FMNV Continuous Non-malleable Encoding Scheme is More Efficient Than Believed

Amir S. Mortazavi
Department of Electrical Engineering
Sharif University of Technology
Tehran, Iran
Email: sa_mortazavi@ee.sharif.edu

Mahmoud Salmasizadeh
Electronics Research Institute
Department of EE as adjunct member
Sharif University of Technology
Tehran, Iran
Email: salmasi@sharif.ir

Amir Daneshgar
Department of Mathematical Sciences
Sharif University of Technology
Tehran, Iran
Email: daneshgar@sharif.ir

***Abstract*—Non-malleable codes are kind of encoding schemes which are resilient to tampering attacks. The main idea behind the non-malleable coding is that the adversary can't be able to obtain any valuable information about the message. Non-malleable codes are used in tamper resilient cryptography and protecting memory against tampering attacks. Several kinds of definitions for the non-malleability exist in the literature. The Continuous non-malleability is aiming to protect messages against the adversary who issues polynomially many tampering queries. The first continuous non-malleable encoding scheme has been proposed by Faust et el. (FMNV) in 2014.**

In this paper, we propose a new method for proving *continuous non-malleability* of FMNV scheme. This new proof leads to an improved and *more efficient* scheme than previous one. The new proof shows we can have the continuous non-malleability with the same security by using a *leakage resilient storage* scheme with about $(k+1)(\log(q)-2)$ bits fewer leakage bound (where k is the output size of the collision resistant hash function and q is the maximum number of tampering queries).

***Index Terms*—non-malleable, continuous non-malleability, tamper-resilient cryptography.**

I. Introduction

Hardware attacks are a dangerous threat for cryptographic devices. These attacks can be divided into active or passive attacks. Passive attacks are based on the measuring of side channel information such as the power consumption of a device or its electromagnetic emanations. While active attacks try to tamper with the devices. In the *tampering attack* the adversary has ability to modify and manipulate some parameters of the system. Tamper-resilient cryptography includes a theoretical study of such attacks.

Non-malleable codes, a kind of encoding schemes, allow a message m to be encoded into a codeword c, such that m can resist against tampering attacks. The main goal of designing non-malleable codes is for resistance against an active adversary that has power to modify the codeword according to a Turing machines family. Note that the tampering attacks only are defined for a specific family of Turing machines and no guarantees are provided for other Turing machines. Non-malleable codes can be used in tamper-resilient cryptography and protecting system memory against tampering attacks.

It is straightforward to show that non-malleable codes do not exist for the family of all efficient tampering Turing machines. Thus we have to restrict the class of tampering attacks. The *split-state model* is the class of tampering Turing machines for which efficient constructions of non-malleable codes are known. A split-state model is one that the codeword has several parts and each part is tampered with independently of each other.

Several kinds of definitions for the non-malleability exist in the literature. The security definition of non-malleability is based on the indistinguishability. The one-time non-malleability considers only one tampering attack and the continuous non-malleability allows polynomially many tamping attacks.

In this paper we study the Faust et al. [7] (FMNV for short) scheme and show that we can prove the continuous non-malleability for this scheme with better efficiency than the original proof. Our proof is based on the fact that it is hard to break the distinguishability of *leakage resilient storage* scheme. Our main contribution is the presentation a new method for finding the *self-destruction* round of tampering queries with $2k+1$ bits of leakage while in the original proof it was $2k\log(q)$ bits of leakage (where q is the number of tampering queries and k is the length of hash function output).

We present in Section II a formal definitions for some required primitives. In Section III we introduce the FMNV scheme for a continuous non-malleable scheme. Finally, in Section IV we show a new proof for the former scheme.

A. Related Works

The non-malleable codes was introduced by Dziembowski et al. [1] for bit-wise family of Turing machines which can tamper with every bit of the codeword independent of other bits. In [2] an efficient non-malleable scheme for bit-wise family of tampering is introduced. The non-malleable code for block-wise tampering introduced in [3]. Liu and Lysyanskaya [4] introduced the first non-malleable codes in the split-state model for all PPT Turing machines. Moreover, they considered the leakage of codeword. The non-malleable code for a family of Turing machines with size $2^{\text{poly}(n)}$ has been studied in [5], [6]. Faust et al. [7] has extended the definition of non-malleability to include the continuity and proposed a scheme in the CRS model. Aggarwal et al. [8] introduced a generalization of non-malleable codes, called non-malleable reductions. Chandran

et al. [9] defined their new notion of lookahead (block-wise) non-malleable codes and proposed a new scheme in this model.

The first information theoretic non-malleable code introduced by Dziembowski et al. [10] for only one-bit messages and in [11] is extended to multi-bit messages. See also [6], [12] for other works in information theoretic model.

