Non-malleable Codes from Authenticated Encryption in Split-State Model

Anit Kumar Ghosal [1] and Dipanwita Roychowdhury [1]

Department of Computer Science and Engineering, IIT Kharagpur, India

Abstract. The secret key of any encryption scheme that are stored in secure memory of the hardwired devices can be tampered using fault attacks. The usefulness of tampering attack is to recover the key by altering some regions of the memory. Such attack may also appear when the device is stolen or viruses has been introduced. Non-malleable codes are used to protect the secret information from tampering attacks. The secret key can be encoded using non-malleable codes rather than storing it in plain form. An adversary can apply some arbitrary tampering function on the encoded message but it guarantees that output is either completely unrelated or original message. In this work, we propose a computationally secure non-malleable code from leakage resilient authenticated encryption along with 1-more extractable hash function in split-state model with no common reference string (CRS) based trusted setup. Earlier constructions of non-malleable code cannot handle the situation when an adversary has access to some arbitrary decryption leakage (i.e., during decoding of the codeword) function to get partial information about the codeword. In this scenario, the proposed construction is capable of handling such decryption leakages along with tampering attacks.

Keywords: Authenticated encryption · Non-malleable codes · Split-state model · Tamper resilient cryptography.

1 Introduction

In cryptography, the security of various primitives like encryption algorithms, message authentication codes (MACs), digital signatures follow a generic template, i.e., the adversary can observe input-output behaviour, and further it can attack the primitives. Usually, the basic assumption is that an adversary is able to learn input-output behaviour only, and the security of cryptographic primitives preserve as long as secret key remains confidential. However, if an adversary gains some partial information about the secret keys, the cryptosystem can be broken easily. The adversary can mount Differential Fault Analysis attack, Template Attack, Timing Attack etc. on the cryptosystem to learn the secret information about keys. In reality, such model does not provide security guarantee when an adversary can tamper the secret key (e.g., flip some bits) and it can analyse the behaviour of the cryptosystem on these tampered keys to break it

completely [1, 5]. Non-malleable codes (NMCs), introduced by Dziembowski et al. [10, 22], are used to protect secret message against related tampering, i.e., the adversary cannot tamper the codeword to make another codeword of a related message. More precisely, an adversary can apply the tampering function on the codeword, and the guarantee is that, encoded message either remains completely unchanged or essentially destroyed, i.e., $\perp$. So, the attack on the tampered codeword is now rendered useless and an adversary can not break the security of the cryptosystem. Usually, NMCs are built for the specific classes of tampering functions. Let us consider a secret message d. Non-malleable code is used to encode the message d and it generates the output as Enc(d). An adversary can apply some tampering function $f_{increment}$ on the message encoded by non-malleable code as $f_{increment}$(Enc(d)). The output of such tampering experiment produces result Enc(d)+1. Further, the adversary decodes the message in the following way Dec(Enc(d)+1). The final result of the experiment is d+1, and it has some relation with the original secret message. Therefore, NMCs can be designed for the specific classes of tampering functions only. The most widely used class of tampering function is the split-state model where the codeword is divided into two different parts and stored into the memory $\mathcal{M}_L$, $\mathcal{M}_R$. Two different functions f_L and f_R act independently but arbitrarily on the codeword [12, 14, 17, 18, 20, 23, 24, 26].

Scheme	Codeword length	Model	Assumption	Security against leakages
[15]	$\mathcal{O}((\|m\|+k)^{\tau}\log^{\tau}(\|m\|+k))$	Information theoretic	NA	Encryption leakages
[17]	$\mathcal{O}(max(\|m\|,k))$	Information theoretic	NA	Encryption leakages
[15] + [20]	$\|m\|+\mathcal{O}(k^7)$	Computational	Authenticated Encryption	Encryption leakages
[6] + [8] + [12] + [20]	$\|m\|+\mathcal{O}(k^2)$	Computational, CRS	Leakage resilient PKE + Robust NIZK	Encryption leakages
[18]	$\|m\|+18k$	Computational, CRS	One-time leakage resilient AE + KEA	Encryption leakages
[23]	$\|m\|+2k$	Computational	Pseudorandom permutation with leakage + Fixed related-key	Encryption leakages
[25]	$\|m\|+5k$	Computational	Entropic fixed related-key	Encryption leakages
[26]	$\|m\|+2k$	Computational	Related-key + PRP with leakage + Leakage resilient CBC-MAC unforgeability	Encryption leakages
Our work	$\|m\|+2k+2log^2(k)$ [1]	Computational	Leakage resilient AE with decryption leakages + 1-more extractable hash	Encryption + Decryption leakages

Table 1: Comparative results of various multi-bit non-malleable codes in the split-state model [18, 23, 26].

Usually, constructions of NMCs can be divided broadly into two domains as information-theoretic [14] and computational [20]. In [15], authors show a construction of non-malleable code of length $\mathcal{O}((|m|+k)^7 log^7(|m|+k))$ in information-theoretic domain, where m and k represent the message length and security parameter respectively. In computational domain, the construction of non-malleable code is shown from public key primitives [8] with robust non interactive Zero knowledge (NIZK) [2] proof [12]. Further, the length of codeword is optimized into $|m|+\mathcal{O}(k^7)$ by combining the idea of [15] and [20]. Subsequently, the size of the codeword is reduced to $|m|+\mathcal{O}(k^2)$ [6, 8, 12, 20]. Usually, non-malleable codes are keyless encoding scheme. Further research work shows that

[1] k is the security parameter.

such codeword can be constructed from symmetric-key primitives, i.e., authenticated encryption, block cipher etc. [18,23,26]. In [18], authors show non-malleable codes from one-time leakage-resilient authenticated encryption along with l-more extractable hash function, and the non-standard assumption, called knowledge of exponent assumption (KEA) is used to prove the security in common reference string (CRS) model. The security margin of [18] is extremely high but the construction is based on CRS which should be generated from honest, trusted parties. It is difficult to manage such CRS setup in practical situation. Later, Fehr et al. [23] show NMCs of optimal codeword length from related-key secure block ciphers such as AES [3], SHACAL-2 [4] without any CRS based trusted setup. Unfortunately, their assumptions are not applicable when the tampering function is cipher-dependent. Hence, a better construction [25] is proposed using entropic fixed-related-key security notion but the length of codeword is not optimal. Recently, Ghosal et al. [26] show a construction of non-malleable code of optimal codeword length from a specific authenticated encryption.

