Abstract. In this report, based on the paper of Dziembowski et al.’s \cite{DPW10}, we explain how we can construct an efficient code that is non-malleable with respect to modifications that effect each bit of the codeword arbitrarily. A variety of modifications of codewords are considered such as flipping each bit of the codeword, leaving it untouched, or setting it to either 0 or 1 but independently of the value of the other bits of the codeword. The main idea for constructing an efficient non-malleable code with respect to bit wise tampering family $\mathcal{F}_{Ba}$ is using Algebraic Manipulation Detection code and a type of Linear Error-Correcting Secret-Sharing schemes.

1 Introduction

Error correction and error detection are traditional ways to protect information against tampering. Indeed correction allows us to recover the original message, and detection allows that in the decoder, we detect that the message has been modified. For the class of additive functions with a bound on the number of tampered codeword components, these protections are done. For example for a codeword $c$ is tampered to $c' = c \oplus e$. More precisely, error correction, would be to require that for any tampering-function $f \in \mathcal{F}$ and any source message $s$, the tampering experiment always produces the correct decoded-message $s' = s$. \cite{DPW10,DKO13}. This notion of correction was first studied in the seminal work of Shannon \cite{Sha79} with respect to random noise, and later by Hamming \cite{Ham50}, who introduced the notion of an error-correcting code with respect to worst-case errors. Error Detection is a weaker guarantee, that requires the tampering experiment always results in either the correct value $s' = s$ or a special symbol $s' = \perp$ showing that tampering has been detected. This notion of error-detection is a weaker guarantee than error-correction, and achievable for larger families $\mathcal{F}$ of tampering functions. The tampering experiment can be used to model several interesting real-world settings, such as data transmitted over a noisy channel, or adversarial tampering of data stored in the memory of a physical device. We would like to build special encoding/decoding procedures ($Enc, Dec$), which give us some meaningful guarantees about the results of the above tampering experiment \cite{LSNCW17}.

Non-Malleable codes (NMC) \cite{DPW10} protect against active adversaries which they (the adversaries) can tamper with coded messages using a function from a family $F$, of tampering functions. Recently NMC were motivated by providing tamper resilience in cryptographic applications such as protection of secret keys that are stored in tamperable storage devices like as smart cards, that can be subjected to physical manipulations that would affect the values.
of the stored secret. NMC ensures the basic security requirement that the tampering (using functions from the function class $F$) cannot be used to generate related cryptographic values for example a digital signature for related keys. Roughly speaking, a coding scheme $(Enc, Dec)$ provides non-malleability with respect to the tampering family $F$ if for any $f \in F$, a codeword $c$ that encodes a message $s$, the decoding of $f(c)$ results in either the original message $s$, or a value $s'$ that is unrelated to $s$, and the probability of which of the two happens is independent of $s$. In the application scenario above, this property will ensure that the tampering with the device (stored codeword of the key) will result in either an unchanged key, or a key that is unrelated to the original key (and hence an unrelated digital signature). A slightly stronger notion is strong non-malleability that effectively requires that the decoded message $s'$ of a modified codeword $f(c) = c'$, where $c' \neq c$, be independent of $s$. NMC found other applications in computational cryptography, including construction of non-malleable commitment [DPW10], domain extension for public key encryption systems [CMTV15].

2 Technical Overview

Below we give a technical overview of the main contribution of [DPW10], namely the construction of a non-malleable with respect to the bitwise independent tampering. Intuitively, one can use the property of two detection codes, Algebraic Manipulation Detection code (AMD) [Jon08] and Linear Error-Correcting Secret-Sharing schemes (LECSS) [CCCX09] and construct an efficient code that is non-malleable with respect to modifications that effect each bit of the codeword arbitrarily (i.e. leave it untouched, flip it or set it to either 0 or 1) but independently of the value of the other bits of the codeword.

3 Preliminaries

We denote by $g(x, r)$ a randomized function which takes as input a string $x \in \{0, 1\}^n$ and a randomness $r \in \{0, 1\}^k$.