Austrin et al. [13] studied the effect of tampering on the randomness of cryptographic algorithms. The works of [4], [7], [14], [15] showed the application of non-malleable codes for tamper-resilient cryptography. Dachman-Soled et al. [16] studied the securing RAM computation against memory tampering and leakage attacks. Coretti et al. [17] showed that non-malleable codes can be used to construct a CCA secure public key.

B. Notations

Given a set $\mathcal{S}$, we write $a \in_R \mathcal{S}$ to denote sampling uniformly an element a from set $\mathcal{S}$. We use $:=$ to denote deterministic assignment and $\leftarrow$ to the probabilistic assignment. We use PPT instead of probabilistic polynomial-time. We denote negl for a negligible function, that grows smaller than $\frac{1}{p(n)}$ for any polynomial $p(n)$ [18]. We denote with $|f|$ to the output size of the function f. We use FMNV as a shorthand for Faust et al. [7].

The leakage oracle $O^l(.)$ indicates an PPT adversary that can query adaptively leakage Turing machines L_i such that $\sum_i(|L_i|) \leq l$.

II. Preliminaries

A. Zero knowledge Proofs

Let a *non-interactive zero knowledge* (NIZK) proof system $\Pi = (\mathrm{Init}, P, V, \mathrm{Sim}_1, \mathrm{Sim}_2)$ for language $\mathcal{L} \in$ NP, $\mathcal{L} = \{x : \exists\ \omega$ such that $R(x,\omega) = 1\}$, where ω is the witness, R is a relation, P, V, Sim_1 and Sim_2 are PPT algorithms such that:

1) **Completeness**: For all $x \in \mathcal{L}$ and all ω such that $R(x,\omega) = 1$ and $\Omega \leftarrow \mathrm{Init}(1^n)$, we have $V(\Omega, x, P(\Omega, x, \omega)) = 1$.
2) **Soundness**: If $x \notin \mathcal{L}$ then for every ω and $\Omega \leftarrow \mathrm{Init}(1^n)$, $\Pr[V(\Omega, x, P(\Omega, x, \omega))] = 1$ be negligible.
3) **Zero-Knowledge**: For all PPT adversaries we have $\mathrm{Real}(n) \approx_c \mathrm{Sim}(n)$, where:

$$\mathrm{Real}(n) = \{\quad \Omega \leftarrow \mathrm{Init}(1^n); X \leftarrow A^{P(\Omega,.,.)}(\Omega) : X\ \},$$

$$\mathrm{Sim}(n) = \left\{ \begin{array}{l} (\Omega, tk) \leftarrow \mathrm{Sim}_1(1^n); \\ Y \leftarrow A^{\mathrm{Sim}_2(\Omega,.,.,tk)}(\Omega) : Y \end{array} \right\}.$$

There are several models of NIZK proofs that have similar definitions as above. In this paper, we use *robust* non-interactive NIZK proofs [19]. This type of NIZK has an extra *Extractability* property which for all PPT adversaries there exist an efficient algorithm Ext such as:

$$\Pr\left[\begin{array}{l} (\Omega, tk, ek) \leftarrow \mathrm{Sim}_1(1^n), \\ (x,\pi) \leftarrow A^{\mathrm{Sim}_2(\Omega,.,.,tk)}(\Omega), \\ \omega \leftarrow \mathrm{Ext}(\Omega, (x,\pi), ek); \\ R(x,\omega) = 1 \ \vee \ (x,\pi) \in Q \\ \vee\ V(\Omega, x, \pi) = 0 \end{array} \right] = 1 - \mathrm{negl}(n).$$

where Q denotes the pairs (x_i, π_i) that Sim_2 has answered $\mathcal{A}$.

Remark II.1. In the definition of NIZK proofs the Init algorithm generates the Ω which is shared between all parties and known as *common reference string* (CRS).

Remark II.2. In this paper, the robust NIZk proofs require to support labels (λ). The labels are a public string as input to P, V, Ext and Sim_2. This property can be achieved by concatenation the label to the statement x. Now we show the NIZK algorithms as P^λ, V^λ, Ext^λ and Sim_2^λ

B. Leakage Resilient Storage

The *leakage resilient storage* encoding system $\Pi = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ is defined in [1]. The Π includes a pair of computable PPT functions where for messages $x \in \{0,1\}^m$:

$$(s_0, s_1) \leftarrow \mathrm{LRS}(x)$$
$$x := \mathrm{LRS}^{-1}(s_0, s_1)$$

It is required that $\Pr[\mathrm{LRS}^{-1}(\mathrm{LRS}(x)) = x] = 1$ for any message x.

The security definition of l-leakage resilient storage system is defined by experiment $\mathrm{Leakage}_{\mathcal{A},l}(n)$ for the security parameter n and every PPT adversary $\mathcal{A}$.