Motivation of our work. Till now, all of the constructions of NMCs are capable of handling encryption leakages but not the leakage during decryption. An adversary can use arbitrary leakage function during decryption and it can obtain partial information about the codeword that compromises the security of the underlying codeword. Since the codeword length of a non-malleable code is denoted as $|m|$[2] + |security-parameter| and the value of such security-parameter is heavily dependent on the maximum amount of leakage supported by the underlying primitive. The goal of our work is not to optimize the codeword length as already optimal length NMCs are available in the literature [23, 26]. Apart from tampering attacks, we want to protect the codeword from encryption and decryption leakages. The main challenge of designing such codeword is to provide security against one-time tampering attack but maintaining the length of codeword optimum as much as possible. In short, the objective of our work is mentioned as follows:

(a) To construct non-malleable codes from authenticated encryption along with 1-more extractable hash function that can handle encryption leakages as well as decryption leakages.
(b) The codeword provides security against one-time tampering attack when an adversary has access to limited bits of leakage.

Our Contribution. In this work, we construct a computationally secure non-malleable code from authenticated encryption [21] along with extractable hash function [18] in split-state model. Though NMCs protect message only for tampering attacks but leakage of information from the stored codeword or leakages during encoding and decoding operation, compromises the security of non-malleable code. Our proposed construction is secure against an adversary with access to some arbitrary leakage function during the decoding step, (i.e., decryption leakages) and the adversary cannot gain much useful information from the codeword. The resulting codeword has length of $|m| + 2k + 2log^2k$ compared to other codewords as shown in Table 1.

[2] $|m|$, $|k|$ denote the message length and security parameter respectively.

Organization. The paper is arranged in the following way. The basics of non-malleable code and various definitions are discussed in Section 2. There after Section 3 illustrates the construction of codeword, while proof of security is explained in Section 4. Finally, Section 5 concludes our work.

2 Preliminaries

Basic Notations. In split-state model, left and right half of the memories are represented by $\mathcal{M}_L$ and $\mathcal{M}_R$. Two tampering functions f_L and f_R are chosen arbitrarily by an adversary working in $\mathcal{M}_L$ and $\mathcal{M}_R$ respectively. $\mathcal{SK}$ denotes the usable key set. If $\mathcal{SK}$ is the key set, $|\mathcal{SK}|$ is the maximum number of key elements in $\mathcal{SK}$. When sk is uniformly chosen at random from $\mathcal{SK}$, we write $sk \xleftarrow{\$} \mathcal{SK}$. Message set is represented by M whereas $\mathcal{C}$ denotes the codeword set. k is the security parameter. m, e and c denote the plain message, encoded message and codeword (m, e, $c \in \{0,1\}^k$). m can be broken into small chunks m_1, m_2 etc. and similarly, e. We denote m^1, m^2 as two different messages. υ denotes the leakage function. A function $\epsilon(k)$ is called negligible in k if it vanishes faster than the inverse of any polynomial in k.

Definition 2.1 (Coding Scheme and Non-malleable Codes). Let (Enc_{sk} , Dec) be a split-state coding scheme. The encoding algorithm Enc_{sk} takes input a message $m \in M$, key $sk \in \mathcal{SK}$ and it outputs a codeword $c \in \mathcal{C}$, divided into $\mathcal{M}_L$ and $\mathcal{M}_R$ parts of the memory respectively. The decoding algorithm Dec takes input a codeword stored in $\mathcal{M}_R$ and $\mathcal{M}_L$, outputs the plain message $m \in M$. Moreover, let $\mathcal{F}$ be a family of tampering functions. The coding scheme is said to satisfy non-malleability if for each tampering function $f \in \mathcal{F}$ and codeword $c \in \mathcal{C}$, the $Dec(f(c))$ produces $m^{'}$, where $m^{'}$ can be m, $\perp$ or completely unrelated to m. The split-state coding scheme which satisfies the non-malleability property is said to be non-malleable codes.

Definition 2.2 (Strong Non-malleability). Let $\mathcal{F}$ be some family of tampering functions. The tampering experiment for each $f = (f_L, f_R) \in \mathcal{F}$ and $m \in M$ is defined as follows:

$$\mathbf{Tamper}^{f}_{m} = \left\{ \begin{array}{c} c \leftarrow Enc_{sk}(m), c = \{\mathcal{M}_L, \mathcal{M}_R\} \\ \{\mathcal{M}^{'}{}_L, \mathcal{M}^{'}{}_R\} = \{f_L(\mathcal{M}_L), f_R(\mathcal{M}_R)\} \\ c^{'} = \{\mathcal{M}^{'}{}_L, \mathcal{M}^{'}{}_R\}, m^{'} = Dec(c^{'}) \\ output : same^{*}, if c^{'} = c, else\ m^{'}. \end{array} \right\},$$

where randomness comes from the encoding algorithm.

We say that the coding scheme (Enc_{sk}, Dec) is strongly non-malleable with respect to some tampering function family $\mathcal{F}$ if the following indistinguishability $\mathbf{Tamper}^{f}_{m^0}(m^0) \underset{c}{\approx} \mathbf{Tamper}^{f}_{m^1}(m^1)$ holds for two arbitrarily chosen messages m^0 and m^1.

Definition 2.3 (Extractable Hash). A hash function H_k is said to be extractable hash [13] if for any PPT algorithm $\mathcal{A}$, there exists a PPT extractor

[1] k denotes the security parameter

$\mathcal{E}_{\mathcal{A}}$, such that for all $k \in \mathcal{N}$, for any input $p \in \{0,1\}^k$ and a negligible function $\epsilon(k)$:

$Pr_{h \leftarrow H_k}[y \leftarrow \mathcal{A}(h,p), \exists x : h(x) = y, x^{'} \leftarrow \mathcal{E}_{\mathcal{A}}(h,p) \wedge h(x^{'}) \neq y] \leq \epsilon(k)$.

Definition 2.4 (1-more Extractable Hash). A hash function H_k is said to be 1-more extractable hash [18] if for any PPT algorithm $\mathcal{A}_v$ and any $p_v \in \{0,1\}^k$, there exists a PPT extractor $\mathcal{E}_v$ and $p_{\mathcal{E}} \in \{0,1\}^k$, such that for all PPT algorithm $\mathcal{A}_s$, $k \in \mathcal{N}$, for any input message $s \in \{0,1\}^k$ and a negligible function $\epsilon(k)$:

$Pr_{h_z \leftarrow H_k}[\mathbf{Exp}^{s,h_z}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_v}(1,p_v,p_{\mathcal{E}}) = 1] \leq \epsilon(k)$,

where $\mathbf{Exp}^{s,h_z}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_v}(1,p_v,p_{\mathcal{E}})$ should satisfy all the four properties [18] as follows:

- (**Hash computation**) : $(h_1) \leftarrow h_z(sk)$
- (**Hash tampering**) : $(h_1^{'}) \leftarrow \mathcal{A}_v(h_z, h_1, p_v)$
- (**Preimage extraction**) : $(\hat{sk}) \leftarrow \mathcal{E}_v(h_z, h_1, p_{\mathcal{E}})$
- (**Preimage tampering**) : $(sk') \leftarrow \mathcal{A}_s(h_z, sk)$

If $h_z(sk^{'}) = h_1^{'} \wedge h_z(\hat{sk}) \neq h_1^{'}$, return 1
else, return 0

The experiment $\mathbf{Exp}^{s,h_z}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_v}(1,p_v,p_{\mathcal{E}})$ works as follows. We use deterministic hash function with no randomness. An adversary $\mathcal{A}_v$ tries to produce a tampered hash $h_1^{'}$, given a hash value h_1 with auxiliary information p_v. Then, extractor $\mathcal{E}_v$ is used to extract the preimage, given h_1 and auxiliary input $p_{\mathcal{E}}$. Finally, an adversary $\mathcal{A}_s$ tries to produce preimage of $h_1^{'}$ with all the information collected in the execution. The experiment outputs 1, if $\mathcal{A}_v$ is able to generate a valid hash $h_1^{'}$, and $\mathcal{A}_s$ produces a valid preimage of $h_1^{'}$, while the extractor algorithm fails.

Definition 2.5 (Pseudorandom Function). Let $F : \mathcal{SK} \times M \rightarrow \mathcal{T}$ be a deterministic function. We say that F is a pseudorandom function (PRF) if for all (q,t) bounded adversary A, the below advantage holds:

$$Pr[A^{F_{sk}(.)} = 1] \text{ - } Pr[A^{f(.)} = 1] \leq \epsilon(k),$$

where $sk \xleftarrow{\$} \mathcal{SK}$, $\epsilon(k)$ denotes a negligible function and the function $f : M \rightarrow \mathcal{T}$ is chosen uniformly at random.

Definition 2.6 (Strong Pseudorandom Function). Let $F : \mathcal{SK} \times M \rightarrow \mathcal{T}$ be a deterministic function. We say that F is a strong pseudorandom function (SPRF) if for all (q,t) bounded adversary A with oracle access to the function and its inverse, the below advantage holds:

$$Pr[A^{F_{sk}(.),F_{sk}^{-1}(.)} = 1] \text{ - } Pr[A^{f(.),f^{-1}(.)} = 1] \leq \epsilon_F(k),$$

where $sk \xleftarrow{\$} \mathcal{SK}$, $\epsilon_F(k)$ denotes a negligible function and the function $f : M \rightarrow \mathcal{T}$ is chosen uniformly at random.

Similarly, a deterministic function F is said to be (q,t,ϵ_F) pseudorandom permutation (PRP) if for all sk, F_{sk} is a permutation, and the above advantage is upper bounded by ϵ_F. The function f is selected among the permutation on $M = \mathcal{T}$ uniformly at random. In case of strong pseudorandom permutation(SPRP), the adversary A has oracle access to the function and its inverse.

Definition 2.7 (Tweakable Pseudorandom Function). Let $F^* : \mathcal{SK} \times \mathcal{TK} \times M \rightarrow \mathcal{T}$ be a deterministic function. It is said to be (q,t,ϵ_{F^*}) tweakable

pseudorandom function (TPRF) if for all (q, t) bounded adversary A, the below advantage holds:

$$Pr[A^{F^{*,(.)}_{sk}(.)} = 1] - Pr[A^{f^{(.)}(.)} = 1] \leq \epsilon_{F^*(k)},$$

where $sk \xleftarrow{\$} \mathcal{SK}$ is selected uniformly at random and f is chosen from the set of functions $\mathcal{TK} \times M \to \mathcal{T}$ uniformly at random.

Definition 2.8 (Strong Tweakable Pseudorandom Function). Let $F^* : \mathcal{SK} \times \mathcal{TK} \times M \to M$ be a deterministic function. We say that F^* is a (q, t, ϵ_{F^*}) strong tweakable pseudorandom function (STPRF) if for all sk, tweaks $F^{*,\mathcal{TK}}_{sk} : M \to M$ and for all (q, t) bounded adversary A that has oracle access to the function and its inverse, the below advantage holds:

$$Pr[A^{F^{*,(.)}_{sk}(.),F^{*,-1,(.)}_{sk}(.)} = 1] - Pr[A^{f^{(.)}(.),f^{-1,(.)}(.)} = 1] \leq \epsilon_{F^*}(k),$$

where $sk \xleftarrow{\$} \mathcal{SK}$ and $f \xleftarrow{\$} f^{tk}$ is chosen uniformly at random from their domain. Further, f^{tk} is selected from an independent uniformly random permutation on M for each value of tk.

Similarly, we say that F^* is a (q, t, ϵ_{F^*}) strong tweakable pseudorandom permutation (STPRP) if $F^{*,tk}_{sk}$ is a permutation for all sk, tk. Moreover, f^{tk} is a random permutation on $M = \mathcal{T}$ and we select it for each value of tk.

Definition 2.9 (Authenticated Encryption). Let $(\mathcal{SK}, \mathsf{Enc}, \mathsf{Dec})$ be an authenticated encryption (AE) that consists of the following algorithms:

- $\mathsf{Enc} : \mathcal{SK} \times M \to \mathcal{C}$ algorithm inputs a key $sk \in \mathcal{SK}$ and message $m \in M$. It produces ciphertext $c \in \mathcal{C}$ as output. We denote it as $c \leftarrow \mathsf{Enc}(sk, m)$.
- $\mathsf{Dec} : \mathcal{SK} \times \mathcal{C} \to M \cup \{\perp\}$ algorithm inputs a key $sk \in \mathcal{SK}$ and ciphertext $c \in \mathcal{C}$. It produces $m \in M$ or $\perp$, if decryption fails. We denote it as $m \leftarrow \mathsf{Dec}(sk, c)$.

Moreover, it should satisfy the correctness property that $\mathsf{Dec}(sk, \mathsf{Enc}\ (sk, m)) = m$, for all $sk \in \mathcal{SK}$, $m \in M$ and $c \in \mathcal{C}$.

Definition 2.10 (Semantically Secure AE). Let υ be the leakage function that outputs λ bits, i.e., $\upsilon : \{0,1\}^* \to \{0,1\}^\lambda$. An authenticated encryption(AE) scheme is said to satisfy semantically secure property with respect to m and m^1 if $\{\mathsf{Enc}(sk, m), \upsilon(sk)\}$ and $\{\mathsf{Enc}(sk, m^1), \upsilon(sk)\}$ are computationally indistinguishable.

