{\Gamma}^{\alpha}$ and any s, $\mathsf{Exp}_0^{f,s} = \mathsf{Exp}_1^{(g_1,g_2),s}$.[18]

[18]For random variables X, Y, $X = Y$ denotes that the random variables are identical.

```
$g_2^{\Sigma}(c_{|_I})$:
    $t \leftarrow 1$, $C_i^* \leftarrow 0^{\mathsf{bs}}$, $i \in [\mathsf{bn}]$.
• (Reconstruct $I$): Compute $I \leftarrow g_1(\Sigma)$.
• (Simulate ciphertext blocks):
  For $i \in [\mathsf{bn}]$, if $i \neq \rho_j$ for $j \in [l]$, $C_i^* \leftarrow (0||e[t : t + (\mathsf{bs} - 1)])$, $t \leftarrow t + (\mathsf{bs} - 1)$.
• (Simulate sensitive blocks):
    * For $i \in [|I|]$, if $\exists j \in [l]$, such that $\mathsf{Ind}(c_{|_I}[i]) = r_j$, set $C_{\rho_j}^* \leftarrow (1||j||c_{|_I}[i])$.
    * Set $C^* := (C_1^*||\ldots||C_{\mathsf{bn}}^*)$ and $\tilde{C}^* := C^*$.
• (Apply $f$): compute $\tilde{C}^*[I_{\mathsf{b}}] \leftarrow f(C^*_{|_{I_{\mathsf{b}}}})$.
• (Output): Output $\tilde{C}^*_{|_{I_{\mathsf{b}}}}$.
```

Fig. 7. The function g_2 that appears in the hybrid experiments of Fig. 9 .

```
$g_3^{\Sigma}(c_{|_I})$:
    $t \leftarrow 1$, $C_i^* \leftarrow 0^{\mathsf{bs}}$, $i \in [\mathsf{bn}]$.
• (Reconstruct $I$): Compute $I \leftarrow g_1(\Sigma)$.
• (Simulate ciphertext blocks):
  For $i \in [\mathsf{bn}]$, if $i \neq \rho_j$, $j \in [l]$, $C_i^* \leftarrow (0||e[t : t + (\mathsf{bs} - 1)])$, $t \leftarrow t + (\mathsf{bs} - 1)$.
• (Simulate sensitive blocks):
    * For $i \in [|I|]$, if $\exists j \in [l]$, such that $\mathsf{Ind}(c_{|_I}[i]) = r_j$, set $C_{\rho_j}^* \leftarrow (1||j||c_{|_I}[i])$.
    * Set $C^* := (C_1^*||\ldots||C_{\mathsf{bn}}^*)$ and $\tilde{C}^* := C^*$.
• (Apply $f$): compute $\tilde{C}^*[I_{\mathsf{b}}] \leftarrow f(C^*_{|_{I_{\mathsf{b}}}})$.
• (Check labels and simulate $\tilde{c}[I]$): For $i \in \{\rho_j | j \in [l]\} \backslash I_{\mathsf{b}}$, $C_i^* \leftarrow (1||i||0)$. If
  $\Pi^{-1}(\tilde{C}^*) = \perp$, set $d \leftarrow 1$, otherwise set $(\tilde{z}^*||\tilde{e}) \leftarrow \Pi^{-1}(\tilde{C}^*)$, $\tilde{c}^* \leftarrow P_{\Sigma}(\tilde{z}^*||\tilde{e})$.
• (Output): If $d = 1$ output $\perp$, otherwise output $\tilde{c}^*_{|_I}$.
```

Fig. 8. The function g_3 that appears in the hybrid experiments of Fig. 9 .

Proof. The main difference between Exp_0 and Exp_1 is that in Exp_1, we introduce the tampering function $g = (g_1, g_2)$ that operates over codewords of $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, and simulates partially the block-wise code. We observe that g_1 simulates perfectly the randomness of the permutation for the block-wise code, denoted as ρ. Thus, the computation $C \leftarrow \Pi_\rho(z||e)$ does not induce any statistical difference between the two experiments. By the definition of g_1, we have that $c_{|_I}$ consists of all ciphertext bits, as well as the indices r_i, for which $\rho_i \in I_{\mathsf{b}}$, $i \in [l]$, i.e., if f requests access to the sensitive block with index ρ_i, containing $z[i]$, g_1 will request access to the r_i-th bit of c, which is $z[i]$. Thus, g_2 will receive as input the entire ciphertext and all the sensitive bits that f will request access to, with respect to I_{b}, thus it can fully simulate $C_{|_{I_{\mathsf{b}}}}$ while being consistent with the distribution of blocks in $C_{|_{I_{\mathsf{b}}^{\mathsf{c}}}}$, as ρ is generated by g_1. Thus, we have that $g_2^{\Sigma}(c_{|_I})$ is identical to $f(C_{|_{I_{\mathsf{b}}}})$, and the proof of the claim is complete. □

Claim 4.10. For any $f \in \mathcal{F}_\Gamma^\alpha$ and any s, $\mathsf{Exp}_1^{(g_1,g_2),s} = \mathsf{Exp}_2^{(g_1,g_3),s}$.

Proof. In Exp_2 we substitute the function g_2 with g_3, and Dec^* with Dec, respectively. By inspecting the code of g_2 and g_3, we observe that latter function executes the code of the former, plus the "Check labels and simulate $\tilde{c}[I]$" step. Thus, the two experiments are identical up to the point of computing $f(C_{|_{I_{\mathsf{b}}}})$. We unfold the code of the two experiments from that point of the computation and on (cf. Figure 10). The idea is that the consistency check on the labels of the block-wise code is transferred from Dec^* in Exp_1 to g_3 in

```
$\mathsf{Exp}_0^{f,s}$ :
$sk \leftarrow \mathsf{KGen}\left(1^k\right), e \leftarrow \mathsf{E}_{sk}(s)$
$z \leftarrow \mathsf{SS}_m(sk||sk^3)$

$\rho := (\rho_1, \ldots, \rho_l) \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$
$C \leftarrow \Pi_\rho(z||e), \tilde{C} \leftarrow C$
$\tilde{C}[I_\mathsf{b}] \leftarrow f(C_{|I_\mathsf{b}})$

$\tilde{s} \leftarrow \mathsf{Dec}^*(\tilde{C})$

Output $\mathsf{same}^*$ if $\tilde{s} = s$ and $\tilde{s}$ otherwise.
```

```
$\mathsf{Exp}_1^{(g_1,g_2),s}$ :
$sk \leftarrow \mathsf{KGen}\left(1^k\right), e \leftarrow \mathsf{E}_{sk}(s)$
$z \leftarrow \mathsf{SS}_m(sk||sk^3)$

$\Sigma \leftarrow \mathsf{Init}(1^k), c \leftarrow P_\Sigma(z||e)$
$I \leftarrow g_1(\Sigma)$
$C \leftarrow \Pi_\rho(z||e), \tilde{C} \leftarrow C$
$\tilde{C}[I_\mathsf{b}] \leftarrow g_2^\Sigma(c_{|I})$

$\tilde{s} \leftarrow \mathsf{Dec}^*(\tilde{C})$

Output $\mathsf{same}^*$ if $\tilde{s} = s$ and $\tilde{s}$ otherwise.
```

```
$\mathsf{Exp}_2^{(g_1,g_3),s}$ :
$\Sigma \leftarrow \mathsf{Init}(1^k)$
$sk \leftarrow \mathsf{KGen}\left(1^k\right), e \leftarrow \mathsf{E}_{sk}(s)$
$z \leftarrow \mathsf{SS}_m(sk||sk^3)$

$c \leftarrow P_\Sigma(z||e)$, $\tilde{c} \leftarrow c$
$I \leftarrow g_1(\Sigma)$
$\tilde{c}[I] \leftarrow g_3^\Sigma(c_{|I})$

$\tilde{s} \leftarrow \mathsf{Dec}(\Sigma, \tilde{c})$

Output $\mathsf{same}^*$ if $\tilde{s} = s$ and $\tilde{s}$ otherwise.
```