The indistinguishability experiment $\mathrm{Leakage}_{\mathcal{A},l}(n)$:

(i) The adversary $\mathcal{A}$ is given public parameters, and outputs a pair of messages m_0, m_1 in the message space.
(ii) A uniform bit $b \in \{0,1\}$ is chosen, and then a codeword $(s_0, s_1) \leftarrow \mathrm{LRS}(m_b)$ is computed.
(iii) Adversary $\mathcal{A}$ can query with the leakage oracles $O^l(s_0)$ and $O^l(s_1)$ independently of each other to maximum l bits.
(iv) $\mathcal{A}$ outputs a bit b'. The output of the experiment is 1 if $b' = b$, and 0 otherwise.

The encoding scheme $\Pi = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ is an l-leakage resilient storage system if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function negl such that

$$Pr[\mathrm{Leakage}_{\mathcal{A},l}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

C. Strong Leakage Resilient Storage

The encoding scheme $\Pi = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ is *strong* l-leakage resilient storage scheme [7], if for $\theta \in \{0,1\}$ and every PPT adversary:

$$\Pr[\mathrm{Leakage}_{\mathcal{A},l,\theta}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

Where indistinguishability experiment $\mathrm{Leakage}_{\mathcal{A},l,\theta}(n)$ is defined as follows:

(i) Adversary $\mathcal{A}$ is given public parameters, and outputs a pair of messages m_0, m_1 in the message space.

(ii) A uniform bit $b \in \{0,1\}$ is chosen, and then a codeword $(s_0, s_1) \leftarrow \mathrm{Enc}(m_b)$ is computed.
(iii) Adversary $\mathcal{A}$ can interact with the leakage oracles $O^l(s_0)$ and $O^l(s_1)$.
(iv) After finishing leakage queries, $\mathcal{A}$ is given s_θ.
(v) $\mathcal{A}$ outputs a bit b'. The output of the experiment is 1 if $b' = b$, and 0 otherwise.

In this definition, one of the two shares is given to the adversary after termination of leakage queries. A good construction for (strong) leakage-resilient is presented in [1], [7] by using the inner product in finite fields. It can be shown the inner product based LRS schemes are also secure in the *information theoretic* model.

D. Non-malleable Codes

We first define an encoding scheme without requiring a key and then define several variant forms of non-malleability for this encoding scheme in the split-state model.

The non-malleable coding $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec})$ is defined in split-state model as:

$$\Omega \leftarrow \mathrm{Init}(1^n),$$

$$(x_0, x_1) \leftarrow \mathrm{Enc}(\Omega, x) \text{ for } x \in \{0,1\}^{n'},$$

$$\tilde{x} := \mathrm{Dec}(x_0, x_1) \text{ for } \tilde{x} \in \{\{0,1\}^{n'} \cup \perp\}.$$

Where n' is a polynomial function of security parameter, $\perp$ is the symbol for indication of the failure and Ω is a public and *untamperable* string for Initialization.
In the split-state model, codeword has two parts, such that each share is tamped with independently.

The *strong non-malleability* for an encoding scheme is defined based on experiment $\mathrm{SNMLR}_{\mathcal{A},l,\mathcal{T}}(n)$ as follows [4]:

Definition II.1. The indistinguishability experiment $\mathrm{SNMLR}_{\mathcal{A},l,\mathcal{T}}(n)$:
(i) $\mathrm{Init}(1^n)$ is run to obtain public parameters Ω.
(ii) Adversary $\mathcal{A}$ is given Ω, and outputs a pair of legal messages m_0, m_1.
(iii) A random bit $b \in \{0,1\}$ is chosen, and $(s_0, s_1) \leftarrow \mathrm{Enc}(m_b)$ is computed.
(iv) The adversary $\mathcal{A}$ has ability to query the leakage oracles $O^l(s_0)$ and $O^l(s_1)$ to l bits.
(v) Send Turing machines (T_0, T_1) for $T_0 \in \mathcal{T}$ and $T_1 \in \mathcal{T}$ as a tampering query.
  a) $x'_0 := T_0(x_0)$, $x'_1 := T_1(x_1)$ and $x' := \mathrm{Dec}(x'_0, x'_1)$ are computed.
  b) If $(x_0, x_1) = (x'_0, x'_1)$ then the adversary is given $same^*$; else, is given x'.
(vi) $\mathcal{A}$ outputs $b' \in \{0,1\}$. The output of the experiment is 1 if $b' = b$, and 0 otherwise.