**Hamming weight.** The Hamming weight of a binary string $x \in \{0, 1\}^n$, denoted $w_H(x)$ is the number of 1’s in the string $x$.

**Hamming distance.** The Hamming distance $d_H(x, y)$ between two strings $x, y \in \{0, 1\}^n$ is the number of positions in which they differ. It is defined as follows,

$$d_H(x, y) := w_H(x - y)$$ (1)

**Statistical Distance.** The statistical distance of two random variables $H_1$ and $H_2$ is defined as,

$$SD(H_1, H_2) := \frac{1}{2} \sum_{x \in X} |Pr[x|H_1] - Pr[x|H_2]|$$ (2)

we say that $H_1$ and $H_2$ are statistically indistinguishable if $SD(H_1, H_2)$ is negligible. We write $H_1 = H_2$ if they are identically distributed.
Coding Scheme. A coding scheme consists of two functions: a randomized encoding function $Enc : \{0, 1\}^k \to \{0, 1\}^n$, and deterministic decoding function $Dec : \{0, 1\}^n \to \{0, 1\}^k \cup \{\perp\}$ such that, for each $s \in \{0, 1\}^k$, $Pr[Dec(Enc(s)) = s] = 1$ (over the randomness of the encoding algorithm).

Non-Malleability. A non-malleable code ensures that either the tampering experiment results in a correct decoded-message $\hat{s} = s$, or the decoded-message $\hat{s}$ is completely independent of and unrelated to the source-message $s$. In particular, if the decoded-message is $\hat{s} \neq s$, then it does not reveal any information about the source-message $s$. We now define non-malleability with respect to some family $\mathcal{F}$ of tampering functions. We say a code is non-malleable, if the result of tampering with a codeword is independent of the encoded message. To do so, we require that for each $f \in \mathcal{F}$ there is a universal distribution $D_f$ which, for all $s$, indicates what the likely outcomes are when applying a tampering function $f$ to a (random) encoding of $s$. The distribution $D_f$ only gets (black-box) access to $f(\cdot)$, but does not get $s$. We allow $D_f$ to output a special same* symbol, to indicate that tampering via $f$ does not change the initial encoded message.

In order to get a better view of malleable and non-malleable codes, we refer to El-Gamal [ElG85] and Cramer-Shoup [CS98] cryptosystems. We recall the structure of these two cryptosystems by the following:

El-Gamal cryptosystem [ElG85]. This cryptosystem consists of the two following phases:

1. Encryption phase: To encrypt message $m$ (encoded as a group element), it selects a randomness $r$, and output $C = (c_1, c_2) = (g^r, m \cdot h^r)$, where $g$ is a group generator, $h = g^x$ is a public key and $x$ is the secret key corresponds to the public key.
2. Decryption phase: To decrypt the ciphertext $C$, by using the secret key $x$ it computes $m = \frac{c_2}{c_1^x}$.

We can see that, El-Gamal cryptosystem is malleable because it is possible for an adversary to transform a ciphertext into another ciphertext which decrypts to a related plaintext. For example, given $(c_1, c_2)$ an adversary can query $(c_1, t \cdot c_2)$, which is a valid decryption for $t \cdot m$.

Cramer-Shoup cryptosystem [CS98]. This cryptosystem also consists of encryption and decryption phases as below:

1. Encryption phase: Firstly, one chooses a randomness $r$. Then to encrypt message $m$, it computes $C = (c_1, c_2, c_3, c_4, c_5) = (g^r, h^r, m \cdot A^r, B^r, H(c_1, c_2, c_3))$ where $h = g^x$, $A = g^y$ and $B = g^z$ are public keys and $x, y, z$ and $w$ are the secret keys corresponds to the public keys. Also the function $H(\cdot)$ is a hash function.
2. Decryption phase: By using the secret keys $x, y, z$ and $w$, it checks whether $c_5 = H(c_1, c_2, c_3)$ then one computes $m = \frac{c_3}{c_1^x c_2^y}$.

