저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer
Homomorphic Primitives in Secret-Key Cryptography for Privacy and Authenticity

Chihong Joo

Department of Electrical and Computer Engineering
Graduate School of UNIST
Homomorphic Primitives in Secret-Key Cryptography for Privacy and Authenticity

A dissertation submitted to the Graduate School of UNIST in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Chihong Joo

6.15.2016

Approved by

[Signature]

Advisor

Aaram Yun
Homomorphic Primitives in Secret-Key Cryptography for Privacy and Authenticity

Chihong Joo

This certifies that dissertation of Chihong Joo is approved.

6. 15. 2016

signature

Advisor: Aaram Yun

Signature

Jaesik Choi

signature

Jin Ho Chung

signature

Jung Hee Cheon

signature

Jooyoung Lee
Abstract

In this thesis, we define various security notions for HMA and HAE and study relations among them. For privacy, we define a homomorphic version of IND-CCA. While for homomorphic encryption, the usual IND-CCA security is not achievable due to the malleability, nevertheless we may define a version of IND-CCA for HAE. It is because that for HAE, encryption of a plaintext is done with respect to a ‘label’, and similarly decryption of a ciphertext is done with respect to a ‘labeled program’. So, while the ciphertext is still malleable by function evaluation, a decryption query should essentially declare how the ciphertext was produced. This allows a homomorphic version of IND-CCA to be defined naturally.

For authenticity, we define UF-CMA for HMA, the homomorphic version of the unforgeability when the adversary has access to the authentication oracle. We also consider UF-CTA, where the adversary not only has the authentication oracle but also the verification oracle. Moreover, we consider strong unforgeability flavors of authenticity and define homomorphic versions accordingly: SUF-CMA and SUF-CTA. These security notions of HMA can be naturally translated to those of HAE such as UF-CPA, UF-CCA, SUF-CPA and SUF-CCA. We investigate relationship between these notions, and, for example, show that SUF-CMA implies SUF-CTA and similarly SUF-CPA implies SUF-CCA. And, we show that IND-CPA and SUF-CPA imply IND-CCA. Together, this shows that a HAE scheme with IND-CPA and SUF-CPA security is in fact IND-CCA and SUF-CCA.

Also, we propose an HAE scheme and an HMA scheme supporting arithmetic circuits. These schemes are not fully homomorphic, but only somewhat homomorphic, but we show that our schemes are fully secure. In case of our HMA scheme, it satisfies SUF-CTA and only needs a weak assumption that a PRF exists. In case of our HAE scheme, it satisfies both IND-CCA and SUF-CCA. And it is a simple and natural construction based on the error-free approximate GCD (EF-AGCD) assumption. EF-AGCD assumption was used before [25, 11, 12, 9, 10] in constructing fully homomorphic encryption schemes supporting boolean circuits, but here we use it to construct a HAE scheme supporting arithmetic circuits on $\mathbb{Z}_Q$ for $Q \in \mathbb{Z}^+$. In case of our HMA scheme, it satisfies SUF-CTA, that is, it is strongly unforgeable even though an adversary is given not only the authentication oracle but also the verification oracle.

Finally, we analyze the security of the homomorphic authenticated encryption schemes obtained by generic compositions of an homomorphic secret-key encryption (HSE) scheme and a homomorphic message authentication (HMA) scheme. There are three possible ways of generic compositions: Encrypt and Authenticate (E&A), Authenticate then Encrypt (AtE), Encrypt then Authenticate (EtA). The E&A composition preserves only unforgeability of HMA. The AtE composition preserves both privacy of HSE and unforgeability of HMA, but not strong unforgeability of HMA. The EtA composition preserves all security properties of HSE and HMA. In particular, if HSE is IND-CPA and HMA is UF-CTA, then their EtA composition achieves IND-CCA.
# Contents

## 1 Introduction

## 2 Preliminary

2.1 Notations ........................................... 5
2.2 Security assumptions ................................ 6
2.3 Pseudo-Random Function ............................ 7
2.4 Labeled Program .................................... 8
2.5 Arithmetic Circuits .................................. 8
2.6 Hash Tree ........................................... 9

## 3 Homomorphic Secret-Key Encryption .......................... 11

3.1 Definition .......................................... 11
3.2 Security Notions .................................... 12
3.3 Construction ....................................... 13
3.4 Security Proof ..................................... 14

## 4 Homomorphic message authentication ....................... 16

4.1 Definition .......................................... 16
4.2 Security Notions .................................... 18
4.3 Relations on Security Notions ........................ 21
4.4 Generic Transformation to TCT ..................... 23
4.5 Construction ....................................... 25
4.6 Security Proof ..................................... 27

## 5 Homomorphic authenticated encryption .................... 30

5.1 Definition .......................................... 30
5.2 Security Notions .................................... 32
5.3 Relations on security notions ........................ 38
5.4 Generic transformation to CCT ..................... 42
5.5 Construction ....................................... 45
5.6 Security Proof .......................................................... 48

6 Generic Compositions of HAE .............................................. 51
   6.1 Encrypt and Authenticate (E&A) .............................. 51
   6.2 Authenticate then Encrypt (AtE) ............................ 54
   6.3 Encrypt then Authenticate (EtA) ............................... 57

Bibliography ................................................................. 63
Chapter 1

Introduction

Homomorphic cryptography allows processing of cryptographically protected data. For example, homomorphic encryption lets a third party which does not have the secret key to evaluate functions implicitly using only ciphertexts so that the computed ciphertext decrypts to the correct function value. Similarly, homomorphic signature allows a third party who is not the signer to derive a signature to the output of a function, given signatures of the inputs. This possibility for secure delegation of computation could potentially be used for many applications including cloud computing, and so it makes homomorphic cryptography a very interesting area, which was recently attracting many focused research activities, especially since Gentry’s first construction [15] of a fully homomorphic encryption scheme in 2009.

While existing fully homomorphic encryption schemes are still many orders slower than ordinary encryption schemes to be truly practical, many progresses are already being made in improving the efficiency of fully homomorphic encryption schemes [25, 24, 6, 7, 16, 11, 12, 3, 5, 17, 18, 9, 4, 10]. Eventually, a truly practical fully homomorphic encryption scheme could be used to implement secure cloud computing services where even the cloud provider cannot break the privacy of the data stored and processed by the cloud.

But, if such user data is important enough to protect its privacy, in many scenarios the authenticity of the data would also be worth protecting simultaneously. Indeed, in secret-key cryptography, the authenticated encryption [23, 2, 13, 22] is exactly such a primitive protecting both privacy and authenticity of data. Therefore, we would like to study homomorphic authenticated encryption (henceforth abbreviated as HAE), which is a natural analogue of the authenticated encryption for homomorphic cryptography. An HAE is a primitive of secret-key cryptography which allows public evaluation of functions using only corresponding ciphertexts.

Just as in the case of homomorphic encryption, one important goal in this area might be to design a fully homomorphic authenticated encryption. Since there are several known constructions of fully homomorphic encryption schemes and fully homomorphic signature [19], we can think that there exist a fully homomorphic secret-key encryption scheme and a fully homomorphic message authentication scheme. And so we may construct a fully homomorphic authenticated encryption scheme by generic composition [2].
Related work

Gennaro and Wichs [14] proposed the first construction of the fully homomorphic message authentication. Their construction uses FHE, and exploits the randomness in the encryption to hide information necessary for authentication. In fact, since their scheme naturally encrypts plaintexts using FHE, it is already a fully homomorphic authenticated encryption. But, their construction essentially does not allow verification queries, so it satisfies only weaker security notions: IND-CPA and UF-CPA, according to our definition.

Catalano and Fiore [8] proposed two somewhat homomorphic message authentication schemes supporting arithmetic circuits on $\mathbb{Z}_p$ for prime modulus $p$. In their construction, a tag for a message $m$ is a polynomial $\sigma(X)$ such that its constant term $\sigma(0)$ is equal to the message $m$, and its value $\sigma(\alpha)$ on a hidden random point $\alpha$ is equal to randomness determined by the ‘label’ $\tau$ of the message $m$. While their construction is very simple and practical for low-degree polynomials, it does not protect privacy of data, and it seems that this cannot be changed by simple modifications, for example by choosing a secret random value $\beta$ as the value satisfying $\sigma(\beta) = m$. Also, in their scheme, the size of the prime modulus $p$ is determined by the security parameter, so it cannot be chosen arbitrarily by the application.

Our homomorphic authenticated encryption scheme is not as efficient as the schemes of Catalano and Fiore, but certainly more efficient than the generically composed HAE of a FHE scheme and the Catalano-Fiore homomorphic message authentication. And our scheme is also very simple and its security is based on the EF-AGCD assumption, an assumption which was used in the context of fully homomorphic encryption schemes before. Moreover, in our construction, the modulus $Q$ does not depend on the security parameter so that it can be chosen depending on the application. And our scheme can also be compared with a homomorphic encryption scheme called IDGHV presented in [9]. It supports encryption of a plaintext vector $(m_1, \ldots, m_\ell)$ where each $m_i$ is an element in $\mathbb{Z}_{Q_i}$. Like our scheme, IDGHV also uses the Chinese remainder theorem, and indeed our construction can be seen as a special-case, secret-key variant of IDGHV where $\ell = 1$, and where encryption randomness is pseudorandomly generated from the label.

The privacy of our homomorphic authenticated encryption scheme is proved using the decisional EF-AGCD assumption, which is suggested by Cheon at al. [9]. Recently, Coron et al. proposed a scale-invariant fully homomorphic encryption scheme over integers [10] in PKC 2014. In the thesis, they showed that the decisional EF-AGCD assumption is equivalent to the (computational) EF-AGCD GCD assumption. Therefore, the privacy of our scheme is in fact based on the computational EF-AGCD assumption. Note that in a previous version of this thesis, the privacy of our scheme was based more directly on the EF-AGCD assumption, but the proof was done only for smooth modulus $Q$.

Security notions of the authenticated encryption was studied before. Bellare and Namprempre [2] studied both privacy and authenticity of authenticated encryption schemes, and the authenticity notions are later studied further by Bellare, Goldreich and Mityagin [1]. Our UF-CPA and SUF-CPA can be considered as homomorphic versions of INT-PTXT-1 and INT-CTXT-1.
of [1], respectively. Our UF-CCA and SUF-CCA are comparable to homomorphic versions of INT-PTXT-M and INT-CTXT-M, respectively, but in our (S)UF-CCA, the adversary has access to the decryption oracle, while in INT-PTXT-M and INT-CTXT-M, the adversary has access to the verification oracle.