The encoding scheme $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec})$ is an strong non-malleable if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function negl such that,

$$\Pr[\mathrm{SNMLR}_{\mathcal{A},l,\mathcal{T}}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

Similar to strong non-malleability we can define l-leakage resilient q-continuous non-malleability [7] (for short (l,q)-CNMLR) based on experiment $\mathrm{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)$ as follows:

Definition II.2. The indistinguishability experiment $\mathrm{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)$:
(i) $\mathrm{Init}(1^n)$ is run to obtain public parameters Ω.
(ii) Adversary $\mathcal{A}$ is given Ω, and outputs a pair of legal messages m_0, m_1.
(iii) A random bit $b \in \{0,1\}$ is chosen, and $(s_0, s_1) \leftarrow \mathrm{Enc}(m_b)$ is computed.
(iv) The adversary $\mathcal{A}$ has ability to query the leakage oracles $O^l(s_0)$ and $O^l(s_1)$ to l bits.
(v) The adversary $\mathcal{A}$ can query the tampering oracle to maximum number of q queries. The one sample query is as follows:
  a) Adversary $\mathcal{A}$ sends Turing machines (T_0, T_1) for $T_0 \in \mathcal{T}$ and $T_1 \in \mathcal{T}$ to the tamping oracle.
  b) $x'_0 := T_0(x_0)$ and $x'_1 := T_1(x_1)$ is computed.
  c) The value of $x' := \mathrm{Dec}(x'_0, x'_1)$ is computed.
  d) If $(x_0, x_1) = (x'_0, x'_1)$ then tampering oracle returns $same^*$; else, outputs x'.
  e) If $x' = \perp$, the tampering oracle goes to the self-destruction mode. (The *self-destruction* meas that the oracle will answer $\perp$ to any other query.)
(vi) $\mathcal{A}$ outputs $b' \in \{0,1\}$. The output of the experiment is 1 if $b' = b$, and 0 otherwise.

The encoding scheme $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec})$ is an l-leakage resilient continuous non-malleable if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function negl such that,

$$\Pr[\mathrm{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

Remark II.3. It is required that the continuous non-malleable scheme have to satisfy the *uniqueness* property. This means that for any share of a codeword x_0 it is hard to find two corresponding shares x_1 and x_2 such that both (x_0, x_1) and (x_0, x_2) make a valid codeword [7].

III. Continuous non-malleable coding scheme

The FMNV scheme [7] and its security are described as follows.

Construction III.1. (FMNV scheme).
The FMNV encoding scheme $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec})$ is based on a strong Leakage resilient storage (SLRS), a collision resistant hash function and a robust non-interactive zero knowledge in the CRS model. We show hash function family with $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^k\}$, robust NIZK proof for the language $\mathcal{L}_{t,\mathcal{H}} = \{h : \exists s \text{ such that } h = H_t(s)\}$ with $\Pi' = (\mathrm{Init}', P, V)$. Let $\Pi'' = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ be strong l'-leakage resilient storage, and q be the maximum number of queries that an adversary can issue to the tampering oracle. This coding scheme is a tuple $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec})$, that is defined as follows:

- $\mathrm{Init}(1^n)$: Choose uniform $t \in_R \{0,1\}^k$ and run $\Omega \leftarrow \mathrm{Init}$.

- $\mathrm{Enc}(\Omega, x)$:
  1) Compute $(s_0, s_1) \leftarrow \mathrm{LRS}(x)$, $h_0 = H_t(s_0)$, $h_1 = H_t(s_1)$, $\lambda_0 = h_0$, $\lambda_1 = h_1$, $\pi_0 = P^{\lambda_1}(\Omega, h_0, s_0)$ and $\pi_1 = P^{\lambda_0}(\Omega, h_1, s_1)$.
  2) Let the two split encoding shares be $X_0 = (s_0, h_1, \pi_0, \pi_1)$ and $X_1 = (s_1, h_0, \pi_0, \pi_1)$.
- $\mathrm{Dec}(X_0, X_1)$:
  1) Parse X_b as $(s_b, h_{1-b}, \pi_0, \pi_1)$ for $b \in \{0, 1\}$;
  2) Run the *local check* as the verification of $V^{\lambda_1}(\Omega, h_0, \pi_0)$ and $V^{\lambda_0}(\Omega, h_1, \pi_1)$.
  3) Run the *cross check* as the verification of $h_0 \stackrel{?}{=} H_t(s_0)$, $h_1 \stackrel{?}{=} H_t(s_1)$ and equality of π_0, π_1 in the two shares.
  4) If each of the verifications fails return $\perp$; else, output $\mathrm{LRS}^{-1}(s_0, s_1)$.

The security of Construction III.1 is defined in Theorem III.1.

