Linear-Time Non-Malleable Codes in the Bit-Wise Independent Tampering Model

Ronald Cramer[1], Ivan Damgård[2], Nico Döttling[3], Irene Giacomelli[4], and Chaoping Xing[5]

[1] CWI Amsterdam & Leiden University, The Netherlands
[2] Aarhus University, Denmark
[3] Friedrich-Alexander-University Erlangen-Nürnberg, Germany
[4] University of Wisconsin-Madison, USA
[5] Nanyang Technological University, Singapore

Abstract. Non-malleable codes were introduced by Dziembowski et al. (ICS 2010) as coding schemes that protect a message against tampering attacks. Roughly speaking, a code is non-malleable if decoding an adversarially tampered encoding of a message $\boldsymbol{m}$ produces the original message $\boldsymbol{m}$ or a value $\boldsymbol{m}'$ (possibly $\perp$) completely unrelated to $\boldsymbol{m}$. It is known that non-malleability is possible only for restricted classes of tampering functions. Since their introduction, a long line of works has established feasibility results of non-malleable codes against different families of tampering functions. However, for many interesting families the challenge of finding "good" non-malleable codes remains open. In particular, we would like to have *explicit constructions* of non-malleable codes with *high-rate* and efficient encoding/decoding algorithms (*i.e.* low computational complexity). In this work we present two explicit constructions: the first one is a natural generalization of the work of Dziembowski et al. and gives rise to the first constant-rate non-malleable code with *linear-time* complexity (in a model including bit-wise independent tampering). The second construction is inspired by the recent works about non-malleable codes of Agrawal et al. (TCC 2015) and of Cheraghchi and Guruswami (TCC 2014) and improves our previous result in the bit-wise independent tampering model: it builds the first non-malleable codes with *linear-time* complexity and *optimal-rate* (*i.e.* rate $1 - o(1)$).

Keywords: non-malleable codes, linear-time, bit-wise independent tampering, secret-sharing.

1 Introduction

Non-malleable codes [32] are a relaxation of error-correcting and error-detecting codes that have useful applications in cryptography. For example, they can be used to protect keys that are stored in non-robust devices against tampering attacks. Recently, they also found application to computational cryptography (*e.g.* construction of non-malleable commitments [7,38] and domain extension for public-key encryption schemes [23,22]). Roughly speaking, a coding scheme

$(\mathsf{Enc}, \mathsf{Dec})$ is non-malleable with respect to the tampering function f if decoding $f(\mathsf{Enc}(\boldsymbol{m}))$ produces the original message $\boldsymbol{m}$ or a value $\boldsymbol{m}'$ (possibly $\perp$) completely unrelated to $\boldsymbol{m}$. Moreover, the probability of which one of these two events happens is also independent of $\boldsymbol{m}$. As an illustration of the notion, consider a key that is stored in a device. The adversary is able to tamper with the key and gets to see the effect of using the device with the tampered key inside. If the key was coded with a non-malleable code and is decoded before use, this attack becomes useless, as the key actually used after tampering is either unchanged or is unrelated to the original key.

Since a tampering function can always try to decode, modify the message, and encode again, it is clear that non-malleable codes are impossible without restrictions on the tampering function. We therefore restrict the adversary to using functions from a specific class $\mathcal{F}$. In this case, we say that we have a non-malleable code with respect to the family $\mathcal{F}$. For example, if the encoding is made by n symbols from a finite field $\mathbb{F}$, then we can restrict the tampering function to be a function with n independent components $(f_1, \ldots, f_n)$ (symbol-wise independent tampering, or bit-wise independent tampering if $\mathbb{F} = \{0, 1\}$). Other important features of the coding scheme are the rate and the computational complexity[6]. Since 2010, a line of works has established increasingly stronger results concerning the feasibility of non-malleable codes against different families of tampering functions. However, for many interesting families the challenge of finding "good" non-malleable codes remains open. In particular, we would like to have *explicit* constructions of non-malleable codes with *high rate* and efficient encoding/decoding algorithm (*i.e. low computational complexity*).

This paper follows this research direction studying the following natural question: can we achieve the optimal properties of linear-time complexity and rate approaching 1 simultaneously (via an explicit constriction)? This is not known, even for the restricted case of bit-wise independent tampering, and even if we only ask for linear-time complexity[7].

Many of the known constructions of non-malleable codes (see for example [32], [16,18], [8],[7]) use *linear secret-sharing schemes* (LSSS) as one of the main building blocks. This holds also for the constructions presented in this paper. Roughly speaking, a secret-sharing scheme is a randomised algorithm that encodes a message $\boldsymbol{m}$ as a longer vector $\boldsymbol{s}$ such that $\boldsymbol{m}$ can be computed from large enough sets of entries in $\boldsymbol{s}$, while smaller set give no information about $\boldsymbol{m}$. LSSS with extra properties (uniformity and distance) are used already by Dziembowski et al. in [32] where they introduce and motivate the formal notion of non-malleable codes and also construct the first family of non-malleable codes in the bit-wise independent tampering model. The computational complexity of

[6] The rate of the coding scheme $(\mathsf{Enc}, \mathsf{Dec})$ is the quotient of the length of the message $\boldsymbol{m}$ over the length of its encoding $\mathsf{Enc}(\boldsymbol{m})$. The computational complexity of the scheme is maximum of the computational complexities of the two algorithm Enc and Dec in function of the length of $\boldsymbol{m}$.

[7] Determining which cryptographic primitives can be instantiated in linear-time is an interesting and challenging program started by Ishai et al. in [39].

the code is quadratic in the size of the input length. Secondly, via the probabilistic method they show that for any family $\mathcal{F}$ of tampering functions such that $|\mathcal{F}| \leq 2^{2^{\alpha n}}$ for some constant $\alpha < 1$ (n is the length of the encoding) there exist constant-rate non-malleable codes with respect to $\mathcal{F}$. In this case, the description of the code is of exponential size, thus the encoding and decoding algorithms are inefficient. More recently, Cheraghchi and Guruswami [15,17] prove that for this kind of families the optimal rate is $1 - \alpha$; they construct non-malleable codes approaching this rate. Again, the construction is non-explicit and gives rise to inefficient codes. For families of single exponential size, *i.e.* $|\mathcal{F}| \leq 2^{p(n)}$ for some polynomial p, efficient (*i.e.* polynomial time) non-malleable codes were constructed in [35]. This construction is also randomized, *i.e.* the construction succeeds with overwhelming probability in providing non-malleable codes achieving optimal rate $1 - o(1)$. On the other hand, in [16] an explicit (deterministic) construction of non-malleable codes with rate arbitrarily close to 1 in the bit-wise independent tampering model is given. The construction is based on the concatenation of a linear error-correcting secret-sharing scheme of rate close to 1 and a constant-size non-malleable code. This construction is instantiated using Reed-Solomon codes and has thus computational complexity at least $O(n\,\mathrm{polylog}(n))$ (super-linear).

In [40], Jafargholi and Wichs introduce *tamper-detection codes* (TD) and use them together with leakage-resilient codes [29] to construct non-malleable codes that achieve optimal rate when $|\mathcal{F}| \leq 2^{2^{\alpha n}}$ and efficient encoding and decoding when $|\mathcal{F}| \leq 2^{p(n)}$. TD codes for the simple family of additive tampering functions are called *algebraic manipulation detection codes* (AMD) and were already introduced by Cramer et al. in 2008 [26].

Our Contribution. In this paper, we study the above question and achieve positive results. In the first part of our work, we push forward the idea of using linear secret sharing, and show that when the family of tampering functions has a clear structure (as in the symbol-wise independent tampering model), then simple constructions based on LSSS can achieve good results: we get constant-rate non-malleable codes with optimal computational complexity $O(k)$, where k is the length of the input message. To obtain this, we also use known results about linear-time encodable error-correcting codes and linear-time computable universal hash functions [39,30].

Building on the first result, we then achieve both linear-time complexity and optimal rate, that is rate $1 - o(1)$, for non-malleable codes in the bit-wise independent tampering model. It is instructive to observe that optimal-rate non-malleable codes with superlinear time complexity were constructed in [16,8], and that these codes are based on secret sharing schemes with (relatively) large privacy and reconstruction thresholds. The problem we face is that there are no constructions of linear secret sharing schemes with linear-time complexity for the required parameter range[8]. We therefore propose a novel construction which is based on slightly weaker primitives which can be instantiated for the rate $1 - o(1)$ and linear-time complexity regime.

Overview of our Constructions. As mentioned, we present two deterministic constructions for linear-time non-malleable codes: Construction 1 (Section 3) can be seen as a generalization of the original construction of [32] and gives rise to the first linear-time non-malleable codes with constant rate in the symbol-wise independent tampering model. More generally, we prove that given a family of TD codes with any computational complexity and rate, it is possible to explicitly construct a family of non-malleable codes with constant rate and linear-time complexity. The other ingredients of this first construction are constant-rate AMD codes and constant-rate LSSS with good privacy (but where one needs almost all shares to reconstruct). We present linear-time instantiations of both these primitives using the results of [30]. Construction 1 encodes a message $\boldsymbol{m}$ with three sequential steps: first $\boldsymbol{m}$ is encoded with an AMD code, then the result is shared by a LSSS with privacy and finally each share is encoded by a tamper-detection code (see Figure 1).

$$\begin{tikzcd}
\mathbb{F}^k \arrow[r, "\text{AMD}"] & \mathbb{F}^{\Theta(k)} \arrow[r, "\text{LSSS}"] & (\mathbb{F}^{\ell})^m \arrow[r, "\text{componet-wise TD}"] & (\mathbb{F}^{\ell'})^m \\
\boldsymbol{m} \arrow[r, mapsto] & \boldsymbol{m}' \arrow[r, mapsto] & \boldsymbol{s} \arrow[d, phantom, "\parallel"] & \\
 & & (\boldsymbol{s}_1, \dots, \boldsymbol{s}_m) \arrow[r, mapsto] & (\boldsymbol{c}_1, \dots, \boldsymbol{c}_m)
\end{tikzcd}$$

Fig. 1. The encoding algorithm of Construction 1 ($m = \Theta(k)$ and ℓ constant).

In particular, in Construction 1 if the tamper-detection code is secure against the family of tampering functions $\mathcal{F}$ with constant error, then the resulting code is non-malleable with respect to the family $\mathcal{F}^+$ of functions of the form $(f_1, \dots, f_m)$ where each f_i is a function from $\mathcal{F}$, a constant function or the identity and it has error negligible in the length of the input. Hence, depending on how one instantiates the components of the construction, one can handle more general tampering models than bit-wise[9]. A key point for the efficiency is that the shares produced by the LSSS used are of constant size. This implies that applying the tamper-detection code to all the shares results only in a constant overhead for the computational complexity.

With Construction 2 (Section 4), we achieve linear-time non-malleable codes with optimal rate approaching 1, still with an explicit (deterministic) construc-

[8] A Monte-Carlo construction by Cramer et al. [24] can be instantiated for a parameter range where the rate of the secret sharing scheme is bounded away from 1 by a constant, but not for rate approaching 1.

[9] The concrete instantiation we give in Corollary 3 leads to bit-wise independent tampering.

tion. The most efficient constructions of optimal rate non-malleable codes in the bit-wise independent tampering model are from [16,8]. Both these constructions require a secret sharing scheme with good privacy and non-trivial reconstruction threshold. Together with the rate close to 1 constraint, these are challenging features to achieve in linear-time. In our construction, we also use a secret-sharing scheme with rate close to 1, but we do not require any reconstruction property for this scheme. Instead, we combine the sharing scheme with two other tailored primitives, each implementable in linear-time, and a short constant-rate non-malleable code. The modular design of our construction makes the security proof much simpler and more intuitive than previous constructions: each primitive takes care of a specific property needed to prove non-malleability. The encoding is done in the following way: first the input message is shared with a sharing scheme that has rate $1 - o(1)$ and t-uniformity (that is, if $\boldsymbol{s}$ is the share vector of $\boldsymbol{m}$, then each set of t components of $\boldsymbol{s}$ are distributed uniformly on $\mathbb{F}^t$). Then we use the two tailored primitives: first, a keyed almost universal function is used to compute the first hash of $\boldsymbol{s}$, $h_{\boldsymbol{k}}(\boldsymbol{s})$. Second, we compute short deterministic hash $\mathsf{Comp}(\boldsymbol{s})$, using a new primitive that we call a *compressor*. This compressed value $\mathsf{Comp}(\boldsymbol{s})$ comes with the guaranty of having high entropy. The two hash values and the key for the almost universal hash function can be thought of as an "authentication tag" of $\boldsymbol{m}$. The final encoding is given by the share vector $\boldsymbol{s}$ and a non-malleable encoding of this tag, this encoding does not have to be high-rate nor linear-time (see Figure 2).

$$\begin{tikzcd}
\mathbb{F}^k \arrow[r, "\text{sharing}"] & \mathbb{F}^{k+o(k)} \arrow[r, "\text{hashing}"] & \mathbb{F}^{k+o(k)} \times \mathbb{F}^{o(k)} \times \mathbb{F}^{o(k)} \arrow[r, "\text{short NM}"] & \mathbb{F}^{k+o(k)} \times \mathbb{F}^{o(k)} \\
\boldsymbol{m} \arrow[r, mapsto] & \boldsymbol{s} \arrow[r, mapsto] & (\boldsymbol{s}, h_{\boldsymbol{k}}(\boldsymbol{s}), \mathsf{Comp}(\boldsymbol{s})) \arrow[d, equal] & \\
& & (\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{c}) \arrow[r, mapsto] & (\boldsymbol{s}, \mathrm{NM}(\boldsymbol{k}, \boldsymbol{h}, \boldsymbol{c}))
\end{tikzcd}$$

Fig. 2. The encoding algorithm of Construction 2.

More related work. Bit-wise independent tampering functions act on each bit of the encoding independently. In the more general, C-split state model the encoding is partitioned into C blocks (C is a constant) and each block can be tampered arbitrarily but independently of the others blocks (*e.g.* [19]). For $C = 10$, an efficient and explicit construction of constant rate non-malleable codes was given in [14]. Several results can be found in the literature when $C = 2$ (split-state model) [43,31,34,16,4,3,5,42]. In [43] the non-malleability property is guaranteed only against computationally bounded adversaries, while the scheme proposed by [31] is secure in the information-theoretic setting, but it can encode only 1-bit messages. The first explicit construction of non-malleable codes with information-theoretic security and message space larger than $\{0, 1\}$ in the split-state model was proposed in [4] and have rate polynomially small (k-bit strings are encoded into codewords of length $\approx k^7$). This result was recently improved in [2], where the codeword length is decreased to $O(k^5)$. In 2015, Aggarwal et al. [3] constructed the first explicit non-malleable codes in the split state model achieving constant-rate. Rate approaching to 1 is achieved in [1,41] in the computational setting.