**Contribution**

Our contribution in this thesis can be summarized as follows. First, we define various security notions for HMA and HAE and study relations among them. For privacy, we define a homomorphic version of IND-CCA. While for homomorphic encryption, the usual IND-CCA security is not achievable due to the malleability, nevertheless we may define a version of IND-CCA for HAE. It is because that for HAE, encryption of a plaintext is done with respect to a ‘label’, and similarly decryption of a ciphertext is done with respect to a ‘labeled program’. So, while the ciphertext is still malleable by function evaluation, a decryption query should essentially declare how the ciphertext was produced. This allows a homomorphic version of IND-CCA to be defined naturally.

For authenticity, we define UF-CMA for HMA, the homomorphic version of the unforgeability when the adversary has access to the authentication oracle. We also consider UF-CTA, where the adversary not only has the authentication oracle but also the verification oracle. Moreover, we consider strong unforgeability flavors of authenticity and define homomorphic versions accordingly: SUF-CMA and SUF-CTA. These security notions of HMA can be naturally translated to those of HAE such as UF-CPA, UF-CCA, SUF-CPA and SUF-CCA. We investigate relationship between these notions, and, for example, show that SUF-CMA implies SUF-CTA and similarly SUF-CPA implies SUF-CCA. And, we show that IND-CPA and SUF-CPA imply IND-CCA. Together, this shows that a HAE scheme with IND-CPA and SUF-CPA security is in fact IND-CCA and SUF-CCA.

The second contribution is that we propose an HAE scheme and an HMA scheme supporting arithmetic circuits. These schemes are not fully homomorphic, but only somewhat homomorphic, but we show that our schemes are fully secure. In case of our HMA scheme, it satisfies SUF-CTA and only needs a weak assumption that a PRF exists. In case of our HAE scheme, it satisfies both IND-CCA and SUF-CCA. And it is a simple and natural construction based on the error-free approximate GCD (EF-AGCD) assumption. EF-AGCD assumption was used before [25, 11, 12, 9, 10] in constructing fully homomorphic encryption schemes supporting boolean circuits, but here we use it to construct a HAE scheme supporting arithmetic circuits on $\mathbb{Z}_Q$ for $Q \in \mathbb{Z}^+$. In case of our HMA scheme, it satisfies SUF-CTA, that is, it is strongly unforgeable even though an adversary is given not only the authentication oracle but also the verification oracle.

The third contribution is that we analyze the security of the homomorphic authenticated encryption schemes obtained by generic compositions of an homomorphic secret-key encryption (HSE) scheme and a homomorphic message authentication (HMA) scheme. There are three
possible ways of generic compositions; Encrypt and Authenticate (E&A), Authenticate then Encrypt (AtE), Encrypt then Authenticate (EtA). The E&A composition preserves only unforgeability of HMA. The AtE composition preserves both privacy of HSE and unforgeability of HMA, but not strong unforgeability of HMA. The EtA composition preserves all security properties of HSE and HMA. In particular, if HSE is IND-CPA and HMA is UF-CTA, then their EtA composition achieves IND-CCA.

Outline

The rest of the thesis is constructed as follows. As a preliminary, some notations, assumptions and definitions are given in Chapter 2. Three homomorphic primitives, a homomorphic secret-key encryption, a homomorphic message authentication and a homomorphic authenticated encryption are considered in Chapter 3, Chapter 4 and Chapter 5, respectively. Each chapter defines the syntax and security notions of the primitive and gives a concrete scheme with security proofs. In Chapter 6, three possible ways of the generic composition of a homomorphic secret-key encryption and a homomorphic message authentication to obtain a homomorphic authenticated encryption are described and analyzed on security.
Chapter 2

Preliminary

In this chapter, we give some notations and a definition of a pseudo-random function (PRF). And we define the error-free approximate greatest common divisor (EF-AGCD) assumptions, which are used as our security assumptions. Also, we give some lemmas and a description on arithmetic circuits, used as a model of computation.

2.1 Notations

In this thesis, we use the following notations for intervals of integers. For any real number $a$ and $b$, we define

$$[a, b] := \{ x \in \mathbb{Z} \mid a \leq x \leq b \},$$
$$\langle a, b \rangle := \{ x \in \mathbb{Z} \mid a < x \leq b \},$$
$$]a, b[ := \{ x \in \mathbb{Z} \mid a \leq x < b \},$$
$$]a, b) := \{ x \in \mathbb{Z} \mid a < x < b \}.$$

For any real number $a$, the nearest integer $\lfloor a \rfloor$ of $a$ is defined as the unique integer in $[a - \frac{1}{2}, a + \frac{1}{2})$. The ring $\mathbb{Z}_n$ of integers modulo $n$ is represented as the set $\langle -\frac{n}{2}, \frac{n}{2} \rangle$. This means that

$$x \mod n := x - \lfloor \frac{x}{n} \rfloor \cdot n$$

for any integer $x$. For example, $\mathbb{Z}_2 = \{0, 1\}$, $\mathbb{Z}_3 = \{-1, 0, 1\}$.

For any positive integers $n$ and $m$ with $\gcd(n, m) = 1$, CRT$_{(n, m)}$ is the isomorphism from $\mathbb{Z}_n \times \mathbb{Z}_m$ onto $\mathbb{Z}_{nm}$, satisfying

$$(\text{CRT}_{(n, m)}(a, b) \mod n, \text{CRT}_{(n, m)}(a, b) \mod m) = (a, b)$$

for any $(a, b) \in \mathbb{Z}_n \times \mathbb{Z}_m$.

In this thesis, the security parameter is always denoted as $\lambda$, and the expression

$$f(\lambda) = \text{negl}(\lambda)$$
means that \( f(\lambda) \) is a negligible function, that is, for any \( c > 0 \), \( f \) satisfies

\[
|f(\lambda)| \leq \lambda^{-c}
\]

for all sufficiently large \( \lambda \in \mathbb{Z}^+ \).

Also, \( \lg \) means the logarithm to base 2. And \( \Delta(D_1, D_2) \) denotes the statistical distance between two distributions \( D_1 \) and \( D_2 \).

### 2.2 Security assumptions

In this section we define security assumptions we are going to use in this thesis. In order to do this, first let us define some distributions.

**Definition 1.** For any positive integers \( p, q_0, \rho \), let us define the following distributions.

\[
D(p, q_0, \rho) := \{ \text{choose } q \leftarrow [0, q_0), r \leftarrow (-2^\rho, 2^\rho) : \text{output } pq + r \},
\]

Clearly, we can efficiently sample from the above distribution for any given parameters. When a distribution is given as an input to an algorithm, it means that a sampling oracle for the distribution is given; we use the same notation for a sampling oracle as that of the distribution from which it samples.

Let \( \text{PRIME} \) be the set of all prime numbers and \( \text{ROUGH}(x) \) the set of all integers having no prime factors less than \( x \).

**Definition 2** (Error-Free Approximate GCD Assumption). The (computational) \((\rho, \eta, \gamma)\)-EF-AGCD assumption is that, for any \( \text{PPT} \) adversary \( A \), we have

\[
\Pr\left[ A(\rho, \eta, \gamma, y_0, D(p, q_0, \rho)) = p \right] = \text{negl}(\lambda),
\]

where \( p \leftarrow [2^{\eta-1}, 2^\eta) \cap \text{PRIME} \), \( q_0 \leftarrow [0, 2^\gamma/p) \cap \text{ROUGH}(2^\lambda) \), and \( y_0 = pq_0 \).

The EF-AGCD assumption is suggested by Coron et al. [11] to prove the security of their variant of the DGHV scheme [25].

There is also a decisional version of the EF-AGCD assumption, which is suggested by Cheon at al. [9].

**Definition 3** (Decisional Error-Free Approximate GCD Assumption). The decisional \((\rho, \eta, \gamma)\)-EF-AGCD assumption is that, for any \( \text{PPT} \) distinguisher \( D \), we have

\[
\left| \Pr \left[ D(\rho, \eta, \gamma, y_0, D(p, q_0, \rho), z) = 1 \mid z \leftarrow D(p, q_0, \rho) \right] - \Pr \left[ D(\rho, \eta, \gamma, y_0, D(p, q_0, \rho), z) = 1 \mid z \leftarrow Z_{y_0} \right] \right| = \text{negl}(\lambda)
\]

where \( p \leftarrow [2^{\eta-1}, 2^\eta) \cap \text{PRIME} \), \( q_0 \leftarrow [0, 2^\gamma/p) \cap \text{ROUGH}(2^\lambda) \), and \( y_0 = pq_0 \).
Recently, Coron et al. proved the equivalence of the EF-AGCD assumption and the decisional EF-AGCD assumption in [10]. Later we will show that our scheme is IND-CPA under the decisional EF-AGCD assumption. Therefore, the security of our scheme is based on the EF-AGCD assumption, according to the equivalence.

2.3 Pseudo-Random Function

A pseudo-random function (PRF) is a basic primitive in cryptography. It can be used in a scheme as a replacement for an ideal random function.

**Definition 4.** Let $F : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \to \{0, 1\}^{\beta(\lambda)}$ be a family of efficiently computable functions parameterized by $\lambda$ for some polynomial $\beta$. $F$ is pseudo-random if and only if, for any PPT adversary $\text{Adv}$ the difference of probabilities

$$\left| \Pr \left[ \text{Adv}^{F(k, \cdot)}(\lambda) = 1 \right] - \Pr \left[ \text{Adv}^{R(\cdot)}(\lambda) = 1 \right] \right|$$

is negligible in $\lambda$, where $k \overset{\$}{\leftarrow} \{0, 1\}^\lambda$ and $R : \{0, 1\}^\lambda \to \{0, 1\}^{\beta(\lambda)}$ is a random function.

A PRF can be constructed from a one-way function, which is also a basic primitive in cryptography. [Reference: Katz and Lindell] Thus, the existence of a one-way function implies the existence of a PRF. And the existence of a one-way function is quite a mild assumption. In the following, we assume that a pseudo-random function (PRF) $F : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \to \{0, 1\}^{\beta(\lambda)}$ is given for some polynomial $\beta$. For each $k \in \{0, 1\}^\lambda$, we define a function $F_k : \{0, 1\}^\lambda \to \{0, 1\}^{\beta(\lambda)}$ as

$$F_k(x) := F(k, x), \forall x \in \{0, 1\}^\lambda.$$

We need the following lemma.

**Lemma 1.** Let $F : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \to \{0, 1\}^{\beta(\lambda)}$ be a PRF for some polynomial $\beta$. For some $n \in \mathbb{Z}^+$, define a function $F' : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \to \mathbb{Z}_n$ as

$$F'(k, x) := F(k, x) \mod n, \forall (k, x) \in \{0, 1\}^\lambda \times \{0, 1\}^\lambda.$$

Then $F'$ is also a PRF if $\beta - \log n = \omega(\log \lambda)$.