Theorem III.1 ([7]). *The scheme of Construction III.1 is l-leakage resilient strong q-continuous non-malleable where $(\mathrm{LRS}, \mathrm{LRS}^{-1})$ be an l'-leakage-resilient strong storage, $\mathcal{H}$ is a family of collision resistant hash functions with output length of k bits, (Init', P, V) is a robust NIZK proof system for language $\mathcal{L}_{t,\mathcal{H}}$, $q = poly(n)$ sufficiently large and $l' \geq 2l + (k+1)\log(q)$.*

IV. An efficient continuous non-malleable encoding scheme

In this section we prove a better result for Theorem III.1 by using a new method of proof. We formalize this via a proof by reduction, in which we show how to use any efficient adversary $\mathcal{A}$ to construct another efficient adversary $\mathcal{A}'$ such that if $\mathcal{A}$ violates the security of $\mathrm{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)$, then $\mathcal{A}'$ breaks the definition of indistinguishability for $\mathrm{Leakage}_{\mathcal{A}',l',\theta}(n)$. The main difficulty of the reduction is how the adversary of LRS can simulate the answers of tampering queries without knowing the challenge codeword. In [7] the proof contains an involved PPT algorithm for finding the round of self-destruction. This algorithm needs to access the leakage oracles and requires rather large amount of leakage of bits. Our new method instead of running an algorithm for finding the exact index of self-destruction makes a guess as to which index (from among the q tampering queries) will correspond to the self-destruction. We can guess with exact probability $1/q$ the correct index of self-destruction. For completion of the proof we require knowing the correctness of our guess.

Theorem IV.1. *Let $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec})$ be a tuple of PPT algorithms as the scheme of Construction III.1, $\Pi'' = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ be an l'-leakage resilient storage, $\mathcal{H}$ a collision resistant hash function family with output length k and q be the maximum number of tampering queries as a polynomial function of security parameter n. Then the scheme Π is (l, q) strong continuous non-malleable encoding scheme for $l' \geq 2l + 2k + 1$.*

Proof. We show that if an adversary $\mathcal{A}$ distinguishes m_0 from m_1 in the experiment strong continuous non-malleability, $\mathrm{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)$, with non-negligible probability, then there exists another adversary $\mathcal{A}'$ that distinguishes the same massages in the leakage-resilient storage experiment, $\mathrm{Leakage}_{\mathcal{A}',l',\theta}(n)$. The formal description of the reduction is as follows.

Let $\mathcal{A}$ be a probabilistic polynomial-time adversary that

$$\Pr[\mathrm{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)] \geq 1/2 + \epsilon(n), \tag{1}$$

for a non-negligible function ϵ.

Consider the following PPT adversary $\mathcal{A}'$ that attempts to solve the $\mathrm{Leakage}_{\mathcal{A}',l',\theta}(n)$.

1) $\mathcal{A}'$ chooses uniformly t as a index of a family of hash functions and runs $(\Omega, tk, ek) \leftarrow \mathrm{Sim}_1(1^n)$.
2) $\mathcal{A}'$ chooses the randomness r.
3) $\mathcal{A}'$ runs the algorithm $\mathcal{A}(\Omega, t, r)$ and gets the two messages m_0 and m_1.
4) $\mathcal{A}'$ runs the strong l'-leakage-resilient storage experiment with messages m_0 and m_1.
5) Adversary $\mathcal{A}'$ is given access to leakage oracles $O^{l'}(s_0)$ and $O^{l'}(s_1)$ for $(s_0, s_1) \leftarrow \mathrm{LRS}(m_b)$ for randomly chosen bit b.
6) $\mathcal{A}'$ with access to its leakage oracles can obtain the $h_0 := H_t(s_0)$ and $h_1 := H_t(s_1)$ (note that this is possible since $l' > k$).
7) $\mathcal{A}'$ sets the $X_\theta = (s_\theta, h_{1-\theta}, \pi_0, \pi_1)$, where $\pi_b \leftarrow \mathrm{Sim}_2^{1-\lambda_b}(\Omega, h_b, tk)$ for $b \in \{0, 1\}$ are simulated robust NIZK proofs (as Construction III.1) for $h_0 := H_t(s_0)$, $\lambda_1 = h_1$, $h_1 := H_t(s_1)$ and $\lambda_0 = h_0$ respectively.
8) $\mathcal{A}'$ runs the algorithm $\mathrm{CalcLeakage}(\Omega, t, h_0, h_1, \pi_0, \pi_1, r)$ and is given two vectors Θ_0, Θ_1 (this algorithm is execute inside of leakage oracles $O^{l'}(s_0)$, $O^{l'}(s_1)$ and simulates the leakage queries of adversary $\mathcal{A}$).
9) $\mathcal{A}'$ chooses $j^* \in_R \{0, 1, \ldots, q\}$ (The index j^* is the first tampering query leading to $\perp$ in the decoding).
10) Check the correctness of our guess for j^*:
    Run the algorithm $\mathrm{VrfyTamper}(\Omega, t, h_0, h_1, \pi_0, \pi_1, \Theta_0, \Theta_1, j^*, r)$ and the output of the algorithm is a True or False.
    a) If the output is False then halt the algorithm and output the randomly chosen bit $b \in_R \{0, 1\}$.
    b) If the output is True then continue.
11) Now the s_θ is given to $\mathcal{A}'$ for $\theta \in \{0, 1\}$ (the access of adversary to the leakage oracle is terminated).
12) $\mathcal{A}'$ answers the ith leakage queries of $\mathcal{A}$ for Turing machines L_0,L_1 with $\Theta_0[i]$ and $\Theta_1[i]$. (Note that if $\Theta_b[i] = \perp^*$ then stop the answering of leakage queries for other steps.)
13) $\mathcal{A}'$ continues interaction with $\mathcal{A}$, answering its ith tampering query T_0,T_1 as follows:
    a) For $i < j^*$, compute $X'_\theta = T_\theta(X_\theta) = (s'_\theta, h'_{1-\theta}, \pi'_0, \pi'_1)$
       i) If $X'_\theta = X_\theta$, return the $same^*$.
       ii) Else compute $s'_{1-\theta} \leftarrow \mathrm{Ext}(\Omega, (h'_{1-\theta}, \pi'_{1-\theta}), ek)$ and define $X'_{1-\theta} = (s'_{1-\theta}, h'_\theta, \pi'_0, \pi'_1)$; finally, return $(X'_\theta, X'_{1-\theta})$.
    b) For $i \geq j^*$ return the $\perp$.