Fig. 9. The hybrid experiments for the proof of Theorem 4.8 .

Exp_2, and Dec^* is substituted by Dec, so that $\mathsf{Exp}_2^{(g_1,g_3),s}$ is the tampering experiment of Lemma 2.6 with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ and (g_1, g_3).

In order to show that $\mathsf{Exp}_1^{(g_1,g_2),s} = \mathsf{Exp}_2^{(g_1,g_3),s}$, is suffices to prove that $\mathsf{Dec}^*(\tilde{C}) = \mathsf{Dec}(\tilde{c})$. By inspecting $\mathsf{Exp}_2^{(g_1,g_3),s}$, we have that $\tilde{c} = \perp$ if and only if $\Pi^{-1}(\tilde{C}^*) = \perp$. By the definition of Π^{-1} (cf. Construction 4.1), $\Pi^{-1}(\tilde{C}^*) = \perp$, if and only if the tampering function creates an inconsistent set of labels, an effect that can be decided by g_3 by only partially accessing C, since it fully simulates the labels for the block-wise code. By Claim 4.9, $C_{|I_\mathsf{b}} = C^*_{|I_\mathsf{b}}$ and thus $\tilde{C}_{|I_\mathsf{b}} = \tilde{C}^*_{|I_\mathsf{b}}$, which implies that $\Pi^{-1}(\tilde{C}^*) = \perp$ if and only if $\Pi^{-1}(\tilde{C}) = \perp$. We conclude that $\tilde{c} = \perp$ if and only if $\Pi^{-1}(\tilde{C}) = \perp$. Let

```
$\mathsf{Exp}_1^{(g_1,g_2),s}$ :
    ⋮
$g_2^{\Sigma}(c_{|_I})$ :
    ⋮
    $\tilde{C}^*[I_{\mathsf{b}}] \leftarrow f(C^*_{|_{I_{\mathsf{b}}}})$
    $\tilde{C}[I_{\mathsf{b}}] \leftarrow \tilde{C}^*_{|_{I_{\mathsf{b}}}}$

$\mathsf{Dec}^*(\tilde{C})$ :
    If $\Pi^{-1}(\tilde{C}) \neq \perp$ :
        $(\tilde{z}||\tilde{e}) \leftarrow \Pi^{-1}(\tilde{C})$
        $(\tilde{sk}||\tilde{sk}') \leftarrow \mathsf{Rec}_m(\tilde{z})$
            If $\tilde{sk}^3 = \tilde{sk}'$ : $\tilde{s} \leftarrow \mathsf{D}_{\tilde{sk}}(\tilde{e})$
            Else: $\tilde{s} \leftarrow \perp$
    Else: $\tilde{s} \leftarrow \perp$
    Output $\tilde{s}$
    ⋮
```

```
$\mathsf{Exp}_2^{(g_1,g_3),s}$ :
    ⋮
$g_3^{\Sigma}(c_{|_I})$ :
    ⋮
    $\tilde{C}^*[I_{\mathsf{b}}] \leftarrow f(C^*_{|_{I_{\mathsf{b}}}})$
    If $\Pi^{-1}(\tilde{C}^*) = \perp : d \leftarrow 1$
    Else:
        $(\tilde{z}^*||\tilde{e}) \leftarrow \Pi^{-1}(\tilde{C}^*)$
        $\tilde{c}^* \leftarrow P_{\Sigma}(\tilde{z}^*||\tilde{e})$
    If $d = 1$: output $\perp$
    Else: output $\tilde{c}^*_{|_I}$

$\mathsf{Dec}(\tilde{c})$ :
    If $\tilde{c} \neq \perp$ :
        $(\tilde{z}||\tilde{e}) \leftarrow P_{\Sigma}^{-1}(\tilde{c})$
        $(\tilde{sk}||\tilde{sk}') \leftarrow \mathsf{Rec}_m(\tilde{z})$
            If $\tilde{sk}^3 = \tilde{sk}'$ : $\tilde{s} \leftarrow \mathsf{D}_{\tilde{sk}}(\tilde{e})$
            Else: $\tilde{s} \leftarrow \perp$
    Else: $\tilde{s} \leftarrow \perp$
    Output $\tilde{s}$
    ⋮
```

Fig. 10. The unfolded code of Exp_1 and Exp_2.

E be the event in which $\tilde{c} \neq \perp$. Clearly, conditioned on $\neg E$ the two experiments are identical, as both output $\perp$. It remains to prove the same conditioned on E.

By inspecting the two experiments, and conditioned on E, we have

$$\tilde{c}_{|_I} = \tilde{c}^*_{|_I} = \left[P_{\Sigma}\left(\Pi^{-1}(\tilde{C}^*)\right)\right]_{|_I} = \left[P_{\Sigma}\left(\Pi^{-1}(\tilde{C})\right)\right]_{|_I}, \qquad (7)$$

where the last equality follows from the fact that $\left[P_{\Sigma}\left(\Pi^{-1}(\tilde{C}^*)\right)\right]_{|_I}$ is independent of the blocks of $\tilde{C}$ that Exp_2 does not have access to. Moreover,

$$\tilde{c}_{|_{I^{\mathsf{c}}}} = c_{|_{I^{\mathsf{c}}}} = \left[P_{\Sigma}\left(\Pi^{-1}(C)\right)\right]_{|_{I^{\mathsf{c}}}} = \left[P_{\Sigma}\left(\Pi^{-1}(\tilde{C})\right)\right]_{|_{I^{\mathsf{c}}}}, \qquad (8)$$

where the last equality follows from the fact that $\tilde{c}_{|_{I^{\mathsf{c}}}}$ is not being accessed by the tampering function. From the above relations, we have that $\tilde{c} = P_{\Sigma}\left(\Pi^{-1}(\tilde{C})\right)$, thus $P_{\Sigma}^{-1}(\tilde{c}) = \Pi^{-1}(\tilde{C})$, and the two executions are identical conditioned on E. □

Claim 4.11. Assuming $(1-\alpha)m = \omega(\log(k))$, for any $f \in \mathcal{F}_{\Gamma}^{\alpha}$ and any s,

$$\Pr\left[\mathsf{Exp}_2^{(g_1,g_3),s} \notin \{\perp, s\}\right] \leq \mathsf{negl}(k),$$

over the randomness of Exp_2.