In [8,7], Agrawal et al. construct explicit and non-malleable codes which are simultaneously resilient against bit-wise independent tampering and permutations. [7] gets optimal rate, but has super-linear computational complexity. In [9] constant-rate and explicit non-malleable codes with respect to the family of functions $f : \{0,1\}^n \to \{0,1\}^n$ such that any output bit depends only on n^δ input bits ($0 \leq \delta < 1$ constant). Finally, notice that many variants of non-malleable codes have been introduced in the literature: *e.g.* continuous non-malleable codes [34,40,6,13], leakage-resilient non-malleable codes [43,5,33], block-wise non-malleable codes [11,37] and local non-malleable codes [28,12,27].

Structure of the paper: In Section 2, we fix the notation and give the basic definitions we need further on in the paper. In Section 3 first we give linear-time construction for AMD codes and LSSS with privacy, then we present Construction 1 in general and finally, we instantiate it for the binary case (bit-wise independent tampering model). Section 4 is also divided in two parts: in the first one we define and instantiate the primitives that are necessary for Construction 2; the latter is described in the second part of the section together with its instantiation in the bit-wise independent tampering model.

2 Preliminaries

For an integer n, we write $[n] = \{1, 2, \dots, n\}$ and, given $A \subseteq [n]$, $|A|$ denotes the cardinality of A, while A^c indicates the complement set of A, i.e. $A^c = [n] \setminus A$. With the notation $(\boldsymbol{z}_1, \dots, \boldsymbol{z}_n)$ we indicate an element of the n-times cartesian product of $\mathbb{F}^\ell$, where $\mathbb{F}$ is a finite field of cardinality q and ℓ is a positive integer. Given $\boldsymbol{z} = (\boldsymbol{z}_1, \dots, \boldsymbol{z}_n) \in (\mathbb{F}^\ell)^n$ and a subset $A \subseteq [n]$, we will use $\boldsymbol{z}_A$ to denote the vector $(\boldsymbol{z}_i)_{i \in A} \in (\mathbb{F}^\ell)^{|A|}$. Given two vectors $\boldsymbol{z} = (\boldsymbol{z}_1, \dots, \boldsymbol{z}_n), \boldsymbol{v} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_n) \in (\mathbb{F}^\ell)^n$, the *generalized Hamming Distance* between $\boldsymbol{z}$ and $\boldsymbol{v}$ is defined by $\mathrm{d}^\ell_{\mathrm{Ham}}(\boldsymbol{z}, \boldsymbol{v}) = |\{i \in [n] \mid \boldsymbol{z}_i \neq \boldsymbol{v}_i\}|$. If Alg is an algorithm (randomized or not) that takes as input a value from $\mathbb{F}^n$, then the computational complexity of Alg is the number of field elementary operations that Alg executes to compute the output. We indicate with id the identity function. We say that a function ε is *negligible* in n ($\varepsilon(n) = negl(n)$) if for every polynomial p there exists a constant c such that $\varepsilon(n) < \frac{1}{p(n)}$ when $n > c$. For a random variable X, the notation $v \leftarrow X$ denotes that v is sampled randomly according to X. For a set S, $v \leftarrow S$ denotes that v is sampled uniformly at random from S. Given two random variables X and Y with finite range S, the *statistical distance* between X and Y is defined as $\mathsf{SD}(X, Y) = \frac{1}{2} \sum_{i \in S} |\Pr[X = i] - \Pr[Y = i]|$. Let $X = (X_1, \dots, X_n)$ be a random variable with range S^n and t be a positive integer less or equal to n. We say that X is *t-wise independent* if for any $A = \{i_1, \dots, i_t\}$ subset of $[n]$ of cardinality t and for any vector $\boldsymbol{b} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_t) \in S^t$, it holds that $\Pr[X_A = \boldsymbol{b}] = \prod_{j=1}^{t} \Pr[X_{i_j} = \boldsymbol{b}_j]$. We say that X is *t-wise uniform* on S^n if for any $A \subseteq [n]$ of cardinality t, X_A has the uniform distribution on S^t. If $t = n$ we simply say that X is an uniform random variable on S^n.

2.1 Tamper-Detection and Non-Malleability

Let $\mathbb{F}$ be a finite field and n, ℓ, k be positive integers. An ℓ-folded *n-code* over $\mathbb{F}$ is a non-empty subset $\mathcal{C}$ of $(\mathbb{F}^\ell)^n$; we will refer to n as the length of the code. Given a set $A \subseteq [n]$, with the notation $\mathcal{C}_A$ we indicate the set $\{\boldsymbol{c}_A \mid \boldsymbol{c} \in \mathcal{C}\}$. If $\psi : \mathcal{C} \to \mathbb{F}^k$ is a regular function, the pair $(\mathcal{C}, \psi)$ is called ℓ-folded (n,k)-*coding scheme*. The rate of a scheme is the ratio $k/\ell n$. If $\mathbb{F} = \{0, 1\}$, the scheme is called binary. When $\ell = 1$, we simply call it (n,k)-coding scheme. If $\mathcal{C}$ is a vector space over $\mathbb{F}$, then the code is called *linear*. The dimension of a linear code is its dimension as vector space over $\mathbb{F}$. Moreover, if the map ψ is an $\mathbb{F}$-linear map, also the scheme $(\mathcal{C}, \psi)$ is called linear.

Remark 1. Given an ℓ-folded (n,k)-coding scheme $(\mathcal{C}, \psi)$, any randomized algorithm $\mathsf{Enc} : \mathbb{F}^k \to \mathcal{C}$ that on input $\boldsymbol{m} \in \mathbb{F}^k$ outputs $\boldsymbol{c} \in \psi^{-1}(\{\boldsymbol{m}\})$ selected uniformly at random is called *encoding algorithm.* On the other side, *decoding algorithm* is the name used for the deterministic algorithm $\mathsf{Dec} : (\mathbb{F}^\ell)^n \to \mathbb{F}^k \cup \{\perp\}$ that maps $\boldsymbol{c}$ to $\boldsymbol{m} = \psi(\boldsymbol{c}) \in \mathbb{F}^k$ if $\boldsymbol{c} \in \mathcal{C}$ and to $\perp$ otherwise. For convenience[10], in the following we will always identify a coding scheme $(\mathcal{C}, \psi)$ with the pair $(\mathsf{Enc}, \mathsf{Dec})$.

While keeping $\mathbb{F}$ fixed, we will assume throughout that $n = n(k)$. The computational complexity (as a function of k) of a coding scheme is the maximum taken over the computational complexities of Enc and Dec, respectively. We say that a coding scheme is *linear-time* if both Enc and Dec have complexity $O(k)$.

Let $(\mathsf{Enc}, \mathsf{Dec})$ be an ℓ-folded (n,k)-coding scheme over $\mathbb{F}$. Given an encoding $\boldsymbol{c} \leftarrow \mathsf{Enc}(\boldsymbol{m})$ for the message $\boldsymbol{m} \in \mathbb{F}^k$, tampering with $\boldsymbol{c}$ can be represented by considering a function $f : (\mathbb{F}^\ell)^n \to (\mathbb{F}^\ell)^n$ that modifies the encoding $\boldsymbol{c}$ in $\tilde{\boldsymbol{c}} = f(\boldsymbol{c})$. The output of $\mathsf{Dec}(\tilde{\boldsymbol{c}})$ now depends on the original message $\boldsymbol{m}$ and also on the tampering function f. To represent this, we consider the following random variable $\mathrm{Real}_f^{\boldsymbol{m}}$.

$$\mathrm{Real}_f^{\boldsymbol{m}} = \begin{cases} \text{sample } \boldsymbol{c} \leftarrow \mathsf{Enc}(\boldsymbol{m}); \\ \text{compute } \tilde{\boldsymbol{c}} = f(\boldsymbol{c}); \\ \text{output } \tilde{\boldsymbol{m}} = \mathsf{Dec}(\tilde{\boldsymbol{c}}); \end{cases}$$

A simple but strong property that we can ask for is that the coding scheme is able to detect with overwhelming probability the tampering caused by all the functions f from a specific family $\mathcal{F}$.

[10] The two definitions are equivalent. Given the pair $(\mathsf{Enc}, \mathsf{Dec})$ such that for any $\boldsymbol{m}$ it holds $\Pr[\mathsf{Dec}(\mathsf{Enc}(\boldsymbol{m})) = \boldsymbol{m}] = 1$, define $\mathcal{C}$ as the image of Enc in $(\mathbb{F}^\ell)^n$ and ψ as the map Dec restricted to $\mathcal{C}$.

Definition 1 (TD Code, [40]). *Given a family $\mathcal{F}$ of functions over $(\mathbb{F}^\ell)^n$, an ℓ-folded (n,k)-*tamper detection code *with respect to $\mathcal{F}$ and with error ϵ is an (n,k)-coding scheme such that* $\Pr[\mathrm{Real}_f^{\boldsymbol{m}} \neq \perp] \leq \epsilon$ *for any $\boldsymbol{m} \in \mathbb{F}^k$ and any $f \in \mathcal{F}$.*

For example, any error-correcting code from coding theory with minimal distance d is a TD code with respect to the family $\mathcal{F}_{dist}$ of functions that modify less than d components in the input vector (*i.e.* $\mathrm{d}^\ell_{\mathrm{Ham}}(f(\boldsymbol{x}), \boldsymbol{x}) < d$). The name *algebraic manipulation detection (**AMD**) code*, introduced by [26], is used for TD codes with respect to the family $\mathcal{F}_{amd}$ of additive tampering functions. That is, functions of the form $f_{\boldsymbol{e}}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{e}$ where the vector $\boldsymbol{e}$ is a non-zero constant vector independent of $\boldsymbol{x}$.

Unfortunately, tampering detection can not be achieved for many natural families. For example, consider the family $\mathcal{F}_{const}$ of all constant functions $f_{\boldsymbol{c}}(\boldsymbol{x}) = \boldsymbol{c}$ for $\boldsymbol{c} \in (\mathbb{F}^\ell)^n$; if $\boldsymbol{c}$ is a valid encoding, then $\Pr[\mathrm{Real}_{f_{\boldsymbol{c}}}^{\boldsymbol{m}} \neq \perp] = 1$ for all $\boldsymbol{m} \in \mathbb{F}^k$. In order to be able to consider larger families of tampering functions, the definition of tampering detection needs to be relaxed. Instead of asking that the tampering is detected, we can ask that the result of the tampering action is independent of the original message. This property, called non-malleability is weaker than tampering-detection, nevertheless it offers enough protection against tampering attacks: an adversary that actively modifies encoded data can not control the practical effect of his action on the encoded message.

Definition 2 (NM Code, [32]). *An ℓ-folded (n,k)-coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is said to be* non-malleable *with respect to a family $\mathcal{F}$ with error ϵ if the following holds for any $f \in \mathcal{F}$. There exists a random variable D_f on $\mathbb{F}^k \cup \{\perp, same\}$ such that, given*

$$\mathrm{Ideal}_f^{\boldsymbol{m}} = \begin{cases} \text{sample } \boldsymbol{m}^* \leftarrow D_f; & \\ \text{if } \boldsymbol{m}^* = \mathit{same} & \text{then } \boldsymbol{m}' = \boldsymbol{m}; \\ & \text{otherwise } \boldsymbol{m}' = \boldsymbol{m}^*; \\ \text{output } \boldsymbol{m}'; & \end{cases}$$

then $\mathsf{SD}(\mathrm{Real}_f^{\boldsymbol{m}}, \mathrm{Ideal}_f^{\boldsymbol{m}}) \leq \epsilon$ *for any $\boldsymbol{m} \in \mathbb{F}^k$.*

In the rest of the paper we will mainly consider the family of *symbol-wise independent tampering* functions. That is, if the encoding has the form $\boldsymbol{c} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n) \in (\mathbb{F}^\ell)^n$, then each component $\boldsymbol{c}_i$ can be modified arbitrarily but independently of the values of the others components. We will use the following notation: $\mathcal{F}^q_{\ell,n} = \{f = (f_1, \ldots, f_n) \mid f_i : \mathbb{F}^\ell \to \mathbb{F}^\ell\}$ and $f(\boldsymbol{c}) = (f_1(\boldsymbol{c}_1), \ldots, f_n(\boldsymbol{c}_n))$. Let q be the cardinality of the field $\mathbb{F}$, note that if $q = 2$ and $\ell = 1$, $\mathcal{F}^2_{1,n}$ is the family considered in the *bit-wise independent tampering* model.

2.2 Secret-Sharing

Suppose that $(\mathsf{Enc}, \mathsf{Dec})$ is an ℓ-folded (n,k)-coding scheme over $\mathbb{F}$. Let t, r be positive integers.

Definition 3. $(\mathsf{Enc}, \mathsf{Dec})$ *has* t-privacy *if the following holds for each set* $A \subset [n]$ *of* $\mathbb{F}^\ell$*-coordinates with* $|A| = t$*. For each* $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{F}^k$*, the distributions of* $(\mathsf{Enc}(\boldsymbol{m}))_A$ *and* $(\mathsf{Enc}(\boldsymbol{m}'))_A$ *on* $(\mathbb{F}^\ell)^t$ *are identical. The scheme has* t-uniformity *if these distributions are the uniform ones on* $(\mathbb{F}^\ell)^t$*.* $(\mathsf{Enc}, \mathsf{Dec})$ *has* r-reconstruction *if the following holds for each set* $A \subset [n]$ *of* $\mathbb{F}^\ell$*-coordinates with* $|A| = r$*. If* $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C}$ *satisfy* $\boldsymbol{c}_A = \boldsymbol{c}'_A$*, then* $\mathsf{Dec}(\boldsymbol{c}) = \mathsf{Dec}(\boldsymbol{c}')$*.*

Note that any scheme has n-reconstruction. Moreover, if the coding scheme has r-reconstruction and t-privacy, then $t < r$.

Remark 2. Given an ℓ-folded linear (n, k)-coding scheme, it is easy to prove that t-privacy and t-uniformity are equivalent to the following conditions, respectively.

- (t-privacy) for each set $A \subsetneq [n]$ of $\mathbb{F}^\ell$-coordinates with $|A| = t$, the map that maps $\boldsymbol{c}$ in $\mathcal{C}$ to the pair $(\mathsf{Dec}(\boldsymbol{c}), \boldsymbol{c}_A)$ is surjective;
- (t-uniformity) the same condition as before holds and moreover $\mathcal{C}_A = (\mathbb{F}^\ell)^t$.