14) $\mathcal{A}$ outputs the bit b' as the result of strong continuous non-malleable experiment and then $\mathcal{A}'$ also outputs the same result as his/her output.

The pseudo code of algorithm CalcLeakage is described in Algorithm 1 and its sub algorithm SubLeakage is described in Algorithm 2.

Algorithm 1: CalcLeakage$(\Omega, t, h_0, h_1, \pi_0, \pi_1, r)$

Set $i_0 \leftarrow 0$, $i_1 \leftarrow 0$.
for $i \leftarrow 0$ **to** q **do**
  $\theta_0[i] = \emptyset$
  $\theta_1[i] = \emptyset$
end
/* Note that Θ_0 and Θ_1 are global vectors. */
/* Note that $\perp^*$ is a special symbol for indication of leakage queries termination. */
Loop
  Query the algorithm SubLeakage$(\Omega, t, , h_0, h_1, \pi_0, \pi_1, 0, r)$ to leakage oracle $O^{l'}(s_0)$ and receives the α and set $\Theta_0[i_0] = \alpha$
  $i_0 = i_0 + 1$
  if $\alpha = \perp^*$ **then**
    Halt and **return** Θ_0 and Θ_1
  end
  Query the algorithm SubLeakage$(\Omega, t, , h_0, h_1, \pi_0, \pi_1, 1, r)$ to leakage oracle $O^{l'}(s_1)$ and receive the α and set $\Theta_1[i_1] = \alpha$
  $i_1 = i_1 + 1$
  if $\alpha = \perp^*$ **then**
    Halt and **return** Θ_0 and Θ_1
  end
EndLoop

Algorithm 2: SubLeakage$(\Omega, t, , h_0, h_1, \pi_0, \pi_1, b, r)$

Set $e \leftarrow 0$
Run following algorithm inside of the oracle $O^{l'}(s_b)$.
Run the $\mathcal{A}(\Omega, t, r)$ and receive m_0 and m_1
Set the $X_b = (s_b, h_{1-b}, \pi_0, \pi_1)$
Answer the ith tampering query T_0, T_1 as follows:
begin Answring Tampering queries:
  compute $X'_b = T_b(X_b) = (s'_b, h'_{1-b}, \pi'_0, \pi'_1)$
  if $X'_b = X_b$ **then**
    return $same^*$ to $\mathcal{A}$
  end
  else if $X'_b \neq X_b$ ***AND*** *local check on* X'_b *fails* **then**
    return $\perp$
  end
  else if $X'_b \neq X_b$ ***AND*** $\pi'_{1-b} \neq \pi_{1-b}$ **then**
    return $\perp$
  end
  else
    Compute $s'_{1-b} \leftarrow \text{Ext}(\Omega, (h'_b, \pi'_b), ek)$ and then return (X'_b, X'_{1-b}) to $\mathcal{A}$, where $X'_{1-b} = (s'_{1-b}, h'_b, \pi'_0, \pi'_1)$
  end
end
Answer the ith leakage query L_0, L_1 as follows:
begin Answring leakage queries:
  if $\Theta_0[i] \neq \emptyset$ *and* $\Theta_1[i] \neq \emptyset$ **then**
    return $\Theta_0[i]$ and $\Theta_0[i]$
  end
  else if $\Theta_b[i] = \emptyset$ **then**
    Compute $\alpha = T_b(s_b, h_{1-b}, \pi_0, \pi_1)$ and **return** α
  end
  else if *We reach to the maximum limit of lekage queires (l)* **then**
    Halt and **return** $\perp^*$
  end
end

The pseudo code of algorithm VrfyTamper is described in Algorithm 3.