Definition 2.11 (Strong Misuse Resistance of AE). An AE scheme is said to be strong misuse resistance [19], if for every (q, t) bounded adversary A, the below advantage holds:

$$[Pr[A^{\mathsf{Enc}(.,.),\mathsf{Dec}(.,.)} = 1] - Pr[A^{R(.),\perp(.)} = 1]] \leq \epsilon(k),$$

where $sk \xleftarrow{\$} \mathcal{SK}$ and $\epsilon(k)$ denotes a negligible function. The function $R(m)$ outputs c and the length of such random bit string is $|\mathsf{Enc}(sk, m)|$. Oracle $\perp(c)$ produces output $\perp$ only if c is generated by $R(m)$ oracle earlier, in that case it returns m.

Definition 2.12 (Encrypt Digest Tag AE). Let $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$ be a collision resistant, range oriented, preimage resistant hash function, $F^* : \{0,1\}^k \times \{0,1\} \times \{0,1\}^k \to \{0,1\}^k$ be a strong tweakable pseudorandom function, F^*_k be a leak-free pseudorandom function and F_k be a pseudorandom function which can leak some information. An AE is called encrypt digest tag

(EDT) [21], if it combines a tweaked versions of $PSVEnc$ [16] with an "hash-then-MAC" scheme. Further, it can minimize the decryption leakages by giving an invalid ciphertext to restrict the impact of differential power analysis (DPA) attacks on message confidentiality.

Definition 2.13 (Pseudorandom Generator). A deterministic polynomial-time algorithm G is called pseudorandom generator (PRG) [9] against an adversary $\mathcal{A}$ if for all $A \in \mathcal{A}$, there exists a stretching function $l : \mathcal{N} \to \mathcal{N}$ (domain of l, i.e., $|\mathcal{N}| = 2\lambda$ and codomain of l, i.e., $|\mathcal{N}| = |m| + k$) such that $\{G_k\}_{k\in\mathcal{N}}$ and $\{U_k\}_{k\in\mathcal{N}}$ are computationally indistinguishable:

(a) The probability distribution G_k is defined as the output of G. The length of G_k is $l(k)$ on a uniformly selected seed in $\{0,1\}^k$.
(b) The probability distribution U_k is defined as the uniform distribution on $\{0,1\}^{l(k)}$, $l(k) > k$.

Let U_k be the uniform distribution over $\{0,1\}^k$, we need that for any PPT algorithm A, any positive polynomial $p(.)$, and for all sufficiently large k, it holds that

$$|Pr[A(G(U_k)) = 1] - Pr[A(U_{l(k)})) = 1]| < \frac{1}{p(k)}.$$

3 Code Construction

We propose the construction of non-malleable code from leakage resilient authenticated encryption [21] along with 1-more extractable hash function. Such authenticated encryption can handle encryption leakages as well as decryption leakages. Initially, a pseudorandom generator (PRG) is used to encode the secret key. The pseudorandom generator : $\{0,1\}^{2\lambda} \to \{0,1\}^{|m|+k}$ of [7] considers secret key $|sk| = 2\lambda/\alpha$, and it can tolerate maximum $\alpha\lambda$ bits of leakage [9], where $\alpha \in [0,1]$ and the value depends on underlying assumption. In our case, we consider the strongest assumption, i.e., $\alpha = 1$ which implies $|sk| = 2\lambda$. The complete encoding and decoding steps are described below:

- **Encoding.** The encoding algorithm takes a message block m, pseudorandom generator with the secret key sk and generates r_1, r_2 as output ($|r_1| = |m|$, $|r_2| = |k|$). The message block m is further divided into small chunks of $m_1, m_2, ..., m_l$ and the randomness r_1 is also broken into $r_1^1, r_1^2, ..., r_1^l$. Then, $PSVEnc_{r_2}(r_1, m)$ [16] is invoked that takes input a message block m, randomness r_1 and secret key r_2, and it generates encoded message $e_1, e_2, ..., e_l$. The $PSVEnc_{r_2}(r_1, m)$ function uses a tweakable leak-free pseudorandom function $F^{*,tw}_{k_{i-1}}(.)$ to generate the master key k_1 and another pseudorandom function $F_k(.)$ that can leak some information, used to generate the remaining keys k_i from the master key, for all $i > 1$. The pseudorandom function is instantiated with AES algorithm. Output of the function is XOR-ed with m_i and r_1^i, and it produces e_i. Next, $Tag_{r_2}(r_1, e)$ is invoked with the encoded message e, randomness r_1 and secret key r_2. Further, the collision resistant hash function H generates a hash value h based on e and r_1. Finally, $F^{*,1}_{r_2}(h)$ returns a tag τ. Apart from that, 1-more extractable hash function h_z is used to generate a hash value for the secret key sk, and the

codeword $C = (sk, e, \tau, h_1)$ is stored as $\mathcal{M}_L = \{sk\}$, $\mathcal{M}_R = \{e, \tau, h_1\}$. The encoding steps are illustrated as follows:

Algorithm 1 Function of $\mathbf{ExpNMC}^{f,m}_{A,1}()$

Input: sk, m
Output: $same^*$, $m^{'}$

1: $sk \leftarrow \{0,1\}^{2\lambda}$
2: $(sk, e, \tau, h_1) \leftarrow Enc_{sk}(m)$
3: $c = (sk, e, \tau, h_1)$
4: $\mathcal{M}_L = (sk)$
5: $\mathcal{M}_R = (e, \tau, h_1)$
6: $sk^{'} = f_L(sk)$
7: $(e^{'}, \tau^{'}, h_1^{'}) = f_R(e, \tau, h_1)$
8: $c^{'} = (sk^{'}, e^{'}, \tau^{'}, h_1^{'})$
9: **if** $c = c^{'}$ **then**
10: output $same^*$
11: **else**
12: output $m^{'} = Dec(c^{'})$
13: **end Function**

Algorithm 2 Function of $\mathbf{ExpNMC}^{f,m}_{A,2}()$

Input: sk, m
Output: $same^*$, $m^{'}$ **or** $\perp$

1: $sk \leftarrow \{0,1\}^{2\lambda}$
2: $(sk, e, \tau, h_1) \leftarrow Enc_{sk}(m)$
3: $c = (sk, e, \tau, h_1)$
4: $\mathcal{M}_L = (sk)$
5: $\mathcal{M}_R = (e, \tau, h_1)$
6: $sk^{'} = f_L(sk)$
7: $(e^{'}, \tau^{'}, h_1^{'}) = f_R(e, \tau, h_1)$
8: $c^{'} = (sk^{'}, e^{'}, \tau^{'}, h_1^{'})$
9: **if** $(h_1, sk, e) = (h_1^{'}, sk^{'}, e^{'})$ **then**
10: **if** $\tau \neq \tau^{'}$ **then**
11: output $\perp$
12: **else**
13: output $same^*$
14: **else**
15: output $m^{'} = Dec(c^{'})$
16: **end Function**

