Strong Continuous Non-malleable Encoding Schemes with Tamper-Detection

Amir S. Mortazavi *Mahmoud Salmasizadeh †Amir Daneshgar ‡

January 18, 2016

***Abstract*—A *non-malleable encoding scheme* is a keyless encoding scheme which is resilient to tampering attacks. Such a scheme is said to be *continuously secure* if the scheme is resilient to attacks containing more than one tampering procedure. Also, such a scheme is said to have *tamper-detection* property if any kind of tampering attack is detected. In [S. Faust, *et al.*, *Continuous non-malleable codes*, TCC Proc., LNCS Vol. 8349, 2014.] a general continuous non-malleable encoding scheme based on NIZK is introduced which is secure in a *strong* model for which the adversary receives a *no-tamper* as a response to its tampering query if the decoding of the tampered codeword is identical to the original message.**

In this article we introduce a new strongly secure continuous non-malleable encoding scheme with tamper-detection property whose security is based on the existence of secure MAC's. Moreover, we introduce and justify the importance of an intermediate security model called *semi-strong* continuous non-malleability, while we provide a secure semi-strong continuous non-malleable encoding scheme whose security is based on the existence of CCA-secure public-key encryption.

Considering the area of applications of encoding schemes in tamper-proof devices, it is instructive to note that our proposed schemes can be used to implement an algorithmic tamper-detection level as well as maintaining the security conditions.

***Index Terms*—encoding schemes, tamper resilient cryptography, tamper-detection, continuous non-malleability.**

I. Motivations and Background

NON-MALLEABLE codes have been introduced by S. Dziembowski et al. in [1] to propose an encoding scheme resilient against an active adversary that may modify the codeword. They studied such schemes for a bit-wise family of functions in [1] (see [2], [3] for more on bit-wise schemes) while non-malleable codes for block-wise tampering have been introduced and studied in [4].

Non-malleable codes have also been studied from an information theoretic point of view [5]–[7]. There are several other variants of non-malleable schemes in the literature (e.g. see [8]–[10]). Also, Non-malleable codes have been used to construct a CCA secure public-key cryptosystem [11]. Note that other tamper-resilient models for cryptographic schemes exist which are in different context than non-malleable encoding (e.g. [12]–[14]).

Security models for non-malleable codes have been improved from one-shot split-state model for arbitrary PPT algorithms introduced in [15] to continuous non-malleability introduced in [16] (also see Definitions I.1 and I.3).

In this article we consider continuous non-malleable coding schemes and we propose some secure variants of them. In order to explain our main results, in the rest of this section, we first go through the basic definitions needed and then we will state our main contributions. To start let us recall the basic definitions of the scheme and the security model.

The split-state model is among the most important variants of non-malleable encoding schemes which explicitly is defined as follows.

Definition I.1. A 2-split-state encoding scheme

A 2-*split-state encoding scheme* which is denoted by $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec}, \kappa, \ell_0, \ell_1)$ consists of

- A nonuniform probabilistic polynomial time Turing machine Init generates necessary public information and sets the initial values of variables.
- A nonuniform probabilistic polynomial time Turing machine Enc such that given $s \in \{0,1\}^\kappa$ outputs a pair $(x_0, x_1) \in \{0,1\}^{\ell_0} \times \{0,1\}^{\ell_1}$.
- A nonuniform deterministic polynomial time Turing machine Dec such that given $(x_0, x_1) \in \{0,1\}^{\ell_0} \times \{0,1\}^{\ell_1}$ returns either an $s \in \{0,1\}^\kappa$ or $\perp$ in a way that

$$\forall\ s \in \{0,1\}^\kappa,\ \ Pr[\mathrm{Dec}(\mathrm{Enc}(s)) = s] = 1,$$

the probability being over the randomness of Enc. In the above definition the special symbol $\perp$ denotes a failure in the decoding algorithm.

Intuitively, an encoding scheme is said to be *non-malleable*, if decoding of any tampered codeword (manipulated by an adversary) gives rise to either the original message or to a totally unrelated message. Clearly, as some important applications of non-malleable coding one may refer to tamper-resilient cryptography and memory protection against tampering attacks.

Historically, non-malleability was first defined in the setting of *one-shot* tampering attacks where the adversary is allowed to apply only one tampering algorithm on a codeword [1]. A generalization of the one-shot setup to the *continuous non-malleability* by S. Faust et al. considers the case when the adversary is allowed to apply different tampering algorithms more than once [16].

*A. S. Mortazavi is with the Department of Electrical Engineering, Sharif university of Technology, Tehran, Iran.
E-mail: sa_mortazavi@ee.sharif.edu

†M. Salmasizadeh is with the Electronics Research Institute and is with the Department of EE as adjunct member, Sharif University of Technology, Tehran, Iran.

‡A. Daneshgar is with the Department of Mathematical Sciences, Sharif university of Technology, Tehran, Iran.

One could also consider the leakage of shares of a codeword and define a leakage-resilient coding scheme (e.g. see [15], [17]).

The next definitions not only present a unification of the above security models but also will give rise to some new security concepts that will be discussed in the rest of this article. To begin, let us define the oracles we need.

Definition I.2. A $(\mathrm{Enc}, \mathrm{Dec}, \mathcal{T}, \mathcal{L}, b, c, tm, lm, \theta)$-oracle

In this type of oracle $\mathcal{T}$ and $\mathcal{L}$ are subclasses of polynomial time Turing machines indexed by a canonical (and pre-fixed) coding. The oracle operates as follows:

- On receiving a pair (s_0, s_1) the oracle computes $\mathrm{Enc}(s_b) = (x_0^b, x_1^b)$.
- On receiving a *leakage query* $(l, \langle T_0\rangle, \langle T_1\rangle)$, where $T_i \in \mathcal{L}$ for $i \in \{0,1\}$ and $\langle T_i\rangle$ stands for the canonical coding of the Turing machine T_i, returns $(T_0(x_0^b), T_1(x_1^b))$.
- On receiving a *tampering query* $(t, \langle T_0\rangle, \langle T_1\rangle)$, where $T_i \in \mathcal{T}$ for $i \in \{0,1\}$ and $\langle T_i\rangle$ stands for the canonical coding of the Turing machine T_i, the oracle computes $\mathrm{Dec}(T_0(x_0^b), T_1(x_1^b)) = s'$. If $s' = s_b$ the oracle returns the special (and pre-fixed) character s^*, if $s' = \perp$ the oracle returns $\perp$, and returns s' otherwise.
- $tm \in \{0,1\}$ is a flag indicating the mode of answering to tampering queries. If $tm = 1$ (normal mode), the oracle answers tampering queries normally (described before). However, if $tm = 0$ (self-destruction mode), on the occasion of returning the first $\perp$ as the answer to a tampering query the oracle always returns $\perp$ to any other tampering query after that.
- The sum of the length of oracle answers to the leakage queries (in bits) must not exceed the bound c. The oracle returns l^* on the occasion of a leakage query whose length will increase the sum strictly greater than c onwards.
- $lm \in \{0,1\}$ is a flag that indicating the behavior of oracle after finishing leakage queries. If $lm = 0$, after finishing leakage queries, the oracle returns the x_θ^b for $\theta \in \{0,1\}$. However, if $lm = 1$ the oracle does not return any part of the codeword.