**Proof.** Let $\beta := \beta(\lambda)$. The set $\{0, 1\}^\beta$ can be identified with the set $[0, 2^\beta)$. Let $X$ and $Y$ be a random variable uniformly distributed over $[0, 2^\beta)$ and $[0, n)$, respectively. And define a random variable $Z := X \mod n$. We only need to show that $\Delta(X, Z)$ is negligible. We can easily find the statistical distance by a direct computation.

$$\Delta(X, Z) = (2^\beta \mod n) \cdot \left( \frac{1}{2^\beta} \cdot \left\lfloor \frac{2^\beta}{n} \right\rfloor - \frac{1}{n} \right) \leq \frac{n}{2^\beta} = \frac{1}{2^{\beta - \log n}}$$

Therefore, the statistical distance is negligible if $\beta - \log n = \omega(\log \lambda)$. This completes the proof. \qed
2.4 Labeled Program

First, let us define labeled programs, a concept first introduced in [14].

For each HAE, a set of admissible functions $\mathcal{F}$ is associated. In reality, $\mathcal{F}$ is not a set of mathematical functions, but a set of representations of mathematical functions; an element $f$ of $\mathcal{F}$ is a concrete representation of a function which can be evaluated in polynomial time, and in general, it is possible for two distinct representations $f \neq f'$ to represent the same mathematical function. It is required that any $f \in \mathcal{F}$ should represent a function of form $f : \mathcal{M}^l \rightarrow \mathcal{M}$ for some $l \in \mathbb{Z}^+$ which depends on $f$. We will simply call an element $f \in \mathcal{F}$ an admissible function. The number $l$ is the arity of $f$.

A HAE encrypts a plaintext $m \in \mathcal{M}$ under a ‘label’ $\tau \in \mathcal{L}$, and a labeled program is an admissible function together with information which plaintexts should be used as inputs. Formally, a labeled program is a tuple $P = (f, \tau_1, \ldots, \tau_l)$, where $f \in \mathcal{F}$ is an admissible function $f : \mathcal{M}^l \rightarrow \mathcal{M}$, and $\tau_i \in \mathcal{L}$ are labels for $i = 1, \ldots, l$ for each input of $f$. The idea is that, if $m_i$ are plaintexts encrypted under the label $\tau_i$, respectively; then the evaluation of the labeled program $P = (f, \tau_1, \ldots, \tau_l)$ is $f(m_1, \ldots, m_l)$.

We also define the identity labeled program with label $\tau$, which is $I_\tau = (\text{id}, \tau)$, where $\text{id} : \mathcal{M} \rightarrow \mathcal{M}$ is the identity function and $\tau \in \mathcal{L}$ is a label.

2.5 Arithmetic Circuits

We adapt the representation of arithmetic circuits as a model of computation, as Catalano and Fiore have done in [8]. For a given ring $R$, an arithmetic circuit on $R$ is an an acyclic directed graph (DAG). Each vertex and each edge of a graph is called a gate and a wire of a circuit, respectively. The in-degree and out-degree of a gate in a circuit is the number of inbound and outbound wires, respectively. We impose the following restrictions on an acyclic directed graph.

- The graph is connected.
- The in-degree of each gate is either 0 or 2.
- There is a unique gate of out-degree 0.

A gate of in-degree 0 is either a constant gate or an input gate. A constant gate is constantly evaluated as some fixed element in $R$. An input gate can be evaluated as arbitrary input $x \in R$. A wire from some gate $g_1$ to another gate $g_2$ passes the value of $g_1$ to $g_2$. A gate of in-degree 2 is either the addition gate or the multiplication gate, which represents the kind of operation taken on values coming from two inbound wires. An evaluation of these gates is naturally the result of the corresponding operation (either addition or multiplication) on values coming from two inbound wires and this value is passed to another gates through outbound wires. The unique gate of out-degree 0 is called an output gate, the value of which represents an output of the entire circuit.
The arity and the size of an arithmetic circuit is the number of input gates and entire gates in the circuit, respectively. And the depth of an arithmetic circuit is the length of the longest directed path from an input gate to an output gate in the circuit. We consider only arithmetic circuit of polynomially bounded size, because such an arithmetic circuit can be evaluated in polynomially bounded time.

Let $f$ be an arithmetic circuit on $R$ of arity $l$. $f$ determines a unique $l$-variate polynomial over $R$, considering each input gate of $f$ as an indeterminate. We use the same notation $f$ for the polynomial determined by an arithmetic circuit $f$. The degree of an arithmetic circuit $f$ is defined by the degree as a polynomial. Also, $f$ determines a function from $R^l$ into $R$, considering each input gate of $f$ as a variable on $R$. By an evaluation of an arithmetic circuit $f$ for any input $(x_1, \ldots, x_l) \in R^l$, we can get the function value $f(x_1, \ldots, x_l) \in R$.

A labeled arithmetic circuit is an arithmetic circuit such that each input gate is labeled with some bit-string distinct from elements in $R$. For an arithmetic circuit on $R$ of arity $l$ and $l$ number of bit-strings $\tau_1, \ldots, \tau_l$, a labeled arithmetic circuit $f(\tau_1, \ldots, \tau_l)$ means that each $i$-th input gate is labeled with a bit-string $\tau_i$. This notation will not make a confusion with an evaluation of $f$ on $(\tau_1, \ldots, \tau_l)$, since labels can be differentiated from elements in $R$.

### 2.6 Hash Tree

Let $H : \{0,1\}^* \to \{0,1\}^n$ be a collision-resistant hash function. We define the hash tree as in [GW2013]. The hash tree $f^H$ of an arithmetic circuit $f : \mathcal{M}^l \to \mathcal{M}$ is a function from $(\{0,1\}^*)^l$ into $\{0,1\}^n$, which takes as input bitstrings $x_i \in \{0,1\}^*$ for each input wire of $f$. For each wire $w$ in the circuit $f$, we define the value of the hash tree $f^H(x_1, \ldots, x_l)$ at $w$ recursively:

- $\text{val}(w) := H(x_i)$, if $w$ is the $i$th input wire of $f$.
- $\text{val}(w) := H(\text{val}(w_1), \ldots, \text{val}(w_t))$, if $w$ is the output wire of some gate with input wires $w_1, \ldots, w_t$.

We define the output of $f^H(x_1, \ldots, x_l)$ as the $\text{val}(w_{\text{out}})$ of the output wire $w_{\text{out}}$ of the circuit $f$. For example, if $f$ consists of only one gate, then $f^H(x_1, \ldots, x_l) = H(H(x_1), \ldots, H(x_l))$.

Also, for each wire $w$ in the circuit $f$, we define the index set $\text{ind}(w)$ associated with the wire $w$ recursively:

- $\text{ind}(w) := \{i\}$, if $w$ is the $i$th input wire of $f$.
- $\text{ind}(w) := \text{ind}(w_1) \cup \cdots \cup \text{ind}(w_t)$, if $w$ is the output wire of some gate with input wires $w_1, \ldots, w_t$.

We say that the $i$th input wire of $f$ is unused in $f$, if $i \notin \text{ind}(w_{\text{out}})$. If the $i$th input wire is unused, then the value of $f$ does not depend on the $i$th input. If the $i$th input wire is not unused, then we say that it is used in $f$. 


In this thesis, a circuit is a DAG where each vertex with positive indegree and positive outdegree is assigned a gate, and there is a unique dedicated wire $w_{out}$ called the output wire. In general, there might be many vertices with zero outdegree, but only one is the outgoing vertex of $w_{out}$. We use this definition to allow possibility of easily representing projection functions $\pi_i : M^l \to M$, for example. But, under this definition, some input wires may be unused.
Chapter 3

Homomorphic Secret-Key Encryption

3.1 Definition

An HSE scheme is a tuple $\Pi = (\text{Gen, Enc, Eval, Dec})$ of the following four PPT algorithms.

- $(ek, sk) \leftarrow \text{Gen}(1^\lambda)$: given a security parameter $\lambda$, Gen outputs a public evaluation key $ek$ and a secret key $sk$.
- $c \leftarrow \text{Enc}(sk, m)$: given a secret key $sk$ and a plaintext $m \in \mathcal{M}$, Enc outputs a ciphertext $c \in \mathcal{C}$.
- $c \leftarrow \text{Eval}(ek, f, c_1, \cdots, c_l)$: given an evaluation key $ek$, an arity-$l$ admissible function $f : \mathcal{M}^l \rightarrow \mathcal{M}$ in $\mathcal{F}$ and $l$ ciphertexts $c_1, \cdots, c_l \in \mathcal{C}$, the deterministic algorithm Eval outputs a ciphertext $c \in \mathcal{C}$.
- $m \leftarrow \text{Dec}(sk, c)$: given a secret key $sk$ and a ciphertext $c \in \mathcal{C}$, the deterministic algorithm Dec outputs a message $m \in \mathcal{M}$.

We assume that evaluation key $ek$ implicitly contains the information about a plaintext space $\mathcal{M}$, a ciphertext space $\mathcal{C}$ and an admissible function space $\mathcal{F}$. And both Eval and Dec are deterministic algorithms.

Compactness.

In order to exclude trivial constructions, we require that there exists some $c > 0$ such that, for any $\lambda \in \mathbb{Z}^+$, the output size of $\text{Eval}(ek, \cdots)$ and $\text{Enc}(sk, \cdot)$ are bounded by $\lambda^c$ for any choice of their input, when $(ek, sk) \leftarrow \text{Gen}(1^\lambda)$. That means that the ciphertext size is independent of the choice of the admissible function $f$ or the arity of $f$. 
Correctness.

An HSE scheme must satisfy the following two correctness properties.

- We should have
  \[ m = \text{Dec}(sk, \text{Enc}(sk, m)), \]
  for any \( \lambda \in \mathbb{Z}^+ \) and \( m \in \mathcal{M} \), when \((ek, sk) \leftarrow \text{Gen}(1^\lambda)\).
- We should have
  \[ f(m_1, \ldots, m_l) = \text{Dec}(sk, c), \]
  for any \( \lambda \in \mathbb{Z}^+, f \in \mathcal{F}, m_i \in \mathcal{M} \) for \( i = 1, \ldots, l \), when \((ek, sk) \leftarrow \text{Gen}(1^\lambda), c_i \leftarrow \text{Enc}(sk, m_i) \) for \( i = 1, \ldots, l \), and \( c \leftarrow \text{Eval}(ek, f, c_1, \ldots, c_l) \).

3.2 Security Notions

The goal of an HSE scheme is the privacy as same as a secret-key encryption scheme. To define a security notion of indistinguishability under chosen plaintext attack (IND-CPA), we use the following security game \( \text{IND-CPA}_{\Pi,A}(1^\lambda) \) for an HSE scheme \( \Pi \) between the challenger and the adversary \( A \), which is a natural adaptation of the corresponding security game of the secret-key encryption.

Indistinguishability under Chosen Plaintext Attack (IND-CPA)

\[ \text{IND-CPA}_{\Pi,A}(1^\lambda): \]

Initialization. A key pair \((ek, sk) \leftarrow \text{Gen}(1^\lambda)\) is generated and then \( ek \) is given to \( A \).