$KeyGen(1^k)$: $sk \leftarrow \{0,1\}^{2\lambda}$

$Enc_{sk}(m)$:

- $(r_1, r_2) \leftarrow PRG(sk)$
- $|r_1| = |m|$, $|r_2| = |k|$
- parse $m = (m_1, m_2, ..., m_l)$
- parse $r_1 = (r_1^1, r_1^2, ..., r_1^l)$
- $e = \{e_1, e_2, ..., e_l\} \leftarrow PSVEnc_{r_2}(r_1, m)$
- $PSVEnc_{r_2}(r_1, m)$
  - $k_1 \leftarrow F^{*,0}_{r_2}(r_1^1)$, also $k_i \leftarrow F^{*,tw}_{k_{i-1}}(p_A)$ and $tw \in \{0,1\}$
  - $e_1 \leftarrow F_{k_1}(p_B) \oplus m_1 \oplus r_1^1$
  - $\forall i = 2$ to l
  - $k_i \leftarrow F_{k_{i-1}}(p_A)$
  - $e_i \leftarrow F_{k_i}(p_B) \oplus m_i \oplus r_1^i$
  - $e = \{e_1, ..., e_l\}$
- $\tau \leftarrow Tag_{r_2}(r_1, e)$
  - $h \leftarrow H(r_1||e)$
  - $\tau \leftarrow F^{*,1}_{r_2}(h)$
- $h_z \leftarrow H_k$, $h_1 \leftarrow h_z(sk)$
- return $C = (sk, e, \tau, h_1)$

– **Decoding.** The decoding algorithm first inputs a secret key sk to the PRG and it generates r_1 and r_2. Next, it parses the codeword C. Collision resistant hash function $H(r_1||e)$ is invoked that returns a hash value h. Inverse tweakable pseudorandom function $(F^{*,1}_{r_2})^{-1}()$ takes τ as input and produces the hash h^c. Further, h^c is checked with h for consistency, if they are *same*, output of 1-more extractable hash $h_z(sk)$ is compared with h_1. Whenever

both the hash values are equal, $PSVEnc_{r_2}(r_1, e)$ returns the message m. Decoding steps are described as follows:

$Dec(C)$:
- Parse $C = (sk, e, \tau, h_1)$
- $(r_1, r_2) \leftarrow PRG(sk)$
- $h \leftarrow H(r_1||e)$
- $h^c \leftarrow (F^{*,1}_{r_2})^{-1}(\tau)$
- If $h \neq h^c$, return $\perp$
- If $h_1 \neq h_z(sk)$, return $\perp$
- $m \leftarrow PSVEnc_{r_2}(r_1, e)$
- return m

Liu et al. [11] observe that an adversary can check the equality between $f_L(\hat{sk})$ and $f_L(sk)$ using the leakage of a universal hash and it generates $log^2 k$ bits as output. Kiayias et al. [18] show that a similar kind of equality check between $f_L(\hat{r}, \hat{sk})$ and $f_L(r, sk)$ can be performed using the leakage of a universal hash that generates $2k+log^2 k$ bits as output, where $|\hat{r}| = 2k$. The proposed split-state non-malleable code is defined as $Enc : m \rightarrow \{sk\}||\{e, \tau, h_1\}$. The length of codeword is $|m|$ +|security-bits|. By setting $\lambda = log^2 k$, we get $|m|+|sk|+|\tau|+|h_1|$, i.e., $|m| + 2k + 2log^2(k)$ (where $|e| = |m|$), assuming the size of $|\tau|$, $|h_1|$ are k bits each. Let $h_z \leftarrow H_k$ be a hash function that is collision resistant, 1-more extractable hash and efficiently samplable as defined in [18]. An adversary chooses $\widetilde{h} \leftarrow \widetilde{H_{\lambda-1}}$ from universal hash function family. The hash function outputs $\lambda - 1$ bits. Let υ be the leakage function defined as follows:

$\upsilon^{\widetilde{h}}(sk)= (0, \widetilde{h}(f_L(sk))$ if $(f_L(sk) = sk)$.

else, $\upsilon^{\widetilde{h}}(sk)= (1, \widetilde{h}(f_L(sk))$, $(f_L(sk) \neq sk)$.

The leakage function outputs $\lambda = \beta(logk)+\omega(k)$ bits. Our experiment checks the leaked value instead of the output generated by f_L.

4 Proof of Security

Theorem 1. *Let H_k be 1-more extractable hash function that generates $\omega(k)$ bits as output, where $\omega(k) = poly(k)$, $(\mathcal{SK}, Enc, Dec)$ be a leakage resilient authenticated encryption that can handle λ bits of leakage, where $\lambda = \beta(logk) + \omega(k)$ and k denotes the security parameter. Then, $(KeyGen(1^k), Enc_{sk}(m), Dec(C))$ is strongly non-malleable.*

Proof. We need to show that for the tampering function $f = (f_L, f_R)$ and messages m, m^1, the experiment $Tamper^f_m(m)$ and $Tamper^f_{m^1}(m^1)$ are computationally indistinguishable, i.e., $Tamper^f_m(m) \underset{c}{\approx} Tamper^f_{m^1}(m^1)$. We define the experiment $\mathbf{ExpNMC}^{f,m}_{A,1}()$ (Algorithm 1) and change it incrementally to $\mathbf{ExpNMC}^{f,m}_{A,2}()$, $\mathbf{ExpNMC}^{f,m}_{A,3}()$, $\mathbf{ExpNMC}^{f,m}_{A,4}()$ and show that they are computationally indistinguishable except with negligible probability. Given a message m and tampering function $f = (f_L, f_R) \in \mathcal{F}$, the experiment $\mathbf{ExpNMC}^{f,m}_{A,1}()$ is exactly same as the original tampering experiment, i.e., $Tamper^f_m(m)$.