Definition 4 (LSSS). *An* ℓ-folded (n, t, r, k)-secret-sharing scheme *over* $\mathbb{F}$ *(with uniformity) is an* ℓ*-folded* (n, k)*-coding scheme over* $\mathbb{F}$ *with* t*-privacy (*t*-uniformity) and* r*-reconstruction. If the coding scheme is linear then we call it* linear *secret-sharing scheme (LSSS).*

Notice that in the existing literature, the algorithms Enc and Dec of a secret-sharing scheme are often indicated with the notation Sh (*sharing algorithm*) and Rec (*reconstruction algorithm*), respectively. Moreover, if $\boldsymbol{c} \leftarrow \mathsf{Sh}(\boldsymbol{m})$, then $\boldsymbol{c}$ is called share vector. Later on in the paper we will use this notation.

In this work, we will use secret-sharing schemes with different parameters and properties as building blocks for constructing efficient NM codes. In particular, for Construction 1 we are interested in the following aspect: what happens if the reconstruction algorithm of a t-private LSSS is applied to a share vector where at most t components have been tampered arbitrarily but independently from the others. The answer is stated in the next lemma.

Lemma 1. *Let* $(\mathsf{Sh}, \mathsf{Rec})$ *be a* t*-private* ℓ*-folded* (n, k)*-LSSS. Fix a set* $A \subsetneq [n]$ *of* $\mathbb{F}^\ell$*-coordinates with* $|A| \leq t$ *and an (eventually randomized) function* $g : (\mathbb{F}^\ell)^n \to (\mathbb{F}^\ell)^n$ *with the following properties. For any* $\boldsymbol{s} \in (\mathbb{F}^\ell)^n$*,* $(g(\boldsymbol{s}))_{A^c} = \boldsymbol{s}_{A^c}$ *and* $(g(\boldsymbol{s}))_A$ *depends only on the entries of* $\boldsymbol{s}_A$*. Then, there exists a random variable* Δ_g *on* $(\mathbb{F}^\ell)^n \cup \{\perp\}$ *such that for any* $\boldsymbol{m} \in \mathbb{F}^k$*,* $\mathsf{Rec}(g(\mathsf{Sh}(\boldsymbol{m})))$ *has the same distribution of* $\boldsymbol{m} + \Delta_g$ *(with the convention that* $\boldsymbol{m} + \perp = \perp$*).*

Proof. Clearly, $g(\mathsf{Sh}(\boldsymbol{m})) = \mathsf{Sh}(\boldsymbol{m}) + [g(\mathsf{Sh}(\boldsymbol{m})) - \mathsf{Sh}(\boldsymbol{m})]$ and it follows from the properties of g that $g(\mathsf{Sh}(\boldsymbol{m})) - \mathsf{Sh}(\boldsymbol{m})$ depends only on the value of $(\mathsf{Sh}(\boldsymbol{m}))_A$. Thus, since $|A| \leq t$, the t-privacy implies that $g(\mathsf{Sh}(\boldsymbol{m})) - \mathsf{Sh}(\boldsymbol{m})$ has the same distribution of $g(\mathsf{Sh}(\mathbf{0})) - \mathsf{Sh}(\mathbf{0})$. If we define $\Delta_g = \mathsf{Rec}(g(\mathsf{Sh}(\mathbf{0})) - \mathsf{Sh}(\mathbf{0}))$, then the lemma follows from the linearity of the map Rec. □

3 Constant-Rate and Linear-Time NM Codes

In this section, we describe our first main result: Construction 1 (Figure 4) combines an AMD code, a LSSS and a TD code with constant error in order to construct a constant-rate NM code (with negligible error) whose computational complexity is controlled by the complexity of the two first schemes used (the AMD code and the LSSS).

3.1 Building Blocks for Construction 1

Before describing Construction 1, we build linear-time and constant-rate AMD codes and LSSSs.

We recall that a coding scheme $(\mathsf{Enc}, \mathsf{Dec})$ (with alphabet $\mathbb{F}$) is an (n, k)-AMD code[11] with error ϵ if $\forall \boldsymbol{m} \in \mathbb{F}^k$ and any non-zero $\boldsymbol{e} \in \mathbb{F}^n$, it holds that $\Pr[\mathsf{Dec}(\mathsf{Enc}(\boldsymbol{m}) + \boldsymbol{e}) \neq \perp] \leq \epsilon$. This special family of TD codes are of particular interest because, despite their simple definition, they can be used as basic tools of generic constructions for coding scheme that achieve security against tampering family larger than $\mathcal{F}_{amd}$ (see for example [32] and our Construction 1). Clearly, the parameters (*i.e.* the rate) and the efficiency of the final schemes depend on the ones of the AMD codes used. In particular, in order to prove our result about constant-rate and linear-time NM codes (Theorem 2), we need to build constant-rate and linear-time AMD codes. Our construction, presented in the following Corollary 1, is based on the family of linear uniform functions from [30].

Lemma 2 (Linear Uniform Family, Theorem 4 in [30]). *For any positive integer c there exists a positive constant b ($b \geq c$) such that for any large enough k there is family of functions $\{g_{\boldsymbol{k}} : \mathbb{F}^k \to \mathbb{F}^{ck}\}_{\boldsymbol{k}}$ with $\boldsymbol{k} \in \mathbb{F}^{bk}$, such that the following holds:*

1. *$g_{\boldsymbol{k}}$ has computational complexity $O(k)$;*
2. *$g_{\boldsymbol{k}}$ is $\mathbb{F}$-linear and $g_{\boldsymbol{k}_1+\boldsymbol{k}_2} = g_{\boldsymbol{k}_1} + g_{\boldsymbol{k}_2}$;*
3. *for any $\boldsymbol{y} \in \mathbb{F}^{ck}$ and $\boldsymbol{x} \in \mathbb{F}^k$ with $\boldsymbol{x} \neq \boldsymbol{0}$, if $\boldsymbol{k}$ is chosen uniformly at random from $\mathbb{F}^{bk}$ then $\Pr[g_{\boldsymbol{k}}(\boldsymbol{x}) = \boldsymbol{y}] = \frac{1}{q^{ck}}$.*

Corollary 1 (Linear-Time and Constant-Rate AMD code). *For any large enough integer k, there exists a linear-time (k', k)-AMD code with error q^{-k} and $k' = \Theta(k)$.*

Proof. (Sketch) Given k, let $\mathcal{G}$ be the family from Lemma 2 with $c = 1$. For the sake of simplicity we assume that $b = 1$ and we define:
$\mathsf{Enc}_{\mathsf{amd}}(\boldsymbol{m}) = (\boldsymbol{m}, \boldsymbol{k}, \boldsymbol{r}, g_{\boldsymbol{k}}(\boldsymbol{m}), g_{\boldsymbol{k}}(\boldsymbol{r}), g_{\boldsymbol{r}}(\boldsymbol{k}))$, where $\boldsymbol{k}, \boldsymbol{r} \in \mathbb{F}^k$ are chosen uniformly at random and

[11] For Construction 1 we need a "strong" AMD code (as in [32]), while AMD codes were introduced in [26] by a slightly different (weaker) notion ($\forall \boldsymbol{m}$ and $\forall \boldsymbol{e}$, $\Pr[\mathsf{Dec}(\mathsf{Enc}(\boldsymbol{m}) + \boldsymbol{e}) \notin \{\perp, \boldsymbol{m}\}] \leq \epsilon$).

$$\mathsf{Dec}_{\mathsf{amd}}(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_6) = \begin{cases} \boldsymbol{v}_1 & \text{if } g_{\boldsymbol{v}_2}(\boldsymbol{v}_1) = \boldsymbol{v}_4, g_{\boldsymbol{v}_2}(\boldsymbol{v}_3) = \boldsymbol{v}_5, g_{\boldsymbol{v}_3}(\boldsymbol{v}_2) = \boldsymbol{v}_6 \\ \perp & \text{otherwise} \end{cases}$$

It is easy to verify that $(\mathsf{Enc}_{\mathsf{amd}}, \mathsf{Dec}_{\mathsf{amd}})$ is a $(6k, k)$-AMD code with error $\frac{1}{q^k}$ and computational complexity $O(k)$. The details of this proof together with its generalization to the case $b > 1$ can be found in [25]. □

For Construction 1, we are interested in linear-time (m, t, m, k)-LSSS with large privacy (*i.e.* $t > m/2$) and constant-rate. Recently [24], the first linear-time constant-rate LSSS was shown, using a construction based on a combination of suitable linear codes and universal hash functions. More concretely, while being linear over a fixed finite field and supporting an unbounded number of players (or shares) m, there are constants $\epsilon_s, \epsilon_t, \epsilon_r$ with $0 < \epsilon_s, \epsilon_t, \epsilon_r < 1$ and an integer ℓ (the share size) such that the length k of the secret satisfies $k \geq e_s \ell m$, the privacy parameter t satisfies $t \geq \epsilon_t m$ and the reconstruction parameter r satisfies $r \leq \epsilon_r m$. Moreover, both the sharing and the reconstruction algorithm have complexity linear in m. Although here we also need constant-rate linear-time sharing scheme, we do not use the result from for Construction 1 and instead we construct our constant-rate linear-time sharing scheme for two reasons. First, the construction in [24] is a Monte-Carlo construction, while in this work we are interested only in explicit (deterministic) constructions. Second, later on (Section 4) we will require constant-rate sharing scheme with t-uniformity (instead of only t-privacy). Our schemes from Corollary 2 have this extra property that is not satisfied by the schemes presented in [24].

We construct the required LSSS using linear codes. Let $\mathcal{D}$ be an ℓ-folded linear m-code of dimension k over the finite field $\mathbb{F}$. The minimum distance of $\mathcal{D}$ is defined as $d = \min\{\mathrm{d}^{\ell}_{\mathrm{Ham}}(\boldsymbol{c}, \boldsymbol{c}') \mid \boldsymbol{c}, \boldsymbol{c}' \in \mathcal{D}, \boldsymbol{c} \neq \boldsymbol{c}'\}$. If $\boldsymbol{G}$ is a $k \times m$ matrix over $\mathbb{F}^{\ell}$, we say that $\boldsymbol{G}$ is a generator matrix for the code $\mathcal{D}$ if $\mathcal{D} = \{\boldsymbol{m} \cdot \boldsymbol{G} \mid \boldsymbol{m} \in \mathbb{F}^k\}$. We say the $\mathcal{D}$ is a *linear-time encodable code* if the map $\boldsymbol{m} \to \boldsymbol{m} \cdot \boldsymbol{G}$ can be computed by an algorithm that has computational complexity $O(k)$.

The following Lemma generalizes and rephrases Theorem 2 in [20] asserting that LSSS with t-uniformity can be obtained from linear codes with distance $t + 1$.

Lemma 3. *Let $\boldsymbol{G}$ be the generator matrix of an ℓ-folded linear code of length m, dimension k and minimum distance d. Assume that $\boldsymbol{G} = (\boldsymbol{I}_k, \boldsymbol{M})$ where $\boldsymbol{I}_k$ is the $k \times k$ identity matrix (systematic form of the code)*[12]*. Then the scheme define in Figure 3 is an ℓ-folded $(m, d-1, m, k)$-LSSS with uniformity. If the code is linear-time encodable, then the LSSS obtained has linear-time complexity.*

Proof. According to Remark 2, showing that the map $\psi_A : \boldsymbol{c} \to (\boldsymbol{c} \cdot \boldsymbol{G}^{\top}, \boldsymbol{c}_A)$ is surjective over $\mathbb{F}^k \times (\mathbb{F}^{\ell})^{d-1}$ for any $A \subseteq [m]$ of size $d-1$ is enough to prove that $(\mathsf{Sh}_1, \mathsf{Rec}_1)$ (see Figure 3) has $d-1$ uniformity. Clearly $\boldsymbol{G}$ (and then $\boldsymbol{G}^{\top}$) has rank k (over $\mathbb{F}$) and the map $\boldsymbol{c} \to \boldsymbol{c} \cdot \boldsymbol{G}^{\top}$ is surjective. Moreover since $\boldsymbol{G}$ generates

[12] With $(\boldsymbol{I}_k, \boldsymbol{M})$ we indicate that we append the columns of $\boldsymbol{M}$ to the ones of the identity matrix $\boldsymbol{I}_k$.

a code of distance d, we can remove any $d-1$ columns of $\boldsymbol{G}$ (*i.e.* $d-1$ rows from $\boldsymbol{G}^\top$) and the punctured matrix still has rank k (as any two distinct codewords differ in at least d coordinates). This means that for any $\boldsymbol{m}$ we can solve in $\boldsymbol{x}$ the linear system $\boldsymbol{x} \cdot \boldsymbol{G}^\top = \boldsymbol{m}$ even when $d-1$ components of $\boldsymbol{x}$ are fixed. This trivially implies that also the map ψ_A is surjective and concludes the proof of the uniformity property. Finally, it follows directly from Tellegen's principle (see Appendix A.3) that if the underlying code is linear-time encodable, then both the algorithms Sh_1 and Rec_1 are linear-time. □

```
Input: $\boldsymbol{m} \in \mathbb{F}^k$

$\mathsf{Sh}_1(\boldsymbol{m})$:

    Sample $\boldsymbol{x}' \leftarrow (\mathbb{F}^\ell)^{m-k}$
    Compute $\boldsymbol{x}'' = \boldsymbol{m} - \boldsymbol{x}' \cdot \boldsymbol{M}^\top$
    Output $\boldsymbol{x} = (\boldsymbol{x}'', \boldsymbol{x}')$
```

```
Input: $\boldsymbol{c} \in (\mathbb{F}^\ell)^m$

$\mathsf{Rec}_1(\boldsymbol{c})$:

    Compute $\boldsymbol{m} = \boldsymbol{c} \cdot \boldsymbol{G}^\top$
    Output $\boldsymbol{m}$
```

Fig. 3. Linear-time and constant-rate LSSS

Instantiating Lemma 3 with ad-hoc linear-time encodable codes (derived by the linear-time encodable codes of [30]) provides us with LSSS with the required properties.