Proof. Assuming $(1-\alpha)m = \omega(\log(k))$, it suffices to prove that for any $f \in \mathcal{F}_{\Gamma}^{\alpha}$, the function $(g_1, g_3) \in \mathcal{F}_{\mathsf{ad}}$ is admissible for $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec},)$, i.e., g_1 will not request to access more that $m-1$ shares for each bit of sk, sk^3, and the proof of the claim will follow by Corollary 4.3 and Lemma 2.6. We prove that for any $f \in \mathcal{F}_{\Gamma}^{\alpha}$, the corresponding (g_1, g_3) will not access the entire z_i, for all $i \in [|2sk|]$, with overwhelming probability. Such an event takes place if and only if $\exists i : |(I_{\mathsf{b}} \cap Z_i)| = m$. We define by E_i the event in which f request access to all blocks in which z_i is stored. Assuming f reads n blocks, we have that for all $i \in [2|sk|]$,

$$\Pr_{\rho}[E_i] = \Pr_{\rho}[\ |I_{\mathsf{b}} \cap Z_i| = m\ ] = \prod_{j=0}^{m-1} \frac{n-j}{\nu-j} \leq \left(\frac{n}{\nu}\right)^m .$$

We have $n = \alpha\nu$ and assuming $\alpha = 1-\epsilon$ for $\epsilon \in (0, 1]$, we have $\Pr[E_i] \leq (1-\epsilon)^m \leq 1/e^{m\epsilon}$ and

$$\Pr[E] = \Pr_{\rho}\left[\bigcup_{i=1}^{2|sk|} E_i\right] \leq \frac{2|sk|}{e^{m\epsilon}},$$

which is negligible when $(1-\alpha)m = \omega(\log(k))$. □

The security of the block-wise code follows from the above claims and the MD-NMC security of $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$. □

Instantiations and rates. By instantiating Construction 4.1, with 2.11, Theorem 4.8, for $m = k\log k, \alpha = 1 - 1/\Omega(\log k)$, yields rate $1 - 1/\Omega(\log k)$ MD-NMC, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s|(1 + 1/O(\log k)) + O(k^2 \log^2 k)$, assuming one-way functions. In addition, by instantiating Construction 4.1, with 2.12, Theorem 4.8, for $m = |s|\log|s|, \alpha = 1 - 1/O(\log(|s|))$, yields unconditionally secure, rate $1/O(|s|\log^2(|s|))$ MD-NMC in the standard model, with access rate $\Omega(1 - 1/\log(|s|))$ and codewords of length $O(|s|^2 \log^2 |s|)$.

5. Continuous MD-NMC with light Updates

In this section, we enhance the block-wise scheme of Sect. 4 with an update mechanism, that uses only shuffling and refreshing operations. The resulting code is secure against continuous attacks, for a notion of security that is weaker than the original one [45], as we need to update the codeword after each round of execution. However, our update mechanism is using cheap operations, avoiding the full decoding and re-encoding of the message, which is the standard way to achieve continuous security [39,55] using a one-time NMC. In addition, our solution avoids the usage of a self-destruction mechanism that produces $\perp$ in all subsequent rounds after the first round in which the attacker creates an invalid codeword, which was originally proposed by [41]. Avoiding the self-destruction mechanism was originally proposed by [41], and it is an important step toward practicality, as (i) the mechanism is subjective to denial of service attacks, and (ii) it renders the device useless in the presence of non-adversarial hardware faults. Our solution enables normal use of the device in the presence of such faults *and* provides security against malicious attacks.[19]

The update mechanism of the proposed scheme works as follows: in each round, it randomly shuffles the blocks and refreshes the randomness of the inner encoding of sk. The idea here is that, due to the continual shuffling and refreshing of the inner encoding scheme, in each round the attacker learns nothing about the secret key, and every attempt to modify the inner encoding, results to an invalid key, with overwhelming probability. Our update mechanism can be made deterministic if we further encode the seed of a PRG together with the secret key, which is similar to the technique presented in [55].

Below we define the update mechanism, which is denoted as Update^*.

Construction 5.1. *Let $k, m \in \mathbb{N}$, and let* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, $(\mathsf{SS}_m, \mathsf{Rec}_m)$, Enc^*, Dec^*, *be as in Construction 4.1. We define the update procedure,* Update^*, *for the encoding scheme of Construction 4.1, as follows:*

– $\mathsf{Update}^*(1^k, \cdot)$*: on input C, parse it as $(C_1||\ldots||C_{\mathsf{bn}})$, set $l \leftarrow 2m|sk|$, $\hat{\mathcal{L}} = \emptyset$, and set $\hat{C} := (\hat{C}_1||\ldots||\hat{C}_{\mathsf{bn}})$ to zeros.*

- **(Secret share $0^{2|sk|}$)**: *Sample $z \leftarrow \mathsf{SS}_m\left(0^{2|sk|}\right)$, where $z = ||_{i=1}^{2|sk|} z_i$, $z \in \{0,1\}^{2m|sk|}$, and for $i \in [2|sk|]$, z_i is an m-out-of-m secret sharing of the 0 bit.*
- **(Shuffle & Refresh)**: *Sample $\rho := (\rho_1, \ldots, \rho_l) \overset{\mathsf{rs}}{\leftarrow} \{0,1\}^{\log(\mathsf{bn})}$. For $i \in [\mathsf{bn}]$,*
  - ∗ **(Sensitive block)** *If $C_i[1] = 1$,*
    - · **(Shuffle)**: *Set $j \leftarrow C_i[2 : \mathsf{bs} - 1]$, $\hat{C}_{\rho_j} \leftarrow C_i$.*
    - · **(Refresh)**: *Set $\hat{C}_{\rho_j}[\mathsf{bs}] \leftarrow \hat{C}_{\rho_j}[\mathsf{bs}] \oplus z[j]$.*
  - ∗ **(Ciphertext block)**

If $C_i[1] = 0$, set $j \leftarrow \min_n \left\{ n \in [\mathsf{bn}] \middle| n \notin \hat{\mathcal{L}}, n \neq \rho_i, i \in [l] \right\}$, and $\hat{C}_j \leftarrow C_i$, $\hat{\mathcal{L}} \leftarrow \hat{\mathcal{L}} \cup \{j\}$.

Output $\hat{C}$.

The following definition of security is along the lines of the one given in [45], adapted to the notion of non-malleability with manipulation detection. Also, after each invocation the codewords are updated, where in our case the update mechanism is only using shuffling and refreshing operations. In addition, there is no need for self-destruct after detecting an invalid codeword [41].