Our indistinguishability adversarial model defined below tries to formulate and unify the existing security models. We will discuss different aspects of this definition and some new consequences after the definition.

Definition I.3. A $(\mathcal{T}, \mathcal{L}, c, q_t, q_l, \tau)$-adversary

A $(\mathcal{T}, \mathcal{L}, c, q_t, q_l, \tau)$-adversary is a nonuniform probabilistic polynomial time oracle Turing machine that uses a $(\mathrm{Enc}, \mathrm{Dec}, \mathcal{T}, \mathcal{L}, b, c, tm, lm, \theta)$-oracle. The adversary operates subject to the following goal and limitations.

- The adversary starts by sending a query (s_0, s_1) to its $(\mathrm{Enc}, \mathrm{Dec}, \mathcal{T}, \mathcal{L}, b, c, tm, lm, \theta)$-oracle and its objective is to guess b, using its further queries, by generating $b' \in \{0, 1, \pitchfork\}$,
- The adversary is limited to ask up to q_l leakage queries and up to q_t tamper queries.
- The running time of the adversary is limited to be bounded by $\tau(\kappa)$ for each input (s_0, s_1) of size κ.

Such adversaries are classified into the following types based on the number of queries and the tampering mode.

- The term *one-shot tampering* is used to refer to the case $q_t = 1$ and the term *continuous tampering* is used for the case $q_t \geq 1$.
- If $q_l \neq 0$ and $lm = 1$ the adversary is said to be of *leakage* type.
- If $q_l \neq 0$ and $lm = 0$ the adversary is said to be of *strong leakage* type.
- The adversary is said to be a *semi-strong* adversary if by receiving the first answer $\perp$ as a response to a query, the adversary halts and outputs the special symbol $\pitchfork$.
- The adversary is said to be a *strong* adversary if it is connected to an oracle operating in the self-destruction mode (i.e. $tm = 0$) and also $lm = 1$.
- The adversary is said to be a *super-strong* adversary if it is connected to an oracle operating in the normal mode (i.e. $tm = 1$) and $lm = 1$.

The random variable standing for the result of the above experiment for a $(\mathcal{T}, \mathcal{L}, c, q_t, q_l, \tau)$-adversary, A, attached to a $(\mathrm{Enc}, \mathrm{Dec}, \mathcal{T}, \mathcal{L}, b, c, tm, lm, \theta)$-oracle, $\mathcal{O}$, is denoted by $\mathrm{Exp}_A^{Type, \mathcal{O}(b)}(1^n) \in \{0, 1, \pitchfork\}$, and n is the security parameter. Based on the type of security we replace $Type$ by the name of the model such as $\mathrm{Exp}_A^{\mathrm{lr}, \mathcal{O}(b)}(1^n)$ for the leakage resilient security, $\mathrm{Exp}_A^{\mathrm{semi}, \mathcal{O}(b)}(1^n)$ for the semi-strong non-malleability, $\mathrm{Exp}_A^{\mathrm{strong}, \mathcal{O}(b)}(1^n)$ for the strong non-malleability and $\mathrm{Exp}_A^{\mathrm{SS}, \mathcal{O}(b)}(1^n)$ for the super strong non-malleability. An encoding scheme $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec}, \kappa, \ell_0, \ell_1)$ is said to be (q_t, q_l, tm, lm)-non-malleable if for all adversaries A the probability $\left|\Pr[\mathrm{Exp}_A^{Type, \mathcal{O}(b)}(1^n) = b] - 1/2\right|$ is a negligible function of n (for definition of the negligible function you may refer to [18]).

Note that the special cases of *leakage resistant* non-malleable, *semi-strong*, *strong* and *supper strong* non-malleable codes are defined based on Definition I.3.

On the other hand, one may find some other related security models in the literature among which in particular, we mention the following model for a *tamper-detection* scheme.

Definition I.4. A tamper-detection encoding scheme

An encoding scheme $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec}, \kappa, \ell_0, \ell_1)$ has $\mathcal{T}$-*tamper-detection* property where $\mathcal{T}$ is a family of nonuniform probabilistic polynomial time Turing machines, if for any pair of machines $T_0 \in \mathcal{T}$ and $T_1 \in \mathcal{T}$ and any message $s \in \{0,1\}^\kappa$ with $\mathrm{Enc}(s) = (x_0, x_1)$, the probability $\Pr[\mathrm{Dec}(T_0(x_0), T_1(x_1)) \notin \{\perp, s^*\}]$ is negligible [19].

It is worthwhile to say that a scheme secure against an adversary of leakage type is said to be a *leakage-resilient encoding scheme* which is also referred to as a *leakage-resilient storage* (LRS) [17]. Similarly, a scheme secure against a strong adversary of leakage type is said to be a *strong leakage-resilient encoding scheme* also referred to as a *strong leakage-resilient storage* (S-LRS) scheme. The notations $\Pi = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ and $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec})$ are interchangeable in this paper for a leakage resilient encoding scheme.

Since there are a number of different definitions in the literature

then different variants of non-malleability studied in our setting. The super-strong non-malleability is defined in [19] and has not been studied in detail so far.

The definition of semi-strong non-malleability is an outcome of our new setting. To justify the importance of this case, consider a computer virus that may tamper with the memory and transmits data to an outsider after gathering all data needed. Note that in this case the virus must send its data to the outside *before* any safeguarding software on the computer detects the presence of the virus and shuts down (kind of self-destruction) the whole system. Also, it is clear by definition that the semi-strong non-malleability is weaker than strong non-malleability in the sense of security, however, based on the above justification the whole model still deserves consideration. As one of the objectives of this article, we will propose a new semi-strong non-malleable encoding scheme and we will prove its security in our proposed setup.

The *one-shot strong* non-malleability is defined in [15] and the *continuous strong* non-malleability is defined in [16] in which the authors use robust non-interactive zero knowledge with supporting label, hash functions, a strong leakage-resilient storage scheme and public untamperable strings to construct their secure scheme.

As our second and main objective in this article, in what follows, we also introduce a strong non-malleable encoding scheme that can be compared to that of S. Faust et al. in the following sense.

- In our scheme we use MAC's instead of NIZK proofs, where the implementing MAC is known to be usually simpler and more efficient.
- In our scheme we use standard leakage-resilient storage instead of *strong* leakage-resilient storage indicating that we use weaker assumptions.
- Our scheme is proved to be continuous non-malleable with tamper-detection property, while the scheme introduced in [16] is just continuous non-malleable.
- Our scheme requires lesser LRS leakage bound than S. Faust's scheme [16]. More precisely, in our scheme the leakage bound is $l_{\text{lrs}} \geq 2l + \log(q_t)$ while in Faust's scheme we have $l_{\text{lrs}} \geq 2l + (k+1)\log(q_t)$ where k is the number of the output bits of the hash function.