Queries. \( A \) may make encryption queries adaptively. For each encryption query \( m \) of \( A \), the challenger returns the answer \( c \leftarrow \text{Enc}(sk, m) \) to \( A \).

Challenge. \( A \) outputs the challenge \((m^*_0, m^*_1)\). The challenger flips a coin \( b \leftarrow \{0, 1\} \) and then gives the corresponding challenge ciphertext \( c^* \leftarrow \text{Enc}(sk, m^*_b) \) to \( A \).

Queries. Again \( A \) may make encryption queries adaptively, and such queries are answered precisely as before.

Finalization. \( A \) outputs a bit \( b' \), and then the challenger returns 1 if \( b = b' \), and 0 otherwise.

The advantage of \( A \) in the game IND-CPA for the scheme \( \Pi \) is defined as

\[ \text{Adv}_{\Pi,A}^{\text{IND-CPA}}(\lambda) := \left| \Pr[\text{IND-CPA}_{\Pi,A}(1^\lambda) = 1] - \frac{1}{2} \right|. \]

We say that an HSE scheme \( \Pi \) satisfies IND-CPA, if the advantage \( \text{Adv}_{\Pi,A}^{\text{IND-CPA}}(\lambda) \) is negligible for any PPT adversary \( A \).
3.3 Construction

As a concrete example of an HSE, we give a description of a secret-key version of somewhat homomorphic public-key encryption proposed [11]. Actually, our construction of a homomorphic authentication encryption in Chapter 5 is just a simple variant of the following HSE scheme. In this scheme, the plaintext space is the ring $\mathbb{Z}_Q$, where the modulus $Q$ can be chosen arbitrarily in the set $[2, 2^\lambda)$ and a function is represented as an arithmetic circuit on $\mathbb{Z}_Q$.

Scheme  The HSE scheme $\text{HSE}_{\text{CMNT}} := (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ is defined as follows.

$$ (pp, sk) \leftarrow \text{Gen}(1^\lambda, Q) $$

Given a security parameter $\lambda$ and a modulus $Q$, let $\eta := \eta(\lambda)$, $\gamma := \gamma(\lambda)$ and $\rho := \rho(\lambda)$. Choose a random prime integer $p$ in the set $[2^{\eta - 1}, 2^\eta)$ and a random $2^{\lambda^2}$-rough integer $n$ in the set $[0, \frac{2^\rho}{p})$. Let $N := pn$. The public parameters $pp := (\lambda, \eta, \gamma, \rho, Q, N)$ and the secret key $sk := (p, n)$. Note that $\gcd(p, Q) = 1$ since $p$ is a prime integer larger than $Q$ if $\eta > \lambda + 1$.

$$ c \leftarrow \text{Enc}(sk, m) $$

Given a secret key $sk = (p, n)$ and a plaintext $m \in \mathbb{Z}_Q$, choose two random integers $r$ and $s$ in the sets $[0, 2^\rho)$ and $[0, n)$, respectively. The ciphertext $c := \text{CRT}(p, n)(rQ + x, s) \in \mathbb{Z}_N$.

$$ c \leftarrow \text{Eval}(f, c_1, \ldots, c_l) $$

Given an arithmetic circuit $f$ of arity $l$ and $l$ ciphertexts $c_1, \ldots, c_l \in \mathbb{Z}_N$, the evaluated ciphertext $c := f(c_1, \ldots, c_l) \mod N \in \mathbb{Z}_N$.

$$ y \leftarrow \text{Dec}(sk, c) $$

Given a secret key $sk = (p, n)$ and a ciphertext $c \in \mathbb{Z}_N$, the decrypted plaintext $m := (c \mod p) \mod Q \in \mathbb{Z}_Q$.

In this scheme, the plaintext space is $\mathbb{Z}_Q$ and the ciphertext space is $\mathbb{Z}_N$. The length of a ciphertext is always $\gamma$ and so the scheme clearly satisfies the compactness. The set $\mathcal{F}$ of admissible functions will be determined by the following argument on correctness.

Correctness  Let $pp := (\lambda, \eta, \gamma, \rho, Q, N)$ and $sk = (p, n)$, where $N := pn$ and $Q \in [2, 2^\lambda)$. And let $f$ be an $l$-arity arithmetic circuit of depth $d$ for some positive integers $l$ and $d$. For any $l$ plaintexts $m_1, \ldots, m_l \in \mathbb{Z}_Q$, let $c_i \leftarrow \text{Enc}(sk, m_i)$, $i = 1, \ldots, l$. Then $c_i \mod p = r_iQ + m_i$ and $r_i \in [0, 2^\rho)$ for each $i = 1, \ldots, l$. And $\tilde{c} := \text{Eval}(f, c_1, \ldots, c_l) = f(c_1, \ldots, c_l) \mod N$. So

$$ \tilde{c} \mod p = f(c_1 \mod p, \ldots, c_l \mod p) \mod p $$

$$ = f(r_1Q + m_1, \ldots, r_lQ + m_l) \mod p $$

$$ = f(r_1Q + m_1, \ldots, r_lQ + m_l) $$
if \( f(r_1Q + m_1, \cdots, r_lQ + m_l) < p \). Then
\[
\text{Dec}(sk, \tilde{c}) = (\tilde{c} \mod p) \mod Q \\
= f(r_1Q + m_1, \cdots, r_lQ + m_l) \mod Q \\
= f(x_1, \cdots, m_l) \mod Q.
\]
Thus, the correctness property holds if \( 2^d < \eta - 1 \), that is, \( d < \log(\eta - 1) \) since the length of an output is
\[
\log f(r_1Q + m_1, \cdots, r_lQ + m_l) \leq 2^d (\lambda + \rho) \leq \eta - 1.
\]
In other words, \( f \) is admissible if the depth of \( f \) is \( d < \log(\eta - 1) \). The scheme is somewhat homomorphic since the depth of an admissible arithmetic circuit is logarithmically bounded.

**Parameters** In the scheme, the parameters \( \rho, \eta \) and \( \gamma \) are given as follows.

- \( \rho := \lambda \) to resist the brute force attack on the EF-AGCD problem.
- \( \eta := \lambda^c \cdot (\rho + \lambda) = \mathcal{O}(\lambda^{c+1}) \) to compute an arithmetic circuit of depth \( \log \lambda^c \) for some constant \( c \geq 1 \). Then \( \eta = \Omega(\lambda^2) \) and so it resists known factoring attacks.
- \( \gamma := \eta^2 \lambda = \mathcal{O}(\lambda^{2c+3}) \) to resist known attacks on the EF-AGCD problem.

### 3.4 Security Proof

**Theorem 1.** The scheme \( \text{HSE}_{\text{CMNT}} \) is IND-CPA under the decisional \( (\eta, \gamma, \rho) \)-EF-AGCD assumption.

**Proof.** Suppose there exists a PPT adversary \( A \) for the game IND-CPA such that
\[
\Pr \left[ \text{IND-CPA}_A(1^\lambda, Q) = 1 \right] \geq 1/2 + \epsilon(\lambda)
\]
for some \( Q \in [2, 2^\lambda) \) and some non-negligible function \( \epsilon \). Then, we can construct a PPT distinguisher \( D \) for the decisional \( (\eta, \gamma, \rho) \)-EF-AGCD problem, by simulating the game IND-CPA\(_A\) as follows.

**Distinguisher** \( D(\eta, \gamma, \rho, N, D(p,n,\rho), z) \)

**Initialization**

Give \( pp := (\lambda, \eta, \gamma, \rho, Q, N) \) to \( A \).

**Queries before Challenge**

For each encryption query \( m \in \mathbb{Z}_Q \) of \( A \), sample \( u \) from the given distribution \( D(p,n,\rho) \) and give \( c := (uQ + m) \mod N \) to \( A \) as an answer for the query.

**Challenge**

For the challenge \( (m_0^*, m_1^*) \) of \( A \), choose a random bit \( b \), give \( c^* := (zQ + m_b^*) \mod N \) to \( A \) as a challenge ciphertext.
Queries after Challenge

For each encryption query \( m \in \mathbb{Z}_Q \) of \( A \), sample \( u \) from the given distribution \( D(p, n, \rho) \) and give \( c := (uQ + m) \mod N \) to \( A \) as an answer for the query.

Finalization For the output \( b' \) of \( A \), return 1 if \( b = b' \). Otherwise, return 0.

Note that \( \gcd(N, Q) = 1 \). Let us consider the distribution of \( c \) produced by \( D \) in the query phase. Since \( c = uQ + m \mod N \) where \( u \leftarrow D(p, n, \rho) \), we have

\[
\begin{align*}
    c &= (sp + r)Q + m \mod N ; r \in [0, 2^p), s \in [0, n) \\
    &= sQp + (rQ + m) \mod N \\
    &= s'p + (rQ + m) \mod N ; s' = sQ \mod n
\end{align*}
\]

Therefore, \( c \mod p = rQ + m \) and \( s' \) is uniform distributed over \( \mathbb{Z}_n \). So, the distribution of \( c \) is identical to that of a real ciphertext and the encryption oracle can be simulated using \( D(p, n, \rho) \).

Now, consider the distribution of \( c^* \) in the Challenge phase. If \( z \leftarrow D(p, n, \rho) \), then the distribution of \( c^* \) is identical to that of original security game by the same reason as the above. But if \( z \not\leftarrow \mathbb{Z}_N \), then \( c^* \) is also uniformly distributed over \( \mathbb{Z}_N \) regardless of a random bit \( b \). Thus, \( c^* \) does not contain any information on the challenge plaintext \( m_b \) in case that \( z \not\leftarrow \mathbb{Z}_N \). So

\[
\Pr \left[ D(p, \eta, \gamma, N, D(p, n, \rho), z) = 1 \mid z \leftarrow D(p, n, \rho) \right] = \Pr \left[ \text{IND-CPA}_A(1^\lambda, Q) = 1 \right] \geq \frac{1}{2} + \epsilon(\lambda)
\]

and

\[
\Pr \left[ D(\eta, \gamma, \rho, N, D(p, n, \rho), z) = 1 \mid z \not\leftarrow \mathbb{Z}_N \right] = \frac{1}{2}
\]

Therefore, the advantage of \( D \) is at least non-negligible \( \epsilon \), and this completes the proof. \( \square \)
Chapter 4

Homomorphic message authentication

4.1 Definition

An HMA scheme is a tuple $\Pi = (\text{Gen}, \text{Auth}, \text{Eval}, \text{Verify})$ of the following four PPT algorithms.

- $(ek, sk) \leftarrow \text{Gen}(1^\lambda)$: given a security parameter $\lambda$, Gen outputs a public evaluation key $ek$ and a secret key $sk$.

- $\sigma \leftarrow \text{Auth}(sk, \tau, m)$: given a secret key $sk$, a label $\tau \in L$ and a message $m \in M$, Auth outputs a tag $\sigma \in T$ for the message $m$.