Lemma 4 (Linear-Time Codes, Theorem 2 in [30]). *For any real number $\delta \in (0,1)$ and large enough integer k, there exist a real number $\rho \in (0,1)$, a positive integer ℓ and a linear code over $\mathbb{F}$ such that the following hold. The code is ℓ-folded; if m is the length of the code and d is its minimum distance, then $\frac{k}{\ell} < m \leq \frac{k}{\ell\rho}$ and $d \geq \delta m$. Furthermore, the code is linear-time encodable.*

Corollary 2 (Linear-Time and Constant-Rate LSSS). *For any real number $\delta \in (0,1)$ there exists a positive integer ℓ such that for any large enough k there exists an (m,k)-coding scheme over $\mathbb{F}$ with the following properties. The scheme is an ℓ-folded linear-time LSSS with δm-uniformity and $m = \Theta(k)$.*

Proof. Given δ and k, let $\boldsymbol{M}$ be the generator matrix of the code of Lemma 4, then the matrix $\boldsymbol{G} = (\boldsymbol{I}_k, \boldsymbol{M})$ defines a ℓ-folded linear code of dimension k, length $m+k$ and distance at least $\delta m + 1$. The Corollary follows from Lemma 3. □

3.2 Construction 1

Finally, we are ready to give the details of Construction 1 and its security proof. All the schemes in the following are defined over the finite field $\mathbb{F}$ and are 1-folded if it is not explicitly stated otherwise. Consider the following building blocks:

- Let $(\mathsf{Enc_{amd}}, \mathsf{Dec_{amd}})$ be a (k', k)-AMD code with error ϵ;
- Let $(\mathsf{Sh}_1, \mathsf{Rec}_1)$ be an ℓ-folded (m, t, m, k')-LSSS with privacy;
- Finally let $(\mathsf{Enc_{td}}, \mathsf{Dec_{td}})$ be an (ℓ', ℓ)-TD codes with respect to the family $\mathcal{F}$ and with error α.

```
Input: $\boldsymbol{m} \in \mathbb{F}^k$

$\mathsf{ENC}_1(\boldsymbol{m})$

  Sample $\boldsymbol{s} \leftarrow \mathsf{Sh}_1(\mathsf{Enc_{amd}}(\boldsymbol{m}))$
  Parse $\boldsymbol{s} = (\boldsymbol{s}_1, \dots \boldsymbol{s}_m)$
  For $i = 1, \dots m$:
    $\boldsymbol{c}_i \leftarrow \mathsf{Enc_{td}}(\boldsymbol{s}_i)$
  Output $\boldsymbol{c} = (\boldsymbol{c}_1, \dots, \boldsymbol{c}_m)$
```

```
Input: $\boldsymbol{c} \in (\mathbb{F}^{\ell'})^m$

$\mathsf{DEC}_1(\boldsymbol{c})$

  Parse $\boldsymbol{c} = (\boldsymbol{c}_1, \dots \boldsymbol{c}_m)$
  For $i = 1, \dots, m$
    $\boldsymbol{s}_i = \mathsf{Dec_{td}}(\boldsymbol{c}_i)$
    If $\boldsymbol{s}_i = \perp$ output $\perp$
  Define $\boldsymbol{s} = (\boldsymbol{s}_1, \dots, \boldsymbol{s}_m)$
  Compute $\boldsymbol{m} = \mathsf{Dec_{amd}}(\mathsf{Rec}_1(\boldsymbol{s}))$
  Output $\boldsymbol{m}$
```

Fig. 4. Construction 1

The new coding scheme $(\mathsf{ENC}_1, \mathsf{DEC}_1)$ is defined in Figure 4. We indicate with $\mathcal{F}^+$ the set of tampering functions $f : (\mathbb{F}^{\ell'})^m \to (\mathbb{F}^{\ell'})^m$ in $\mathcal{F}^q_{\ell',m}$ such each f_i is a function from $\mathcal{F} \cup \mathcal{F}_{const} \cup \{id\}$. That is, each block $\boldsymbol{c}_i$ of the encoding is modified by the adversary using a function $f_i : \mathbb{F}^{\ell'} \to \mathbb{F}^{\ell'}$, which can be any function from $\mathcal{F} \cup \mathcal{F}_{const} \cup \{id\}$ provided that it doesn't depend on the others blocks of the encoding.

Theorem 1. *If $t > \frac{m}{2}$, then $(\mathsf{ENC}_1, \mathsf{DEC}_1)$ defined in Figure 4 is an ℓ'-folded (m, k)-NM code with respect to the family $\mathcal{F}^+$ with error less than or equal to $\max\{\epsilon, \alpha^{2t-m}\}$. Moreover, if ρ is the rate of $(\mathsf{Enc_{amd}}, \mathsf{Dec_{amd}})$ and ρ' is the rate of the sharing scheme, then the rate $k/m\ell'$ of the new scheme is $\rho\rho'\frac{\ell}{\ell'}$.*

Proof. The correctness of the scheme $(\mathsf{ENC}_1, \mathsf{DEC}_1)$ (*i.e.* $\Pr[\mathsf{DEC}_1(\mathsf{ENC}_1(\boldsymbol{m})) = \boldsymbol{m}] = 1$ for any $\boldsymbol{m} \in \mathbb{F}^k$) and the statement about the rate are easy to verify and follow directly from the construction (Figure 4). Fix $f = (f_1, f_2, \dots, f_m) \in \mathcal{F}^+$, to prove the non-malleability property, we have to define D_f as in Definition 2 and bound the error $\mathsf{SD}(\mathrm{Real}^{\boldsymbol{m}}_f, \mathrm{Ideal}^{\boldsymbol{m}}_f)$ for any $\boldsymbol{m} \in \mathbb{F}^k$. Let $\boldsymbol{c} = (\boldsymbol{c}_1, \dots, \boldsymbol{c}_m) = \mathsf{ENC}_1(\boldsymbol{m})$ and $\boldsymbol{s} = (\boldsymbol{s}_1, \dots, \boldsymbol{s}_m) = \mathsf{Sh}_1(\mathsf{Enc_{amd}}(\boldsymbol{m}))$. Notice that a valid encoding in the new scheme is a vector $\boldsymbol{c} = (\boldsymbol{c}_1, \dots, \boldsymbol{c}_m)$ of m blocks each of which is an encoding done by the constant-size tamper-detection code $(\mathsf{Enc_{td}}, \mathsf{Dec_{td}})$. Each block is independently tampered by the function $f_i : \mathbb{F}^{\ell'} \to \mathbb{F}^{\ell'}$ and since $(\mathsf{Enc_{td}}, \mathsf{Dec_{td}})$ is an TD code, for any block such that $f_i \in \mathcal{F}$ we know that the outputs of $\mathsf{Dec_{td}}(f_i(\boldsymbol{c}_i))$ is $\perp$ with probability greater or equal to $1 - \alpha$. Using this and the t-privacy property, in the following we will show that we can have enough information on the output of $\mathsf{DEC_1}(f(\mathsf{ENC_1}(\boldsymbol{m})))$ only looking at how many blocks have been tampered by functions not in $\mathcal{F}$. More precisely, define the following sets: $I \subseteq [m]$ is the set of indices i such that f_i is the identity function, $C \subseteq [m]$ is the set of indices i such that f_i is a constant function on $\mathbb{F}^{\ell'}$ and $J = [m] \setminus (I \cup C) = (I \cup C)^c$. Consider now the following cases:

1) Suppose that many blocks are tampered using constant functions (*i.e.* $|C| \geq m - t$). Then, the t-privacy implies that the distribution of the blocks not touched by a constant function is the same for any input message $\boldsymbol{m}$, while all the other blocks are fixed to known constants. Hence, we define D_f as
   - sample $\boldsymbol{d}$ accordingly to the distribution of $\mathsf{ENC_1}(\mathbf{0})$ and output the result of $\mathsf{DEC_1}(f(\boldsymbol{d}))$.

   Because of the t-privacy, $\mathsf{DEC_1}(f(\boldsymbol{d}))$ has the same distribution of $\mathsf{DEC_1}(f(\boldsymbol{c}))$ and thus we have that $\mathsf{SD}(\mathrm{Real}_f^{\boldsymbol{m}}, \mathrm{Ideal}_f^{\boldsymbol{m}}) = 0$.
2) Otherwise we can assume that few blocks are tampered by constant functions (*i.e.* $|J| + |I| > t$) and we consider two sub-cases.

   2.a) Suppose that few blocks are tampered (*i.e.* $|I| \geq m - t$) and look at what happens during the execution of $\mathsf{DEC_1}$ on input $f(\boldsymbol{c})$. If there exists $i \in I^c$ such that $\mathsf{Dec_{td}}(f_i(\boldsymbol{c}_i)) = \perp$, then the entire decoding outputs $\perp$. Otherwise, we have the situation described by Lemma 1 with[13] $g = \mathsf{Dec_{td}} \circ f \circ \mathsf{Enc_{td}}$. Indeed, in the decoding phase the algorithm $\mathsf{Rec_1}$ is applied to a share vector $\tilde{\boldsymbol{s}}$ where at most t components have been modified respect to the original share vector $\boldsymbol{s}$. It follows by Lemma 1 that $\mathsf{Rec_1}(\tilde{\boldsymbol{s}})$ has the same distribution as $\mathsf{Enc_{amd}}(\boldsymbol{m}) + \Delta_g$. Moreover, by definition of AMD code, if $\Delta_g = \mathbf{0}$, then $\mathsf{DEC_1}(f(\boldsymbol{c}))$ outputs the original message $\boldsymbol{m}$, else it outputs $\perp$ with probability grater than or equal to $1 - \epsilon$. Thus, in this case we define D_f by the following steps:
   - sample $\boldsymbol{r} = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m)$ accordingly to the distribution of $\mathsf{Sh_1}(\mathbf{0})$. If there exists $i \in I^c$ such that $\mathsf{Dec_{td}}(f_i(\mathsf{Enc_{td}}(\boldsymbol{r}_i))) = \perp$, then output $\perp$. Otherwise continue with the next step;
   - sample $\boldsymbol{e}$ accordingly to the distribution of Δ_g. If $\boldsymbol{e} = \mathbf{0}$, D_f outputs *same*; otherwise it outputs $\perp$.

   Because of the t-privacy, the probability that there exists $i \in I^c$ such that $\mathsf{Dec_{td}}(f_i(\boldsymbol{c}_i)) = \perp$ is equal to the probability that there exists $i \in I^c$ such that $\mathsf{Dec_{td}}(f_i(\mathsf{Enc_{td}}(\boldsymbol{r}_i))) = \perp$. Moreover, Lemma 1 implies that $\mathsf{SD}(\mathrm{Real}_f^{\boldsymbol{m}}, \mathrm{Ideal}_f^{\boldsymbol{m}}) = \Pr[\mathsf{Dec_{amd}}(\mathsf{Enc_{amd}}(\boldsymbol{m}) + \Delta_g)) \neq \perp]$ and we know the latter to be less than or equal to ϵ.

   2.b) Else we can use the assumption on t and m and say that more than $2t - m$ blocks are tampered by functions in $\mathcal{F}$. That is, $|J| > t - m + t = 2t - m > 0$. Independently for all these blocks, the tamper-detection code outputs a message different from $\perp$ with probability less than or equal to α. Thus, $\mathsf{DEC_1}(f(\boldsymbol{c})) = \perp$ with probability less than or equal to α^{2t-m}.

[13] Abuse of notation, with $g = \mathsf{Dec_{td}} \circ f \circ \mathsf{Enc_{td}}$ we mean the randomized function $g : (\mathbb{F}^\ell)^m \to (\mathbb{F}^\ell)^m$ such that $(g(\boldsymbol{v}))_i = \mathsf{Dec_{td}}(f_i(\mathsf{Enc_{td}}(\boldsymbol{v}_i)))$ for all $i \in [m]$.

Therefore, in this last case we define D_f to output $\perp$ and we have that $\mathsf{SD}(\mathrm{Real}_f^{\boldsymbol{m}}, \mathrm{Ideal}_f^{\boldsymbol{m}}) = \Pr[\mathrm{Real}_f^{\boldsymbol{m}} \neq \perp] \leq \Pr[\mathsf{Dec}_{\mathsf{td}}(f_i(\boldsymbol{c}_i)) \neq \perp \ \forall\, i \in J] \leq \alpha^{2t-m}$. □

We are now ready to state the first of the results about linear-time NM codes that we present in this paper:

Theorem 2 (Linear-Time and Constant-Rate NM codes). *If for infinitely many integers b, there exists an (b', b)-TD code with respect of a family $\mathcal{F}$ and with constant error α, then there exist a positive integer ℓ' such that the following holds. For any large enough integer k there exists an ℓ'-folded (m, k)-NM code $(\mathsf{ENC}_1, \mathsf{DEC}_1)$ with respect of the family $\mathcal{F}^+$ and $m = \Theta(k)$. Furthermore, the NM code has error negligible in k and linear-time computational complexity.*

Proof. Given k, instantiate Construction 1 (Figure 4) with the (k, k')-AMD code given by Corollary 1 and with the ℓ-folded $(m, \delta m, m, k')$-LSSS given by Corollary 2 (with $\delta > 1/2$). Notice that $k' = \Theta(k)$ and $m = \Theta(k') = \Theta(k)$ and ℓ is constant respect to k. Finally, use the first TD code from the family stated in the thesis such that the input length is at least ℓ to complete the instantiation. Let ℓ' be the output length of the TD code used. Notice that also ℓ' is constant respect to k. It follows from Theorem 1 that the obtained scheme is non-malleable with respect to $\mathcal{F}^+$ and has constant rate. Moreover, the error $\epsilon + \alpha^{2\delta m - m} = q^{-\Theta(k)} + \alpha^{(2\delta-1)\Theta(k)}$ is negligible in k. Finally, since ℓ, ℓ' are constant and the AMD code and the LSSS are both linear-time, the computational complexity of the algorithms $\mathsf{ENC}_1, \mathsf{DEC}_1$ is $O(k)$. □

In [16] an infinite family of TD code with respect the family $\mathcal{F}$ of bit-wise independent tampering functions that are neither the identity nor constant functions is given. Each code in the family has an error less than or equal to 2/3.

Lemma 5 (Lemma 3.5 in [16]). [14] *For any $\beta \in (0, 1)$ and any large enough ℓ' (i.e. $\ell' \geq \ell'(\beta) = O(\log^2(1/\beta)/\beta)$), there exists a binary (ℓ, ℓ')-TD code respect to the family $\mathcal{F} = \mathcal{F}_{1,n}^2 \setminus (\mathcal{F}_{const} \cup \{id\})$ with error 2/3 and with $\ell \geq (1 - \beta)\ell'$.*

The previous lemma together with Theorem 2 implies the following result in bit-wise independent tampering model.

Corollary 3 (Binary Case for Construction 1). *For any large enough integer k, there exists a linear-time binary (N, k)-NM code with respect of the family $\mathcal{F}_{1,N}^2$ and with error negligible in k. Furthermore $N = \Theta(k)$.*

4 Optimal-Rate and Linear-Time NM Codes

In this section, we will construct a linear-time non-malleable code with rate approaching 1 (Construction 2).

[14] The construction presented in [16] is randomised, but since in our Construction 1 the parameter ℓ is constant (respect to k) we can exhaustively search for the proper TD code.

4.1 Building Blocks for Construction 2

Before showing our second main result (Construction 2), we present the required building blocks.