The organization of the paper is as follows. In Section II we introduce the concept of *semi-strong non-malleability* and we provide an encoding scheme with continuous semi-strong non-malleability where we prove its security in the proposed model. In Section III we present a leakage-resilient continuous strong non-malleable scheme with tamper-detection and prove its security in the standard model.

II. A SECURE SEMI-STRONG NON-MALLEABLE SCHEME

We define *continuous semi-strong* non-malleability formally in previous section and now describe a construction in this model. We present an encoding scheme in the split-state model with continuous semi-strong non-malleability. Our proposed scheme is based on an l-leakage CCA indistinguishable public key encryption cryptosystem and also we assume the existence of untamperable public strings. We can imagine this public strings hardwired to the encoding function. We first define the l-leakage CCA indistinguishable encryption cryptosystem and then present the construction.

The encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ is an l-leakage CCA, if it has standard CCA indistinguishability and also sustainable against l bits leakage of the secret key. We define formally the l-leakage CCA in Definition II.1 .

Definition II.1. A l-leakage CCA public key encryption Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a public key encryption scheme, $n \in \mathbb{N}$ be the security parameter, $\mathcal{L}$ be a family of nonuniform probabilistic polynomial time Turing machines and also let A be a nonuniform probabilistic polynomial time Turing machine as the adversary. The encryption of message m is shown with $\text{Enc}_{pk}(m)$ and decryption of ciphertext c is shown with $\text{Dec}_{sk}(c)$. The adversary A has access to a leakage oracle $\mathcal{O}(\mathcal{L}, sk)$ and may send a Turing machine $T \in \mathcal{L}$ as a query to the leakage oracle where it receives $T(sk)$ in response as the answer. Note that the leakage oracle is allowed to return at most l bits as the whole number of bits of its answers during the process of adversary A. The adversary A also has access to the decryption oracle that may return the plaintexts corresponding to the ciphertexts queried by A. Let the random variable $\text{Exp}_A^{CCA}(b, 1^n)$, for any bit $b \in \{0,1\}$, denote the result of following experiment for an adversary A.

Experiment 1: $\text{Exp}_A^{CCA}(b, 1^n)$

1. Run $(pk, sk) \leftarrow \text{Gen}(1^n)$ and let A have pk while sk is kept secret.
2. A may query the leakage oracle.
3. A may query the decryption oracle.
4. A selects two arbitrary messages m_0 and m_1.
5. A is given $c = Enc_{pk}(m_b)$.
6. A may query the leakage oracle again.
7. A may query the decryption oracle except for the challenge ciphertext c.
8. A outputs $b' \in \{0,1\}$ (as a guess for the value of b).

The scheme Π is said to have the l-leakage CCA security if

$$\text{Exp}_A^{\text{CCA}}(0, 1^n) \approx_c \text{Exp}_A^{\text{CCA}}(1, 1^n).$$

The security definition and proposed schemes for the leakage-resilient CCA indistinguishability can be found for example in [20].

Now we define the construction for a continuous semi-strong non-malleable scheme as the following.

Construction II.1. Let $\Pi' = \{\text{Gen}', \text{Enc}', \text{Dec}'\}$ be an l-leakage CCA indistinguishable public key encryption scheme, and let $\Pi_H = (\text{Gen}_H, \mathcal{H}, l_H)$ be a collision resistant family of hash functions [18] with output length l_H bits. Our construction for a semi-strong encoding scheme

$$\Pi = (\text{Init}, \text{Enc}, \text{Dec}, \kappa, \ell_0, \ell_1)$$

is as follows:

- $\text{Init}(1^n)$: Set $(sk, pk) \leftarrow \text{Gen}(1^n)$ and $s \leftarrow \text{Gen}_H(1^n)$. Note that sk is the secret key where $h = H_s(sk)$, s and pk are public values.

- $\mathrm{Enc}(x)$: set $x_0 \leftarrow sk$ and $x_1 \leftarrow \mathrm{Enc}'_{pk}(x)$.
- $\mathrm{Dec}(x_0, x_1)$: If $h \neq H_s(x_0)$ then return $\perp$, otherwise return $c' = \mathrm{Dec}'_{x_0}(x_1)$.

The semi-strong non-malleability experiment for a $(\mathcal{T}, \mathcal{L}, c, q_t, q_l, \tau)$-adversary, A and $(\mathrm{Enc}, \mathrm{Dec}, \mathcal{T}, \mathcal{L}, b, c, tm, lm, \theta)$-oracle, $\mathcal{O}$, with $q_l = 0$ is referred to as $\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n)$ and an encoding scheme $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec}, \kappa, \ell_0\ell_1)$ is said to be (q_t, q_l, tm, lm)-non-malleable, if for all adversaries A the success probability of the adversary A in the above experiment, i.e.

$$\left|\Pr[\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b] - 1/2\right|$$

is negligible.

Lemma II.1. *For the encoding scheme* Π *as defined in Construction* II.1*, let* Coll *be the event that there exists an adversary tampering query for which we have a collision for the hash of the first encoding share* x_0*. Then*

$$\Pr[\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b] - \Pr[(\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b) \mid \overline{\mathrm{Coll}}]$$

is a negligible function of n*.*

Proof.

$$\begin{aligned}
&\Pr[\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b] \\
&= \Pr[(\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b) \cap \mathrm{Coll}] \\
&+ \Pr[(\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b) \cap \overline{\mathrm{Coll}}] \\
&\leq \Pr[\mathrm{Coll}] + \Pr[(\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b) \cap \overline{\mathrm{Coll}}] \\
&\leq \Pr[\mathrm{Coll}] + \Pr[(\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b) \mid \overline{\mathrm{Coll}}] \times \Pr[\overline{\mathrm{Coll}}] \\
&\leq \Pr[\mathrm{Coll}] + \Pr[(\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b) \mid \overline{\mathrm{Coll}}] \\
&\leq \mathrm{negl}(n) + \Pr[(\mathrm{Exp}_A^{\mathrm{semi},\mathcal{O}(b)}(1^n) = b) \mid \overline{\mathrm{Coll}}]
\end{aligned}$$

Where in the last inequality we have used the collision resistant of the hash function. ■

Note that restricted to the event $\overline{\mathrm{Coll}}$, a change in first share of the encoding scheme causes $\perp$ as a response of the oracle to the corresponding tampering query which gives rise to an immediate halt and $\Lightning$ in the output.

Theorem II.1. *Let* $\Pi' = \{\mathrm{Gen}', \mathrm{Enc}', \mathrm{Dec}'\}$ *be an* l*-leakage* CCA *indistinguishable public key encryption scheme,* $\Pi_H = (\mathrm{Gen}_H, \mathcal{H}, l_H)$ *a family of collision resistant hash functions and* $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec}, \kappa, \ell_0, \ell_1)$ *be the scheme described in Construction* II.1*. Then the encoding scheme* Π *has semi-strong non-malleability for* $l \geq l_H$*.*

Proof. Assume that there exists a semi-strong $(\mathcal{T}, \mathcal{L}, c, q_t, q_l, \tau)$-adversary A for the scheme Π having access to an $(\mathrm{Enc}, \mathrm{Dec}, \mathcal{T}, \mathcal{L}, b, c, tm, lm, \theta)$-oracle with $q_l = 0$.