- $\sigma \leftarrow \text{Eval}(ek, f, \sigma_1, \cdots, \sigma_l)$: given an evaluation key $ek$, an arity-$l$ admissible function $f : M^l \rightarrow M$ in $F$ and $l$ tags $\sigma_1, \cdots, \sigma_l \in T$, Eval outputs a tag $\sigma \in T$.

- $b \leftarrow \text{Verify}(sk, (f, \tau_1, \cdots, \tau_l), m, \sigma)$: given a secret key $sk$, a labeled program $(f, \tau_1, \cdots, \tau_l)$, a message $\hat{m} \in M$ and a tag $\sigma \in T$, Verify outputs a bit $b \in \{0, 1\}$.

We assume that evaluation key $ek$ implicitly contains the information about a message space $M$, a tag space $T$, a label space $L$ and an admissible function space $F$. And both Eval and Verify are deterministic algorithms.

Compactness.

In order to exclude trivial constructions, we require that there exists some $c > 0$ such that, for any $\lambda \in \mathbb{Z}^+$, the output size of $\text{Eval}(ek, \cdots)$ and $\text{Auth}(sk, \cdot, \cdot)$ are bounded by $\lambda^c$ for any choice of their input, when $(ek, sk) \leftarrow \text{Gen}(1^\lambda)$. That means that the tag size is independent of the choice of the admissible function $f$ or the arity of $f$.

Correctness.

An HMA scheme must satisfy the following two correctness properties.
1. We should have
   \[ 1 = \text{Verify}(sk, l, m, \text{Auth}(sk, \tau, m)), \]
   for any \( \lambda \in \mathbb{Z}^+, \tau \in \mathcal{L} \) and \( m \in \mathcal{M} \), when \((ek, sk) \leftarrow \text{Gen}(1^\lambda)\).

2. We should have
   \[ 1 = \text{Verify}(sk, (f, \tau_1, \ldots, \tau_l), f(m_1, \ldots, m_l), \sigma), \]
   for any \( \lambda \in \mathbb{Z}^+, f \in \mathcal{F}, \tau_i \in \mathcal{L}, m_i \in \mathcal{M} \) for \( i = 1, \ldots, l \), when \((ek, sk) \leftarrow \text{Gen}(1^\lambda), \sigma_i \leftarrow \text{Auth}(sk, \tau_i, m_i) \) for \( i = 1, \ldots, l \), and \( \sigma \leftarrow \text{Eval}(ek, f, \sigma_1, \ldots, \sigma_l) \).

**Constant testability.**

Given an HMA \( \Pi \), an admissible function \( f : \mathcal{M}^l \rightarrow \mathcal{M} \) of arity \( l \), a subset \( I \) of the index set \( \{1, \ldots, l\} \), messages \((m_i)_{i \in I} \in \mathcal{M}^{\lvert I\rvert} \), and their corresponding tags \((\sigma_i)_{i \in I} \in \mathcal{T}^{\lvert I\rvert} \), consider the following functions:

\[
\tilde{m}_{f,(m_i)_{i \in I}} := f(m_i)_{i \in I}
\]
\[
\tilde{\sigma}_{f,(\sigma_i)_{i \in I}} := \text{Eval}(ek, f, (\sigma_i)_{i \in I}).
\]

More explicitly, \( \tilde{m}_{f,(m_i)_{i \in I}} \) is a function from \( \mathcal{M}^{\lvert I\rvert} \) to \( \mathcal{M} \) defined by

\[
\tilde{m}_{f,(m_i)_{i \in I}}(m_j)_{j \not\in I} := f(m_1, \ldots, m_l),
\]

for any \((m_j)_{j \not\in I} \in \mathcal{M}^{\lvert I\rvert} \). And \( \tilde{\sigma}_{f,(\sigma_i)_{i \in I}} \) is a function from \( \mathcal{T}^{\lvert I\rvert} \) to \( \mathcal{T} \) defined by

\[
\tilde{\sigma}_{f,(\sigma_i)_{i \in I}}(\sigma_j)_{j \not\in I} := \text{Eval}(ek, f, \sigma_1, \ldots, \sigma_l),
\]

That is, messages or tags for indices in \( I \) are fixed, and messages or tags for indices in \( \{1, \ldots, l\} \setminus I \) are considered as variables. In short, \( \tilde{m} = \tilde{m}_{f,(m_i)_{i \in I}} \) and \( \tilde{\sigma} = \tilde{\sigma}_{f,(\sigma_i)_{i \in I}} \) are partially evaluated functions. In particular, \( \tilde{m}_{f,(m_i)_{i \in I}} \) and \( \tilde{\sigma}_{f,(\sigma_i)_{i \in I}} \) are constant functions if \( I = \{1, \ldots, l\} \).

We may need to determine whether such a function \( \tilde{m} = \tilde{m}_{f,(m_i)_{i \in I}} \) or \( \tilde{\sigma}_{f,(\sigma_i)_{i \in I}} \) is constant or not. So we define a property called ‘constant testability’ as follows. Depending whether we are working on messages or tags, we define two versions of constant testability accordingly.

**Definition 5.** We say that a HMA scheme \( \Pi \) satisfies the **message constant testability (MCT)** if there exists a PPT algorithm that determines if the function \( \tilde{m} = \tilde{m}_{f,(m_i)_{i \in I}} \) is constant or not with overwhelming probability, for any evaluation key \( ek \) generated by \( \Pi.Gen \), any admissible function \( f : \mathcal{M}^l \rightarrow \mathcal{M} \) of arity \( l \), any subset \( I \) of the index set \( \{1, \cdots, l\} \) and any \((m_i)_{i \in I} \in \mathcal{M}^{\lvert I\rvert} \).

**Definition 6.** We say that a HMA scheme \( \Pi \) satisfies the **tag constant testability (TCT)** if there exists a PPT algorithm that determines if the function \( \tilde{\sigma} = \tilde{\sigma}_{f,(\sigma_i)_{i \in I}} \) is constant or not with overwhelming probability, for any evaluation key \( ek \) generated by \( \Pi.Gen \), any admissible function \( f : \mathcal{M}^l \rightarrow \mathcal{M} \) of arity \( l \), any subset \( I \) of the index set \( \{1, \cdots, l\} \) and any \((\sigma_i)_{i \in I} \in \mathcal{T}^{\lvert I\rvert} \).
When the set of admissible functions supported by a HMA is simple, both MCT and TCT may be satisfied. But, the message constant testability might be a difficult property to be satisfied in general; for example, if a HMA supports general boolean circuits, then MCT implies that the CIRCUIT-SAT problem can be solved in polynomial time with overwhelming probability, therefore the polynomial hierarchy $\text{PH}$ collapses.

On the other hand, we claim that a HMA to satisfy the tag constant testability is a relatively mild requirement: unlike the message space $\mathcal{M}$, often the tag space $\mathcal{T}$ might be a large ring, and $\tilde{\sigma}_{f, (\sigma_i)_{i \in I}}$ is a polynomial on the ring $\mathcal{T}$, in which case we may use the Schwartz-Zippel lemma to perform the polynomial identity testing. This applies to our HMA scheme to be presented in this thesis, as shown in Theorem 16.

Moreover, we show that if $\Pi$ is a HMA which does not necessarily satisfy TCT, then there is a simple generic transformation which turns it into another HMA $\Pi'$ which satisfies TCT, while preserving original security properties satisfied by $\Pi$. This will be shown in Theorems 13, 14 and 15. Without loss of generality, we assume the TCT property to be an additional requirement for a HMA to satisfy.

### 4.2 Security Notions

The goal of an HMA scheme is the authenticity as same as a message authentication. To define a security game for authenticity of an HMA scheme, we consider two possible attack models; one is the chosen-message attack (CMA) and the other is the chosen-tag attack (CTA). In the CMA, we allow that an adversary adaptively makes authentication queries. In the CTA, we allow that an adversary adaptively makes not only authentication queries but also verification queries. There is one trivial restriction on authentication queries; a label used to generate a tag for some message can not be used again to generate a tag for another message. In other words, for each label $\tau \in \mathcal{L}$, either $\tau$ is used, that is, there exists a unique message $m \in \mathcal{M}$ such that exactly one tag $\sigma \in \mathcal{T}$ has generated by the authentication algorithm $\text{Auth}(sk, \tau, m)$, or not used. To prevent a used label from being reused in the authentication algorithm $\text{Auth}$, we can maintain a history of authentication queries as follows.

**Authentication History** An authentication history $H : \mathcal{L} \to \{\bot\} \cup (\mathcal{M} \times \mathcal{T})$ is a function, which is dynamically changed as authentication queries made by an adversary.

- At first, $H$ is initialized as constantly $\bot$. That is, $H(\tau) = \bot$ for all $\tau \in \mathcal{L}$.
- For each authentication query $(\tau, m) \in \mathcal{L} \times \mathcal{M}$, if $\tau$ is new, that is, $H(\tau) = \bot$, then the query is accepted and we update $H(\tau) := (m, \sigma)$, where $\sigma \leftarrow \text{Auth}(sk, \tau, m)$. Otherwise, if $\tau$ is used, that is, $H(\tau) \neq \bot$, then the query is rejected.

In the following, we assume that an adversary does not make an authentication query for a used label.
The goal of an adversary in a security game for authenticity is to make a forgery. We define a forgery as follows.

**Forgery** Let \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) be a forgery attempt given by an adversary. It is a forgery if and only if \(1 = \text{Verify}(sk, (f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) and satisfies one of the following two conditions.

- \(\tilde{m}_{f, (m_i)_{i \in I}}\) is not constant; a forgery of type 1.
- \(\tilde{m}_{f, (m_i)_{i \in I}}\) is constantly \(\tilde{m}\) but \(\tilde{m} \neq \hat{m}\); a forgery of type 2.

where \(I\) is the set of indices of used labels in \(\tau_1, \cdots, \tau_l\).

Also, we define a strong forgery to obtain a notion of a stronger security,

**Strong Forgery** Let \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) be a forgery attempt given by an adversary. It is a strong forgery if and only if \(1 = \text{Verify}(sk, (f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) and satisfies one of the following two conditions.

- Either \(\tilde{m}_{f, (m_i)_{i \in I}}\) or \(\tilde{\sigma}_{f, (\sigma_i)_{i \in I}}\) is not constant; a strong forgery of type 1.
- \(\tilde{m}_{f, (m_i)_{i \in I}}\) is constantly \(\tilde{m}\) and \(\tilde{\sigma}_{f, (\sigma_i)_{i \in I}}\) is constantly \(\tilde{\sigma}\) but \((\tilde{m}, \tilde{\sigma}) \neq (\hat{m}, \hat{\sigma})\); a strong forgery of type 2.

where \(I\) is the set of indices of used labels in \(\tau_1, \cdots, \tau_l\).

