The ISC Int'l Journal of Information Security

ISeCure

January 2017, Volume 9, Number 1 (pp. 41–51)

http://www.isecure-journal.org

A New Security Proof for FMNV Continuous Non-malleable Encoding Scheme☆

Amir S. Mortazavi [1], Mahmoud Salmasizadeh [2,*], and Amir Daneshgar [3]

[1] *Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran*

[2] *Electronics Research Institute, Department of EE as adjunct member, Sharif University of Technology, Tehran, Iran*

[3] *Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 13 November 2016
Revised: 15 January 2017
Accepted: 31 January 2017
Published Online: 31 January 2017

Keywords:
Non-malleable, Continuous Non-malleability, Tamper-resilient Cryptography, Leakage-resilient.

ABSTRACT

A non-malleable code is a variant of an encoding scheme which is resilient to tampering attacks. The main idea behind non-malleable coding is that the adversary should not be able to obtain any valuable information about the message. Non-malleable codes are used in tamper-resilient cryptography and protecting memories against tampering attacks. Many different types of non-malleability have already been formalized and defined in current literature, among which continuous non-malleability is the setup in which the messages are protected against adversaries who may issue polynomially many tampering queries. The first continuous non-malleable encoding scheme has been proposed by Faust *et al.* (FMNV) in 2014. In this article, we propose a new proof of *continuous non-malleability* of the FMNV scheme. The new proof will give rise to an improved and *more efficient* version of this scheme. Also, the new proof shows that one may achieve continuous non-malleability of the same security by using a *leakage resilient storage* scheme with fewer bits for the leakage bound. This shows that the new scheme is more efficient and practical for tamper-resilient applications.

1 Introduction

Hardware attacks are considered to be dangerous for cryptographic devices, which are usually divided into active or passive attacks. Passive attacks are based on a 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 itself. In a *tampering attack* the adversary is assumed to have the ability to modify and manipulate some parameters of the system, where tamper-resilient cryptography considers a theoretical study of such attacks. Tampering attacks can be implemented with malwares, viruses and adversaries with the ability of access to memory or injection faults and related key attacks as variants of such attacks. An adversary applying a tampering attack can replace a parameter (e.g. secret key) of a cryptographic scheme with another related value and based on the new outputs of the scheme can infer some information about that parameter (e.g. secret key).

Non-malleable coding is among countermeasures against tampering attacks as a keyless coding scheme

☆ This article is an extended version of an ISCISC'13 paper.

* Corresponding author.

Email addresses: sa_mortazavi@ee.sharif.edu (A.S. Mortazavi), salmasi@sharif.ir (M. Salmasizadeh), daneshgar@sharif.ir (A. Daneshgar)

ISSN: 2008-2045

designed to protect some critical parameters of the scheme against the attack. The main goal of non-malleable coding is to resist against an active adversary that has the ability to modify a codeword according to a family of Turing machines (i.e. algorithms). In non-malleable coding, as a variant of encoding schemes, one allows a message m to be encoded into a codeword c, such that c can resist against tampering attacks. A coding scheme is non-malleable if the *tampered codeword* does not provide any valuable information about the original message (i.e. the message embedded in the codeword). It is worthwhile to say that non-malleable codes can be used in tamper-resilient cryptography or protecting memories against tampering attacks. To point out some applications one may note that non-malleable coding schemes provide an algorithmic procedure against tampering attacks for cryptographic schemes embedded in hardware implementations such as smart cards and hardware security modules whose soundness is dependent on satisfaction of required assumptions of non-malleable schemes if they exist. Hence, it is recommended to use this type of tamper proofs in an extra layer of security accompanied by hardware tamper proof solutions.

From an abstract viewpoint, tampering attacks can only be defined for a specific subclass of Turing machines (rather than the whole set of such machines). Informally, the concept of security as non-malleability is defined based on an experiment between an adversary and a simulator Sim. For this, let's assume that Enc and Dec are encoding and decoding algorithms for a non-malleable encoding scheme. The adversary selects its message m and a codeword c is computed within the experiment such that $c \leftarrow \mathrm{Enc}(m)$. The adversary then issues a tampering Turing machine T (of the predefined type) and within the experiment either one computes $m' := \mathrm{Dec}(T(c))$ or $m' \leftarrow \mathrm{Sim}(T)$ and then m' is given to the adversary. The encoding scheme is called secure if for all such adversaries there exist an efficient simulator such that the adversary cannot win the above experiment for non-trivial tampering queries except with negligible probability.