In order to complete our proof, consider these points:

1) We replace NIZK with $(\text{Sim}_1, \text{Sim}_2)$ because the zero-knowledge property of NIZK proof system [7].
2) We fix the randomness of adversary A by choosing randomness r, and then this adversary will be a deterministic algorithm.
3) The behavior of algorithm CalcLeakage is a precise simulator for the adversary $\mathcal{A}$ with the randomness r in the experiment $\text{Leakage}_{\mathcal{A}',l',\theta}(n)$. Hence we can conclude that vectors Θ_0 , Θ_1 are exact results of leakage queries. (Note that this part of proof is similar to [7].)
4) $\mathcal{A}'$ with probability exact $1/q$ guess the index of tampering queries leading to $\perp$.
5) $\mathcal{A}'$ answers the $i < j^*$ tampering queries by using values of X_θ and X'_θ.
6) The answers of $i < j^*$ tampering query is $same^*$, when $X_\theta = X'_\theta$. Note that to the *uniqueness* property of encoding scheme. Also, the answers of tampering queries are not $\perp$.
7) The answers of $i < j^*$ tampering query is $x' \notin \{same^*, \perp\}$, when $X_\theta \neq X'_\theta$ and local checks verify. Note that the answer of tampering query is not $\perp$ and we can use the Ext algorithm to obtain s'_0 and s'_1.
8) The algorithm VrfyTamper verifies the correctness of our guess for index j^*. Our guess with probability $1/q$ is correct and with probability $(q-1)/q$ is incorrect and in this case we output a random output.
9) The algorithm VrfyTamper requires $2k+1$ bits of leakage for verifying the correctness of self-destruction index. Note

Algorithm 3: VrfyTamper$(\Omega, t, h_0, h_1, \pi_0, \pi_1, \Theta_0, \Theta_1, j^*, r)$

Sample a hash function $H_t \leftarrow \mathcal{H}$.
Run $\mathcal{A}(\Omega, t, r)$ inside of the oracle $O^{l'}(s_0)$.
begin Answring leakage and tampering queries:
    Answer the leakage quires with Θ_0 and Θ_1.
    Answer the tampering queries similar to Algorithm 2.
    Compute the hash value of a vector of $j^* - 1$ tampering queries by using H_t and set it in η_0.
    Return η_0.
    Note that this step of algorithm requires k bits.
end
Run $\mathcal{A}(\Omega, t, r)$ inside of the oracle $O^{l'}(s_1)$.
begin Answring leakage and tampering queries:
    Answer the leakage quires with Θ_0 and Θ_1.
    Answer the tampering queries similar to Algorithm 2.
    Compute the hash value of a vector of $j^* - 1$ tampering queries by using H_t and set it in η_1.
    If $\eta_0 \neq \eta_1$ halt the Algorithm 3 and return False.
    Compute the hash value of j^*th tampering query by using H_t and set it in ζ_1.
    Return ζ_1.
    Note that this step of algorithm requires at most k bits.
end
Run $\mathcal{A}(\Omega, t, r)$ inside of the oracle $O^{l'}(s_0)$.
begin Answring leakage and tampering queries:
    Answer the leakage quires with Θ_0 and Θ_1.
    Answer the tampering queries similar to Algorithm 2.
    Compute the hash value of j^*th tampering query by using H_t and set it in ζ_0.
    If $\zeta_0 \neq \zeta_1$ halt the Algorithm 3 and return True.
    If $\zeta_0 = \zeta_1$ halt the Algorithm 3 and return False.
    Note that this step of algorithm requires 1 bit.
end

that the decoding of Construction III.1 is $\perp$, when the two shares of codeword in our reduction decode to different answers. Algorithm VrfyTamper checks the equality of $j^* - 1$ tampering queries and inequality of j^*th tampering query.

10) If $\mathcal{A}$ wins then $\mathcal{A}'$ also wins.

Based on above notes, we can conclude that:

$$\begin{aligned} \Pr[\text{Leakage}_{\mathcal{A}',l',\theta}(n)] &= 1/2 \times (q-1)/q \\ &+ 1/q \times \Pr[\text{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)]. \end{aligned} \tag{2}$$

Using Equations 1 and 2, we thus have

$$\Pr[\text{Leakage}_{\mathcal{A}',l',\theta}(n)] \geq \frac{(q-1)}{2q} + 1/q(1/2+\epsilon(n)) = 1/2+\epsilon/q.$$

Because the q is a polynomial function then ϵ/q is a non-negligible function and this is in contradiction to the assumption that the problem $\text{Leakage}_{\mathcal{A}',l',\theta}(n)$ is hard. ■

V. Conclusion

Tamper-resilient cryptography is a method to provably protect memory and cryptographic functionalities against a specific class of tampering and leakage attacks. The non-malleable encoding schemes are a keyless cryptographic primitive for handling tampering attacks. This paper shows that different viewpoints to a specific problem can lead to different results for the same problem. In this paper, we use another method for proving the security of Construction III.1 which leads to a more efficient scheme than previous. Our new proof shows that the FMNV scheme can be constructed with a more effective leakage resilient storage scheme.