In what follows we construct an adversary A' for an encryption scheme Π' that initializes and runs A just once and answers its queries appropriately to gain enough information. Our proof is based on proving that the success probability of A' is at least equal to that of A. The adversary A' operates as follows:

- Initialization of A': The adversary A' is given pk and access to its leakage oracle for at most l bits. Then by receiving m_0 and m_1 as the oracle initialization step for A, the adversary A' choses the same plaintexts m_0 and m_1 and is given $c = \mathrm{Enc}_{pk}(m_b)$ within its experiment for $b \in \{0, 1\}$.
- Initialization of A: The adversary A' computes $s \xleftarrow{A'} \mathrm{Gen}_H(1^n)$ and then queries H_s to its leakage oracle and gets $h \xleftarrow{A'} H_s(sk)$(`it is possible because the leakage bound` $l \geq l_H$). Then A' sends s, pk and h to A.
- Answering a typical tampering query (T_0, T_1) of A: By receiving (T_0, T_1) (through A), the adversary A' computes $c' = T_1(c)$. If $c' = c$, it returns s^*, otherwise queries c' to its decryption oracle and returns the result to A.
- Output: A outputs a bit b' and A' outputs the same bit.

First, note that for the above experiment the probability space generated by the randomness of both adversaries A and A' are the same. Also, by Lemma II.1, we may assume that the event Coll does not occur, since this assumption can affect the success probability at most by a negligible function of the security parameter.

On the other hand, we have

$$\Pr[\mathrm{Exp}_{A'}^{\mathrm{CCA}}(b, 1^n) = b] \geq \Pr[(\mathrm{Exp}_A^{\mathcal{O}(b)}(1^n) = b) \mid \overline{\mathrm{Coll}}],$$

since the event of success in the experiment related to the adversary A is a subset of the event of success for the adversary A'. This is clear since the event of success for A conditioned to $\overline{\mathrm{Coll}}$ can be described as follows,

1) There is no change in the first share x_0 (because of $\overline{\mathrm{Coll}}$ event, any changes in first share resulted to the $\perp$).
2) There is no $\perp$ as the response of a tampering query for the second share.
3) $b = b'$.

While the event of success for A' can be described as,

1) A succeeds.
2) $b = b'$.

Note that because of the definition of semi-strong non-malleability if $\perp$ is returned to the adversary as the response to a tampering query then the experiment is terminated with $\Lightning$ and the adversary cannot apply the distinguishability test. ■

III. A STRONGLY SECURE NON-MALLEABLE SCHEME

The experiment $\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)$ for a $(\mathcal{T}, \mathcal{L}, c, q_t, q_l, \tau)$-adversary, A, attached to a $(\mathrm{Enc}, \mathrm{Dec}, \mathcal{T}, \mathcal{L}, b, c, tm, lm, \theta)$-oracle, $\mathcal{O}$, with $tm = 0$ defines *strong non-malleability* in which the self-destruction may occur, i.e. if the oracle returns $\perp$ in response to a tampering query then the other tampering queries also will be answered with a $\perp$ onwards (independent of the queries themselves). Recall that if $tm = 0$ and $q_t \leq 1$ we have one-shot strong non-malleability and if $tm = 0, q_t \geq 1$ we have continuous strong non-malleability in this model. The leakage-resilient strong non-malleability is also defined for $q_l \neq 0$.

The first secure strong *continuous* non-malleable encoding scheme has been introduced in [16] that uses robust non-interactive zero knowledge proofs. In this section we introduce

a new secure strong continuous non-malleable encoding scheme with *tamper-detection* having the following properties:

- Our scheme uses MAC functions instead of NIZK proofs.
- Our scheme uses standard leakage-resilient storage instead of strong leakage-resilient.
- Our scheme also has the extra advantage of having tamper-detection.
- As will be shown later our scheme has a better leakage-resilient bound.

For the security analysis of our proposed scheme we will need the definition for the l-leakage-resilient strong existentially unforgeable MAC.

Definition III.1. Let $\Pi = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ be a *message authentication code* (MAC) scheme, $n \in \mathbb{N}$ be the security parameter, $\mathcal{L}$ be a family of nonuniform probabilistic polynomial time Turing machines and also let A be a nonuniform probabilistic polynomial time Turing machine as the adversary. If $t = \mathrm{Mac}_k(m)$ where k is the secret key and m is the message then t is referred to as the tag of the message m. For the predicate $\mathrm{Vrfy}_k(t, m)$ a pair (t, m) is said to be a *valid* pair if $\mathrm{Vrfy}_k(t, m) = 1$.

The adversary A has access to leakage and MAC oracles. The adversary can send the coding of a Turing machine $T \in \mathcal{L}$ as a query to the leakage oracle $\mathcal{O}(\mathcal{L}, k)$ for the secret key k, where it receives $T(k)$ in response as the answer. Note that the leakage oracle is allowed to return at most l bits as the whole number of bits of its answers during the process of A. The adversary A also has access to a $\mathcal{MAC}_k(.)$ oracle that may return the tags corresponding to the messages queried by A.

Let the random variable $\mathrm{Exp}_A^{\mathrm{MAC}}(1^n)$ denotes the result of the following experiment for an adversary A.

Experiment 2: $\mathrm{Exp}_A^{\mathrm{MAC}}(1^n)$

1 Run $k \leftarrow \mathrm{Gen}(1^n)$ where k is kept secret.
2 A may query the leakage oracle $(\mathcal{O}(\mathcal{L}, k))$.
3 A may query the $\mathcal{MAC}_k(.)$ oracle. (The set of queried messages by the adversary is denoted by M and the set of corresponding answers is denoted by R).
4 A outputs (m', t').
5 If $[(m' \notin M$ or $(m' \in M$ and $t' \notin R))$ and $t' = \mathrm{Vrfy}_k(m')]$ the output of the experiment is 1 and otherwise is 0.

Then the scheme Π is said to be a secure l-leakage-resilient strong existentially unforgeable scheme under an adaptive chosen message attack if

$$\Pr[\mathrm{Exp}_A^{\mathrm{MAC}}(1^n) = 1] \leq \mathrm{negl}(n).$$

The details of the security model and some examples for leakage-resilient MAC schemes can be found for example in [21]. Now we define our scheme for leakage-resilient continuous strong non-malleability as follows.

Construction III.1. Let $\Pi_H = (\mathrm{Gen}_H, \mathcal{H}, l_H)$ be a family of collision resistant hash functions with output length l_H bits. Also, let $\Pi_L = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ be an l_{lrs}-leakage-resilient storage, $\Pi' = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ be a secure l_{mac}-leakage-resilient strong existentially unforgeable MAC algorithm under an adaptive chosen message attack. Our coding scheme is a tuple $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec}, \kappa, \ell_0, \ell_1)$, which is defined as follows.