It is straightforward to show that non-malleable codes do not exist for the family of all efficient tampering Turing machines because a decoding algorithm is always a member of such family. Thus we have to restrict the class of tampering attacks in a well-defined setup. The *split-state model* is a class of tampering Turing machines for which efficient constructions of non-malleable codes are known, in which the codeword has several parts and each part is tampered with independently. If we can satisfy the independence of tampering attacks on each part of the codeword, then this type of coding can be used in practical purposes. Using two separated and isolated memories can mitigate and relax this required assumption.

There exists a variety of definitions for non-malleability in current literature, where the security is based on the indistinguishability in general. For instance, the one-time non-malleability considers only one tampering where continuous non-malleability allows polynomially many tamping attacks.

In an information theoretical setting for continuous non-malleability which is presented in [2], the adversary is allowed to issue an arbitrary function $f : \{0,1\}^n \to \{0,1\}^n$ as a tampering query, and in [2] it is shown that an unbounded adversary can always break the security of non-malleability. Therefore, this result shows that the computational type of continuous non-malleability is the only scenario in which one may think of a secure continuously non-malleable encoding scheme.

In this paper we study the Faust et al. [2] (FMNV for short) scheme and show that we can prove the continuous non-malleability for this scheme with a 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 of a new method for finding the *self-destruction* round of tampering queries with $2k + 1$ bits of leakage while the original proof uses $2k \log(q)$ bits of leakage from both oracles (where q is the number of tampering queries and k is the length of hash function output). The amount of the leakage in the new scheme is independent of the maximum allowed number of tampering queries (q) and is a constant value. It means that for a fixed and same security level, our scheme is more efficient, memory-wise, and requires a cryptographic primitive with lower resistance against the leakage.

In Section 2 we present a formal definition for some required primitives. In Section 3 we introduce the FMNV scheme for a continuous non-malleable scheme, and finally, in Section 4 we compare our proof with the original one and present a new proof for the former scheme.

1.1 Related Work

In [3] the arithmetic manipulation detection codes (AMD codes) are introduced as a keyless and special type of non-malleable codes against a family of tampering Turing machines with an algebraic structure. The general and formal definition of non-malleable codes is first introduced by Dziembowski et al. [4] for the bit-wise family of Turing machines which can tamper with every bit of the codeword independent of other bits. In [5] an efficient non-malleable scheme

for bit-wise family of tampering is introduced.

The first non-malleable code for a block-wise tampering is introduced in [6]. Liu and Lysyanskaya [7] introduced the first non-malleable code in the split-state model for all PPT Turing machines and also considered the leakage of the codeword. Non-malleable codes for a family of Turing machines of size $2^{\text{poly}(n)}$ has been studied in [8, 9]. Faust et al. [2] extended the definition of non-malleability in a way to include continuity and proposed a scheme in the CRS model. Aggarwal et al. [10] introduced a generalization of non-malleable codes, called non-malleable reductions. Chandran et al. [11] 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 is introduced by Dziembowski et al. [12] for only one-bit messages, in [13] it is extended to multi-bit messages and in [14, 15] is extended to constant rates. See also [9, 16] for other contributions related to the information theoretic model.

Austrin et al. [17] studied the effect of tampering on randomness of cryptographic algorithms. In [18, 19], [20] and [21] constructions for secure and non-malleable public key, symmetric key and commitment schemes are presented, respectively. Some contributions as [2, 7, 22, 23] also focus on the applications of non-malleable codes for tamper-resilient cryptography. Dachman-Soled et al. [24] studied the problem of securing RAM computation against memory tampering and leakage attacks. Coretti et al. [25] showed that non-malleable codes can be used to construct a CCA secure public key.