Definition 5.2. (*Continuous* MD-NMC *with* light *updates*) Let $\mathsf{CS} = (\mathsf{Enc}, \mathsf{Dec})$ be an encoding scheme, $\mathcal{F}$ be a function class and $k, q \in \mathbb{N}$. Then, CS is a q-continuously non-malleable code with manipulation detection (q-MD-CNMC) with light updates, if

[19]We assume that the attacks against the memory are non-persistent.

for every, sufficiently large $k \in \mathbb{N}$, any pair of messages $s_0, s_1 \in \{0,1\}^{\text{poly}(k)}$, and any algorithm $\mathcal{A}$,

$$\left\{\mathsf{Tamper}_{s_0}^{\mathcal{A}}(k)\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{Tamper}_{s_1}^{\mathcal{A}}(k)\right\}_{k\in\mathbb{N}},$$

where,

```
$\mathsf{Tamper}_s^{\mathcal{A}}(k)$ :
$C \leftarrow \mathsf{Enc}(1^k, s), \tilde{s} \leftarrow 0$
For $\tau \in [q]$ :
    $f \leftarrow \mathcal{A}(\tilde{s}), \tilde{C} \leftarrow f(C), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{C})$
    If $\tilde{s} = s$ : $\tilde{s} \leftarrow \mathsf{same}^*$
    $C \leftarrow \mathsf{Update}^*(1^k, C)$
$out \leftarrow \mathcal{A}(\tilde{s})$
**Return** : $out$
```

and for each round the output of the decoder is not in $\{s, \perp\}$ with negligible probability in k, over the randomness of $\mathsf{Tamper}_s^{\mathcal{A}}$.

Below we prove that the scheme of Construction 5.1 is continuously non-malleable with manipulation detection and light updates.

Theorem 5.3. *Let $q, k, m, \in \mathbb{N}$, $\Gamma = \{0,1\}^{O(\log(k))}$ and $\alpha \in [0,1)$. Assuming* $(\mathsf{SS}_m, \mathsf{Rec}_m)$ *is an m-out-of-m secret sharing scheme and* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ *is a* 1-IND-CPA*, authenticated encryption scheme, the scheme of Construction 5.1 is a* MD-CNMC *with light updates, against $\mathcal{F}_{\Gamma}^{\alpha}$, for any α, m, such that* $(1-\alpha)m = \omega(\log(k))$.

Proof. Let $\mathcal{A}$ be any adversary playing against $\mathsf{Tamper}_s^{\mathcal{A}}$, for any s. Let I_{b} be the set of indices chosen by the attacker in each round and $I^{\mathsf{c}} = [\nu]\backslash I$. The tampered components of the codeword will be denoted using the symbol "~" on top of the original symbol. Our proof follows the strategy of the one given in Theorem 3.2, using a series of hybrid experiments that are depicted in Fig. 11. Below, we describe the hybrids.

- $\mathsf{Exp}_0^{\mathcal{A},s,q}$: For any $\mathcal{A}$, s, q, the experiment $\mathsf{Exp}_0^{\mathcal{A},s,q}$ is the experiment $\mathsf{Tamper}_s^{\mathcal{A}}$, of Definition 5.2.
- $\mathsf{Exp}_1^{\mathcal{A},s,q}$: In the second experiment and for each round of the execution, we define Z_i, $i \in [2|sk|]$, to be the set of indices in which z_i is stored, $|Z_i| = m$. Intuitively, in each round, by calling the Update^* procedure that permutes the blocks using a fresh permutation key and updates the shares of sk and sk^3, we achieve the following: in each round, the attacker finds all shares for a bit of sk, and sk^3, with negligible probability in k, thus the tampering function is not accessing any bit of sk and sk^3, even if it is given access to $(1-o(1))\nu$ blocks of the codeword. Thus, the indistinguishability between the current experiment and the previous one comes from a claim analogous to Claim 3.3, made in the proof of Theorem 3.2. In particular, we have the following claim.

```
$\mathsf{Exp}_0^{\mathcal{A},s,q}$ :
$C \leftarrow \mathsf{Enc}(s), \tilde{s} \leftarrow 0, \tilde{C} \leftarrow C$
For $\tau \in [q]$ :
    $f \leftarrow \mathcal{A}(\tilde{s})$

    $\tilde{C}_{|I_{\mathsf{b}}} \leftarrow f(C_{|I_{\mathsf{b}}}), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{C})$
    If $\tilde{s} = s$ :
        $\tilde{s} \leftarrow \mathsf{same}^*$
    $C \leftarrow \mathsf{Update}^*(1^k, C)$

$out \leftarrow \mathcal{A}(\tilde{s})$
Return : $out$
```

```
$\mathsf{Exp}_1^{\mathcal{A},s,q}$ :
$C \leftarrow \mathsf{Enc}(s), \tilde{s} \leftarrow 0, \tilde{C} \leftarrow C$
For $\tau \leq [q]$ :
    $f \leftarrow \mathcal{A}(\tilde{s})$, $\tilde{s} \leftarrow \perp$
    If $\forall i : |(I_{\mathsf{b}} \cap Z_i)| < m$:
        $\tilde{C}_{|I_{\mathsf{b}}} \leftarrow f(C_{|I_{\mathsf{b}}}), \tilde{s} \leftarrow \mathsf{Dec}(\tilde{C})$
        If $\tilde{s} = s$ :
            $\tilde{s} \leftarrow \mathsf{same}^*$
    $C \leftarrow \mathsf{Update}^*(1^k, C)$

$out \leftarrow \mathcal{A}(\tilde{s})$
Return : $out$
```

```
$\mathsf{Exp}_2^{\mathcal{A},s,q}$ :
$\rho \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$
$sk \leftarrow \mathsf{KGen}(1^k), e \leftarrow \mathsf{E}_{sk}(s)$
$\tilde{s} \leftarrow 0$
For $\tau \in [q]$ :
    $f \leftarrow \mathcal{A}(\tilde{s}), \tilde{s} \leftarrow \perp$
    $z^* \leftarrow \bar{\mathsf{S}}\mathsf{S}_m^{f,\rho}(sk), C \leftarrow \Pi_\rho(z^* || e)$
    If $\forall i : |(I_{\mathsf{b}} \cap Z_i)| < m$:
        $\tilde{C}_{|I_{\mathsf{b}}} \leftarrow f(C_{|I_{\mathsf{b}}})$
        $w_i \leftarrow \bigoplus_{j \in (I_{\mathsf{b}} \cap Z_i)} C_j[\mathsf{bs}]$
        $\tilde{w}_i \leftarrow \bigoplus_{j \in (I_{\mathsf{b}} \cap \tilde{Z}_i)} \tilde{C}_j[\mathsf{bs}]$

        If $\forall i : w_i = \tilde{w}_i$:
            $\tilde{s} \leftarrow \mathsf{D}_{sk}(\tilde{e})$

        If $\tilde{s} = s$ :
            $\tilde{s} \leftarrow \mathsf{same}^*$
    $C \leftarrow \mathsf{Update}^*(1^k, C)$

$out \leftarrow \mathcal{A}(\tilde{s})$
Return : $out$
```

```
$\mathsf{Exp}_3^{\mathcal{A},s,q}$ :
$\rho \xleftarrow{\mathsf{rs}} \{0,1\}^{\log(\mathsf{bn})}$
$sk \leftarrow \mathsf{KGen}(1^k), e \leftarrow \mathsf{E}_{sk}(s)$
$\tilde{s} \leftarrow 0$

For $\tau \in [q]$ :
    $f \leftarrow \mathcal{A}(\tilde{s}), \tilde{s} \leftarrow \perp$
    $z^* \leftarrow \bar{\mathsf{S}}\mathsf{S}_m^{f,\rho}(sk), C \leftarrow \Pi_\rho(z^* || e)$
    If $\forall i : |(I_{\mathsf{b}} \cap Z_i)| < m$:
        $\tilde{C} \leftarrow f(C_{|I_{\mathsf{b}}})$
        $w_i \leftarrow \bigoplus_{j \in (I_{\mathsf{b}} \cap Z_i)} C_j[\mathsf{bs}]$
        $\tilde{w}_i \leftarrow \bigoplus_{j \in (I_{\mathsf{b}} \cap \tilde{Z}_i)} \tilde{C}_j[\mathsf{bs}]$

        If $\forall i : w_i = \tilde{w}_i$ :
            If $\tilde{e} = e$:
                $\tilde{s} \leftarrow \mathsf{same}^*$

    $C \leftarrow \mathsf{Update}^*(1^k, C)$

$out \leftarrow \mathcal{A}(\tilde{s})$
Return : $out$
```