In order to achieve linear-time and optimal-rate NM codes, we will employ linear-time (n, t, n, k)-secret-sharing schemes again, however we will need stronger assumptions regarding the rate and the privacy property of the used scheme. Namely, besides linear-time complexity, we require that the rate is not merely constant but that it approaching 1, *i.e.*, length of a full share-vector divided by the length of the secret tends to 1 when the n tends to infinity. By general bounds on secret sharing, this implies that the privacy parameter t is sublinear in the number of players n and that reconstruction is essentially by the full player set only. But that is still fine for our purposes here (as long as privacy is nonconstant). Moreover, we note that we do not require linearity of the scheme either. Besides, we require that any t shares are uniformly and independently distributed over the share-space (t-uniformity). Below we show how to construct the schemes required here by combining results on t-wise independence generators and constant-rate secret sharing. A t-wise independence generator is a deterministic algorithm that expands a short random seed in a longer t-uniform vector. More precisely:

Definition 5 (t-wise Independence Generator, [36]). *Let k, k' and t be positive integers. A function $\mathsf{Gen} : \mathbb{F}^{k'} \to \mathbb{F}^{k}$ is a* t-wise independence generator *if the following holds. For each uniform random variable X over $\mathbb{F}^{k'}$ (called the* seed*), $\mathsf{Gen}(X)$ is t-wise uniform over $\mathbb{F}^{k}$.*

In [25] (Lemma 12) we provide an independence generator with seed-length and independence sub-linear in the output length. Moreover the proposed independence generator has computational complexity linear in the seed-length. Lemma 6 shows how to use the t-wise independence generator to build a linear-time secret-sharing scheme with t'-uniformity, $t' = \Theta(t)$ and rate $1 - o(1)$. The high-level idea (Figure 5) is simple, to share a secret $\boldsymbol{m} \in \mathbb{F}^{k}$ we do the following. First, we mask $\boldsymbol{m}$ using $\mathsf{Gen}(\boldsymbol{s})$ where $\boldsymbol{s}$ is a uniformly random seed for Gen. Then, we share the seed $\boldsymbol{s}$ with a constant-rate sharing scheme (for example, the scheme from Corollary 2). The final share vector is defined by the concatenation of $\boldsymbol{m} + \mathsf{Gen}(\boldsymbol{s})$ and the share vector of $\boldsymbol{s}$.

Lemma 6 (Linear-Time and Optimal-Rate LSSS). *For any real number $\epsilon \in (0, 1)$ and any large enough k, there exists a linear-time (n, t, n, k)-LSSS with uniformity such that $t = \Omega(k^{1-\epsilon})$ and $n = k + o(k)$.*

Proof. Given $\epsilon \in (0, 1)$ and k large enough, there exists a t-wise independence generator $\mathsf{Gen} : \mathbb{F}^{k'} \to \mathbb{F}^{k}$ with $t = \Omega(k^{1-\epsilon})$ and $k' = \Theta(k^{1-\delta})$ ($\delta \leq \epsilon$, see Lemma 12 in [25]). Let $(\mathsf{Sh}_1, \mathsf{Rec}_1)$ be the (m, t', m, k')-LSSS from Corollary 2. Notice[15] that $m = \Theta(k')$ and that the scheme is t'-uniform with $t' = \Theta(k')$.

[15] The family of LSSSs from Corollary 2 is ℓ folded, where ℓ is a constant respect to k'. Thus, the scheme $(\mathsf{Sh}_1, \mathsf{Rec}_1)$ can be "unfolded" and still it remains a constant-rate scheme.

```
$\mathsf{Sh}_2(\boldsymbol{m})$:
    Sample $\boldsymbol{s} \leftarrow \mathbb{F}^{k'}$
    Compute $\boldsymbol{c}_1 = \boldsymbol{m} + \mathsf{Gen}(\boldsymbol{s})$
    Compute $\boldsymbol{c}_2 \leftarrow \mathsf{Sh}_1(\boldsymbol{s})$
    Output $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$
```

```
$\mathsf{Rec}_2(\boldsymbol{c})$:
    Parse $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$
    Compute $\boldsymbol{s} = \mathsf{Rec}_1(\boldsymbol{c}_2)$
    If $\boldsymbol{s} = \perp$, then output $\perp$
    Otherwise output $\boldsymbol{c}_1 - \mathsf{Gen}(\boldsymbol{s})$
```

Fig. 5. Linear-time and optimal-rate LSSS

Consider the scheme in Figure 5 and define $s = \min\{t, t'\}$. It is easy to verify that $(\mathsf{Sh}_2, \mathsf{Rec}_2)$ is a linear-time (n, s, n, k)-LSSS with uniformity. Moreover, $s = \Omega(k^{1-\epsilon})$ and $n = k + m = n + O(k^{1-\delta})$. □

We introduce a novel primitive, a *compressor.* Suppose we are given a vector whose coordinates are t-wise independent random variables. A compressor is a deterministic function that, when applying it to the given vector, results in a shorter vector with nontrivial entropy[16], assuming that the original vector contains at least t coordinates with nontrivial entropy[17].

Definition 6 (Compressor). *Let t, n, n' be positive integers and r a positive real number. A function* $\mathsf{Comp} : \mathbb{F}^n \to \mathbb{F}^{n'}$ *is a* (t, r)-compressor *if the following holds. Suppose that $X = (X_1, \ldots, X_n)$ is a t-wise independent random variable on $\mathbb{F}^n$ such that there is a set $A \subseteq [n]$ of cardinality t and a real number $c > 0$ for which $\mathrm{H}_\infty(X_i) \geq c$ for all $i \in A$. Then $\mathrm{H}_\infty(\mathsf{Comp}(X)) \geq rc$.*

This primitive is used in the security proof of Construction 2 to handle the case of a component-wise tampering function that has many non-constant components. More precisely, we will use the following fact:

Lemma 7. *Let $f = (f_1, \ldots, f_n) \in \mathcal{F}^q_{1,n}$ be a function such that least t of the of the functions $f_i : \mathbb{F} \to \mathbb{F}$ are non-constant. If $\mathsf{Comp} : \mathbb{F}^n \to \mathbb{F}^{n'}$ is a (t, r)-compressor and X is a t-wise uniform random variable on $\mathbb{F}^n$, then for any vector $\boldsymbol{b} \in \mathbb{F}^{n'}$, $\Pr[\mathsf{Comp}(f(X)) = \boldsymbol{b}] \leq \left(\frac{q-1}{q}\right)^r$.*

Proof. By the conditions on f, there is a set $A \subseteq [n]$ of cardinality t such that, for each $i \in A$ it holds that $\mathrm{H}_\infty(f_i(X_i)) \geq \log_2(q/(q-1))$. Since X is t-wise independent, it follows by definition of compressor that $\mathrm{H}_\infty(\mathsf{Comp}(f(X))) \geq r \log_2(q/(q-1))$. □

We show a simple construction of Comp suitable for our purposes later on.

[16] The *min-entropy* of a random variable X is $\mathrm{H}_\infty(X) = -\log_2(\max_{\boldsymbol{b}} \Pr[X = \boldsymbol{b}])$

[17] Since we require compressors to be deterministic, generic methods for privacy amplification do not apply here.

Lemma 8 (Linear-Time Compressor). *For any real number $\epsilon \in (0,1)$ and for any large enough positive integer n there exists an (r^2, r)-compressor $\mathsf{Comp} : \mathbb{F}^n \to \mathbb{F}^{n'}$ with $r^2 = \Omega(n^{1-\epsilon})$ and $n' = o(n)$. Moreover Comp has computational complexity $O(n)$.*

Proof. Given ϵ, for any $n \geq 1$ define $r = \lceil n^{(1-\epsilon)/2} \rceil$ and $n' = \lfloor n/r \rfloor$. Notice that $n'r \leq n$ and, if n large enough, $r^2 \leq n$. Consider the function $\mathsf{Comp} : \mathbb{F}^n \to \mathbb{F}^{n'}$, $(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \mapsto (\boldsymbol{y}_1, \dots, \boldsymbol{y}_{n'})$ defined by $\boldsymbol{y}_i = \sum_{j=1}^{r} \boldsymbol{x}_{(i-1)r+j}$ for $i = 1, \dots, n'$. Thus, a vector in the domain is viewed as comprising n' consecutive blocks of r coordinates and, for $i = 1, \dots, n'$, the sum taken over the coordinates in the i-th block gives the i-th coordinate in the image of the vector under Comp. We now verify that Comp is a (r^2, r)-compressor. Suppose $X = (X_1, \dots, X_n)$ be a r^2-wise independent random variable on $\mathbb{F}^n$ and suppose $A \subset [n]$ with $|A| = r^2$ satisfies $\mathrm{H}_\infty(X_i) \geq c > 0$ for each $i \in A$. Define $(Y_1, \dots, Y_{n'}) = \mathsf{Comp}(X)$. By the pigeonhole principle, there exists a $B \subseteq [n']$ with $|B| = r$ such that each Y_i with $i \in B$ is sum of at least one X_i with $i \in A$. This, together with r^2-independence of X, implies that the corresponding random variable $Y_B = (Y_i)_{i \in B}$ has the properties that $\mathrm{H}_\infty(Y_i) \geq c$ for each $i \in B$ and that the Y_i's are independent. In conclusion, $\mathrm{H}_\infty(\mathsf{Comp}(X)) \geq \mathrm{H}_\infty(Y_B) \geq rc$. By inspection, the computational complexity of Comp is $O(n)$. □

Our Construction 2 that we present later on in Section 4.2 depends in particular on *universal hash functions*.

Definition 7 (Almost Universal Family). *Let $\mu \in (0,1)$ be a real number and let n, m be positive integers. Suppose $\mathcal{H}$ is a family of functions $h_{\boldsymbol{k}} : \mathbb{F}^n \to \mathbb{F}^m$, one for each $\boldsymbol{k} \in \mathbb{F}^a$. Then $\mathcal{H}$ is μ-*almost universal *if the following holds. For any pair of distinct $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{F}^n$, if $\boldsymbol{k}$ is chosen uniformly at random from $\mathbb{F}^a$ then $\Pr[h_{\boldsymbol{k}}(\boldsymbol{x}) = h_{\boldsymbol{k}}(\boldsymbol{x}')\}] \leq \mu$.*

For our purposes, we require that these functions are linear-time computable and have vanishingly small key- and output-lengths. Hence, the linear uniform family of [30] (see Lemma 2) does not apply directly due to its linear key-length. Note that, besides linear-time, the uniform output property of this particular family enables arbitrary output-length. In [25] we show an easy adaptation of the family from [30] suitable for our purposes. It is a μ-almost universal family. But since μ is very small, it is good enough for our purposes.

Lemma 9. *For any real number $\beta \in (0,1)$ and any positive integer n, there exists a μ-universal family $\mathcal{H} = \{h_{\boldsymbol{k}} : \mathbb{F}^n \to \mathbb{F}^m\}_{\boldsymbol{k} \in \mathbb{F}^a}$ with $a = o(n)$, $m = o(n)$ and $\mu = \Theta(q^{-n^{(1-\beta)}})$. Moreover, each function $h_{\boldsymbol{k}}$ has complexity $O(n)$.*

4.2 Construction 2

Finally, we are ready to give the details of Construction 2 and its security proof. Consider the following ingredients (all the scheme are over the finite field $\mathbb{F}$):

- Let $(\mathsf{Sh}_2, \mathsf{Rec}_2)$ an (n, t, n, k)-SSS with uniformity;

- Let $\mathsf{Comp} : \mathbb{F}^n \to \mathbb{F}^{n'}$ be a (t, r)-compressor;
- Let $\mathcal{H} = \{h_{\boldsymbol{k}} : \mathbb{F}^n \to \mathbb{F}^m\}$ be a μ-almost universal family with key-space $\mathbb{F}^a$;
- Let $(\mathsf{Enc}, \mathsf{Dec})$ be a (b', b)-NM code with respect to a family $\mathcal{F}$ with error ϵ. We require that $b = a + m + n'$.

Let $N = n + b'$, the new (N, k)-coding scheme $(\mathsf{ENC}_2, \mathsf{DEC}_2)$ is defined in Figure 6.

```
Input: $\boldsymbol{m} \in \mathbb{F}^k$

$\mathsf{ENC}_2(\boldsymbol{m})$:

    Compute $\boldsymbol{c}^{(1)} \leftarrow \mathsf{Sh}_2(\boldsymbol{m})$
    Sample $\boldsymbol{k} \leftarrow \mathbb{F}^a$
    Compute $\boldsymbol{h} = h_{\boldsymbol{k}}(\boldsymbol{c}^{(1)})$
    Compute $\boldsymbol{c} = \mathsf{Comp}(\boldsymbol{c}^{(1)})$
    Compute $\boldsymbol{c}^{(2)} = \mathsf{Enc}(\boldsymbol{k}, \boldsymbol{h}, \boldsymbol{c})$
    Output $(\boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)})$
```

```
Input: $\boldsymbol{c} \in \mathbb{F}^N$

$\mathsf{DEC}_2(\boldsymbol{c})$:

    Parse $\boldsymbol{c} = (\boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}) \in \mathbb{F}^n \times \mathbb{F}^{b'}$
    Compute $\boldsymbol{z} = \mathsf{Dec}(\boldsymbol{c}^{(2)})$
    If $\boldsymbol{z} = \perp$ output $\perp$
    Otherwise
        Parse $\boldsymbol{z} = (\boldsymbol{k}, \boldsymbol{h}, \boldsymbol{c})$
        If $\boldsymbol{h} \neq h_{\boldsymbol{k}}(\boldsymbol{c}^{(1)})$ output $\perp$
        If $\boldsymbol{c} \neq \mathsf{Comp}(\boldsymbol{c}^{(1)})$ output $\perp$
    Output $\boldsymbol{m} = \mathsf{Rec}_2(\boldsymbol{c}^{(1)})$
```

Fig. 6. Construction 2

Theorem 3. *The coding scheme* $(\mathsf{ENC}_2, \mathsf{DEC}_2)$ *is an* (N, k)*-non-malleable code with respect to the family* $\mathcal{F}^q_{1,n} \times \mathcal{F}$ *with error less than or equal to*

$$\max\left\{\left(\frac{q-1}{q}\right)^t + \mu, \left(\frac{q-1}{q}\right)^r\right\} + \epsilon$$