1.2 Notations

Given a set $\mathcal{S}$, we write $a \in_R \mathcal{S}$ to denote sampling an element a from set $\mathcal{S}$ uniformly at random. We use $:=$ to denote deterministic assignment and $\leftarrow$ for the probabilistic assignment. We use the shorthand PPT for the phrase *probabilistic polynomial-time*. The symbol negl is used to refer to a typical negligible function, that grows smaller than $\frac{1}{p(n)}$ for any polynomial $p(n)$ [26]. Hereafter, $|f|$ stands for the output size of the function f and FMNV is used as a shorthand to refer to the reference Faust et al. [2].

The leakage oracle $O^l(k)$ interacts with the adversary that can query adaptively leakage Turing machines L_i and receive $L_i(k)$ until $\sum_i(|L_i(k)|) \leq l$ for the secret parameter k.

2 Preliminaries

2.1 Zero knowledge Proofs

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

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

$$\text{Real}(n) = \left\{ \ \Omega \leftarrow \text{Init}(1^n); X \leftarrow A^{P(\Omega,\ldots)}(\Omega) : X \ \right\},$$

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

There are several models of NIZK proofs that have similar definitions as above. In this paper, we use *robust* non-interactive NIZK proofs [27]. This type of NIZK has an extra *Extractability* property which for all PPT adversaries there exist an efficient algorithm Ext such as:

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

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

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

Remark 2. In [2], the robust NIZK proofs require to support an extra condition in terms of excessive data called labels. The labels are public strings and can be used as inputs to P, V, Ext and Sim_2. This property can be achieved by concatenating the label λ to the statement x, in which case we show the NIZK algorithms as P^λ, V^λ, Ext^λ and Sim_2^λ.

Remark 3. There exists constructions for robust NIZK's as is proposed in [27]. The ingredients of this construction are commitment schemes, pseudo-

random functions and one-time signatures.

2.2 Leakage Resilient Storage

A *leakage resilient storage* encoding system $\Pi = (\text{LRS}, \text{LRS}^{-1})$ is defined in [4]. In this, Π contains a pair of computable PPT algorithms where for messages $x \in \{0,1\}^m$:

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

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

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

The indistinguishability experiment $\text{Leakage}_{\mathcal{A},l}(n)$ is as follows.

(1) The adversary $\mathcal{A}$ is given public parameters, and outputs a pair of messages m_0, m_1 in the message space.
(2) A uniform bit $b \in \{0,1\}$ is chosen, and then a codeword $(s_0, s_1) \leftarrow \text{LRS}(m_b)$ is computed.
(3) 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.
(4) $\mathcal{A}$ outputs a bit b'. The output of the experiment is 1 if $b' = b$, and 0 otherwise.

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

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

2.3 Strong Leakage Resilient Storage

The encoding scheme $\Pi = (\text{LRS}, \text{LRS}^{-1})$ is an *strong* l-leakage resilient storage scheme [2], if for $\theta \in \{0,1\}$ and every PPT adversary:

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

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

(1) Adversary $\mathcal{A}$ is given public parameters, and outputs a pair of messages m_0, m_1 in the message space.
(2) A uniform bit $b \in \{0,1\}$ is chosen, and then a codeword $(s_0, s_1) \leftarrow \text{Enc}(m_b)$ is computed.
(3) Adversary $\mathcal{A}$ can interact with the leakage oracles $O^l(s_0)$ and $O^l(s_1)$.
(4) After finishing leakage queries, $\mathcal{A}$ is given s_θ.
(5) $\mathcal{A}$ 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 [2, 4] by using *inner product* in finite fields, while it can be shown that the inner product based LRS schemes are also secure in the *information-theoretic* model.

2.4 Non-malleable Codes

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

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

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

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

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

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, a 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 $\text{SNMLR}_{\mathcal{A},l,\mathcal{T}}(n)$ as follows [7]:

Definition 2.1. The indistinguishability experiment $\text{SNMLR}_{\mathcal{A},l,\mathcal{T}}(n)$:

(1) $\text{Init}(1^n)$ is run to obtain public parameters Ω.
(2) Adversary $\mathcal{A}$ is given Ω, and outputs a pair of legal messages m_0, m_1.
(3) A random bit $b \in \{0,1\}$ is chosen, and $(x_0, x_1) \leftarrow \text{Enc}(m_b)$ is computed.
(4) The adversary $\mathcal{A}$ has ability to query the leakage oracles $O^l(x_0)$ and $O^l(x_1)$ to l bits.
(5) Send Turing machines (T_0, T_1) for $T_0 \in \mathcal{T}$ and $T_1 \in \mathcal{T}$ as a tampering query (note that this step is performed once by adversary).
   (a) $x'_0 := T_0(x_0)$, $x'_1 := T_1(x_1)$ and $x' := \text{Dec}(x'_0, x'_1)$ are computed.
   (b) If $(x_0, x_1) = (x'_0, x'_1)$ then the adversary is given $same^*$; else, is given x'.
(6) $\mathcal{A}$ outputs $b' \in \{0,1\}$. The output of the experiment is 1 if $b' = b$, and 0 otherwise.