Fig. 11. Hybrids for the proof of Theorem 5.3. The gray part signifies the portion of the code of an experiment that differs from the previous one.

Claim 5.4. For $k, q, m \in \mathbb{N}$, assume $(1-\alpha)m = \omega(\log(k))$. Then, for any $\mathcal{A}$ that chooses its tampering strategy from $\mathcal{F}_{\Gamma}^{\alpha}$, and any message s, we have $\mathsf{Exp}_0^{\mathcal{A},s,q} \approx \mathsf{Exp}_1^{\mathcal{A},s,q}$, where the probability runs over the randomness used by Enc^*, Update^*.

– $\mathsf{Exp}_2^{\mathcal{A},s,q}$: In the third experiment, we define by $\tilde{Z}_i$ to be the set of indices in which $\tilde{z}_i$ is stored, $|\tilde{Z}_i| = m$. The main difference with the previous experiment is that we unfold the encoding procedure, and in addition, we substitute the secret sharing procedure SS_m with $\bar{\mathsf{SS}}_m^{f,\rho}$, defined as follows:

$\bar{\mathsf{SS}}_m^{f,\rho}(sk)$:

1. Sample $\left(z_1^*, \ldots, z_{2|sk|}^*\right) \leftarrow \mathsf{SS}_m\left(sk||sk^3\right)$.
2. For $i \in [2|sk|]$:

$$l_i := \max_d \left\{d \in [m] \wedge \mathsf{Ind}\left(z_i[d]\right) \notin I_{\mathsf{b}}\right)\},$$

   where Ind returns the index of $z_i[d]$ in C, i.e., l_i is the largest index in $[m]$ such that the codeword block containing $z_i[l_i]$, is not accessed by f.
3. (**Output**): For all i set $z_i^*[l_i] = *$, and output $z^* :=\|_{i=1}^{2|sk|} z_i^*$.

In $\mathsf{Exp}_1^{\mathcal{A},s,q}$, we have $z = \|_{i=1}^{2|sk|} z_i$, and each z_i is an m-out-of-m secret sharing for a bit of sk or sk^3. From the first transition we have that for all i, $|I_{\mathsf{b}} \cap Z_i| < m$ with overwhelming probability, and the current experiment is identical to the previous one up to the point of computing $f(C|_{I_{\mathsf{b}}})$, as $C|_{I_{\mathsf{b}}}$ and $f(C|_{I_{\mathsf{b}}})$ depend only on z^*, that gives no information about sk and sk^3.

Another difference between the two experiments is that, after applying the tampering function, $\mathsf{Exp}_1^{\mathcal{A},s,q}$ makes a call on the decoder while $\mathsf{Exp}_2^{\mathcal{A},s,q}$, checks if the tampering function has modified the shares in a way such that the reconstruction procedure will give $\tilde{sk} \neq sk$ or $\tilde{sk}' \neq sk'$. In case modification is detected the current experiments sends $\perp$ to the attacker. The main idea here is that, a tampering function that modifies the shares of sk and sk^3, creates shares of $\tilde{sk}$ and $\tilde{sk}'$, such that $\tilde{sk}^3 = \tilde{sk}'$, with negligible probability in k. We prove this by a reduction to the 1-IND-CPA security of the encryption scheme in the presence of z^*, that as we have already stated, it gives no information about the secret key. The indistinguishability between the two experiments comes from the following claim, whose proof is similar to the one given in Claim 3.4.

Claim 5.5. Assuming $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is 1-IND-CPA secure (cf. Definition 2.9), for any $\mathcal{A}$ choosing its tampering strategy from $\mathcal{F}_{\Gamma}^{\alpha}$, and any message s, $\mathsf{Exp}_1^{\mathcal{A},s,q} \approx \mathsf{Exp}_2^{\mathcal{A},s,q}$, where the probability runs over the randomness used by Enc^*, Update^*.

– $\mathsf{Exp}_3^{\mathcal{A},s,q}$: In the final experiment, in each round of the execution, instead of calling the decryption $\mathsf{D}_{sk}(\tilde{e})$, we first check if the attacker has modified the ciphertext, in which case the current experiment outputs $\perp$, otherwise it outputs same^*. This part of the program is reached only if the tampering function does not modify the secret key. Thus, the indistinguishability between the two experiments follows from the authenticity property of the encryption scheme in the presence of z^*, which is updated in each round depending on the set I_{b}. Clearly, requesting z^* adaptively in each round does not compromise the security of the encryption scheme, as z^* carries no information about sk. Thus, in each round, given that $\tilde{sk} = sk$ and $\tilde{sk}' = sk'$, we have that if the attacker modifies the ciphertext, then with overwhelming probability $\mathsf{D}_{sk}(\tilde{e}) = \perp$, otherwise, $\mathsf{D}_{sk}(\tilde{e}) = s$, and the current experiment correctly sends $\tilde{s} = \mathsf{same}^*$ to the attacker. Thus, we have the following claim.

Claim 5.6. Assuming the authenticity property of $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, for any $\mathcal{A}$ choosing its tampering strategy from $\mathcal{F}_{\Gamma}^{\alpha}$, and any message s, $\mathsf{Exp}_2^{\mathcal{A},s,q} \approx \mathsf{Exp}_3^{\mathcal{A},s,q}$, where the probability runs over the randomness used by KGen, E and Update^*.