Now, we can define four security notions, UF-CMA, SUF-CMA, UF-CTA and SUF-CTA.

**Unforgeability under Chosen Message Attack (UF-CMA)**

\(\text{UF-CMA}_{A}(1^\lambda)\):

**Initialization**

Given a security parameter \(\lambda\), generate a pair \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\). Give \(pp\) to \(A\).

**Queries**

\(A\) can make authentication queries adaptively. For each authentication query \((\tau, m)\) of \(A\), give \(\sigma \leftarrow \text{Auth}(sk, \tau, m)\) to \(A\) as an answer for the query.

**Finalization**

For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) of \(A\), if it is indeed a forgery, then output 1. Otherwise, output 0.

The advantage of \(A\) in the game UF-CMA is defined as

\[
\text{Adv}^\text{UF-CMA}_{A}(1^\lambda) := \Pr\left[\text{UF-CMA}_{A}(1^\lambda) = 1\right]
\]

We say that an HMA satisfies UF-CMA, if \(\text{Adv}^\text{UF-CMA}_{A}(1^\lambda)\) is negligible for any PPT adversary \(A\).
Strong Unforgeability under Chosen Message Attack (SUF-CMA)

\textbf{SUF-CMA}_A(1^\lambda):

\textbf{Initialization}
Given a security parameter \( \lambda \), generate a pair \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\). Give pp to \( A \).

\textbf{Queries}
\( A \) can make authentication queries adaptively. For each authentication query \((\tau, m)\) of \( A \), give \( \sigma \leftarrow \text{Auth}(sk, \tau, m) \) to \( A \) as an answer for the query.

\textbf{Finalization}
For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) of \( A \), if it is indeed a strong forgery, then output 1. Otherwise, output 0.

The advantage of \( A \) in the game SUF-CMA is defined as
\[
\text{Adv}^\text{SUF-CMA}_A(1^\lambda) := \Pr[\text{SUF-CMA}_A(1^\lambda) = 1]
\]
We say that an HMA satisfies SUF-CMA, if \( \text{Adv}^\text{SUF-CMA}_A(1^\lambda) \) is negligible for any PPT adversary \( A \).

Unforgeability under Chosen Tag Attack (UF-CTA)

\textbf{UF-CTA}_A(1^\lambda):

\textbf{Initialization}
Given a security parameter \( \lambda \), generate a pair \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\). Give pp to \( A \).

\textbf{Queries}
\( A \) can make authentication queries and verification queries adaptively. For each authentication query \((\tau, m)\) of \( A \), give \( \sigma \leftarrow \text{Auth}(sk, \tau, m) \) to \( A \) as an answer for the query. For each verification query \(((f, \tau_1, \cdots, \tau_l), m, \sigma)\) of \( A \), give \( b \leftarrow \text{Verify}(sk, (f, \tau_1, \cdots, \tau_l), m, \sigma) \) to \( A \) as an answer for the query.

\textbf{Finalization}
For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) of \( A \), if it is indeed a forgery, then output 1. Otherwise, output 0.

The advantage of \( A \) in the game UF-CTA is defined as
\[
\text{Adv}^\text{UF-CTA}_A(1^\lambda) := \Pr[\text{UF-CTA}_A(1^\lambda) = 1]
\]
We say that an HMA satisfies UF-CTA, if \( \text{Adv}^\text{UF-CTA}_A(1^\lambda) \) is negligible for any PPT adversary \( A \).
Strong Unforgeability under Chosen Tag Attack (SUF-CTA)

SUF-CTA\(_A(1^\lambda)\):

Initialization
Given a security parameter \(\lambda\), generate a pair \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\). Give pp to A.

Queries
A can make authentication queries and verification queries adaptively. For each authentication query \((\tau, m)\) of A, give \(\sigma \leftarrow \text{Auth}(sk, \tau, m)\) to A as an answer for the query. For each verification query \(((f, \tau_1, \cdots, \tau_l), m, \sigma)\) of A, give \(b \leftarrow \text{Verify}(sk, (f, \tau_1, \cdots, \tau_l), m, \sigma)\) to A as an answer for the query.

Finalization
For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) of A, if it is indeed a strong forgery, then output 1. Otherwise, output 0.

The advantage of A in the game SUF-CTA is defined as

\[
\text{Adv}^{\text{SUF-CTA}}_A(1^\lambda) := \Pr\left[\text{SUF-CTA}_A(1^\lambda) = 1\right]
\]

We say that an HMA satisfies SUF-CTA, if \(\text{Adv}^{\text{SUF-CTA}}_A(1^\lambda)\) is negligible for any PPT adversary A.

4.3 Relations on Security Notions

In this section, we prove some relations on four security notions, UF-CMA, UF-CTA, SUF-CMA and SUF-CTA, which are defined in the previous section. First, we have trivial implication.

**Theorem 2.** UF-CTA implies UF-CMA. And SUF-CTA implies SUF-CMA.

**Proof.** In chosen-message attack (CMA), an adversary can make only authentication queries. But an adversary can make both authentication queries and verification queries in chosen-tag attack (CTA). Clearly, a stronger power is given to an adversary in CTA than in CMA. So, a security notion in CTA is stronger than that in CMA. \(\square\)

**Theorem 3.** SUF-CMA implies UF-CMA. And SUF-CTA implies UF-CTA.

**Proof.** By definition, a forgery is a strong forgery. Thus, an adversary that can not produce a strong forgery, can not produce a forgery. \(\square\)

The following is the main theorem in this section.

**Theorem 4.** SUF-CMA together with TCT imply SUF-CTA.
Proof. We prove the theorem by a hybrid argument to transform the game SUF-CTA into another game that is essentially the same as the game SUF-CMA.

Let $A$ be any PPT adversary for the game SUF-CTA. Without loss of generality, we may assume that $A$ makes exactly $q = q(\lambda)$ number of verification queries. For each $k \in \{0, \ldots, q\}$, define $\text{SUF-CTA}^k$ to be the game that is identical to SUF-CTA except that the first $k$ verification queries are answered by the following verification simulation.

**Verification Simulation**

For a verification query $((f, \tau_1, \cdots, \tau_l), m, \sigma)$ of $A$, let $I$ be the set of indices of used labels in $\tau_1, \cdots, \tau_l$. If $\tilde{\sigma}_{f,(\sigma_i)_{i \in I}}$ is constantly $\sigma$ and $\tilde{m}_{f,(\sigma_i)_{i \in I}}(m_j)_{j \notin I} = m$ for $(m_j)_{j \notin I} \in \mathcal{M}^{l-|I|}$, then give 1 to $A$ as an answer for the query. Otherwise, give 0 as an answer for the query.

Clearly, SUF-CTA$^0$ is equal to the original security game SUF-CTA. So,

$$\text{Adv}_{A}^{\text{SUF-CTA}^0}(\lambda) = \text{Adv}_{A}^{\text{SUF-CTA}}(\lambda).$$

And, note that the verification simulation does not use any secret information and is efficiently computable by the TCT property. So, the adversary $A$ does not obtain any useful information by the verification queries at all in the game SUF-CTA$^q$. Formally, we can easily construct an adversary $A'$ which plays SUF-CMA game and makes the same number of encryption queries as $A$ does, and satisfying

$$\text{Adv}_{A}^{\text{SUF-CTA}^q}(\lambda) = \text{Adv}_{A'}^{\text{SUF-CMA}}(\lambda).$$

For each $k \in \{1, \ldots, q\}$, the difference between $\text{Adv}_{A}^{\text{SUF-CTA}^{k-1}}(\lambda)$ and $\text{Adv}_{A}^{\text{SUF-CTA}^k}(\lambda)$ is bounded by the probability that the verification simulation on the $k$-th verification query made by $A$ fails (that is, is different from the real verification). Let us show that the verification simulation fails for the verification query $\hat{t} := ((f, \tau_1, \cdots, \tau_l), \tilde{m}, \tilde{\sigma})$ only if $\hat{t}$ is a strong forgery. If so, we may construct a PPT adversary $A''$ for the game SUF-CMA using the $i$th verification query made by $A$. Specifically, $A''$ runs the adversary $A$ until it makes the $i$th verification query, while answering the authentication queries using its own authentication queries and answering the previous verification queries by the verification simulation. Then $A''$ aborts the running of $A$, and outputs the $i$th verification query of $A$ as its own forgery attempt. So, we can conclude that

$$\left| \text{Adv}_{\Pi,A}^{\text{SUF-CTA}^{i-1}}(\lambda) - \text{Adv}_{\Pi,A}^{\text{SUF-CTA}^i}(\lambda) \right| \leq \text{Adv}_{\Pi,A''}^{\text{SUF-CMA}}(\lambda) = \text{negl}(\lambda).$$

For a verification query $\hat{t} := ((f, \tau_1, \cdots, \tau_l), \tilde{m}, \tilde{\sigma})$, suppose that $\tilde{\sigma} := \text{Eval}(ek, f, (\sigma_i)_{i \in I})$ is nonconstant. In this case, verification simulation fails if and only if $\text{Verify}(sk, \hat{t}) = 1$. So $\hat{t}$ is a strong forgery if verification simulation fails since $\text{Verify}(sk, \hat{t}) = 1$ and $\tilde{\sigma}$ is nonconstant. On the another hand, suppose that $\tilde{\sigma}$ is constant. By the correctness, $\text{Verify}(sk, \hat{t}) = 1$, where $\hat{t} := ((f, \tau_1, \cdots, \tau_l), \tilde{m}, \tilde{\sigma})$. If $(\tilde{m}, \tilde{\sigma}) \neq (\hat{m}, \hat{\sigma})$, then $\hat{t} = \tilde{t}$ and $\text{Verify}(sk, \hat{t}) = 1$. In this case, verification simulation is exact. But, if $(\tilde{m}, \tilde{\sigma}) = (\hat{m}, \hat{\sigma})$, then verification simulation fails if and
only if \( \text{Verify}(sk, \hat{t}) = 1 \). So \( \hat{t} \) is a strong forgery if verification simulation fails since \( \text{Verify}(sk, \hat{t}) = 1 \) and \( \hat{\sigma} \) is constant but \((\hat{m}, \hat{\sigma}) \neq (m, \sigma)\).

Therefore,

\[
\text{Adv}_{\Pi, A}^{\text{SUf-CTA}}(\lambda) \leq \text{Adv}_{\Pi, A}^{\text{SUf-CTA}}(\lambda) + \sum_{i=1}^{q} \left| \text{Adv}_{\Pi, A}^{\text{SUf-CTA}}(\lambda) - \text{Adv}_{\Pi, A}^{\text{SUf-CTA}^{-1}}(\lambda) \right|
\]

\[
\leq \text{Adv}_{\Pi, A'}^{\text{SUf-CMA}}(\lambda) + \sum_{i=1}^{q} \text{Adv}_{\Pi, A''}^{\text{SUf-CMA}}(\lambda)
\]

\[
\leq (q + 1) \cdot \text{Adv}_{\Pi}^{\text{SUf-CMA}}(\lambda)
\]

\( = \text{negl}(\lambda) \).