The encoding scheme $\Pi = (\text{Init}, \text{Enc}, \text{Dec})$ is strong non-malleable if for all probabilistic polynomial-time adversaries $\mathcal{A}$ if 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 [2] (for short (l,q)-CNMLR) based on experiment $\mathrm{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)$ as follows:

Definition 2.2. The indistinguishability experiment $\mathrm{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)$:

(1) $\mathrm{Init}(1^n)$ is run to obtain public parameters Ω.
(2) Adversary $\mathcal{A}$ is given Ω, and outputs a pair of legal messages m_0, m_1.
(3) A random bit $b\in\{0,1\}$ is chosen, and $(x_0,x_1)\leftarrow\mathrm{Enc}(m_b)$ is computed.
(4) The adversary $\mathcal{A}$ has ability to query the leakage oracles $O^l(x_0)$ and $O^l(x_1)$ to l bits.
(5) 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.)
(6) $\mathcal{A}$ 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 l-leakage resilient and 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 4. It is required that continuous non-malleable schemes have to satisfy the *uniqueness* property. This means that for any share of a codeword x_0 it must be 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 [2]. If we assume that one can find two valid codewords (x_0,x_1) and (x_0,x_2) with $x_1\neq x_2$, then he/she can successfully recover X_1 from the target codeword (X_0,X_1) with asking a polynomially bounded number of queries to the tampering oracle. The adversary for finding the ith bit of the X_1 issues the following tampering query:
The adversary replaces the first share of the codeword with x_0 and replaces the second share of codeword with x_1 if the ith bit of the second part is 0, otherwise replaces with x_2. Based on the answer of this query the value of the ith bit is revealed.
The adversary can repeat this algorithm to find all bits of the second share. This shows that the uniqueness is required for Definition 2.2.

Remark 5. It is straightforward to show that it is impossible to create an encoding scheme in the information-theoretical setting of continuous non-malleable coding (Definition 2.2) according to the uniqueness property [2, Lemma 2]. In the information-theoretical definition of non-malleability the unbounded adversary can query a tampering Turing machine as follows:
Assume that the challenge codeword is (x_0,x_1). The adversary issues (T_0,T_1) as a tampering query such that T_0, knowing x_0, tries all possible x_1^* until $\mathrm{Dec}(x_0,x_1^*)\neq\perp$ and at this step the distinguishability can be easily signaled to the adversary.

3 Continuous non-malleable coding schemes

The FMNV scheme [2] and its security are described as follows.

Construction 3.1. (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. The hash function family is $\mathcal{H}=\{h:\{0,1\}^n\to\{0,1\}^k\}$, and the robust NIZK proof for the language $\mathcal{L}_{t,\mathcal{H}}=\{h:\exists s \text{ such that } h=H_t(s)\}$ is referred to as $\Pi'=(\mathrm{Init}_{NIZK},P,V)$. Let $\Pi''=\left(\mathrm{LRS},\mathrm{LRS}^{-1}\right)$ be a 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 $\Omega\leftarrow\mathrm{Init}_{NIZK}$.
- $\mathrm{Enc}(\Omega,x)$:
  (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 3.1 is established in Theorem 1.