Algorithm 3 Function of $\mathbf{ExpNMC}^{f,m}_{A,3}()$

Input: sk, m
Output: $same^*$, $m^{'}$ **or** $\perp$

$sk \leftarrow \{0,1\}^{2\lambda}$
$(sk, e, \tau, h_1) \leftarrow Enc_{sk}(m)$
$c = (sk, e, \tau, h_1)$
$\mathcal{M}_L = (sk)$
$\mathcal{M}_R = (e, \tau, h_1)$
$sk^{'} = f_L(sk)$
$(e^{'}, \tau^{'}, h_1^{'}) = f_R(e, \tau, h_1)$
if $(h_1, sk, e) = (h_1^{'}, sk^{'}, e^{'})$ **then**
  if $\tau \neq \tau^{'}$ **then**
    output $\perp$
  else
    output $same^*$
if $h_1 \neq h_1^{'}$ **then**
  $\hat{sk} \leftarrow \mathcal{E}(h_z, h_1)$
  if $sk^{'} = \hat{sk}$ **then**
    if $h_1^{'} = h_z(\hat{sk})$ **then**
      output $m^{'} = Dec(c^{'})$
    else
      output $\perp$
  else
    output $\perp$
end Function

Algorithm 4 Function of $\mathbf{ExpNMC}^{f,m}_{A,4}()$

Input: sk, m
Output: $same^*$, $m^{'}$ **or** $\perp$

$sk \leftarrow \{0,1\}^{2\lambda}$
$(sk, e, \tau, h_1) \leftarrow Enc_{sk}(m)$
$c = (sk, e, \tau, h_1)$
$\mathcal{M}_L = (sk)$
$\mathcal{M}_R = (e, \tau, h_1)$
$sk^{'} = f_L(sk)$
$(e^{'}, \tau^{'}, h_1^{'}) = f_R(e, \tau, h_1)$
$(b, h_1^{l}) \leftarrow v^{\widetilde{h}}(sk)$
if $(h_1^{l}, sk, e) = (h_1^{'}, sk^{'}, e^{'})$ **then**
  if $\tau = \tau^{'}$ **then**
    output $same^*$
  else
    output $\perp$
$\hat{sk} \leftarrow \mathcal{E}(h_z, h_1)$
if $\widetilde{h}(\hat{sk}) \neq h_1^{l}$ **then**
  if $sk^{'} = \hat{sk}$ **then**
    if $h_1^{'} = h_z(\hat{sk})$ **then**
      output $m^{'} = Dec(c^{'})$
    else
      output $\perp$
  else
    output $\perp$
end Function

In $\mathbf{ExpNMC}^{f,m}_{A,2}()$ (Algorithm 2), we check whether an adversary has modified the hash value of sk and e. As the hash function H_k is collision resistant, if the adversary does not modify the hash h_1, secret key is not changed at all, i.e., $(h_z(sk) = h_1)$ and the condition $(h_1, sk, e) = (h_1^{'}, sk^{'}, e^{'})$ should be satisfied. Next, tag is calculated from $Enc_{sk}(m)$. New tag and modified tag are compared and if they are $equal$, output is set to $same^*$. Whenever $(\tau \neq \tau^{'})$ output is set to $\perp$, otherwise, it breaks the authenticity property under leakage. If $(h_1, sk, e) \neq (h_1^{'}, sk^{'}, e^{'})$, it outputs $m^{'}$ which is same as $\mathbf{ExpNMC}^{f,m}_{A,1}()$.

$\mathbf{ExpNMC}^{f,m}_{A,3}()$ (Algorithm 3) does not use the real decoding procedure but it uses extractor $\mathcal{E}_v$ (i.e., in short $\mathcal{E}$) of 1-more extractable hash function to get preimage of the hash value h_1, i.e., $\hat{sk}$. Using the preimage, it again calculates $h_z(\hat{sk})$ and compares with $h_1^{'}$. Whenever the condition $(h_1 \neq h_1^{'})$ is satisfied, tampered secret key $sk^{'}$ is compared with $\hat{sk}$, and the new hash value is computed for $\hat{sk}$ to check consistency with $h_1^{'}$. This part only differs with $\mathbf{ExpNMC}^{f,m}_{A,2}()$ and output $m^{'}$ is generated, otherwise, it returns $\perp$. To illustrate the working strategy of $\mathcal{E}_v$, we define $\mathcal{A}_v$, p_v, with respect h_z, (e, τ), h_1, and the tampering function $f = (f_L, f_R)$.

- (**Define** $\mathcal{A}_v$) : $\mathcal{A}_v(h_z, h_1, p_v) = ([f_R(h_1, p_v)])$
- (**Auxiliary info for** $\mathcal{A}_v$) : $p_v = (e, \tau)$
- (**Existence of extractor** $\mathcal{E}_v$**, and auxiliary input** $p_{\mathcal{E}}$): Given $\mathcal{A}_v$ and p_v, using 1-more extractability of the hash function H_k, an extractor $\mathcal{E}_v$ can be constructed, with hardwired auxiliary info $p_{\mathcal{E}}$, and it computes $\hat{sk} \leftarrow \mathcal{E}_v(h_z, h_1)$. We denote $\mathcal{E}_v$ as $\mathcal{E}$ for brevity.

In $\mathbf{ExpNMC}^{f,m}_{A,4}()$ (Algorithm 4), we perform consistency check procedure through leakage. Let $\widetilde{h} \leftarrow \widetilde{H_{\lambda-1}}$ be selected from universal hash function family.

The underlying hash function generates $\lambda-1$ bits as output. The leakage function is defined as follows:

$v^{\widetilde{h}}(sk)= (0, \widetilde{h}(f_L(sk))$ if $(f_L(sk) = sk)$

else, $v^{\widetilde{h}}(sk)= (1, \widetilde{h}(f_L(sk))$, $(f_L(sk) \neq sk)$.

An adversary has access to the leakage function $v^{\widetilde{h}}(sk)$ to calculate h_1^l. A random variable b is used to store the output, and b is set to 0 if $(f_L(sk) = sk)$. Next, $(h_1^l, sk, e) = (h_1^{'}, sk^{'}, e^{'})$ is checked. If there is a collision against $\widetilde{h}$, it induces statistical difference only. Since $\widetilde{h}$ is a universal hash function and it is chosen independently for the current experiment, the probability of occurrence is negligible. Further, the tampered *tag* $\tau^{'}$ and the original *tag* τ are compared and output is set to $same^*$, if they are equal. Whenever $(\widetilde{h}(\hat{sk}) \neq h_1^l)$, it compares tampered secret key $sk^{'}$ with the extracted secret key $\hat{sk}$ and the hash value $(h_1^{'} = h_z(\hat{sk}))$. If it is successful, output $m^{'} = Dec(c^{'})$, otherwise, it outputs $\perp$.

Lastly, we show that $\mathbf{ExpNMC}_{A,4}^{f,m}()$ and $\mathbf{ExpNMC}_{A,4}^{f,m^1}()$ are computationally indistinguishable for any two arbitrarily chosen messages m and m^1.