For any PPT adversary \( \text{Adv}, A_{\Pi, A}^{\text{SUf-CTA}}(\lambda) = A_{\Pi, A}^{\text{SUf-CTA}}(\lambda) \) is negligible since \( \text{Adv}_{\Pi}^{\text{SUf-CMA}}(\lambda) \) is negligible. This completes the proof. \( \square \)

### 4.4 Generic Transformation to TCT

Suppose that \( \Pi \) is a HMA which is not necessarily tag constant testable. We describe a generic construction that transforms a HMA \( \Pi \) into another HMA \( \Pi' \) satisfying TCT while preserving SUF-CMA of the original scheme \( \Pi \). Our construction is based on the Merkle hash tree technique used by Gennaro and Wichs. For concreteness, here we assume that \( \Pi \) represents admissible SUF-CMA of the original scheme \( \Pi \). Our construction is based on the Merkle hash tree technique used by Gennaro and Wichs. For concreteness, here we assume that \( \Pi \) represents admissible functions as arithmetic circuits. In such a case, we also assume that \( \text{Eval} \) function becomes an arithmetic circuit on the tag space.

Now, using a pseudo-random function \( F \) and a family \( \mathcal{H} \) of collision-resistant hash functions, we can transform a HMA scheme \( \Pi = (\text{Gen}, \text{Auth}, \text{Eval}, \text{Verify}) \) to another HMA scheme \( \Pi' = (\text{Gen}', \text{Auth}', \text{Eval}', \text{Verify}') \) as follows.

**Scheme \( \Pi' = (\text{Gen}', \text{Auth}', \text{Eval}', \text{Verify}') \):**

- \((pp', sk') \leftarrow \text{Gen}'(1^\lambda)\): Generate keys \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\) and \( k \leftarrow \{0, 1\}^\lambda \) and \( H \leftarrow \mathcal{H} \). Return \( pp' := (pp, H) \) and \( sk' := (sk, k) \).

- \( \sigma' \leftarrow \text{Auth}'(sk', \tau, m)\): Let \( h := H(F_k(\tau)) \) and \( \sigma \leftarrow \text{Auth}(sk, \tau, m) \). Return \( \sigma' := (h, \sigma) \).

- \( \sigma' \leftarrow \text{Eval}'(f, \sigma_1', \cdots, \sigma_l')\): Let \( f : \mathcal{M} \rightarrow \mathcal{M} \) be a circuit. For each \( i = 1, \cdots, l \), parse \( \sigma_i' \) as \((h_i, \sigma_i)\). Let \( h := f^H(h_1, \cdots, h_l) \) and \( \sigma \leftarrow \text{Eval}(f, \sigma_1, \cdots, \sigma_l) \). Return \( \sigma' := (h, \sigma) \).

- \( b \leftarrow \text{Verify}'(sk', (f, \tau_1, \cdots, \tau_l), m, \sigma')\): Let \( f : \mathcal{M} \rightarrow \mathcal{M} \) be a circuit. Parse \( \sigma' \) as \((h, \sigma)\). For each \( i = 1, \cdots, l \), let \( h_i := H(F_k(\tau_i)) \). If \( h = f^H(h_1, \cdots, h_l) \) and \( 1 = \text{Verify}(sk, (f, \tau_1, \cdots, \tau_l), m, \sigma) \), then return 1. Otherwise, return 0.

Above, we assume that \( F_k : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \) and \( H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda \).
II’ and II share the same message space $\mathcal{M}$, the same label space $\mathcal{L}$ and the same set $\mathcal{F}$ of admissible functions. And the tag space of II’ is $\{0, 1, \ldots, l\}$, where $\mathcal{T}$ is the ciphertext space of II. The correctness and compactness of II’ can easily be concluded from that of II. Below, we show that the constructed scheme II’ satisfies the TCT property, and also this generic transformation preserves SUF-CMA.

**Theorem 5.** The HMA scheme II’ satisfies TCT.

**Proof.** Let $f: \mathcal{M}^I \rightarrow \mathcal{M}$ be an admissible function of arity $I$, $I$ a subset of the index set $\{1, \ldots, l\}$, $(m_i)_{i \in I} \in \mathcal{M}^{|I|}$ and $(\sigma_i)_{i \in I} = (f_i, \sigma_i)_{i \in I} \in \{0, 1\}^|\{1, \ldots, l\}|$ some messages and their corresponding tags. Consider the function $\tilde{\sigma}_{f, (\sigma_i)_{i \in I}} : \{0, 1\}^l \rightarrow \{0, 1\}^|\{1, \ldots, l\}|$. Then

$$\tilde{\sigma}_{f, (\sigma_i)_{i \in I}} = Eval(f, (h_i, \sigma_i)_{i \in I}) = (f^H(h_i)_{i \in I}, Eval(f, (\sigma_i)_{i \in I})).$$

The above expression $\tilde{\sigma}$ is a function of the values for the ‘missing’ indices: $(h_i)_{i \notin I}$ and $(c_i)_{i \notin I}$.

If $I = \{1, \ldots, l\}$, then $\tilde{\sigma}_{f, (\sigma_i)_{i \in I}}$ is clearly constant. Now, suppose that there exist an index $i \in \{1, \ldots, l\}$ such that $i \notin I$. Since the underlying hash function $H$ is collision-resistant, the hash tree $f^H$ cannot be constant on each variable except with negligible probability. So, $\tilde{\sigma}_{f, (\sigma_i)_{i \in I}}$ is not constant except with negligible probability regardless of the scheme II. This means that we can trivially determine if $\tilde{\sigma}_{f, (\sigma_i)_{i \in I}}$ is constant or not. Therefore, II’ satisfies TCT.

**Theorem 6.** If II is SUF-CMA, then II’ is also SUF-CMA.

**Proof.** Let $A'$ be a PPT adversary engaged in the security game SUF-CMA$_{II', A'}$. Using $A'$, we construct a PPT adversary $A$ for the security game SUF-CMA$_{II, A}$ which simulates the game SUF-CMA$_{II', A'}$.

**Adversary** $A(1^\lambda)$:

1. **Initialization.** Given pp in the scheme II, the adversary $A$ picks $H \leftarrow \mathcal{H}$, $k \leftarrow \{0, 1\}^\lambda$, and gives pp' := (pp, H) to $A'$. And $A$ keeps the PRF key $k$ by himself.

2. **Queries.** Whenever $A'$ makes an authentication query $(\tau, m)$, $A$ makes the same authentication query. Receiving the answer $\sigma$, the adversary $A$ computes $h := H(F_k(\tau))$, $A$ gives $\sigma' := (h, \sigma)$ to $A'$ as an answer for the query.

3. **Forgery.** Given a forgery attempt $((f, \tau_1, \ldots, \tau_l), \hat{m}, (\hat{h}, \hat{\sigma}))$ output by $A'$, $A$ outputs $((f, \tau_1, \ldots, \tau_l), \hat{m}, \hat{\sigma})$.

Now, let us show that if the forgery attempt $((f, \tau_1, \ldots, \tau_l), \hat{m}, (\hat{h}, \hat{\sigma}))$ of $A'$ is a strong forgery for SUF-CMA$_{II', A'}$, then $((f, \tau_1, \ldots, \tau_l), \hat{m}, \hat{\sigma})$ is also a strong forgery for SUF-CMA$_{II, A}$, except with negligible probability. Let $I$ be the set of indices $i \in \{1, \ldots, l\}$ such that $(\tau_i, m_i, c_i) \in S$.

Since $((f, \tau_1, \ldots, \tau_l), \hat{c})$ is valid in II’, $((f, \tau_1, \ldots, \tau_l), \hat{c})$ is also valid in II. Also, we have $f^H(H(F_k(\tau_1)), \ldots, H(F_k(\tau_l))) = \hat{h}$. 

24
Suppose $I \neq \{1, \ldots, l\}$, and assume there exists at least an index $j \not\in I$ which is used in the circuit $f$. In this case, we claim that $f^H(H(F_k(\tau_1)), \cdots, H(F_k(\tau_l))) = \hat{h}$ only with negligible probability. Since $j$ is new, $F_k(\tau_j)$ is computationally indistinguishable to a random number $r_j \leftarrow \{0,1\}^\lambda$. Due to the collision resistance of $H$, the probability

$$\Pr \left[ f^H(H(F_k(\tau_1)), \cdots, r_j, \cdots, H(F_k(\tau_l))) = \hat{h} \right]$$

should be negligible. So, $f^H(H(F_k(\tau_1)), \cdots, H(F_k(\tau_l))) = \hat{h}$ holds only with negligible probability. Therefore, since we already have $f^H(H(F_k(\tau_1)), \cdots, H(F_k(\tau_l))) = \hat{h}$, we may assume with negligible exception that any $i \not\in I$ is unused in the circuit $f$. In this case, both $\tilde{c}'$ and $\tilde{c}$ are constants, where

$$\tilde{c}' = (\tilde{h}, \tilde{c}),$$

with

$$\tilde{h} := f^H((H(F_k(\tau_i)))_{i \in I}), \tilde{c} = \text{Eval}(ek, f, (c_i)_{i \in I}).$$

But then $((f, \tau_1, \cdots, \tau_l), \tilde{c}')$ must be a strong forgery of type 2. So we have

$$\tilde{c}' = (\tilde{h}, \tilde{c}) \neq (\hat{h}, \hat{c}) = \hat{c}.'$$

Therefore, $\hat{c} \neq \hat{c}$, and $((f, \tau_1, \cdots, \tau_l), \hat{c})$ is a strong forgery of type 2 for $\Pi$. This shows that

$$\text{Adv}_{\Pi, A}^{\text{SUF-CPA}}(\lambda) \leq \text{Adv}_{\Pi, A}^{\text{SUF-CPA}}(\lambda) = \text{negl}(\lambda).$$

Hence $\Pi'$ is SUF-CMA.

4.5 Construction

In this section, we describe the HMA scheme given in [21] and prove its security of SUF-CTA in the next section.