Theorem 1 ([2]). *The scheme of Construction 3.1 is l-leakage resilient strong q-continuous non-malleable when $(\mathrm{LRS}, \mathrm{LRS}^{-1})$ is 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 the language $\mathcal{L}_{t,\mathcal{H}}$, $q = poly(n)$ for sufficiently large n and $l' \geq 2l + (k+1)\log(q)$.*

4 An efficient continuous non-malleable encoding scheme

In this section we improve Theorem 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 $\mathcal{A}'$ does not satisfy the definition of indistinguishability for $\mathrm{Leakage}_{\mathcal{A}',l',\theta}(n)$. The main difficulty of the reduction is to present a method that the adversary of LRS can simulate the answers of tampering queries without knowing the challenge codeword. In [2] the proof contains an involved PPT algorithm for finding the round of self-destruction. This algorithm needs to access the leakage oracles and requires a rather large amount of leakage bits. In our approach, instead of running an algorithm to find the exact index of self-destruction, we choose a randomized approach to make a guess for the index (from among the q tampering queries) that will correspond to self-destruction. Note that in this setting, the probability of such a guess is exactly $1/q$. To complete the proof we, of course, we need to verify the correctness of our guess which is an important part of the proof.

4.1 Outline of the original proof

First, we present a high-level overview of the original proof appeared in [2]. The proof is based on a reduction from the security of LRS scheme to the security of continuous non-malleability of the encoding scheme. It is shown that if the adversary $\mathcal{A}$ can break the security of continuous non-malleability, then there exists another adversary $\mathcal{A}'$ which can break the security of the LRS scheme. Note that based on the LRS experiment the adversary $\mathcal{A}'$ only has access to leakage oracles $O^l(s_0)$ and $O^l(s_1)$ (Section 2.2). the adversary $\mathcal{A}'$ should run $\mathcal{A}$ as a subroutine and simulate the leakage and tampering queries of $\mathcal{A}$. Since $\mathcal{A}'$ has no direct access to the challenge codeword, the simulation of the tampering queries of $\mathcal{A}$ is the most challenging part of the proof. The only relation of $\mathcal{A}'$ with the challenge codeword is via $O^l(s_0)$ and $O^l(s_1)$, independent of each other. For simulation of the leakage queries of $\mathcal{A}$, $\mathcal{A}'$ runs $\mathcal{A}(r)$ with randomness r several times inside of the leakage oracles to get the total leakage queries. To simulate the tampering queries of $\mathcal{A}$, $\mathcal{A}'$ answers based on the self-destruction point j^*. The adversary $\mathcal{A}'$ knows that for queries before j^* the tampering codeword is valid and therefore with knowing only one part of the codeword (e.g. $X_0 = (s_0, h_1, \pi_0, \pi_1)$) and based on the values of π_0 and π_1 and the extractability of NIZK the values of the other part are known and simulation is completed. In j^*th tampering query the local checks or global checks of the challenge codeword are not satisfied and this is the main point for finding j^*. Then $\mathcal{A}'$ knowing leakage queries runs $\mathcal{A}(r)$ with the same randomness as before inside of $O^l(s_0)$ and also inside of $O^l(s_1)$. Note that by the definition of self-destruction, j^* is an index for which the answer of the j^*th tampering query within $O^l(s_0)$ is different from the j^*th tampering query within $O^l(s_1)$. Therefore, $\mathcal{A}'$ runs the *standard binary search* algorithm to find the point at which the answers to the tampering queries are not equal. According to the complexity of standard binary search algorithm this step requires $\log(q)$ queries to the leakage oracles.

we summarize main steps of the proof as follow:

(1) According to the security of NIZK, the Init_{NIZK} is replaced with $(\Omega, tk, ek) \leftarrow \mathrm{Sim}_1$ and P with $\mathrm{Sim}_2^{\lambda}(\Omega, ., tk)$.
(2) $\mathcal{A}'$ runs $\mathcal{A}(r)$ with randomness r.
(3) $\mathcal{A}'$ with access to leakage oracles $O^l(s_0)$ and $O^l(s_1)$ computes the leakage queries of $\mathcal{A}$ by $2l$ bits of leakage.
   (a) $\mathcal{A}'$ runs the $\mathcal{A}$ with the same randomness r inside of leakage oracle $O^l(s_0)$. Note that inside of the $O^l(s_0)$ the adversary knows the value of s_0.
   (b) $\mathcal{A}'$ uses a simulated algorithm to answer the tampering queries inside of the $O^l(s_0)$ without knowing the value of the s_1. (This simulated algorithm is described in Algorithm 1.)
   (c) $\mathcal{A}'$ repeat the above algorithm inside of the leakage oracle $O^l(s_1)$.
   (d) $\mathcal{A}'$ runs the above steps alternatively to obtain all the leakage queries of $\mathcal{A}$.
(4) $\mathcal{A}'$ runs the $\mathcal{A}(r)$ inside of the leakage oracles and runs the standard binary search for finding the index in which the answers of tampering query from two leakage oracles are not equal with each other.
   (a) This step requires at most $\log(q)$ queries to the oracles.