Proof. It is trivial to verify that the scheme $(\mathsf{DEC}_2, \mathsf{ENC}_2)$ is correct, that is $\Pr[\mathsf{DEC}_2(\mathsf{ENC}_2(\boldsymbol{m})) = \boldsymbol{m}] = 1$ for all $\boldsymbol{m} \in \mathbb{F}^k$. In order to prove non-malleability, for each tampering function F we have to show a simulator which only depends on F and whose output distribution is statistically close to the one of $\mathsf{DEC}_2(F(\mathsf{ENC}_2(\boldsymbol{m})))$ for any given $\boldsymbol{m} \in \mathbb{F}^k$. More precisely, according to Definition 2 for any $F = (f, g) \in \mathcal{F}^q_{1,n} \times \mathcal{F}$, we have to define a random variable D_F and bound the error $\epsilon' = \mathsf{SD}(\mathrm{Real}^{\boldsymbol{m}}_F, \mathrm{Ideal}^{\boldsymbol{m}}_F)$ for any $\boldsymbol{m} \in \mathbb{F}^k$. Given F and $\boldsymbol{m} \in \mathbb{F}^k$, we write $\mathsf{ENC}_2(\boldsymbol{m}) = (\boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)})$. Notice that the left part of the encoding, $\boldsymbol{c}^{(1)}$, is tampered by the function $f \in \mathcal{F}^q_{1,n}$, while the right part, $\boldsymbol{c}^{(2)}$, by the function g from $\mathcal{F}$. Since $(\mathsf{Enc}, \mathsf{Dec})$ is a NM-code, there exists the random variable D_g such that $\mathsf{SD}(\mathrm{Real}^{\boldsymbol{z}}_g, \mathrm{Ideal}^{\boldsymbol{z}}_g) \le \epsilon$ for all $\boldsymbol{z} \in \mathbb{F}^b$. That is, we can simulate the output of decoding the right part, $\mathsf{Dec}(g(\boldsymbol{c}^{(2)}))$, using the random variable $\mathrm{Ideal}_g^{\boldsymbol{z}}$. Specifically, we define the random variable $\mathrm{Hyb}_F^{\boldsymbol{m}}$ as detailed in Figure 7. Notice that by construction the output of $\mathrm{Hyb}_F^{\boldsymbol{m}}$ depends on $\boldsymbol{c}^{(1)}$ (the output of $\mathsf{Sh}_2(\boldsymbol{m})$) and on the output of $\mathrm{Ideal}_g^{\boldsymbol{z}}$, and the output of $\mathrm{Real}_F^{\boldsymbol{m}}$ depends on $\boldsymbol{c}^{(1)}$ and the output of $\mathrm{Real}_g^{\boldsymbol{z}}$ in the same way. Thus, we have that $\mathsf{SD}(\mathrm{Real}_F^{\boldsymbol{m}}, \mathrm{Hyb}_F^{\boldsymbol{m}}) \leq \mathsf{SD}(\mathrm{Real}_g^{\boldsymbol{z}}, \mathrm{Ideal}_g^{\boldsymbol{z}})$. Given this, defining the random variable D_F in such a way that we can bound $\epsilon'' = \mathsf{SD}(\mathrm{Hyb}_F^{\boldsymbol{m}}, \mathrm{Ideal}_F^{\boldsymbol{m}})$ will conclude the proof. Indeed, we have $\epsilon' \leq \epsilon + \epsilon''$. To define D_F, first sample $\boldsymbol{z}^*$ randomly according to D_g. The results of the sampling can be classified in three cases: $\perp$, *same* or some vector $(\boldsymbol{k}^*, \boldsymbol{h}^*, \boldsymbol{c}^*)$. Then, we proceed in the definition of D_F in a different way for each one of the three aforementioned cases. In each case, we will bound the error ϵ''. In the following, we will write $f = (f_1, \dots, f_n) \in \mathcal{F}_{1,n}^q$. Remember that the value of $\boldsymbol{z}^*$ determines the output $\boldsymbol{z}'$ of $\mathrm{Ideal}_g^{\boldsymbol{z}}$.

1) Assume that $\boldsymbol{z}^* = \perp$, then $\boldsymbol{z}' = \perp$. We know that $\Pr[\mathrm{Hyb}_F^{\boldsymbol{m}} = \perp \mid D_g = \perp] = 1$, thus we define D_F to output $\perp$ and we get that $\epsilon'' = 0$.

2) If $\boldsymbol{z}^* = \mathit{same}$, then $\boldsymbol{z}' = (\boldsymbol{k}, h_{\boldsymbol{k}}(\boldsymbol{c}^{(1)}), \mathsf{Comp}(\boldsymbol{c}^{(1)}))$. Define $I \subseteq [n]$ the set of indices i such that f_i is the identity function on $\mathbb{F}$. Consider the following two situations.
   - First, assume that many f_i are the identity function (*i.e.* $|I| \geq n - t$). Then the difference $f(\boldsymbol{c}^{(1)}) - \boldsymbol{c}^{(1)}$ depends only on the vector $(\boldsymbol{c}^{(1)})_{I^c}$ whose entries are independent of $\boldsymbol{m}$ (because of the t-uniformity property). In particular, both the event $f(\boldsymbol{c}^{(1)}) = \boldsymbol{c}^{(1)}$ and its complement occur with the same probability for any message $\boldsymbol{m}$. If $f(\boldsymbol{c}^{(1)}) = \boldsymbol{c}^{(1)}$, then $\mathrm{Hyb}_F^{\boldsymbol{m}}$ obviously outputs the original message $\boldsymbol{m}$. Otherwise, we have $f(\boldsymbol{c}^{(1)}) \neq \boldsymbol{c}^{(1)}$ and the check done via the hash function $h_{\boldsymbol{k}}$ fails with probability at least $1 - \mu$. If the check fails, $\mathrm{Hyb}_F^{\boldsymbol{m}}$ outputs $\perp$. Given this, we define D_F in the following way:
     - sample $\boldsymbol{r}_i \leftarrow \mathbb{F}$ for all $i \in I^c$; if $f_i(\boldsymbol{r}_i) = \boldsymbol{r}_i$ for all $i \in I^c$ then outputs *same*, otherwise output $\perp$.

     As we have already argued before, the t-uniformity property implies that the event $f_i(\boldsymbol{r}_i) = \boldsymbol{r}_i$ for all $i \in I^c$ has the same probability as the event $f(\boldsymbol{c}^{(1)}) = \boldsymbol{c}^{(1)}$ and therefore, as a consequence of the check involving the hash function, we can bound the error in the following way:

$$\begin{aligned}\epsilon'' &\leq \Pr[\mathrm{Hyb}_F^{\boldsymbol{m}} \neq \perp \mid D_g = \mathit{same} \text{ and } f(\boldsymbol{c}^{(1)}) \neq \boldsymbol{c}^{(1)}] \\ &\leq \Pr[h_{\boldsymbol{k}}(f(\boldsymbol{c}^{(1)})) = h_{\boldsymbol{k}}(\boldsymbol{c}^{(1)}) \mid f(\boldsymbol{c}^{(1)}) \neq \boldsymbol{c}^{(1)}] \leq \mu\end{aligned}$$

   - In the second case, assume that many f_i are not the identity function (*i.e.* $|I| < n - t$). Then, there exists a set $A \subseteq I^c$ of size t, and it follows again from the uniformity property that the events $f_i(\boldsymbol{c}_i^{(1)}) \neq \boldsymbol{c}_i^{(1)}$ with $i \in A$ are independent and each of them occurs with probability at least $1/q$. Therefore, very likely and independently of $\boldsymbol{m}$, $f(\boldsymbol{c}^{(1)}) \neq \boldsymbol{c}^{(1)}$ and $\mathrm{Hyb}_F^{\boldsymbol{m}}$ outputs $\perp$ because of the check done using the hash function $h_{\boldsymbol{k}}$. For this reason, in this case we define D_F to always output $\perp$ and we can bound the error as follows.

$$\epsilon'' \leq \Pr[\mathrm{Hyb}^{\boldsymbol{m}}_F \neq \perp |\ D_g = \mathit{same}] \leq \Pr[h_{\boldsymbol{k}}(f(\boldsymbol{c}^{(1)})) = h_{\boldsymbol{k}}(\boldsymbol{c}^{(1)})]$$
$$\leq \Pr[f(\boldsymbol{c}^{(1)}) = \boldsymbol{c}^{(1)}] + \mu \leq \left(\frac{q-1}{q}\right)^t + \mu$$

3) If $\boldsymbol{z}^* = (\boldsymbol{k}^*, \boldsymbol{h}^*, \boldsymbol{c}^*)$, then we have that $\boldsymbol{z'} = \boldsymbol{z}^*$. Let $C \subseteq [n]$ be the set of all indices i such that f_i is a constant function on $\mathbb{F}$. Consider the following two situations.
   - If many f_i are constant functions (*i.e.* $|C| \geq n-t$), then the value of vector $f(\boldsymbol{c}^{(1)})$ is independent of $\boldsymbol{m}$. Indeed, the t-uniformity makes the value of $(f(\boldsymbol{c}^{(1)}))_{C^c}$ independent of $\boldsymbol{m}$, while $(f(\boldsymbol{c}^{(1)}))_C$ is fixed equal to a constant defined only by f. It follows that, if we define D_F in this way:
     - sample $\boldsymbol{r} \leftarrow \mathbb{F}^n$, if $\boldsymbol{h}^* \neq h_{\boldsymbol{k}^*}(f(\boldsymbol{r}))$ or $\boldsymbol{c}^* \neq \mathsf{Comp}(f(\boldsymbol{r}))$ output $\perp$; otherwise output $\mathsf{Rec}_2(f(\boldsymbol{r}))$.

     then we have that $\epsilon'' = 0$.
   - Otherwise more than t components f_i are not constant functions (*i.e.* $|C| < n-t$) and it follows from Lemma 7 that $\mathsf{Comp}(f(\mathsf{Sh}_2(\boldsymbol{m})))$ is a random variable with min-entropy at least $r\log_2(q/(q-1))$. Moreover, $\mathsf{Comp}(f(\mathsf{Sh}_2(\boldsymbol{m})))$ is independent of the random variable D_g. Therefore, in this case the probability that the check done using the compressor is satisfied is less than or equal to $\left(\frac{q-1}{q}\right)^r$. Remember that if the check is not satisfied then, $\mathrm{Hyb}^{\boldsymbol{m}}_F$ outputs abort. Thus, we can define D_F to output always $\perp$ and we get an error bounded by:

$$\epsilon'' \leq \Pr[\mathrm{Hyb}^{\boldsymbol{m}}_F \neq \perp |\ D_g = (\boldsymbol{k}^*, \boldsymbol{h}^*, \boldsymbol{c}^*)] \leq \Pr[\mathsf{Comp}(f(\boldsymbol{c}^{(1)}) = \boldsymbol{c}^*]$$
$$\leq \left(\frac{q-1}{q}\right)^r$$

□

We are now ready to state the main result about linear-time NM codes that we present in this paper:

Theorem 4 (Linear-Time and Optimal-Rate NM codes). *Suppose that there exists real number $\alpha \in (0,2)$ such that for any positive integer b there exists a (b', b)-NM-code* $(\mathsf{Enc}, \mathsf{Dec})$ *with respect of a family $\mathcal{F}$, with error $\epsilon(b) = negl(b)$ (the error is a negligible function of the message length) and $b' = O(b^\alpha)$, then the following holds. For any positive integer k large enough, there exists an (N, k)-NM code* $(\mathsf{ENC}_2, \mathsf{DEC}_2)$ *with respect of the family $\mathcal{F}^q_{1,n} \times \mathcal{F}$ and with error negligible in k. Furthermore $N = k + o(k)$ and, if the computational complexity of* $(\mathsf{Enc}, \mathsf{Dec})$ *is sub-quadratic in b, then* $(\mathsf{ENC}_2, \mathsf{DEC}_2)$ *is linear-time.*

Proof. Instantiate Construction 2 (Figure 6) with the t-uniform sharing scheme from Lemma 6, the compressor from Lemma 8 (with $\epsilon \leq \frac{2}{\alpha} - 1$) and the universal family from Lemma 9 (with $\beta \geq 2 - \frac{2}{\alpha}$). It easy to check that $b = (a+m) + n' = O(n^{1-\beta/2}) + O(n^{(1+\epsilon)/2})$ and $n = k + o(k)$. Thus, $b' = O(n^{1-\beta/2})^\alpha +$

```
Real$^{\boldsymbol{m}}_F$:
    Compute $\boldsymbol{c}^{(1)} \leftarrow \mathsf{Sh}_2(\boldsymbol{m})$
    Sample $\boldsymbol{k} \leftarrow \mathbb{F}^a$
    Compute $\boldsymbol{z} = (\boldsymbol{k}, h_{\boldsymbol{k}}(\boldsymbol{c}^{(1)})), \mathsf{Comp}(\boldsymbol{c}^{(1)})$
    Compute $\boldsymbol{z}' \leftarrow \mathrm{Real}^{\boldsymbol{z}}_g$
    If $\boldsymbol{z}' = \perp$ output $\perp$
    Otherwise
        Parse $\boldsymbol{z}' = (\boldsymbol{k}', \boldsymbol{h}', \boldsymbol{c}')$
        If $\boldsymbol{h}' \neq h_{\boldsymbol{k}'}(f(\boldsymbol{c}^{(1)}))$ output $\perp$
        If $\boldsymbol{c}' \neq \mathsf{Comp}(f(\boldsymbol{c}^{(1)}))$ output $\perp$
    Output $\boldsymbol{m} = \mathsf{Rec}_2(f(\boldsymbol{c}^{(1)}))$
```

```
Hyb$^{\boldsymbol{m}}_F$:
    Compute $\boldsymbol{c}^{(1)} \leftarrow \mathsf{Sh}_2(\boldsymbol{m})$
    Sample $\boldsymbol{k} \leftarrow \mathbb{F}^a$
    Compute $\boldsymbol{z} = (\boldsymbol{k}, h_{\boldsymbol{k}}(\boldsymbol{c}^{(1)})), \mathsf{Comp}(\boldsymbol{c}^{(1)})$
    Compute $\boldsymbol{z}' \leftarrow \mathrm{Ideal}^{\boldsymbol{z}}_g$
    If $\boldsymbol{z}' = \perp$ output $\perp$
    Otherwise
        Parse $\boldsymbol{z}' = (\boldsymbol{k}', \boldsymbol{h}', \boldsymbol{c}')$
        If $\boldsymbol{h}' \neq h_{\boldsymbol{k}'}(f(\boldsymbol{c}^{(1)}))$ output $\perp$
        If $\boldsymbol{c}' \neq \mathsf{Comp}(f(\boldsymbol{c}^{(1)}))$ output $\perp$
    Output $\boldsymbol{m} = \mathsf{Rec}_2(f(\boldsymbol{c}^{(1)}))$
```

Fig. 7. On the right, the definition of the random variable $\mathrm{Hyb}^{\boldsymbol{m}}_F$ for an input message $\boldsymbol{m} \in \mathbb{F}^k$ and a tampering function $F = (f, g) \in \mathcal{F}^q_{1,n} \times \mathcal{F}$. On the left, for a quick reference, the random variable $\mathrm{Real}^{\boldsymbol{m}}_F$ (defined in Section 2) for the scheme $(\mathsf{ENC}_2, \mathsf{DEC}_2)$.