Lemma 1. *Let H be a collision resistant, range-oriented, preimage resistant hash function, $(\mathcal{SK}, Enc, Dec)$ is a leakage resilient authenticated encryption with decryption leakages, $f = (f_L,\ f_R) \in \mathcal{F}$ be a tampering function and for any message m, $\boldsymbol{ExpNMC}_{A,1}^{f,m}()$ and $\boldsymbol{ExpNMC}_{A,2}^{f,m}()$ are computationally indistinguishable.*

Proof. $\mathbf{ExpNMC}_{A,1}^{f,m}()$ and $\mathbf{ExpNMC}_{A,2}^{f,m}()$ are different in the following branch condition:

- $(h_1, sk, e) = (h_1^{'}, sk^{'}, e^{'}) \wedge (\tau \neq \tau^{'})$
- $(h_1, sk, e) = (h_1^{'}, sk^{'}, e^{'}) \wedge (\tau = \tau^{'})$

Let the branch condition $(h_1, sk, e) = (h_1^{'}, sk^{'}, e^{'}) \wedge (\tau \neq \tau^{'})$ be denoted by the event C and $\mathbf{ExpNMC}_{A,2}^{f,m}()$ experiment returns $\perp$ when the event C occurs. So, $\mathbf{ExpNMC}_{A,2}^{f,m}()$ and $\mathbf{ExpNMC}_{A,1}^{f,m}()$ output $same^*$ conditioned on the event $\sim C$. Let F be the event that $(\tau = \tau^{'})$. Now, $Pr[C] = Pr[C \wedge F] + Pr[C \wedge \sim F]$. We have to show that $Pr[C \wedge F]$, $Pr[C \wedge \sim F]$ occurs with negligible probability. Here, we follow proof by contradiction technique. Let us consider $Pr[C \wedge \sim F] > \epsilon(k)$, for some negligible function $\epsilon(k)$. Then, there exists a PPT adversary A which can break the collision resistance property of hash function H. Further, the adversary simulates $\mathbf{ExpNMC}_{A,2}^{f,m}()$ and outputs τ, $\tau^{'}$, (sk, e), $(sk^{'}, e^{'})$ $(h \leftarrow H(r_1||e)$ and $\tau \leftarrow F_{r_2}^{*,1}(h))$. The function f is polynomial time computable. So, the running time of the adversary is also polynomial and it wins the event $Pr[C \wedge \sim F]$, where the assumption is that $Pr[C \wedge \sim F] > \epsilon(k)$. Hence, the adversary breaks the collision resistance property of hash function with non-negligible probability. Let us consider that $Pr[C \wedge F] > \epsilon(k)$, for some negligible function $\epsilon(k)$. An adversary with access to tampering function $f = (f_L,\ f_R)$ with $(e^{'}, \tau^{'}, h_1^{'}) = f_R(e, \tau, h_1)$ breaks the authenticity property under leakage. Firstly, $Enc_{sk}(m)$ is invoked and the *tag* is recomputed using $h \leftarrow H(r_1||e)$, $\tau \leftarrow F_{r_2}^{*,1}(h)$. Assuming $Pr[C \wedge F] > \epsilon(k)$, the inequality $(\tau \neq \tau^{'})$ generates a valid ciphertext with respect to sk and authenticity property under leakage breaks with non-negligible probability $\epsilon(k)$. Hence, the proof of lemma concludes.

Lemma 2. *Let H_k be 1-more extractable hash function and $f = (f_L, f_R)$ be a tampering function, for any message m, $\boldsymbol{ExpNMC}^{f,m}_{A,2}()$ and $\boldsymbol{ExpNMC}^{f,m}_{A,3}()$ are computationally indistinguishable.*

Proof. In $\mathbf{ExpNMC}^{f,m}_{A,3}()$, we do not use real decoding procedure. Here, the role of extractor function in the 1-more extractable hash is used, and $\hat{sk}$ is compared with the tampered key $sk^{'}$ if $(h_1 \neq h_1^{'})$. This part only differs with $\mathbf{ExpNMC}^{f,m}_{A,2}()$. If $(h_1^{'} = h_z(\hat{sk}))$ and $(sk^{'} = \hat{sk})$ is true, output of the two experiments are equal. We need to show that two experiments are indistinguishable, i.e., the probability of occurrence of the following two conditions $(sk^{'} \neq \hat{sk}) \wedge (h_1^{'} = h_z(\hat{sk}))$ and $(sk^{'} = \hat{sk}) \wedge (h_1^{'} \neq h_z(\hat{sk}))$ are negligible. From the property of 1-more extractable hash function, the probability of occurrence $(sk^{'} = \hat{sk}) \wedge (h_1^{'} \neq h_z(\hat{sk}))$ is negligible. Let E be the event that $(sk^{'} \neq \hat{sk})$. Consider the below events, denoted as E_1, E_2.

- $E \wedge (h_z(sk^{'}) = h_z(\hat{sk}) = h_1^{'})$: It happens when there is a collision and by the property of hash function used in our scheme $Pr[E_1] \leq \epsilon(k)$.
- $E \wedge (h_z(sk^{'}) = h_1^{'} \wedge h_z(\hat{sk}) \neq h_1^{'})$: Since the hash function is 1-more extractable [11], we can conclude that $Pr[E_2] \leq \epsilon(k)$. Now, we can relate $\mathbf{ExpNMC}^{f,m}_{A,3}()$ with $Exp^{s,h_z}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_v}(1, p_v, p_{\mathcal{E}})$, for some message $m^{'}$, algorithm $\mathcal{A}_v$, $\mathcal{A}_s$, extractor $\mathcal{E}_v$ and inputs p_v, $p_{\mathcal{E}}$.

Therefore, we get $Pr[E_1] + Pr[E_2] \leq \epsilon(k)$. Hence, $\mathbf{ExpNMC}^{f,m}_{A,2}()$ and $\mathbf{ExpNMC}^{f,m}_{A,3}()$ are computationally indistinguishable. This concludes the proof of lemma.