Our proof is similar to the above proof except that instead of standard binary search we use guess and verification method. We guess the index j^* with prob-

ability $1/q$, however, for every guess we should also know whether our guess is true or not. If our guess is false we return a random bit as the answer and if our guess is true we do normally as before. We will be needing $2k+1$ bits for testing the correctness of j^*.

Theorem 2. *Let* $\Pi = (\text{Init}, \text{Enc}, \text{Dec})$ *be a tuple of* PPT *algorithms as in the scheme of Construction* 3.1, $\Pi'' = (\text{LRS}, \text{LRS}^{-1})$ *be an* l'*-leakage resilient storage,* $\mathcal{H}$ *a family of collision resistant hash functions 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 an* (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 of strong continuous non-malleability, $\text{CNMLR}_{\mathcal{A},l,\mathcal{T},q}(n)$, with non-negligible probability, then there exists another adversary $\mathcal{A}'$ that distinguishes the same messages in the leakage-resilient storage experiment, $\text{Leakage}_{\mathcal{A}',l',\theta}(n)$. The formal description of the reduction is as follows.

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

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

for a non-negligible function ϵ.

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

(1) $\mathcal{A}'$ chooses uniformly t as a index of a family of hash functions and runs $(\Omega, tk, ek) \leftarrow \text{Sim}_1(1^n)$.
(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 $(s_0, s_1) \leftarrow \text{LRS}(m_b)$ 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 $\pi_b \leftarrow \text{Sim}_2^{1-\lambda_b}(\Omega, h_b, tk)$ for $b \in \{0,1\}$ are simulated robust NIZK proofs (as Construction 3.1) 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 $\text{CalcLeakage}(\Omega, t, h_0, h_1, \pi_0, \pi_1, r)$ 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 $\text{VrfyTamp}(\Omega, t, h_0, h_1, \pi_0, \pi_1, \Theta_0, \Theta_1, j^*, r)$ 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
    $$s'_{1-\theta} \leftarrow \text{Ext}(\Omega, (h'_{1-\theta}, \pi'_{1-\theta}), ek)$$
    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 subalgorithm SubLeakage is described in Algorithm 2.

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

In order to complete our proof, consider these points:

(1) Replacement of the NIZK P with $(\text{Sim}_1, \text{Sim}_2)$ is justified by the zero-knowledge property of NIZK proof system assumed in[2].
(2) We fix the randomness of adversary A by choosing randomness r, and then this adversary will operate as a deterministic algorithm.
(3) The algorithm CalcLeakage exactly simulates 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 [2].)
(4) $\mathcal{A}'$ guesses the index of tampering queries leading to $\perp$ with probability $1/q$.

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 the 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

(5) $\mathcal{A}'$ answers the $i < j^*$ tampering queries by using values of X_θ and X'_θ.
(6) The answer of $i < j^*$ tampering query is *same**, when $X_\theta = X'_\theta$. Since the answers of tampering queries are not $\perp$ and the encoding scheme satisfies the *uniqueness* property.
(7) The answer of $i < j^*$ tampering query is $x' \notin \{same^*, \perp\}$, when $X_\theta \neq X'_\theta$ and local checks are passed. 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 VrfyTamp 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 bit.
(9) The algorithm VrfyTamp requires $2k+1$ bits of leakage to verify the correctness of self-destruction index. Note that the decoding of Construction 3.1 is $\perp$, when the two shares of the codeword in our reduction decode to different answers. Algorithm VrfyTamp 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.

Therefore, we can conclude that:

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

Using Equations 1 and 2, we have

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

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

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

Since q is a polynomial function of n, we conclude that 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. □

5 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 keyless cryptographic primitives for handling tampering attacks. In this paper, we proposed a new method of proof for the security of Construction 3.1 which leads to a more efficient scheme than the previous one. Our new proof shows that the FMNV scheme can be constructed with a more effective leakage resilient storage scheme such that the amount of leakage is independent of the number of tampering queries and therefore the required memory is constant and more practical for use on tamper-proof devices.