$O(n^{(1+\epsilon)/2})^\alpha = O(n)$. It follows from Theorem 3 that $(\mathsf{ENC}_2, \mathsf{DEC}_2)$ is (N, k)-NM with respect of the family $\mathcal{F}^q_{1,n} \times \mathcal{F}$ and that $N = n + b' = k + o(k)$. Moreover, since $t, r^2 = \Omega(k^{1-\epsilon})$ and b tends to infinity as k tends to infinity, the error written in Theorem 3 is negligible in k. Finally, since all the building blocks mentioned before are linear-time, then the computational complexity of the new scheme is $O(k) + O(b^\alpha) = O(k)$. □

Corollary 4 (Binary Case for Construction 2). *For any large enough k, there exists linear-time binary (N, k)-NM code with respect of the family $\mathcal{F}^2_{1,N}$ and with error negligible in k. Furthermore, $N = k + o(k)$.*

Acknowledgements. Ivan Damgård and Irene Giacomelli acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61361136003) for the Sino-Danish Center for the Theory of Interactive Computation and from the Center for Research in Foundations of Electronic Markets (CFEM), supported by the Danish Strategic Research Council. Ivan Damgård acknowledges support from the Advanced ERC grant MPCPRO.

References

1. Aggarwal, D., Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Optimal computational split-state non-malleable codes. In: Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II, pp. 393–417 (2016)

2. Aggarwal, D., Briët, J.: Revisiting the sanders-bogolyubov-ruzsa theorem in fp^{n} and its application to non-malleable codes. In: IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016, pp. 1322–1326 (2016)
3. Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: Non-malleable reductions and applications. In: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015, pp. 459–468 (2015)
4. Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC '14, pp. 774–783. ACM, New York, NY, USA (2014)
5. Aggarwal, D., Dziembowski, S., Kazana, T., Obremski, M.: Leakage-resilient non-malleable codes. In: Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I, pp. 398–426 (2015)
6. Aggarwal, D., Kazana, T., Obremski, M.: Inception makes non-malleable codes stronger. IACR Cryptology ePrint Archive **2015**, 1013 (2015)
7. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Explicit non-malleable codes against bit-wise tampering and permutations. In: Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I, pp. 538–557 (2015)
8. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In: Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I, pp. 375–397 (2015)
9. Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes for bounded depth, bounded fan-in circuits. In: Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II, pp. 881–908 (2016)
10. Capalbo, M.R., Reingold, O., Vadhan, S.P., Wigderson, A.: Randomness conductors and constant-degree lossless expanders. In: Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002, p. 15 (2002)
11. Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. In: 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, pp. 31:1–31:14 (2016)
12. Chandran, N., Kanukurthi, B., Raghuraman, S.: Information-theoretic local non-malleable codes and their applications. In: Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II, pp. 367–392 (2016)
13. Chattopadhyay, E., Goyal, V., Li, X.: Non-malleable extractors and codes, with their many tampered extensions. In: Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016, pp. 285–298 (2016)
14. Chattopadhyay, E., Zuckerman, D.: Non-malleable codes against constant split-state tampering. In: 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014, pp. 306–315 (2014)

15. Cheraghchi, M., Guruswami, V.: Capacity of non-malleable codes. In: Proceedings of the 5th Conference on Innovations in Theoretical Computer Science, ITCS '14, pp. 155–168. ACM, New York, NY, USA (2014)
16. Cheraghchi, M., Guruswami, V.: Non-malleable coding against bit-wise and split-state tampering. In: Theory of Cryptography, pp. 440–464. Springer (2014)
17. Cheraghchi, M., Guruswami, V.: Capacity of non-malleable codes. IEEE Trans. Information Theory **62**(3), 1097–1118 (2016)
18. Cheraghchi, M., Guruswami, V.: Non-malleable coding against bit-wise and split-state tampering. J. Cryptology **30**(1), 191–241 (2017)
19. Choi, S.G., Kiayias, A., Malkin, T.: Bitr: Built-in tamper resilience. In: D. Lee, X. Wang (eds.) Advances in Cryptology – ASIACRYPT 2011, *Lecture Notes in Computer Science*, vol. 7073, pp. 740–758. Springer Berlin Heidelberg (2011)
20. Chor, B., Goldreich, O., Hasted, J., Freidmann, J., Rudich, S., Smolensky, R.: The bit extraction problem or t-resilient functions. In: Foundations of Computer Science, 1985., 26th Annual Symposium on, pp. 396–407. IEEE (1985)
21. Christiani, T., Pagh, R.: Generating k-independent variables in constant time. In: 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS2014, Philadelphia, PA, USA, October 18-21, 2014, pp. 196–205 (2014)
22. Coretti, S., Dodis, Y., Tackmann, B., Venturi, D.: Non-malleable encryption: Simpler, shorter, stronger. In: Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I, pp. 306–335 (2016)
23. Coretti, S., Maurer, U., Tackmann, B., Venturi, D.: From single-bit to multi-bit public-key encryption via non-malleable codes. In: Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I, pp. 532–560 (2015)
24. Cramer, R., Damgård, I., Dottling, N., Fehr, S., Spini, G.: Linear secret sharing scheme from error correcting codes and universal hash function. In: Advances in Cryptology–EUROCRYPT 2015. Springer (2015)
25. Cramer, R., Damgård, I., Döttling, N., Giacomelli, I., Xing, C.: Linear-time non-malleable codes in the bit-wise independent tampering model. IACR Cryptology ePrint Archive 2016/397
26. Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Advances in Cryptology–EUROCRYPT 2008, pp. 471–488. Springer (2008)
27. Dachman-Soled, D., Kulkarni, M., Shahverdi, A.: Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. In: Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I, pp. 310–332 (2017)
28. Dachman-Soled, D., Liu, F., Shi, E., Zhou, H.: Locally decodable and updatable non-malleable codes and their applications. In: Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I, pp. 427–450 (2015)
29. Davì, F., Dziembowski, S., Venturi, D.: Leakage-resilient storage. In: J.A. Garay, R. De Prisco (eds.) Security and Cryptography for Networks, *Lecture Notes in Computer Science*, vol. 6280, pp. 121–137. Springer Berlin Heidelberg (2010)
30. Druk, E., Ishai, Y.: Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In: Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014, pp. 169–182 (2014)

31. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: R. Canetti, J.A. Garay (eds.) Advances in Cryptology – CRYPTO 2013, *Lecture Notes in Computer Science*, vol. 8043, pp. 239–257. Springer Berlin Heidelberg (2013)

32. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings, pp. 434–452 (2010)

33. Faonio, A., Nielsen, J.B.: Non-malleable codes with split-state refresh. In: Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I, pp. 279–309 (2017)

34. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. In: Y. Lindell (ed.) Theory of Cryptography, *Lecture Notes in Computer Science*, vol. 8349, pp. 465–488. Springer Berlin Heidelberg (2014)

35. Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In: P. Nguyen, E. Oswald (eds.) Advances in Cryptology – EUROCRYPT 2014, *Lecture Notes in Computer Science*, vol. 8441, pp. 111–128. Springer Berlin Heidelberg (2014)

36. Goldreich, O.: Modern Cryptography, Probabilistic Proofs and Pseudorandomness, *Algorithms and Combinatorics*, vol. 17. Springer (1998)

37. Goyal, V., Khurana, D., Sahai, A.: Breaking the three round barrier for non-malleable commitments. In: IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey,USA, pp. 21–30 (2016)

38. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016, pp. 1128–1141 (2016)

39. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: STOC, pp. 433–442 (2008)

40. Jafargholi, Z., Wichs, D.: Tamper detection and continuous non-malleable codes. In: Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I, pp. 451–480 (2015)

41. Kiayias, A., Liu, F., Tselekounis, Y.: Practical non-malleable codes from l-more extractable hash functions. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, pp. 1317–1328 (2016)

42. Li, X.: Improved non-malleable extractors, non-malleable codes and independent source extractors. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017, pp. 1144–1156 (2017)

43. Liu, F.H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: R. Safavi-Naini, R. Canetti (eds.) Advances in Cryptology – CRYPTO 2012, *Lecture Notes in Computer Science*, vol. 7417, pp. 517–532. Springer Berlin Heidelberg (2012)

44. Siegel, A.: On universal classes of extremely random constant-time hash functions. SIAM J. Comput. **33**(3), 505–543 (2004)

45. Tellegen, B.D.H.: A general network theorem, with applications. Philips Research Reports **7**, 259–269 (1952)

A Appendix

A.1 Proofs for Section 3

In Section 3.1 we build linear-time and constant-rate AMD codes using the family of linear uniform functions from [30] (Lemma2). The full proof of this construction can be found here.

Corollary 1 (Linear-Time and Constant-Rate AMD code) *For any large enough integer k, there exists a linear-time (k', k)-AMD code with error q^{-k} and $k' = \Theta(k)$.*

Proof. Given k, let $\mathcal{G}$ be the family from Lemma 2 with $c = 1$. For the sake of simplicity we consider separately the cases $b = 1$ and $b > 1$.
First, assume that $b = 1$ and define

$\mathsf{Enc}_{\mathsf{amd}}(\boldsymbol{m}) = (\boldsymbol{m}, \boldsymbol{k}, \boldsymbol{r}, g_{\boldsymbol{k}}(\boldsymbol{m}), g_{\boldsymbol{k}}(\boldsymbol{r}), g_{\boldsymbol{r}}(\boldsymbol{k}))$, where $\boldsymbol{k}, \boldsymbol{r} \in \mathbb{F}^k$ are chosen uniformly at random.

$$\mathsf{Dec}_{\mathsf{amd}}(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_6) = \begin{cases} \boldsymbol{v}_1 & \text{if } g_{\boldsymbol{v}_2}(\boldsymbol{v}_1) = \boldsymbol{v}_4, g_{\boldsymbol{v}_2}(\boldsymbol{v}_3) = \boldsymbol{v}_5, g_{\boldsymbol{v}_3}(\boldsymbol{v}_2) = \boldsymbol{v}_6 \\ \perp & \text{otherwise} \end{cases}$$

We will show that $(\mathsf{Enc}_{\mathsf{amd}}, \mathsf{Dec}_{\mathsf{amd}})$ is a $(6k, k)$-AMD code with error $\frac{1}{q^k}$. That is, given a non-zero error vector $\boldsymbol{e} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4, \boldsymbol{e}_5, \boldsymbol{e}_6) \in \mathbb{F}^{6k}$, we will prove that $\Pr[\mathsf{Dec}_{\mathsf{amd}}(\mathsf{Enc}_{\mathsf{amd}}(\boldsymbol{m}) + \boldsymbol{e}) \neq \perp] \leq \frac{1}{q^k}$. For this purpose notice that

$$\mathsf{Dec}_{\mathsf{amd}}(\mathsf{Enc}_{\mathsf{amd}}(\boldsymbol{m}) + \boldsymbol{e}) = \mathsf{Dec}(\boldsymbol{m} + \boldsymbol{e}_1, \boldsymbol{k} + \boldsymbol{e}_2, \boldsymbol{r} + \boldsymbol{e}_3, g_{\boldsymbol{k}}(\boldsymbol{m}) + \boldsymbol{e}_4, g_{\boldsymbol{k}}(\boldsymbol{r}) + \boldsymbol{e}_5, g_{\boldsymbol{r}}(\boldsymbol{k}) + \boldsymbol{e}_6)$$

and that $\Pr[\mathsf{Dec}_{\mathsf{amd}}(\mathsf{Enc}_{\mathsf{amd}}(\boldsymbol{m}) + \boldsymbol{e}) \neq \perp]$ is equal to the probability that the following equations are all satisfied:

$$\begin{cases} g_{\boldsymbol{k}+\boldsymbol{e}_2}(\boldsymbol{m} + \boldsymbol{e}_1) = g_{\boldsymbol{k}}(\boldsymbol{m}) + \boldsymbol{e}_4 \\ g_{\boldsymbol{k}+\boldsymbol{e}_2}(\boldsymbol{r} + \boldsymbol{e}_3) = g_{\boldsymbol{k}}(\boldsymbol{r}) + \boldsymbol{e}_5 \\ g_{\boldsymbol{r}+\boldsymbol{e}_3}(\boldsymbol{k} + \boldsymbol{e}_2) = g_{\boldsymbol{r}}(\boldsymbol{k}) + \boldsymbol{e}_6 \end{cases}$$

The above system is equivalent to

$$\begin{cases} g_{\boldsymbol{k}}(\boldsymbol{e}_1) = \boldsymbol{e}_4 - g_{\boldsymbol{e}_2}(\boldsymbol{m} + \boldsymbol{e}_1) \\ g_{\boldsymbol{k}}(\boldsymbol{e}_3) = \boldsymbol{e}_5 - g_{\boldsymbol{e}_2}(\boldsymbol{r} + \boldsymbol{e}_3) \\ g_{\boldsymbol{r}}(\boldsymbol{e}_2) = \boldsymbol{e}_6 - g_{\boldsymbol{e}_3}(\boldsymbol{k} + \boldsymbol{e}_2) \end{cases} \tag{1}$$

If at least one among the vectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ is different from zero (w.l.o.g. assume that $\boldsymbol{e}_1 \neq 0$) then

$$\Pr[\mathsf{Dec}_{\mathsf{amd}}(\mathsf{Enc}_{\mathsf{amd}}(\boldsymbol{m}) + \boldsymbol{e}) \neq \perp] \leq \Pr[g_{\boldsymbol{k}}(\boldsymbol{e}_1) = \boldsymbol{e}_4 - g_{\boldsymbol{e}_2}(\boldsymbol{m} + \boldsymbol{e}_1)] = 1/q^k$$

where that the last inequality holds as $\mathcal{G}$ is a linear uniform family (property 3 in Lemma 2). On the other hand, if $\boldsymbol{e}_1 = \boldsymbol{e}_2 = \boldsymbol{e}_3 = 0$, then system (1) is

satisfied if and only if also $\boldsymbol{e}_4 = \boldsymbol{e}_5 = \boldsymbol{e}_6 = 0$. But this situation is not possible since $\boldsymbol{e} \neq \mathbf{0}$. Thus, the proof in this first case is concluded.
If $b > 1$, the previous construction is still possible, but with worse rate. To see this, split both the vectors $\boldsymbol{r}$ and $\boldsymbol{k}$ in b pieces of length k and, in the encoding algorithm $\mathsf{Enc}_{\mathsf{amd}}$, substitute the vector $g_{\boldsymbol{k}}(\boldsymbol{r})$ with the vectors $g_{\boldsymbol{k}}(\boldsymbol{r}_1), \ldots, g_{\boldsymbol{k}}(\boldsymbol{r}_b)$ and the vector $g_{\boldsymbol{r}}(\boldsymbol{k})$ with the vectors $g_{\boldsymbol{r}}(\boldsymbol{k}_1), \ldots, g_{\boldsymbol{r}}(\boldsymbol{k}_b)$, respectively. It is easy to verify that in this case we obtain an (k', k)-AMD code with $k' = (3+2b)k$ and error $\frac{1}{q^k}$. By inspection, the computational complexity of the scheme is $O(k)$. □

Notice that for Construction 1 we need a "strong" AMD code (that is for any $\boldsymbol{m}$ and any non-zero $\boldsymbol{e}$, it holds that $\Pr[\mathsf{Dec}(\mathsf{Enc}(\boldsymbol{m}) + \boldsymbol{e}) \neq \perp] \leq \epsilon$). In the literature, there exists also another (weaker) notion of AMD codes: for any $\boldsymbol{m}$ and any $\boldsymbol{e}$, it holds that $\Pr[\mathsf{Dec}(\mathsf{Enc}(\boldsymbol{m}) + \boldsymbol{e}) \notin \{\perp, \boldsymbol{m}\}] \leq \epsilon$. If the latter is the intended definition, then our construction from Corollary 1 could be simplified as follows.

$$\mathsf{Enc}_{\mathsf{amd}}(\boldsymbol{m}) = (\boldsymbol{m}, \boldsymbol{k}, g_{\boldsymbol{k}}(\boldsymbol{m})), \text{ where } \boldsymbol{k} \leftarrow \mathbb{F}^k$$

$$\mathsf{Dec}_{\mathsf{amd}}(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3) = \begin{cases} \boldsymbol{v}_1 & \text{if } g_{\boldsymbol{v}_2}(\boldsymbol{v}_1) = \boldsymbol{v}_3 \\ \perp & \text{otherwise} \end{cases}$$

A.2 Proofs for Section 4

We provide here a t-wise independence generator with seed-length and independence sub-linear in the output length. The construction combines results of Christiani and Pagh [21] and Siegel [44]. Note that the parameter regime we are interested in here differs from that in [21].