References

[1] F. Davì, S. Dziembowski, and D. Venturi, "Leakage-resilient storage," in *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Proceedings*, vol. 6280 of *Lecture Notes in Computer Science*, pp. 121–137, Springer, 2010.

[2] M. Cheraghchi and V. Guruswami, "Non-malleable coding against bit-wise and split-state tampering," in *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, Proceedings*, vol. 8349 of *Lecture Notes in Computer Science*, pp. 440–464, Springer, 2014.

[3] S. G. Choi, A. Kiayias, and T. Malkin, "Bitr: Built-in tamper resilience," in *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, vol. 7073 of *Lecture Notes in Computer Science*, pp. 740–758, Springer, 2011.

[4] F. Liu and A. Lysyanskaya, "Tamper and leakage resilience in the split-state model," in *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Proceedings*, vol. 7417 of *Lecture Notes in Computer Science*, pp. 517–532, Springer, 2012.

[5] M. Cheraghchi and V. Guruswami, "Capacity of non-malleable codes," *CoRR*, vol. abs/1309.0458, 2013.

[6] S. Faust, P. Mukherjee, D. Venturi, and D. Wichs, "Efficient non-malleable codes and key-derivation for poly-size tampering circuits," in *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, vol. 8441 of *Lecture Notes in Computer Science*, pp. 111–128, Springer, 2014.

[7] S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi, "Continuous non-malleable codes," in *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, Proceedings*, vol. 8349 of *Lecture Notes in Computer Science*, pp. 465–488, Springer, 2014.

[8] D. Aggarwal, Y. Dodis, T. Kazana, and M. Obremski, "Non-malleable reductions and applications," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pp. 459–468, ACM, 2015.

[9] N. Chandran, V. Goyal, P. Mukherjee, O. Pandey, and J. Upadhyay, "Block-wise non-malleable codes," *IACR Cryptology ePrint Archive*, 2015.

[10] S. Dziembowski, T. Kazana, and M. Obremski, "Non-malleable codes from two-source extractors," in *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Proceedings, Part II*, vol. 8043 of *Lecture Notes in Computer Science*, pp. 239–257, Springer, 2013.

[11] D. Aggarwal, Y. Dodis, and S. Lovett, "Non-malleable codes from additive combinatorics," in *Symposium on Theory of Computing, STOC 2014*, pp. 774–783, ACM, 2014.

[12] S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran, "Explicit non-malleable codes resistant to permutations," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 21, p. 69, 2014.

[13] P. Austrin, K. Chung, M. Mahmoody, R. Pass, and K. Seth, "On the impossibility of cryptography with tamperable randomness," in *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Proceedings, Part I*, vol. 8616 of *Lecture Notes in Computer Science*, pp. 462–479, Springer, 2014.

[14] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner, "Private circuits II: keeping secrets in tamperable circuits," in *Advances in Cryptology - EUROCRYPT* 2006*,* 25*th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, vol. 4004 of *Lecture Notes in Computer Science*, pp. 308–327, Springer, 2006.

[15] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology - CRYPTO* 2003*,* 23*rd Annual International Cryptology Conference, Proceedings*, vol. 2729 of *Lecture Notes in Computer Science*, pp. 463–481, Springer, 2003.

[16] D. Dachman-Soled, F. Liu, E. Shi, and H. Zhou, "Locally decodable and updatable non-malleable codes and their applications," in *Theory of Cryptography* - 12th Theory of Cryptography Conference, TCC 2015, Proceedings, Part I, vol. 9014 of *Lecture Notes in Computer Science*, pp. 427–450, Springer, 2015.

[17] S. Coretti, U. Maurer, B. Tackmann, and D. Venturi, "From single-bit to multi-bit public-key encryption via non-malleable codes," in *Theory of Cryptography* - 12*th Theory of Cryptography Conference, TCC* 2015*, Proceedings, Part I*, vol. 9014 of *Lecture Notes in Computer Science*, pp. 532–560, Springer, 2015.

[18] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, ch. 3. Chapman & Hall/CRC Press, 2 ed., 2015.

[19] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai, "Robust non-interactive zero knowledge," in *Advances in Cryptology - CRYPTO* 2001*,* 21*st Annual International Cryptology Conference, Santa Barbara,* 2001, vol. 2139 of *Lecture Notes in Computer Science*, pp. 566–598, Springer, 2001.