6 Acknowledgment

This research is partially supported by grant number 7100/7293 by the research office of Sharif University of Technology.

References

[1] Amir S. Mortazavi, Mahmoud Salmasizadeh, and Amir Daneshgar. FMNV continuous non-malleable encoding scheme is more efficient than believed. In *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, pages 72–78, Sept 2016.

[2] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC* 2014, *Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 465–488. Springer, 2014.

[3] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 2008.

[4] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *Security and Cryptography for Networks, 7th International Conference, SCN* 2010*, Proceedings*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2010.

[5] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC* 2014*, Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 440–464. Springer, 2014.

[6] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: Built-in tamper resilience. In *Advances in Cryptology - ASIACRYPT* 2011 - 17*th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 740–758. Springer, 2011.

[7] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *Advances in Cryptology - CRYPTO* 2012 - 32*nd Annual Cryptology Conference, Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2012.

[8] Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC* 2015*, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 451–480. Springer, 2015.

[9] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In *Advances in Cryptology - EUROCRYPT* 2014 - 33*rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 111–128. Springer, 2014.

[10] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015* [18], pages 459–468.

[11] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. *IACR Cryptology ePrint Archive*, 2015.

[12] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In *Advances in Cryptology - CRYPTO* 2013 - 33*rd Annual Cryptology Conference, Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 239–257. Springer, 2013.

[13] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *Symposium on Theory of Computing, STOC* 2014, pages 774–783. ACM, 2014.

[14] Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In 55*th IEEE Annual Symposium on Foundations of Computer Science, FOCS* 2014, pages 306–315. IEEE Computer Society, 2014.

[15] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015,*, pages 459–468. ACM, 2015.

[16] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes resistant to permutations. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:69, 2014.

[17] Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. In *Advances in Cryptology - CRYPTO* 2014 - 34*th Annual Cryptology Conference, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 462–479. Springer, 2014.

[18] *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. ACM, 2015.

[19] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2011.

[20] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, 2003.

[21] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryp-*

tology Conference, Proceedings, Part I, volume 9215 of *Lecture Notes in Computer Science*, pages 538–557. Springer, 2015.

[22] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: keeping secrets in tamperable circuits. In *Advances in Cryptology - EUROCRYPT* 2006*, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, 2006.

[23] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO* 2003*, 23rd Annual International Cryptology Conference, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[24] Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng 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, volume 9014 of *Lecture Notes in Computer Science*, pages 427–450. Springer, 2015.

[25] Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele 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*, volume 9014 of *Lecture Notes in Computer Science*, pages 532–560. Springer, 2015.

[26] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*, chapter 3. Chapman & Hall/CRC Press, 2 edition, 2015.

[27] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology - CRYPTO* 2001*, 21st Annual International Cryptology Conference, Santa Barbara,* 2001, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598. Springer, 2001.

Seyyed Amir Mortazavi received his B.S. in Electrical Engineering from Tabriz University in 2008 and also received his M.S. degree in 2010 from Sharif University of Technology. He is now a Ph.D. candidate at the Sharif University of Technology. His research interests include computer science and information security.

Mahmoud Salmasizadeh received the B.S. and M.S. degrees in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1972 and 1989, respectively. He also received the Ph.D. degree in information technology from Queensland University of Technology, Australia, in 1997. Currently, he is an associate professor in the Electronics Research Institute and adjunct associate professor in the Electrical Engineering Department, Sharif University of Technology. His research interests include design and cryptanalysis of cryptographic algorithms and protocols, e-commerce security, and information theoretic secrecy. He is a founding member of Iranian Society of Cryptology.

Amir Daneshgar received his B.S. in digital electronics & computer hardware from Sharif University of Technology in 1990 and also received his M.S. and Ph.D. degrees in mathematics from Iran University of Science and Technology and Sharif University of Technology in 1992 and 1995, respectively. He joined the Department of Mathematical Sciences of Sharif University of Technology in 1995 and is a professor of mathematical sciences there since 2006 while his research interests fall mainly in combinatorics and computer science.