– Finally, we have that for any $\mathcal{A}$ choosing its tampering strategy from $\mathcal{F}_{\Gamma}^{\alpha\nu}$, and any message s, $\mathsf{Exp}_3^{\mathcal{A},s,q}$ is indistinguishable from $\mathsf{Exp}_3^{\mathcal{A},\mathsf{0},q}$, where $\mathsf{0}$ denotes the zero message. This follows by the semantic security of the encryption scheme in the presence of z^*, for the multi-round case.

Claim 5.7. Assuming $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ is semantically secure, for any $\mathcal{A}$ choosing its tampering strategy from $\mathcal{F}_{\Gamma}^{\alpha}$, $\mathsf{Exp}_3^{\mathcal{A},s,q} \approx \mathsf{Exp}_3^{\mathcal{A},\mathsf{0},q}$, where the probability runs over the randomness used by KGen, E, Update, "$\approx$" may refer to statistical or computational indistinguishability, and $\mathsf{0}$ denotes the zero message.

The above claims conclude our proof. Clearly, the manipulation detection property follows from the fact that the output of Exp_3 is in $\{\mathsf{same}^*, \perp\}$, with overwhelming probability. □

In the above theorem, q can be polynomial (resp. exponential) in k, assuming the underlying encryption scheme is computationally (resp. unconditionally) secure.

Instantiations and rates. By instantiating Construction 5.1, with 2.11, Theorem 5.3, for $m = k \log k$, $\alpha = 1 - 1/\Omega(\log k)$, yields rate $1 - 1/\Omega(\log k)$ MD-NMC, with access rate $1 - 1/\Omega(\log k)$ and codewords of length $|s|(1 + 1/O(\log k)) + O(k^2 \log^2 k)$, assuming one-way functions. In addition, by instantiating Construction 5.1, with 2.12, Theorem 5.3, for $m = |s| \log |s|$, $\alpha = 1 - 1/O(\log(|s|))$, yields unconditionally secure, rate $1/O(|s| \log^2(|s|))$ MD-NMC in the standard model, with access rate $\Omega(1 - 1/\log(|s|))$ and codewords of length $O(|s|^2 \log^2 |s|)$.

Acknowledgements

Feng-Hao Liu was supported by the NSF award CNS 2402031. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

[1] D. Aggarwal, S. Agrawal, D. Gupta, H.K. Maji, O. Pandey, M. Prabhakaran, Optimal computational split-state non-malleable codes, in E. Kushilevitz, T. Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*. Lecture Notes in Computer Science, vol. 9563 (Springer, Heidelberg, 2016), pp. 393–417, Tel Aviv, Israel, Jan. 10–13, 2016

[2] D. Aggarwal, Y. Dodis, T. Kazana, M. Obremski, Non-malleable reductions and applications, in R.A. Servedio, R. Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing* (ACM Press, Portland, 2015), pp. 459–468

[3] D. Aggarwal, Y. Dodis, S. Lovett. Non-malleable codes from additive combinatorics, in D.B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing* (ACM Press, New York, 2014), pp. 774–783

[4] S. Agrawal, D. Gupta, H.K. Maji, O. Pandey, M. Prabhakaran, Explicit non-malleable codes against bit-wise tampering and permutations, in R. Gennaro, M.J.B. Robshaw, editors, *Advances in Cryptology—CRYPTO 2015, Part I*. Lecture Notes in Computer Science, vol. 9215, Santa Barbara, CA, USA, Aug. 16–20, 2015 (Springer, Heidelberg, 2015), pp. 538–557

[5] S. Agrawal, D. Gupta, H.K. Maji, O. Pandey, M. Prabhakaran, A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations, in Y. Dodis, J.B. Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*. Lecture Notes in Computer Science, vol. 9014, Warsaw, Poland, Mar. 23–25, 2015 (Springer, Heidelberg, 2015), pp. 375–397

[6] J. Alwen, S. Coretti, Y. Dodis, Y. Tselekounis, Security analysis and improvements for the IETF MLS standard for group messaging, in D. Micciancio, T. Ristenpart, editors, *Advances in Cryptology—CRYPTO 2020* (Springer, Cham, 2020), pp. 248–277

[7] J. Alwen, S. Coretti, Y. Dodis, Y. Tselekounis, Modular design of secure group messaging protocols and the security of MLS, in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21* (Association for Computing Machinery, New York, 2021), pp. 1463–1483

[8] J. Alwen, M. Mularczyk, Y. Tselekounis, Fork-resilient continuous group key agreement, in H. Handschuh, A. Lysyanskaya, editors, *Advances in Cryptology—CRYPTO 2023* (Springer, Cham, 2023), pp. 396–429

[9] G. Ateniese, A. Kiayias, B. Magri, Y. Tselekounis, D. Venturi, Secure outsourcing of circuit manufacturing. Cryptology ePrint Archive, Paper 2016/527, 2016. https://eprint.iacr.org/2016/527

[10] G. Ateniese, A. Kiayias, B. Magri, Y. Tselekounis, D. Venturi, Secure outsourcing of cryptographic circuits manufacturing, in J. Baek, W. Susilo, J. Kim, editors, *Provable Security* (Springer, Cham, 2018), pp. 75–93

[11] M. Ball, E. Chattopadhyay, J.-J. Liao, T. Malkin, L.-Y. Tan, Non-malleability against polynomial tampering, in D. Micciancio, T. Ristenpart, editors, *Advances in Cryptology—CRYPTO 2020, Part III*. Lecture Notes in Computer Science, vol. 12172, Santa Barbara, CA, USA, Aug. 17–21, 2020 (Springer, Heidelberg, 2020), pp. 97–126

[12] M. Ball, D. Dachman-Soled, S. Guo, T. Malkin, L.-Y. Tan, Non-malleable codes for small-depth circuits, in M. Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, Paris, France, Oct. 7–9, 2018 (IEEE Computer Society Press, 2018), pp. 826–837

[13] M. Ball, D. Dachman-Soled, M. Kulkarni, H. Lin, T. Malkin, Non-malleable codes against bounded polynomial time tampering, in Y. Ishai, V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*. Lecture Notes in Computer Science, vol. 11476, Darmstadt, Germany, May 19–23, 2019 (Springer, Heidelberg, 2019), pp. 501–530

[14] M. Ball, D. Dachman-Soled, M. Kulkarni, T. Malkin, Non-malleable codes for bounded depth, bounded fan-in circuits, in M. Fischlin, J.-S. Coron, editors, *Advances in Cryptology—EUROCRYPT 2016, Part II*.