In Cramer-Shoup cryptosystem, an adversary can not change a ciphertext such that it decrypts to a related plaintext which it returns that Cramer-Shoup cryptosystem is non-malleable.

4 Tampering Families: bit-wise independent tampering

By $\mathcal{F}_{\text{Bit}}$ we denote the family which contains all tampering functions that tamper every bit independently. Formally, this family contains all functions,
that are defined by \( n \) functions,

\[
f_i : \{0, 1\} \rightarrow \{0, 1\} \quad \text{(for } i = 1, \ldots, n \text{)}
\]

Here each of the functions \( f_i \) is applied independently to each bit of codeword \( c \) as

\[
f(c_1, \ldots, c_n) = (f_1(c_1), \ldots, f_n(c_n)).
\]

The function \( f_i \) have only 4 possible following choices:

- **Set to 0**: It means that \( f_i \) can set any bit of codeword to 0.
- **Set to 1**: Each \( f_i \) is able to set independently each bit to 1.
- **Keep**: It means \( f_i \) does nothing.
- **Flip**: By flipping, each \( f_i \) can flip each bit of the codeword, (i.e. in the case binary, flipping means that changing bit 0 to 1 and vice versa).

We call the above family the bit-wise independent tampering family.

## 5 Construction of an Efficient Non-Malleable code

In this section we show that by using AMD codes and a type of LECSS scheme, one can construct an efficient non-malleable code with respect to bit wise tampering family \( F_{\text{Bit}} \). Before proceeding with the construction, we first define two primitives, AMD codes and LECSS scheme with a large distance.

### 5.1 Algebraic Manipulation Detection Code

Let \( m, n \in \mathbb{N}^+ \), with \( m \leq n \), and let \( 0 < \epsilon < 1 \).

**Definition 1** \(((m, n)-\text{AMD code}).

An \((m, n)\)-AMD code for short, is a probabilistic encoding function \( E : S \rightarrow G \) from a set \( S \) of size \( m \) into a finite abelian group \( G \) of order \( n \), together with a decoding function \( D : G \rightarrow S \cup \{\perp\} \) such that \( D(g) = s \) if \( g = E(s) \) for some \( s \in S \), and \( \perp \) otherwise [Jon08].

The idea behind the decoding function \( D \) is that if we decode an encoding of an element \( s \in S \), we always get \( s \), since we want unique decodeability. However, if we try to decode an element \( g \in G \) that is not an encoding, we know that the stored data has been corrupted and we output the \( \perp \). Note that if we want the adversary’s chances of fooling us to be small, the probability that \( E(s) + \delta = E(s') \) must be small. In other words, if the adversary adds an element \( \delta \) to the encoding, the chance that the result is an encoding of a different source state \( s' \) must be small. This idea is written down more precisely in the following definition:

**Definition 2** (\( \epsilon \)-AMD code). An \((m,n)\)-AMD code is called \( \epsilon \)-secure if the following holds: For \( s \) sampled at random from \( S \), and for \( \delta \) sampled from \( G \) according to some distribution independent of \( s \) and \( E(s) \), the probability that \( D(E(s) + \delta) = s' \neq s \) is at most \( \epsilon \) [Jon08].
Note that in this definition, the source state \( s \) is uniformly distributed from the point of view of our adversary. Therefore, his distribution on \( \delta \) is independent of not only \( E(s) \), but also of \( s \). To get a better view of AMD codes, we present a simple example of an AMD code and its parameters:

Let \( \mathbb{F} \) be a finite field of characteristic \( q \neq 2 \). Consider the following AMD code:

\[
E : \mathbb{F} \to \mathbb{F} \times \mathbb{F} \\
s \mapsto (s, s^2)
\]

(4)

**Theorem 1.** The above \((q, q^2)\)-AMD code \( E \) is weakly \( \epsilon \)-secure, where \( \epsilon = \frac{1}{q} \).