- $\mathrm{Init}(1^n)$: Set $s \leftarrow \mathrm{Gen}_H(1^n)$, $k_0 \leftarrow \mathrm{Gen}(1^n)$ and $k_1 \leftarrow \mathrm{Gen}(1^n)$. Then compute $h_0 = H_s(k_0)$, $h_1 = H_s(k_1)$ and publish h_0, h_1 and s as untamperable public strings.
- $\mathrm{Enc}(x)$: Compute $(c_0, c_1) \leftarrow \mathrm{LRS}(x)$, then compute the two split encoding shares as $x_0 = (c_0, t_0, k_1)$ and $x_1 = (c_1, t_1, k_0)$ where $t_0 = \mathrm{Mac}_{k_0}(c_0)$ and $t_1 = \mathrm{Mac}_{k_1}(c_1)$.
- $\mathrm{Dec}(x_0, x_1)$: Pars x_b for the format of (c_b, t_b, k_{1-b}) for $b \in \{0, 1\}$, and then check the correctness of $h_0 \stackrel{?}{=} H_s(k_0)$ and $h_1 \stackrel{?}{=} H_s(k_1)$, $t_0 \stackrel{?}{=} \mathrm{Mac}_{k_0}(c_0)$ and $t_1 \stackrel{?}{=} \mathrm{Mac}_{k_1}(c_1)$. If any one of the verifications fails, output $\perp$; otherwise, output $x' = \mathrm{LRS}^{-1}(c_0, c_1)$.

The strong non-malleability experiment for an adversary A is referred to as $\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}}(1^n)$ and an encoding scheme $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec}, k, \ell_0 \ell_1)$ is said to be (q_t, q_l, tm, lm)-non-malleable for $tm = 0$ if for all adversaries A, the probability

$$\left| \Pr[\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n) = b] - 1/2 \right|,$$

is negligible.

Theorem III.1. *Let $\Pi_L = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ be an l_{lrs}-leakage-resilient storage scheme, $\Pi_H = (\mathrm{Gen}_H, \mathcal{H}, l_H)$ a family of collision resistant hash functions and $\Pi' = (\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ be a secure l_{mac}-leakage-resilient strong existentially unforgeable* MAC *and $\Pi = (\mathrm{Init}, \mathrm{Enc}, \mathrm{Dec}, \kappa, \ell_0, \ell_1)$ be the scheme described in Construction* III.1. *If $l_{mac} \geq \log(q_t) + l_H + l$ and $l_{\mathrm{lrs}} \geq \log(q_t) + 2l$, then Π is an l-leakage-resilient continuous strong non-malleable encoding scheme with tamper-detection property.*

Proof. The intuition behind the proof is as follows. Intuitively, no adversary can tamper with the values of keys k_0, k_1 as far as the verification for the hash of keys hold, since existence of a modification that passes the verifications, contradicts the collision resistance of the family $\mathcal{H}$. Hence, assuming that the keys k_0, k_1 are not changed by the adversary, if the adversary changes the value of c_b to c_b', then the adversary must also be able to compute a new tag for c_b which is verifiable. This also contradicts the strong security of the MAC function.

Now we go through the details of the proof. Note that, hereafter, we assume that a probabilistic Turing machine T is a machine with a random tape such that for any input x and a random string r on its random tape computes $A(r, x)$ using a deterministic procedure.

Considering the existence of an adversary A operating within the procedure of the experiment $\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)$, let us define two events Coll and Tamper, where Coll is the event that the adversary A, using the tampering oracle, can change the values of k_0 and k_1 within the framework of Construction III.1 such that the hash verifications hold, and also Tamper is

defined as the event that A can tamper with codewords and receive a value other than s^* or $\perp$.

Claim III.1. $Pr[\text{Tamper} \mid \overline{\text{Coll}}] \leq \text{negl}(n)$.

Proof. We show that if A receives $x' \notin \{s^*, \perp\}$ in tampering queries, then there exists an adversary A' attacking the leakage-resilient MAC, $\Pi' = (\text{Gen}, \text{Mac}, \text{Vrfy})$, who is able to forge a valid tag on some messages for the MAC function.

The adversary A' operates as follows.

- Generate the key k using $\text{Gen}(1^n)$.
- From this stage onwards, A' may access the oracles $\mathcal{MAC}_k(.)$ and $\mathcal{O}(\mathcal{L}, k)$.
- A' computes $k_1 \xleftarrow{A'} \text{Gen}(1^n)$, $s \xleftarrow{A'} \text{Gen}_H(1^n)$, $h_1 \xleftarrow{A'} H_s(k_1)$ and also receives $h_0 \xleftarrow{A'} H_s(k)$ using queries from its leakage oracle (`note that this is possible since by the hypothesis we have` $l_{\text{mac}} > l_H$).
- A' chooses the bits b, b' and b'' as well as the binary string r uniformly at random.
- A' send the coding of $F_{\text{m}}(h_0, h_1, s, k_1, b, b', b'', r)$ to the leakage oracle and receives j as the result of executing F_{m} within the oracle environment (`see Algorithm 1 and the comments proceeding this algorithm description`) and halts if $j = 0$. (j `is the index of tampering query resulting to` $\perp$.)
- A' runs $A(r, h_0, h_1, s)$ and receives m_0 and m_1.
- A' computes $(c_0, c_1) \xleftarrow{A'} \text{LRS}(m_{b''})$ and requests $t_b = \text{Mac}_k(c_b)$ from its MAC oracle.
- A' set $x_{b'} = (c_b, t_b, k_1)$.
- Whenever A queries its leakage oracle, A' answers this query using its own leakage oracle $\mathcal{O}(\mathcal{L}, k)$ (`note that this is also possible because of the inequality` $l_{\text{mac}} > l$).
- When A requests its ith tampering query, then if $i < j$, A' sends s^* to A as the answer.
- For the jth tampering query by (T_0, T_1), A' computes $x' = T_{b'}(x_{b'}) = (c'_b, t'_b, k'_1)$ for $x_{b'} = (c_b, t_b, k_1)$ and halts where c'_b is a valid forged message for the MAC function with the valid tag t'_b.

Note that we have used a method described in [16] to handle the using of the key k which is only available to the leakage oracle $\mathcal{O}(\mathcal{L}, k)$. In this sense recall that the function (here the function F_{m}) is passed to the oracle as its coding where the code is run within the oracle and the answer is passes to the main program. The description of the function F_{m} is as follows (see Algorithm 1).