Lecture Notes in Computer Science, vol. 9666, Vienna, Austria, May 8–12, 2016 (Springer, Heidelberg, 2016), pp. 881–908

[15] M. Ball, D. Dachman-Soled, M. Kulkarni, T. Malkin, Non-malleable codes from average-case hardness: AC^0, decision trees, and streaming space-bounded tampering, in J.B. Nielsen, V. Rijmen, editors, *Advances in Cryptology—EUROCRYPT 2018, Part III*. Lecture Notes in Computer Science, vol. 10822, Tel Aviv, Israel, Apr. 29 – May 3, 2018 (Springer, Heidelberg, 2018), pp. 618–650

[16] F. Bao, R.H. Deng, Y. Han, A. Jeng, A.D. Narasimhalu, T. Ngair, *Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults* (Springer, Berlin, 1998), pp. 115–124

[17] M. Bellare, S. Tessaro, A. Vardy. Semantic security for the wiretap channel, in R. Safavi-Naini, R. Canetti, editors, *Advances in Cryptology—CRYPTO 2012*. Lecture Notes in Computer Science, vol. 7417, Santa Barbara, CA, USA, Aug. 19–23, 2012 (Springer, Heidelberg, 2012), pp. 294–311

[18] P. Bhatotia, M. Kohlweiss, L. Martinico, Y. Tselekounis, Steel: composable hardware-based stateful and randomised functional encryption, in J.A. Garay, editor, *Public-Key Cryptography—PKC 2021* (Springer, Cham, 2021), pp. 709–736

[19] E. Biham, A. Shamir, Differential fault analysis of secret key cryptosystems, in B.S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO'97*. Lecture Notes in Computer Science, vol. 1294, Santa Barbara, CA, USA, Aug. 17–21, 1997 (Springer, Heidelberg, 1997), pp. 513–525

[20] D. Boneh, R.A. DeMillo, R.J. Lipton, On the importance of checking cryptographic protocols for faults (extended abstract), in W. Fumy, editor, *Advances in Cryptology—EUROCRYPT'97*. Lecture Notes in Computer Science, vol. 1233, Konstanz, Germany, May 11–15, 1997 (Springer, Heidelberg, 1997), pp. 37–51

[21] D. Boneh, R.A. DeMillo, R.J. Lipton, On the importance of eliminating errors in cryptographic computations. *J. Cryptol.* **14**(2), 101–119 (2001)

[22] V. Boyko, On the security properties of OAEP as an all-or-nothing transform, in M.J. Wiener, editor, *Advances in Cryptology—CRYPTO'99*. Lecture Notes in Computer Science, vol. 1666, Santa Barbara, CA, USA, Aug. 15–19, 1999 (Springer, Heidelberg, 1999), pp. 503–518

[23] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, A. Sahai, Exposure-resilient functions and all-or-nothing transforms, in B. Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*. Lecture Notes in Computer Science, vol. 1807, Bruges, Belgium, May 14–18, 2000 (Springer, Heidelberg, 2000), pp. 453–469

[24] N. Chandran, V. Goyal, P. Mukherjee, O. Pandey, J. Upadhyay, Block-wise non-malleable codes, in I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, D. Sangiorgi, editors, *ICALP 2016: 43rd International Colloquium on Automata, Languages and Programming*. LIPIcs, vol. 55, Rome, Italy, July 11–15, 2016 (Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016), pp. 31:1–31:14

[25] N. Chandran, B. Kanukurthi, S. Raghuraman, Information-theoretic local non-malleable codes and their applications, in E. Kushilevitz, T. Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*. Lecture Notes in Computer Science, vol. 9563, Tel Aviv, Israel, Jan. 10–13, 2016 (Springer, Heidelberg, 2016), pp. 367–392.

[26] E. Chattopadhyay, D. Zuckerman, Non-malleable codes against constant split-state tampering, in *55th Annual Symposium on Foundations of Computer Science*, Philadelphia, PA, USA, Oct. 18–21, 2014 (IEEE Computer Society Press, 2014), pp. 306–315

[27] M. Cheraghchi, V. Guruswami, Capacity of non-malleable codes, in M. Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, Princeton, NJ, USA, Jan. 12–14, 2014 (Association for Computing Machinery, 2014), pp. 155–168

[28] S.G. Choi, A. Kiayias, T. Malkin, BiTR: built-in tamper resilience, in D.H. Lee, X. Wang, editors, *Advances in Cryptology—ASIACRYPT 2011*. Lecture Notes in Computer Science, vol. 7073, Seoul, South Korea, Dec. 4–8, 2011 (Springer, Heidelberg, 2011), pp. 740–758

[29] S. Coretti, U. Maurer, B. Tackmann, D. Venturi, From single-bit to multi-bit public-key encryption via non-malleable codes, in Y. Dodis, J.B. Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*. Lecture Notes in Computer Science, vol. 9014, Warsaw, Poland, Mar. 23–25, 2015 (Springer, Heidelberg, 2015), pp. 532–560.

[30] R. Cramer, Y. Dodis, S. Fehr, C. Padró, D. Wichs, Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In N. P. Smart, editor, *Advances in Cryptology—EUROCRYPT 2008*. Lecture Notes in Computer Science, vol. 4965, Istanbul, Turkey, Apr. 13–17, 2008 (Springer, Heidelberg, 2008), pp. 471–488

[31] D. Dachman-Soled, Y.T. Kalai, Securing circuits against constant-rate tampering, in *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology—CRYPTO 2012* vol. 7417 (2012), pp. 533–551

[32] D. Dachman-Soled, Y.T. Kalai, Securing circuits and protocols against 1/poly(k) tampering rate, in Y. Lindell, editor, *Theory of Cryptography: 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24–26, 2014. Proceedings* (2014)

[33] D. Dachman-Soled, I. Komargodski, R. Pass, Non-malleable codes for bounded parallel-time tampering, in T. Malkin, C. Peikert, editors, *Advances in Cryptology—CRYPTO 2021, Part III*. Lecture Notes in Computer Science, vol. 12827, Virtual Event, Aug. 16–20, 2021 (Springer, Heidelberg, 2021), pp. 535–565

[34] D. Dachman-Soled, M. Kulkarni, A. Shahverdi, Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes, in S. Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*. Lecture Notes in Computer Science, vol. 10174, Amsterdam, The Netherlands, Mar. 28–31, 2017 (Springer, Heidelberg, 2017), pp. 310–332

[35] D. Dachman-Soled, M. Kulkarni, A. Shahverdi, Local non-malleable codes in the bounded retrieval model, in M. Abdalla, R. Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*. Lecture Notes in Computer Science, vol. 10770, Rio de Janeiro, Brazil, Mar. 25–29, 2018 (Springer, Heidelberg, 2018), pp. 281–311

[36] D. Dachman-Soled, F.-H. Liu, E. Shi, H.-S. Zhou, Locally decodable and updatable non-malleable codes and their applications, in In Y. Dodis, J.B. Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, Lecture Notes in Computer Science, vol. 9014, Warsaw, Poland, Mar. 23–25, 2015 (Springer, Heidelberg, 2015), pp. 427–450

[37] N. Döttling, J.B. Nielsen, M. Obremski, Information theoretic continuously non-malleable codes in the constant split-state model. Cryptology ePrint Archive, Report 2017/357, 2017. https://eprint.iacr.org/2017/357