**Proof.** It is clear that \( E \) is an \((q, q^2)\)-AMD code. Now, we need to determine \( \epsilon \).

Fix an arbitrary translation \((\Delta s, \Delta s^2) \in \mathbb{F} \times \mathbb{F} \) with \( \Delta s \neq 0 \). Now, we need to count the number of different \( s \) that solve the following equation,

\[
(s + \Delta s)^2 = s^2 + \Delta s^2 \\
2s\Delta s = \Delta s^2 - (\Delta s)^2 \\
s = \frac{\Delta s^2 - (\Delta s)^2}{2\Delta s}
\]

(5)

There is only one solution for this equation, since \( \Delta s^2 \) and \( (\Delta s)^2 \) are fixed. Therefore, the error probability \( \epsilon \) becomes \( \frac{1}{q} \) [Jon08].

### 5.2 Linear Error-Correcting Secret-Sharing Schemes

**Definition 3** \(((d, t)\)-LECSS scheme [CCCX09]). Let \((E, D)\) be a coding scheme with source-messages \( s \in \{0, 1\}^k \) and codewords \( c \in \{0, 1\}^n \). We say that the scheme is a \((d, t)\)-LECSS scheme if the following properties hold:

1. **Linearity:** For all \( c \in \{0, 1\}^n \), such that \( D(c) \neq \perp \), all \( \Delta \in \{0, 1\}^n \), we have

\[
D(c + \Delta) = \begin{cases} 
\perp & \text{if } D(\Delta) = \perp \\
D(c) + D(\Delta) & \text{otherwise}
\end{cases}
\]

(6)

2. **Distance \( d \):** For all non-zero \( \hat{c} \in \{0, 1\}^n \) with hamming-weight \( w_H(\hat{c}) < d \), we have \( D(\hat{c}) = \perp \).

3. **Secrecy \( t \):** For any fixed \( s \in \{0, 1\}^k \), we define the random variables \( C = (C_1, \ldots, C_n) = E(s) \), where \( C_i \) denotes the bit of \( C \) in position \( i \) (and where the randomness comes from the encoding procedure). Then the random variables \( \{C_i\}_{i=1}^n \), are individually uniform over \( \{0, 1\} \) and \( t \)-wise independent.
5.3 An Efficient Non-Malleable Code

Based on AMD codes and a type of LECSS scheme, we construct an efficient code that is non-malleable. This construction is secure against bit wise independent tampering that effect each bit of the codeword arbitrarily (i.e. leave it untouched, flip it or set it to either 0 or 1), but independently of the value of the other bits of the codeword. By the following definition, we show that by composing an AMD code with a LECSS scheme, one obtains a non-malleable code for the bit wise independent tampering $F_{\text{Bit}}$ [DPW10].

**Definition 4 (ϵ-NMC scheme [DPW10]).**

Let $(E, D)$ be a $(d, t)$-LECSS with,

$$E : \{0, 1\}^k \rightarrow \{0, 1\}^m$$

and with hamming distance $d > \frac{n}{4}$

Also let $(A, V)$ be a ϵ-secure AMD code where,

$$A : \{0, 1\}^m \rightarrow \{0, 1\}^n$$

Define the composed code $(\text{Enc}, \text{Dec})$ as follows,

$$\text{Enc}(s) = E(A(s))$$

$$\text{Dec}(c) = \begin{cases} \bot & \text{if } D(c) = \bot \\ V(D(c)) & \text{otherwise} \end{cases}$$

Then $(\text{Enc}, \text{Dec})$ is non-malleable with respect to the family $F_{\text{Bit}}$ with security

$$\epsilon' \leq \max(\epsilon, 2^{-\Omega(t)})$$

6 Conclusions

In this report we introduce the notation of non-malleable codes. First, we introduce bit-wise independent tampering family which contains all tampering functions that tamper every bit independently. Then by using [DPW10] we describe two different families of detection codes, Algebraic Manipulation Detection code and Linear Error-Correcting Secret-Sharing schemes. Finally, we explain that how we can construct an efficient non-malleable code with respect to bit wise tampering family $F_{\text{Bit}}$.

References