Note that the behavior of adversary A in the above algorithm is identical to a real continuous strong non-malleable adversary until the jth tampering query. If $j \neq 0$ then the event Tamper occurs within the experiment $\text{Exp}_A^{\text{strong},\mathcal{O}(b)}(1^n)$. Let Forge be the event that in $\text{Exp}_A^{\text{strong},\mathcal{O}(b)}(1^n)$ at least one part of x_0 changes in the tampering query, resulting to $x' \notin \{s^*, \perp\}$. We claim that

$$\Pr[\text{Forge} \mid \overline{\text{Coll}}] \geq \frac{1}{2} \times \Pr[\text{Tamper} \mid \overline{\text{Coll}}].$$

Algorithm 1: $j \longleftarrow F_{\text{m}}(h_0, h_1, s, k_1, b, b', b'', r)$

1 Set $k_0 \overset{\text{def}}{=} k$ using the oracle database (`note that inside of the leakage oracle, we know the secret key` k).
2 Invoke the adversary $A(r, h_0, h_1, s)$ to obtain two messages m_0 and m_1.
3 Set $x_{b'} = (c_b, t_b, k_1)$ and $x_{1-b'} = (c_{1-b}, t_{1-b}, k_0)$, where $(c_0, c_1) \leftarrow \text{LRS}(m_{b''})$, $t_b = \text{Mac}_{k_0}(c_b)$ and $t_{1-b} = \text{Mac}_{k_1}(c_{1-b})$.
4 This algorithm (F_m) can answer the leakage and tampering queries of A using x_0 and x_1.
5 In the ith tampering query,
- If A is given $x' \notin \{s^*, \perp\}$ as the answer to a tampering query then $j \leftarrow i$ and halt.
- If A is given $\perp$ as the answer to a tampering query, then $j \leftarrow 0$ and halt.
- If A halts, output $j \leftarrow 0$ and halt.
- Otherwise answer the $i + 1$th tampering query.

(`Note that this is possible because of the leakage bound` $l_{\text{mac}} > \log(q_t)$.)

We know that in $\overline{\text{Coll}}$ the values of k_0 and k_1 may not change, and also c_0, c_1, k_0 and k_1 are located uniformly at random in shares of x_0 and x_1 in algorithm F_{m} (because the bits b, b' as input of the algorithm F_{m} is chosen uniformly at random). Consequently, since Tamper depends uniformly on changes in x_0 and x_1 the claim is proved.

On the other hand, if the event $(\text{Forge} \mid \overline{\text{Coll}})$ occurs in the experiment $\text{Exp}_A^{\text{strong},\mathcal{O}(b)}(1^n)$, the security of the MAC function may be broken. Hence,

$$\begin{aligned}
&\frac{1}{2} \times \Pr[\text{Tamper} \mid \overline{\text{Coll}}] \\
&\leq \Pr[\text{Forge} \mid \overline{\text{Coll}}] \\
&= \Pr[\text{Exp}_{A'}^{\text{MAC}}(1^n) = 1] \\
&\leq \text{negl}_0(n) \\
&\Rightarrow \Pr[\text{Tamper} \mid \overline{\text{Coll}}] \leq 2 \times \text{negl}_0(n) = \text{negl}(n).
\end{aligned}$$

Note that this proves the tamper-detection property of our scheme. ■

Claim III.2. *For the encoding scheme* Π *as defined in Construction* III.1

$$\begin{aligned}
&\Pr[\text{Exp}_A^{\text{strong},\mathcal{O}(b)}(1^n) = b] \\
&- \Pr[(\text{Exp}_A^{\text{strong},\mathcal{O}(b)}(1^n) = b) \mid (\overline{\text{Coll}} \cap \overline{\text{Tamper}})
\end{aligned}$$

is a negligible function of n.

Proof.

$$
\begin{aligned}
&\Pr[\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b]\\
&=\Pr[(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b)\cap(\mathrm{Coll}\cup\mathrm{Tamper})]\\
&+\Pr[(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b)\cap\overline{\mathrm{Coll}\cup\mathrm{Tamper}}]\\
&\leq\Pr[\mathrm{Coll}\cup\mathrm{Tamper}]\\
&+\Pr[(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b)\cap(\overline{\mathrm{Coll}}\cap\overline{\mathrm{Tamper}})]\\
&\leq\Pr[\mathrm{Coll}]+\Pr[\mathrm{Tamper}\cap\overline{\mathrm{Coll}}]\\
&+\Pr[(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b)\cap(\overline{\mathrm{Coll}}\cap\overline{\mathrm{Tamper}})]\\
&\leq\Pr[\mathrm{Coll}]+\Pr[\mathrm{Tamper}\mid\overline{\mathrm{Coll}}]\\
&+\Pr[(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b)\mid(\overline{\mathrm{Coll}}\cap\overline{\mathrm{Tamper}})]\\
&\leq\mathrm{negl}_0(n)+\mathrm{negl}_1(n)\\
&+\Pr[(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b)\mid(\overline{\mathrm{Coll}}\cap\overline{\mathrm{Tamper}})].
\end{aligned}
$$

Where in the last inequality we have used the collision resistance of the hash function and Claim III.1. ■

Claim III.3. $\Pr[(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b)\mid\overline{\mathrm{Coll}}\cap\overline{\mathrm{Tamper}}]\leq\frac{1}{2}+\mathrm{negl}(n)$.

Proof. We show that if the adversary A distinguishes m_0 from m_1 in the experiment $(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)=b\mid\overline{\mathrm{Coll}}\cap\overline{\mathrm{Tamper}})$ with non-negligible probability, then there exists another adversary A' that distinguishes the same massages in the leakage-resilient storage experiment. The reduction is as follows.

- A' computes $k_0\xleftarrow{A'}\mathrm{Gen}(1^n)$, $k_1\xleftarrow{A'}\mathrm{Gen}(1^n)$, $s\xleftarrow{A'}\mathrm{Gen}_H(1^n)$, $h_0=H_s(k_0)$ and $h_1=H_s(k_1)$.
- A' generates a random string r.
- A' runs $A(r,h_0,h_1,s)$ and then A outputs m_0 and m_1.
- A' chooses m_0 and m_1 as its own messages. (Note that A' has access to its leakage oracle.)
- A' sends the coding of $F(k_0,k_1,l_{\mathrm{lrs}},h_0,h_1,s,r)$ to the leakage oracle and receives j and two arrays γ_0 and γ_1 as the result of executing F within the oracle (see Algorithm 2) and halts if $j=0$. (Here j is the index of tampering query resulting to $\perp$.)
- When A requests its ith leakage query (T_0,T_1), then A' answers this query using $\gamma_0[i]$ and $\gamma_1[i]$.
- When A requests its ith tampering query (T_0,T_1), then A' answers $i<j$ tampering queries with s^* and answers $\perp$ to the tampering queries when $i\geq j$.
- A' outputs whatever is output by A.

The algorithm $F(k_0,k_1,l_{\mathrm{lrs}},h_0,h_1,s,r)$ is a Turing machine that is queried to the leakage oracle of adversary A. This algorithm runs the adversary $A(r,h_0,h_1,s)$ with randomness r inside of the leakage oracle. This method is similar to the one used in [16].