Lemma 3. *Let H be a collision resistant, range-oriented, preimage resistant hash function, $\widetilde{h} \leftarrow \widetilde{H_{\lambda-1}}$ is chosen from universal hash function family that generates $\lambda - 1$ bits as output, where $\lambda = \beta(logk)$, for any message m, $\boldsymbol{ExpNMC}^{f,m}_{A,3}()$ and $\boldsymbol{ExpNMC}^{f,m}_{A,4}()$ are computationally indistinguishable.*

Proof. In $\mathbf{ExpNMC}^{f,m}_{A,4}()$, we present h_1 as leakage over sk and such thing does not make any statistical difference. $\widetilde{h} \leftarrow \widetilde{H_{\lambda-1}}$ is chosen from universal hash function family and we compare $(h_1^l, sk, e) = (h_1^{'}, sk^{'}, e^{'})$, whereas in $\mathbf{ExpNMC}^{f,m}_{A,3}()$, we compare $(h_1, sk, e) = (h_1^{'}, sk^{'}, e^{'})$. The remaining part $(\tau = \tau^{'})$ is exactly same as previous experiment. If $(h_1^l, sk, e) = (h_1^{'}, sk^{'}, e^{'})$ and $(\tau = \tau^{'})$ is satisfied, output is set to $same^*$. The only difference between current and previous experiment is that the calculation of hash value of the secret key over leakage. Let B be the event $(\widetilde{h}(\hat{sk}) = h_1^l) \wedge (sk^{'} \neq \hat{sk})$. It is clear that the statistical difference between the two experiments are upper bounded by $Pr[B]$. We choose the universal hash function $\widetilde{h}$ independently from its input, and the probability of collision is bounded by $\lambda - 1 = \epsilon(k)$. The collision event B is exactly same and we have $Pr[B] \leq \epsilon(k)$. Further, the probability of occurrence of the condition $(h_z(\hat{sk}) \neq h_1^{'}) \wedge (sk^{'} = \hat{sk})$, denoted as $B1$, is negligible, i.e., $Pr[B1] \leq \epsilon(k)$, since the hash function is deterministic. Therefore, the proof of lemma completes.

Lemma 4. *Let $(\mathcal{SK}, Enc, Dec)$ be a leakage resilient authenticated encryption scheme, $f = (f_L, f_R)$ be a tampering function and for two arbitrarily chosen*

messages m *and* m^1, $\boldsymbol{ExpNMC}^{f,m}_{A,4}()$ *and* $\boldsymbol{ExpNMC}^{f,m^1}_{A,4}()$ *are computationally indistinguishable.*

Proof. Here, we use proof by contradiction approach. Let us assume that for the two arbitrarily chosen messages m, m^1, there exists a tampering function $f = (f_L, f_R)$ and PPT distinguisher D such that $|Pr[D(\mathbf{ExpNMC}^{f,m}_{A,4}()) = 1]$ - $Pr[D(\mathbf{ExpNMC}^{f,m^1}_{A,4}()) = 1]| > \epsilon$, where $\epsilon = 1/poly(k)$. The adversary A is able to break the semantic security of encryption scheme (**Definition 2.10**) under leakage. It picks up the leakage function $v^{\widetilde{h}}(sk)$, connect the function with hardware, and performs the experiment $\mathbf{ExpNMC}^{f,m}_{A,4}()$ for two arbitrarily chosen messages m and m^1. It is straightforward to see that A simulates $\mathbf{ExpNMC}^{f,m}_{A,4}()$ and the advantage of breaking semantic security is same as distinguisher D, in distinguishing $\mathbf{ExpNMC}^{f,m}_{A,4}()$ and $\mathbf{ExpNMC}^{f,m^1}_{A,4}()$, which is non-negligible by assumption. Therefore, we arrive at the contradiction and it completes the proof of lemma.

From the above analysis, it is evident that for the tampering function f, two arbitrarily chosen messages m, m^1, $Tamper^f_m(m)$ and $Tamper^f_{m^1}(m^1)$ are computationally indistinguishable, i.e., $Tamper^f_m(m) \underset{c}{\approx} Tamper^f_{m^1}(m^1)$.

5 Conclusion

In this work, we construct a computationally secure non-malleable code from authenticated encryption with 1-more extractable hash function. Our proposed construction removes the requirement of common reference string based trusted setup. The codeword provides security against one-time tampering attack with a limited bits of leakage.

References

1. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. J. Cryptology 14(2), 101-119 (2001)
2. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566-598. Springer, Heidelberg (2001)
3. Joan, D., Vincent R.: The Design of Rijndael. Springer-Verlag New York, Inc., Secaucus (2002)
4. Handschuh, H., Naccache, D.: SHACAL: A Family of Block Ciphers. Submission to the NESSIE project (2002)
5. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side channel(s): Attacks and assessment methodologies. In: Kaliski Jr., B.S., Koc, C.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29-45. Springer, Heidelberg (2003)
6. Groth, J., Sahai, A.: Efficient Non-Interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415-432. Springer, Heidelberg (2008)
7. K. Pietrzak. Advances in Cryptology - EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings, chapter A Leakage-Resilient Mode of Operation. 2009.

8. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18-35. Springer, Heidelberg (2009)
9. F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. chapter Leakage Resilient Cryptography in Practice, pages 99-134. 2010.
10. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: Yao, A.C.-C. (ed.) ICS 2010, Beijing, China, January 5-7, pp. 434-452. Tsinghua University Press (2010)
11. F.-H. Liu, A. Lysyanskaya, Tamper and leakage resilience in the split-state model, in CRYPTO (2012), pp. 517-532.
12. Liu, F.-H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 517-532. Springer, Heidelberg (2012)
13. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS, pp. 326-349 (2012)
14. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 239-257. Springer, Heidelberg (2013)
15. Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. In: STOC, pp. 774-783 (2014)
16. Pereira, O., Standaert, F.X., Vivek, S.: Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In: ACM CCS 15. ACM Press
17. 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, pp. 459–468. ACM (2015)
18. A. Kiayias, F.-H. Liu, Y. Tselekounis, Practical non-malleable codes from l-more extractable hash functions, in CCS (2016), pp. 1317-1328.
19. Berti, F., Koeune, F., Pereira, O., Peters, T., Standaert, F.X.: Leakage-resilient and misuse-resistant authenticated encryption. Cryptology ePrint Archive, Report 2016/996 (2016).
20. Aggarwal, D., Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Optimal computational split-state non-malleable codes. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 393-417. Springer, Heidelberg (2016)
21. Berti, F., Pereira, O., Peters, T., Standaert, F.X.: On leakage-resilient authenticated encryption with decryption leakages. IACR Trans. Symmetric Cryptol. 2017(3), 271-293.
22. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. J. ACM 65(4), 20:1-20:32 (2018)
23. Fehr, S., Karpman, P., Mennink, B.: Short Non-Malleable Codes from Related-Key Secure Block Ciphers. IACR Transactions on Symmetric Cryptology, 336-352, (2018)
24. Aggarwal, D., Obremski, M.: A constant-rate non-malleable code in the split-state model. IEEE 61st Annual Symposium on Foundations of Computer Science, FOCS (2020).
25. Brian, G., Faonio, A., Ribeiro, L., Venturi, D.: Short Non-Malleable Codes from Related-Key Secure Block Ciphers, Revisited. IACR Transactions on Symmetric Cryptology, 1-19, (2022)
26. Ghosal, A.K., Ghosh, S., Roychowdhury, D.: Practical Non-malleable Codes from Symmetric-Key Primitives in 2-Split-State Model. In: Ge, C., Guo, F. (eds) Provable and Practical Security, (2022).