Definition 8 (Unique Neighbour Expander Graph). *Let $\Gamma = (L, R, E)$ be a finite undirected bipartite graph, with left-vertex set L, right-vertex set R and edge set E. Let n, m, d, e be positive integers. Then Γ is an (n, m, d, e)-*unique neighbour expander *if the following holds. First, $|L| = n$ and $|R| = m$ and each vertex $v \in L$ has degree d. Second, for each set $S \subseteq L$ of size at most e, there exists a vertex $v \in R$ that has a unique neighbour in S.*

Siegel [44] showed how such an expander can be used to extend the output length of an independence generator at a constant factor loss in the independence. Precisely:

Lemma 10 (Lemma 2.6, Corollary 2.11 in [44]). *Let $\Gamma = (L, R, E)$ be a (n, m, d, e)-unique neighbour expander. Then there exists a $\mathbb{F}$-linear function $\mathsf{Expand}_\Gamma : \mathbb{F}^m \to \mathbb{F}^n$ such that the following holds: if X is a (de)-wise uniform random variable over $\mathbb{F}^m$, then $\mathsf{Expand}_\Gamma(X)$ is an e-wise uniform random variable on $\mathbb{F}^n$. Moreover, Expand_Γ has computational complexity $O(n)$.*

Proof. Given Γ as in the lemma, the function $\mathsf{Expand}_\Gamma : \mathbb{F}^m \to \mathbb{F}^n$, $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \mapsto (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ is defined by

$$\boldsymbol{y}_i = \sum_{j \in \Gamma(i)} \boldsymbol{x}_i$$

where $\Gamma(i) \subseteq R$ indicates the neighbours of $i \in L$.

Assume that $A = \{a_1, \ldots, a_e\}$ is a subset of $[n]$ of size e. Since Γ be a (n, m, d, e)-unique neighbour expander, there is $v_1 \in R$ that has a unique neighbour in A, wlog we can assume that this neighbour is a_1. Now we consider $A \setminus \{a_1\}$ and applying the definition again we obtain $v_1 \in R$ that has a unique neighbour a_2 in $A \setminus \{a_1\}$. Continuing in this way we can prove that for any $i \in [e]$ there is $v_i \in R$ such that $v_i \in \Gamma(a_i)$ and $v_i \notin \Gamma(a_j)$ for all $j \geq i+1$.

Now let $X = (X_1, \ldots, X_m)$ be a (de)-wise uniform random variable over $\mathbb{F}^m$ and $Y = (Y_1, \ldots, Y_n) = \mathsf{Expand}_\Gamma(X)$. Consider Y_A, because of the previous observation, we have that Y_{a_i} is independent of $(Y_{a_{i+1}}, \ldots, Y_{a_e})$ for any $i = 1, \ldots, e-1$. Indeed, $Y_{a_i} = X_{v_i} + Z_i$ for some uniform random variable Z_i and $(Y_{a_{i+1}}, \ldots, Y_{a_e})$ is trivially independent of X_{v_i} because $v_i \notin \Gamma(a_j)$ for all $j \geq i+1$. It follows by induction that $Y_{a_1}, \ldots, Y_{a_e}$ are independent. Therefore, Y_A has the uniform distribution on $\mathbb{F}^e$. □

In order to be able to use this lemma iteratively, as we will do shortly, an explicit finite family $\{\Gamma_i\}_i$ is required such that $n_i = m_{i+1}$ and $e_i = de_{i+1}$ for all indices i. Christiani and Pagh [21] observed that a construction of Capalbo et al. (Theorem 7.1 in [10]) in fact has this property.

Lemma 11 (Lemma 3 in [21]). *For each positive integer c, there are positive integers d and m' and a real number α such that the following holds. For any integer $m \geq m'$ there exists a (cm, m, d, e)-unique neighbour expander Γ with $e \geq \alpha m/d$. The construction of Γ is explicit, i.e. Γ can be constructed in time* $\mathsf{poly}(m)$.

These results give immediate rise to the t-wise independence generator, which will be used in Lemma 6 to construct the secret-sharing scheme we require to implement our Construction 2. Note that the generator we construct here is $\mathbb{F}$-linear.

Lemma 12. *For each real number $\epsilon \in (0,1)$, there exists a real number $\delta \in (0, \epsilon)$ such that the following holds. For any sufficiently large integer k there exists an explicit t-wise independence generator* $\mathsf{Gen} : \mathbb{F}^{k'} \to \mathbb{F}^k$, *where $t = \Omega(k^{1-\epsilon})$ and $k' = \Theta(k^{1-\delta})$. Moreover,* Gen *is a $\mathbb{F}$-linear function and has computational complexity $O(k)$.*

Proof. For concreteness, set $c = 2$ in Lemma 11 and let d, m' and α be as given in that lemma. Let ℓ and t be positive integers with $\ell \geq 1$ and $t \geq \max\{1/\alpha, m'\}$ and define $m_i := 2^i d^\ell t$ for any integer $i \geq 0$; notice that $m_i = 2m_{i+1}$. Since $m_i \geq t \geq m'$ for any $i \geq 0$, Lemma 11 implies that there exists an explicit family $\{\Gamma_i\}_{i \geq 0}$ where each Γ_i is a $(2m_i, m_i, d, e_i)$-unique neighbour expander with $e_i \geq \alpha m_i/d = \alpha 2^i d^{\ell-1} t$. For any $0 \leq i \leq \ell - 1$ define $e'_i := d^{\ell-1-i}\lfloor \alpha t \rfloor$ and notice that $e'_i \geq 1$ and $e'_i = de'_{i+1}$; moreover, since $e'_i \leq \alpha d^{\ell-1-i} t \leq \alpha 2^i d^{\ell-1} t \leq e_i$, the graph Γ_i is also a $(2m_i, m_i, d, e'_i)$-unique neighbour expander. Therefore the family $\{\Gamma_i\}_{i=0,\ldots,\ell-1}$ has the required parameters.

Now start with a de'_0-wise uniform random variable on $\mathbb{F}^{m_0} = \mathbb{F}^{d^\ell t}$ (e.g. taking the uniform random variable on $\mathbb{F}^{m_0}$) and apply the map $\mathsf{Expand}_{\Gamma_i}$ from Lemma

10 for all $0 \leq i \leq \ell - 1$. In this way we obtain a $\lfloor \alpha t \rfloor$-wise independence generator $\mathsf{Gen} : \mathbb{F}^{d^\ell t} \to \mathbb{F}^{(2d)^\ell t}$ with computational complexity $\sum_{i=0}^{\ell-1} O(2m_i) = O((2d)^\ell t)$.

Finally, given ϵ and k as in the statement of the lemma, we choose $t = \lceil k^{1-\epsilon} \rceil$ and $\ell = \lceil \epsilon \log_{2d} k \rceil$. Notice that for k large enough $t \geq \max\{1/\alpha, m'\}$, $\ell \geq 1$ and moreover, the output length of the generator satisfies $(2d)^\ell t \geq (2d)^{\epsilon \log_{2d} k} k^{1-\epsilon} = k$ and $(2d)^\ell < (2d)(2d)^{\epsilon \log_{2d} k}(k^{1-\epsilon}+1) = 2d(k+k^\epsilon)$. Therefore, after truncating if necessary the original output of Gen we obtain a t-wise independence generator of output of length k and computational complexity $O(k)$. Write $z = \log_{2d} d$. The seed length k' satisfies $k' = d^\ell t = (2d)^{z\ell} t < (2d)^{z(1+\epsilon \log_{2d} k)}(1 + k^{1-\epsilon}) = (2d)^z k^{z\epsilon}(1 + k^{1-\epsilon})$, which is of the order $k^{1-(1-z)\epsilon}$. Moreover, $k' = d^\ell t = (2d)^{z\ell} t \geq (2d)^{z\epsilon \log_{2d} k} k^{1-\epsilon} = k^{z\epsilon} k^{1-\epsilon}$. Thus, choosing $\delta = (1-z)\epsilon$ concludes the proof. □

Lemma 9 (Linear-Time Universal Family) *For any real number $\beta \in (0,1)$ and any positive integer n, there exists a μ-universal family $\mathcal{H} = \{h_{\boldsymbol{k}} : \mathbb{F}^n \to \mathbb{F}^m\}_{\boldsymbol{k} \in \mathbb{F}^a}$ with $a = o(n)$, $m = o(n)$ and $\mu = \Theta(q^{-n^{(1-\beta)}})$. Moreover, each function $h_{\boldsymbol{k}}$ has computational complexity $O(n)$.*

Proof. Given $\beta \in (0,1)$ and $n \geq 1$, define $k = \lfloor n^{1-\beta/2} \rfloor$ and $k' = \lfloor n^{1-\beta} \rfloor$. It is immediate to verify that in Lemma 2 the range dimension ck of the linear uniform family $\mathcal{G}$ may be replaced by $k' \leq k$ and the result still holds. Therefore, we can assume that there exist a positive integer b and μ-almost universal family $\mathcal{G} = \{g_{\boldsymbol{k}} : \mathbb{F}^k \to \mathbb{F}^{k'}\}_{\boldsymbol{k} \in \mathbb{F}^{bk}}$ with $\mu = 1/q^{k'}$. Moreover, $g_{\boldsymbol{k}}$ has computational complexity $O(k)$. Now define $n' = \lceil n/k \rceil$ and $h_{\boldsymbol{k}} : \mathbb{F}^n \to \mathbb{F}^{k'n'}$ as follows:

$$h_{\boldsymbol{k}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = (g_{\boldsymbol{k}}(\boldsymbol{y}_1), \ldots, g_{\boldsymbol{k}}(\boldsymbol{y}_{n'}))$$

where[18] $\boldsymbol{y}_i = (\boldsymbol{x}_{(i-1)k+1}, \boldsymbol{x}_{(i-1)k+2}, \ldots, \boldsymbol{x}_{ik})$ for any $i = 1, 2, \ldots, n'$.
Define $m = k'n'$ and $a = bk$. Then $0 < m < n^{1-\beta}(n/k+1)$, which has order $n^{1-\beta/2}$, and $0 < a \leq bn^{1-\beta/2}$. The computational complexity of $h_{\boldsymbol{k}}$ is $n'O(k) = O(n)$. Finally, for any distinct $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{F}^n$ there is $i \in [n']$ such that $\boldsymbol{y}_i \neq \boldsymbol{y}'_i$. Then, if $\boldsymbol{k}$ is chosen uniformly at random from $\mathbb{F}^a$

$$\Pr[h_{\boldsymbol{k}}(\boldsymbol{x}) = h_{\boldsymbol{k}}(\boldsymbol{x}')] \leq \Pr[g_{\boldsymbol{k}}(\boldsymbol{y}_i) = g_{\boldsymbol{k}}(\boldsymbol{y}'_i)] \leq 1/q^{k'}$$

□

A.3 Tellegen's Principle

We will briefly discuss a technique know as Tellegen's principle. Assume that we are given a linear algorithm T computing the function $f(\boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{A}$, where $\boldsymbol{A}$ is a $m \times n$ matrix over some ring R and $\boldsymbol{x}$ is a vector from R^n. Then we can transform T into an algorithm T' computing the function $f'(\boldsymbol{y}) = \boldsymbol{y} \cdot \boldsymbol{A}^\top$, where $\boldsymbol{y} \in R^m$ and $\boldsymbol{A}^\top$ is the transpose of the matrix $\boldsymbol{A}$, which has the same computational

[18] Notice that $n'k \geq n$. If $n'k > n$, the vector $\boldsymbol{y}_{n'}$ is obtained from the last components of $\boldsymbol{x}$ padded with zeros.

complexity as T. We will discuss this transformation for arithmetic circuits. We can decompose a circuit into a sequence of elementary instructions ϕ_i, where each ϕ_i is a linear transformation on all the wires. We can thus write the matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} = \phi_n \cdot \phi_{n-1} \cdots \phi_2 \cdot \phi_1.$$

Transposing $\boldsymbol{A}$ immediately yields

$$\boldsymbol{A}^\top = \phi_1^\top \cdot \phi_2^\top \cdots \phi_{n-1}^\top \cdot \phi_n^\top.$$

Thus, we only have to consider the effect of transposition to the elementary instructions ϕ_i.

- Instruction ϕ_i multiplies a wire $\boldsymbol{x}$ with a constant $\alpha \in R$ and writes the output in the same register. In this case $\phi_i^\top = \phi_i$, as the transformation matrix ϕ_i is diagonal and thus symmetric.
- Instruction ϕ_i adds wire $\boldsymbol{y}$ to wire $\boldsymbol{x}$. In this case $\phi_i^\top$ adds wire $\boldsymbol{x}$ to wire $\boldsymbol{y}$.

These two instructions are sufficient to implement any linear transformation. For instance, to clear an (auxiliary) register, simply multiply it by 0. We summarize this in the following Lemma.

Lemma 13 (Tellegen's Principle [45]). *Let $\mathsf{T}(\boldsymbol{x})$ be a linear arithmetic circuit or linear RAM algorithm computing the function $\boldsymbol{x} \cdot \boldsymbol{A}$. Then there exists a linear arithmetic circuit $\mathsf{T}'(\boldsymbol{y})$ that computes the function $\boldsymbol{y} \cdot \boldsymbol{A}^\top$ and has the same computational complexity as T.*