The algorithm $F(k_0,k_1,l_{\mathrm{lrs}},h_0,h_1,s,r)$ outputs j, γ_0 and γ_1. The number j is the index of tampering query resulting to $\perp$ where γ_0 and γ_1 are two arrays containing the responses of leakage oracle to the adversary A. Note that γ_0 and γ_1 are arrays of size at most q_t. The algorithm $F(k_0,k_1,l_{\mathrm{lrs}},h_0,h_1,s,r)$ contains two sub-algorithms F_t and F_ℓ where F_t handles

Algorithm 2: $[j,\gamma_0,\gamma_1]\longleftarrow F(k_0,k_1,l_{\mathrm{lrs}},h_0,h_1,s,r)$

Set $j_0\leftarrow 0$, $j_1\leftarrow 0$, $e_0\leftarrow 0$, $e_1\leftarrow 0$.
for $i\leftarrow 0$ **to** q_t **do**
  $\gamma_0[i]=-1$
  $\gamma_1[i]=-1$
end
/* Note that γ_0 and γ_1 are global vectors */
Loop
  Form $x_0=(c_0,\mathrm{Mac}_{k_0}(c_0),k_1)$ and $x_1=(c_1,\mathrm{Mac}_{k_1}(c_1),k_0)$
  (This is possible since inside of the leakage oracle c_0 and c_1 are known)
  if $j_0=0$ **then**
    Run $[y,\alpha_0,\alpha_1]\leftarrow F_\ell(h_0,h_1,s,x_0,0,l_{\mathrm{lrs}},r)$
    if $\alpha_0\neq -1$ **then**
      $e_0\leftarrow(e_0+1)$
      Set $\gamma_0[e_0]\leftarrow\alpha_0$
    end
    if $y\neq 0$ **then**
      $j_0\leftarrow y$
    end
  end
  if $j_1=0$ **then**
    Run $[y,\alpha_0,\alpha_1]\leftarrow F_\ell(h_0,h_1,s,x_1,1,l_{\mathrm{lrs}},r)$
    if $\alpha_1\neq -1$ **then**
      $e_1\leftarrow(e_1+1)$
      Set $\gamma_1[e_1]\leftarrow\alpha_1$
    end
    if $y\neq 0$ **then**
      $j_1\leftarrow y$
    end
  end
  if $j_0>0$ **and** $j_1>0$ **then**
    Halt the algorithm and **return** $j\leftarrow\min(j_0,j_1)$, γ_0,γ_1
  end
  if $j_0=0$ **and** $j_1>0$ **then**
    Halt the algorithm and **return** $j\leftarrow j_1$,γ_0,γ_1
  end
  if $j_1=0$ **and** $j_0>0$ **then**
    Halt the algorithm and **return** $j\leftarrow j_0$, γ_0,γ_1
  end
  if $e_0\geq q_t$ **or** $e_1\geq q_t$ **then**
    Halt the algorithm F and **return** $j\leftarrow 0$, γ_0,γ_1
  end
EndLoop

Algorithm 3: $[y, \alpha_0, \alpha_1] \longleftarrow F_\ell(h_0, h_1, s, x, b, l, r)$

Set $y \leftarrow 0, \alpha_0 \leftarrow -1, \alpha_1 \leftarrow -1$
Invoke the adversary $A(r, h_0, h_1, s)$ with randomness r
Answer tampering queries of A (if there is any at this stage) by receiving the ith tampering query (T_0, T_1) (through A),
Run F_t and compute $y \leftarrow F_t(T_b, i, x)$ for each tampering query (T_0, T_1)
if $y \neq 0$ **then**
  Return s^* to the adversary A
  Halt the algorithm and **return** y, α_0, α_1
end
else
  Return $\perp$ to the adversary A
end
Answer leakage queries of A by receiving the ith leakage query (T_0, T_1) (through A) as:
begin Answring leakage queries:
  if $\gamma_0[i] \neq -1$ *and* $\gamma_1[i] \neq -1$ **then**
    Return $\gamma_0[i]$ and $\gamma_1[i]$ to the adversary A
  end
  if $\gamma_{1-b}[i] = -1$ **then**
    Halt the algorithm F_ℓ and **return** y, α_0, α_1
  end
  if $\gamma_b[i] = -1$ **then**
    Compute $L \leftarrow T_b(x)$
    if $((\sum_{j=0}^{i} |\gamma_b[j]|) + |T_b(x)|) \leq l$ **then**
      Set $\alpha_b \leftarrow L$, halt and **return** y, α_0, α_1
    end
    if $((\sum_{j=0}^{i} |\gamma_b[j]|) + |T_b(x)|) > l$ **then**
      Set $\alpha_b \leftarrow l^*$, halt and **return** y, α_0, α_1
      (l^* is a special symbol)
    end
  end
  if $\gamma_b[i] = l^*$ **then**
    Output the special symbol l^* to the adversary A as response of the leakage query
  end
  if $\gamma_{1-b}[i] = l^*$ **then**
    Output the special symbol l^* to the adversary A as response of the leakage query
  end
end
If A halts then halt the algorithm F_ℓ and **return** y, α_0, α_1

Algorithm 4: $j \longleftarrow F_t(T, i, x)$

Compute $x' = T(x)$.
Parse the format of x as (c, t, k) and x' as (c', t', k')
if $k' \neq k$ **or** $\mathrm{Vrfy}_k(c', t') = 0$ **then**
  Set $j \leftarrow i$
end
else
  $j \leftarrow 0$
end
return j

tampering queries and F_ℓ handles leakage queries. Turing machine F is described in Algorithm 2 and its sub-algorithms.

The output of F is (j, γ_0, γ_1) while $|j| = \mathcal{O}(\log(q_t))$, $|\gamma_0| = l$ and $|\gamma_1| = l$. Note that the maximum number of queries for the algorithm F to compute the vectors γ_i for $i \in \{0, 1\}$, is $2l$, showing that $l_{\mathrm{lrs}} \geq \log(q_t) + 2l$.

Now, we show that subject to the occurrence of the event $\overline{\mathrm{Coll}} \cap \overline{\mathrm{Tamper}}$, the algorithm F outputs the correct value of j. To see this, note that when the events Tamper and Coll do not occur, the possible answers of tampering oracle are s^* and $\perp$. The output of tampering oracle is $\perp$ when at least one of the verifications of the encoding shares is false, when $j = \min(j_0, j_1)$ is the index of the query at which that oracle outputs $\perp$. Therefore, the view of algorithm F is similar to the view of A in experiment $\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)$ until the jth tampering query.

Consequently,

$$\begin{aligned}
&\Pr[(\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n) = b) \mid \overline{\mathrm{Coll}} \cap \overline{\mathrm{Tamper}}] \\
&\leq \Pr[\mathrm{Exp}_{A'}^{\mathrm{lr},\mathcal{O}(b)}(1^n) = b] \\
&\leq \frac{1}{2} + \mathrm{negl}(n).
\end{aligned}$$

■

Finally, we compute the success probability of the adversary A in the experiment $\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n)$ using Claims III.1, III.2 and III.3 as follows.

$$\begin{aligned}
&\Pr[\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n) = b] \\
&= \Pr[\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n) = b \ \cap (\mathrm{Coll} \cup \mathrm{Tamper})] \\
&\quad + \Pr[\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n) = b \ \cap (\overline{\mathrm{Coll} \cup \mathrm{Tamper}})] \\
&\leq \mathrm{negl}_0(n) + \Pr[\mathrm{Exp}_A^{\mathrm{strong},\mathcal{O}(b)}(1^n) = b \mid \overline{\mathrm{Coll}} \cap \overline{\mathrm{Tamper}}] \\
&\leq \mathrm{negl}_0(n) + \frac{1}{2} + \mathrm{negl}_1(n) \\
&= \frac{1}{2} + \mathrm{negl}(n).
\end{aligned}$$

■

IV. Conclusion and Remarks

As one of our main contributions, we proposed a secure scheme for leakage-resilient continuous strong non-malleability with tamper-detection. We would like to emphasize that *tamper-detection* is an important property of our scheme that makes it possible to detect any tampering attacks to the system. We believe that this type of encoding scheme with tamper-detection may have applications in devices requiring the tamper proof mechanisms in an algorithmic level. Note that most of current tamping proof methods are hardware based while using our approach one may obtain security by implementing an extra independent security layer as *defense in depth*.