[38] S. Dziembowski, T. Kazana, M. Obremski, Non-malleable codes from two-source extractors, in R. Canetti, J.A. Garay, editors, *Advances in Cryptology—CRYPTO 2013, Part II*. Lecture Notes in Computer Science, vol. 8043, Santa Barbara, CA, USA, Aug. 18–22, 2013 (Springer, Heidelberg, 2013), pp. 239–257

[39] S. Dziembowski, K. Pietrzak, D. Wichs, Non-malleable codes, in A.C.C. Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, Tsinghua University, Beijing, China, Jan. 5–7, 2010 (Tsinghua University Press, 2010), pp. 434–452

[40] N. Ephraim, C. Freitag, I. Komargodski, R. Pass, Non-malleable time-lock puzzles and applications. Cryptology ePrint Archive, Report 2020/779, 2020. https://eprint.iacr.org/2020/779

[41] A. Faonio, J.B. Nielsen, Non-malleable codes with split-state refresh, in S. Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, Lecture Notes in Computer Science, vol. 10174, Amsterdam, The Netherlands, Mar. 28–31, 2017 (Springer, Heidelberg, 2017), pp. 279–309

[42] S. Faust, K. Hostáková, P. Mukherjee, D. Venturi, Non-malleable codes for space-bounded tampering, J. Katz, H. Shacham, editors, *Advances in Cryptology—CRYPTO 2017, Part II*. Lecture Notes in Computer Science, vol. 10402, Santa Barbara, CA, USA, Aug. 20–24, 2017 (Springer, Heidelberg, 2017), pp. 95–126

[43] S. Faust, P. Mukherjee, J.B. Nielsen, D. Venturi, Continuous non-malleable codes, in Y. Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*. Lecture Notes in Computer Science, vol. 8349, San Diego, CA, USA, Feb. 24–26, 2014 (Springer, Heidelberg, 2014), pp. 465–488

[44] S. Faust, P. Mukherjee, J.B. Nielsen, D. Venturi, A tamper and leakage resilient von neumann architecture, in J. Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*. Lecture Notes in Computer Science, vol. 9020, Gaithersburg, MD, USA, Mar. 30 – Apr. 1, 2015 (Springer, Heidelberg, 2015), pp. 579–603

[45] S. Faust, P. Mukherjee, D. Venturi, D. Wichs, Efficient non-malleable codes and key-derivation for poly-size tampering circuits, in P.Q. Nguyen, E. Oswald, editors, *Advances in Cryptology—EUROCRYPT 2014*. Lecture Notes in Computer Science, vol. 8441, Copenhagen, Denmark, May 11–15, 2014 (Springer, Heidelberg, 2014), pp. 111–128

[46] S. Faust, K. Pietrzak, D. Venturi, Tamper-proof circuits: How to trade leakage for tamper-resilience, in *Automata, Languages and Programming: 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I* (2011), pp. 391–402

[47] D. Genkin, Y. Ishai, M. Prabhakaran, A. Sahai, E. Tromer, Circuits resilient to additive attacks with applications to secure computation, in D.B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, New York, NY, USA, May 31–June 3, 2014 (ACM Press, 2014), pp. 495–504

[48] Y. Ishai, M. Prabhakaran, A. Sahai, D. Wagner, Private circuits ii: keeping secrets in tamperable circuits, in *Advances in Cryptology—EUROCRYPT 2006, St. Petersburg, Russia, May 28 - June 1, 2006. Proceedings* (Springer, Berlin, 2006), pp. 495–504

[49] Z. Jafargholi, D. Wichs, Tamper detection and continuous non-malleable codes, in Y. Dodis, J.B. Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*. Lecture Notes in Computer Science, vol. 9014, Warsaw, Poland, Mar. 23–25, 2015 (Springer, Heidelberg, 2015), pp. 451–480

[50] J. Katz, Y. Lindell, *Introduction to Modern Cryptography*, 2nd edition (Chapman & Hall/CRC, 2014).

[51] A. Kiayias, F.-H. Liu, Y. Tselekounis, Practical non-malleable codes from l-more extractable hash functions, in E.R. Weippl, S. Katzenbeisser, C. Kruegel, A.C. Myers, S. Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, Vienna, Austria, Oct. 24–28, 2016 (ACM Press, 2016), pp. 1317–1328

[52] A. Kiayias, F.-H. Liu, Y. Tselekounis, Non-malleable codes for partial functions with manipulation detection, in H. Shacham, A. Boldyreva, editors, *Advances in Cryptology—CRYPTO 2018* (Springer, Cham , 2018), pp. 577–607

[53] A. Kiayias, F.-H. Liu, Y. Tselekounis, Leakage resilient l-more extractable hash and applications to non-malleable cryptography. *Cryptology ePrint Archive* (2022)

[54] A. Kiayias, Y. Tselekounis, Tamper resilient circuits: the adversary at the gates, in K. Sako, P. Sarkar, editors, *Advances in Cryptology—ASIACRYPT 2013* (Springer, Berlin, 2013), pp. 161–180

[55] F.-H. Liu, A. Lysyanskaya, Tamper and leakage resilience in the split-state model, in R. Safavi-Naini, R. Canetti, editors, *Advances in Cryptology—CRYPTO 2012*. Lecture Notes in Computer Science, vol. 7417, Santa Barbara, CA, USA, Aug. 19–23, 2012 (Springer, Heidelberg, 2012), pp. 517–532

[56] S. Micali, L. Reyzin, Physically observable cryptography (extended abstract), in M. Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*. Lecture Notes in Computer Science, vol. 2951, Cambridge, MA, USA, Feb. 19–21, 2004 (Springer, Heidelberg, 2004), pp. 278–296

[57] L.H. Ozarow, A.D. Wyner, Wire-tap channel ii. *AT T Bell Lab. Tech. J.*

[58] T. Rabin, M. Ben-Or, Verifiable secret sharing and multiparty protocols with honest majority (extended abstract), in *21st Annual ACM Symposium on Theory of Computing*, Seattle, WA, USA, May 15–17, 1989 (ACM Press, 1989), pp. 73–85

[59] J.K. Resch, J.S. Plank, AONT-RS: blending security and performance in dispersed storage systems, in *FAST'11* (2011)

[60] R.L. Rivest, All-or-nothing encryption and the package transform, in E. Biham, editor, *Fast Software Encryption—FSE'97*. Lecture Notes in Computer Science, vol. 1267, Haifa, Israel, Jan. 20–22, 1997 (Springer, Heidelberg, 1997), pp. 210–218

[61] R. Shaltiel, J. Silbak, Explicit list-decodable codes with optimal rate for computationally bounded channels, in *APPROX/RANDOM 2016* (2016)

[62] D.R. Stinson, Something about all or nothing (transforms). *Designs Codes Cryptogr.* **22**(2), 133–138 (2001)

[63] M. Tunstall, D. Mukhopadhyay, S. Ali, *Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault* (Springer, Berlin, 2011), pp. 224–233

[64] A.D. Wyner, The wire-tap channel. *Bell Syst. Tech. J.* (1975)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.