Our proposed scheme for strong non-malleable codes also can be used to protect stateful and stateless cryptographic functionalities against tampering and leakage attacks. With some inspirations from [15], [16], our proposed encoding scheme can combined with other cryptographic schemes to

achieve some aspects of tamper resilient cryptography. To combining non-malleable codes with cryptographic functions, the secret state of the system is encoded by a non-malleable encoding scheme and the cryptographic function executes only when the self-destruction does not occur.

We also define a new *semi-strong* security model for continuous tampering encoding. The semi-strong model assumes that adversary is active until being detected by the system and this makes it possible to realize some real world attackers as viruses in this new model.

Note that the design of non-malleable encoding schemes *without any public information* is a very attractive problem that asks for further research.

REFERENCES

[1] S. Dziembowski, K. Pietrzak, and D. Wichs, "Non-malleable codes," in *Innovations in Computer Science - ICS* 2010*, Proceedings*, A. C. Yao, Ed. Tsinghua University Press, 2010, pp. 434–452. [Online]. Available: http://conference.itcs.tsinghua.edu.cn/ICS2010/content/papers/34.html

[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*, ser. Lecture Notes in Computer Science, Y. Lindell, Ed., vol. 8349. Springer, 2014, pp. 440–464. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-54242-8_19

[3] S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran, "A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations," in *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC* 2015*, Proceedings, Part I*, ser. Lecture Notes in Computer Science, Y. Dodis and J. B. Nielsen, Eds., vol. 9014. Springer, 2015, pp. 375–397. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-46494-6_16

[4] S. G. Choi, A. Kiayias, and T. 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*, ser. Lecture Notes in Computer Science, D. H. Lee and X. Wang, Eds., vol. 7073. Springer, 2011, pp. 740–758. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-25385-0_40

[5] S. Dziembowski, T. Kazana, and M. Obremski, "Non-malleable codes from two-source extractors," in *Advances in Cryptology - CRYPTO* 2013 - 33*rd Annual Cryptology Conference, Proceedings, Part II*, ser. Lecture Notes in Computer Science, R. Canetti and J. A. Garay, Eds., vol. 8043. Springer, 2013, pp. 239–257. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40084-1_14

[6] D. Aggarwal, Y. Dodis, and S. Lovett, "Non-malleable codes from additive combinatorics," in *Symposium on Theory of Computing, STOC* 2014, D. B. Shmoys, Ed. ACM, 2014, pp. 774–783. [Online]. Available: http://doi.acm.org/10.1145/2591796.2591804

[7] E. Chattopadhyay and D. Zuckerman, "Non-malleable codes against constant split-state tampering," in 55*th IEEE Annual Symposium on Foundations of Computer Science, FOCS* 2014. IEEE Computer Society, 2014, pp. 306–315. [Online]. Available: http://dx.doi.org/10.1109/FOCS.2014.40

[8] H. Chabanne, G. D. Cohen, J. Flori, and A. Patey, "Non-malleable codes from the wire-tap channel," in 2011 *IEEE Information Theory Workshop, ITW*. IEEE, 2011, pp. 55–59. [Online]. Available: http://dx.doi.org/10.1109/ITW.2011.6089565

[9] H. Chabanne, G. D. Cohen, and A. Patey, "Secure network coding and non-malleable codes: Protection against linear tampering," in *Proceedings of the* 2012 *IEEE International Symposium on Information Theory, ISIT* 2012. IEEE, 2012, pp. 2546–2550. [Online]. Available: http://dx.doi.org/10.1109/ISIT.2012.6283976

[10] N. Chandran, V. Goyal, P. Mukherjee, O. Pandey, and J. Upadhyay, "Block-wise non-malleable codes," *IACR Cryptology ePrint Archive*, vol. 2015, p. 129, 2015. [Online]. Available: http://eprint.iacr.org/2015/129

[11] 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 - 12th Theory of Cryptography Conference, TCC* 2015*, Proceedings, Part I*, ser. Lecture Notes in Computer Science, Y. Dodis and J. B. Nielsen, Eds., vol. 9014. Springer, 2015, pp. 532–560. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-46494-6_22

[12] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology - CRYPTO* 2003*, 23rd Annual International Cryptology Conference, Proceedings*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, 2003, pp. 463–481. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45146-4_27

[13] Y. Ishai, M. Prabhakaran, A. Sahai, and D. 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*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed., vol. 4004. Springer, 2006, pp. 308–327. [Online]. Available: http://dx.doi.org/10.1007/11761679_19

[14] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin, "Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering," in *Theory of Cryptography, First Theory of Cryptography Conference, TCC* 2004*, Proceedings*, ser. Lecture Notes in Computer Science, M. Naor, Ed., vol. 2951. Springer, 2004, pp. 258–277. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24638-1_15

[15] F. Liu and A. Lysyanskaya, "Tamper and leakage resilience in the split-state model," in *Advances in Cryptology - CRYPTO* 2012 - 32*nd Annual Cryptology Conference, Proceedings*, ser. Lecture Notes in Computer Science, R. Safavi-Naini and R. Canetti, Eds., vol. 7417. Springer, 2012, pp. 517–532. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32009-5_30

[16] 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*, ser. Lecture Notes in Computer Science, Y. Lindell, Ed., vol. 8349. Springer, 2014, pp. 465–488. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-54242-8_20

[17] F. Davì, S. Dziembowski, and D. Venturi, "Leakage-resilient storage," in *Security and Cryptography for Networks, 7th International Conference, SCN* 2010*, Proceedings*, ser. Lecture Notes in Computer Science, J. A. Garay and R. D. Prisco, Eds., vol. 6280. Springer, 2010, pp. 121–137. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15317-4_9

[18] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Chapman & Hall/CRC Press, 2014.

[19] Z. Jafargholi and D. Wichs, "Tamper detection and continuous non-malleable codes," in *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC* 2015*, Proceedings, Part I*, ser. Lecture Notes in Computer Science, Y. Dodis and J. B. Nielsen, Eds., vol. 9014. Springer, 2015, pp. 451–480. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-46494-6_19

[20] M. Naor and G. Segev, "Public-key cryptosystems resilient to key leakage," *SIAM J. Comput.*, vol. 41, no. 4, pp. 772–814, 2012. [Online]. Available: http://dx.doi.org/10.1137/100813464

[21] C. Hazay, A. López-Alt, H. Wee, and D. Wichs, "Leakage-resilient cryptography from minimal assumptions," in *Advances in Cryptology - EUROCRYPT* 2013*, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings*, ser. Lecture Notes in Computer Science, T. Johansson and P. Q. Nguyen, Eds., vol. 7881. Springer, 2013, pp. 160–176. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38348-9_10