Scheme

We assume that a PRF $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^{\beta(\lambda)}$ is given for some polynomial $\beta$. The message space is $\mathbb{Z}_Q$, where the modulus $Q$ can be chosen arbitrarily in the set $[2, 2^\lambda]$. The HMA scheme $\text{HMA}_{\text{JY}} := (\text{Gen, Auth, Eval, Verify})$ is defined as follows.

$$(\text{pp, sk}) \leftarrow \text{Gen}(1^\lambda, Q)$$

Given a security parameter $\lambda$ and a modulus $Q$, let $\gamma := \gamma(\lambda)$ and $\beta := \beta(\lambda)$. Choose a random prime integer $p$ in the set $[2^\lambda, 2^{\lambda+1}]$ and a random string $k$ in the set $\{0,1\}^\lambda$. Let $F_k := F(k, \cdot) : \{0,1\}^\lambda \rightarrow [0, 2^\beta)$. The set of public parameters $\text{pp} := (\lambda, \gamma, \beta, F, Q)$ and the secret key $\text{sk} := (p, k)$. Note that $\gcd(p, Q) = 1$ since $p$ is a prime integer larger than $Q$. 
\[ \sigma \leftarrow \text{Auth}(sk, \tau, x) \]

Given a secret key \( sk = (p, k) \), a label \( \tau \in \{0, 1\}^\lambda \) and a message \( x \in \mathbb{Z}_Q \), compute \( a := \text{CRT}_{(p, Q)}(s, x) \) where \( s := F_k(\tau) \mod p \). Choose a random integer \( r \) in the set \( [0, \frac{2^\lambda}{pQ}) \). The tag \( \sigma := rpQ + a \in \mathbb{Z} \).

\[ \tilde{\sigma} \leftarrow \text{Eval}(f, \sigma_1, \cdots, \sigma_l) \]

Given an arithmetic circuit \( f \) of arity \( l \) and \( l \) tags \( \sigma_1, \cdots, \sigma_l \in \mathbb{Z} \), the evaluated tag \( \tilde{\sigma} := f(\sigma_1, \cdots, \sigma_l) \) in the integer ring \( \mathbb{Z} \).

\[ b \leftarrow \text{Verify}(sk, f(\tau_1, \cdots, \tau_l), y, \tilde{\sigma}) \]

Given a secret key \( sk = (p, k) \), a labeled arithmetic circuit \( f(\tau_1, \cdots, \tau_l) \), a message \( y \in \mathbb{Z}_Q \) and a tag \( \tilde{\sigma} \in \mathbb{Z} \), compute \( v := f(F_k(\tau_1) \mod p, \cdots, F_k(\tau_l) \mod p) \mod p \). If \( v = \tilde{\sigma} \mod p \) and \( y = \tilde{\sigma} \mod Q \), then \( b := 1 \). Otherwise, \( b := 0 \).

The message space is \( \mathbb{Z}_Q \) and the tag space is \( \mathbb{Z} \) and the label space is \( \{0, 1\}^\lambda \). And the set \( \mathcal{F} \) is arithmetic circuits of depth \( d \), where \( d(\lambda) \) is logarithmically bounded. So the scheme \( \text{HMA}_{\lambda^y} \) is somewhat homomorphic. Note that the evaluation algorithm \( \text{Eval} \) and the verification algorithm \( \text{Verify} \) are deterministic.

**Correctness** Let \( pp := (\lambda, \beta, F, Q) \) and \( sk = (p, k) \) where \( Q \in [2, 2^\lambda) \). And let \( f \) be an arithmetic circuit of arity \( l \) for some positive integer \( l \). For each \( i = 1, \cdots, l \), let \( \sigma_i \leftarrow \text{Auth}(sk, \tau_i, x_i) \) where \( x_i \in \mathbb{Z}_Q \) and \( \tau_i \in \{0, 1\}^\lambda \). Then, for each \( i = 1, \cdots, l \), \( \sigma_i \mod p = F_k(\tau_i) \mod p \) and \( \sigma_i \mod Q = x_i \). Let \( \tilde{\sigma} := \text{Eval}(f, \sigma_1, \cdots, \sigma_l) \), then \( \tilde{\sigma} = f(\sigma_1, \cdots, \sigma_l) \). So,

\[
\tilde{\sigma} \mod p = f(\sigma_1 \mod p, \cdots, \sigma_l \mod p) \mod p
= f(F_k(\tau_1) \mod p, \cdots, F_k(\tau_l) \mod p) \mod p
= v
\]

and

\[
\tilde{\sigma} \mod Q = f(\sigma_1 \mod Q, \cdots, \sigma_l \mod Q) \mod Q
= f(x_1, \cdots, x_l) \mod Q.
\]

Therefore, the correctness holds for any arithmetic circuit.

**Parameters** In the scheme, the parameters \( \gamma, \beta \) and \( d \) are given as follows.

- \( \gamma := 3\lambda \) to satisfy the condition \( \gamma - 2\lambda = \omega(\log \lambda) \) in the security proof.
- \( \beta := 2\lambda \) to obtain a PRF with codomain \( \mathbb{Z}_p \) for any \( p \) of length \( \lambda \).
- \( d := \log \lambda^c \) for some constant \( c \geq 1 \) to satisfy the compactness. The length of tags is at most \( 2^d\gamma = 3\lambda^{c+1} \).
4.6 Security Proof

In this section, we prove that the scheme $\text{HMA}_JY$ is fully secure, that is, SUF-CTA. We need the following two theorems, $\text{HMA}_JY$ is TCT and SUF-CMA.

**Theorem 7.** The scheme $\text{HMA}_JY$ satisfies TCT.

*Proof.* Let $ek$ be an evaluation key generated by $\text{Gen}(1^\lambda, Q)$ for some $\lambda \in \mathbb{Z}^+$ and a modulus $Q$, $f$ any admissible arity-$l$ arithmetic circuit for some $l \in \mathbb{Z}^+$ and $(\sigma_i)_{i \in I}$ any element in $\mathbb{Z}^{1|I|}$ for some subset $I$ of the index set $\{1, \ldots, l\}$. We construct an algorithm $\text{ALG-TCT}$ that determines if $\tilde{\sigma} = \text{Eval}(ek, f, (\sigma_i)_{i \in I})$ is constant or not with overwhelming probability, as follows.

**procedure** $\text{ALG-CCT}(ek, f, (\sigma_i)_{i \in I})$:

if $I = \{1, \ldots, l\}$ then
    return 1
else
    $(\sigma_j^0)_{j \notin I}, (\sigma_j^1)_{j \in I} \leftarrow (\mathbb{Z})^{l-|I|}$
    if $\tilde{\sigma}(\sigma_j^0)_{j \notin I} = \tilde{\sigma}(\sigma_j^1)_{j \in I}$ then
        return 1
    else
        return 0

The algorithm $\text{ALG-TCT}$ is essentially the usual probabilistic polynomial identity testing. In the scheme $\Pi$, $\tilde{\sigma}$ can be considered as an $(l - |I|)$-variate polynomial over $\mathbb{Z}$ of degree not greater than $\text{deg } f$. We have

$$\tilde{\sigma} = f(\sigma_i)_{i \in I} : \mathbb{Z}^{l-|I|} \to \mathbb{Z}$$

In case $I = \{1, \ldots, l\}$, $\tilde{\sigma}$ is clearly constant and the algorithm outputs 1 correctly. In case $I \subseteq \{1, \ldots, l\}$, consider the function $\tilde{\sigma}' := \tilde{\sigma} - \tilde{\sigma}(\sigma_j^0)_{j \notin I} \mod y_0$ for any $(\sigma_j^0)_{j \notin I} \in \mathbb{Z}^{l-|I|}$. If $\tilde{\sigma}$ is constant, then $\tilde{\sigma}'$ is constantly zero and $\tilde{\sigma}'(\sigma_j^1)_{j \in I} = \tilde{\sigma}(\sigma_j^1)_{j \in I} - \tilde{\sigma}(\sigma_j^0)_{j \notin I} = 0$ for any $(\sigma_j^1)_{j \in I} \in \mathbb{Z}^{l-|I|}$. So, $\tilde{\sigma}(\sigma_j^0)_{j \notin I} = \tilde{\sigma}(\sigma_j^1)_{j \in I}$ and the algorithm outputs 1 correctly. If $\tilde{\sigma}$ is not constant, then $\tilde{\sigma}'$ is not constantly zero and the algorithm outputs the incorrect answer 1 when $\tilde{\sigma}(\sigma_j^0)_{j \notin I} \equiv \tilde{\sigma}(\sigma_j^1)_{j \in I}$, that is, $\tilde{\sigma}'(\sigma_j^1)_{j \in I} = 0$. This is the only case that the algorithm outputs an incorrect answer. So the error probability of the algorithm is

$$\Pr[\tilde{\sigma}'(\sigma_j^1)_{j \in I} = 0 \mid (\sigma_j^1)_{j \in I} \leftarrow \mathbb{Z}^{l-|I|}],$$

when $\tilde{\sigma}'$ is not constantly zero.

To find an upper bound on the error probability of the algorithm using Schwartz-Zippel lemma,

$$\Pr[\tilde{\sigma}'(\sigma_j^1)_{j \in I} = 0 \mid (\sigma_j^1)_{j \in I} \leftarrow \mathbb{Z}^{l-|I|}] \leq \Pr[\tilde{\sigma}'(\sigma_j^1)_{j \notin I} \equiv 0 \mod p \mid (\sigma_j^1)_{j \notin I} \leftarrow \mathbb{Z}_p^{l-|I|}] \leq \frac{\deg f}{p} \leq \frac{d}{2\lambda} = \text{negl}(\lambda)$$
where $\tilde{d}$ is the upper bound on degrees of the admissible functions in our scheme, which is polynomially bounded. Therefore, the error probability of the algorithm is negligible and we can efficiently determine if $\tilde{d}$ is constant or not with overwhelming probability.

\begin{theorem}
The scheme HMA\textsubscript{JY} is SUF-CMA.
\end{theorem}

\begin{proof}
We prove the theorem by a hybrid argument to transform the game SUF-CMA into another game with a negligible advantage for any PPT adversary $A$. Let $pp := (\lambda, \beta, F, Q)$ and $sk := (p, k)$ for any $Q \in [2, 2^{\lambda-1}]$ and $F'_k : \{0,1\}^\lambda \rightarrow \mathbb{Z}_p$ be defined as $F'_k(\tau) := F_k(\tau) \mod p$ for any $\tau \in \{0,1\}^\lambda$. We know that if $\beta - \lambda = \omega(\lambda)$, then $F'_k : \{0,1\}^\lambda \rightarrow \mathbb{Z}_p$ is also a PRF for any $p \in [2^{\lambda-1}, 2^\lambda)$. Define GAME\textsubscript{0} as the game that is identical to SUF-CMA except that the PRF $F'_k : \{0,1\}^\lambda \rightarrow \mathbb{Z}_p$ is replaced with a real random function $G : \{0,1\}^\lambda \rightarrow \mathbb{Z}_p$. This make a negligible difference of advantages since $F'_k$ is a PRF. That is,

$$\left| \text{Adv}^{\text{SUF-CMA}}_A(\lambda, Q) - \text{Adv}^{\text{GAME}\textsubscript{0}}_A(\lambda, Q) \right| \leq \text{negl}(\lambda)$$

Define GAME\textsubscript{1} as the game that is identical to GAME\textsubscript{0} except that all of the encryption queries are answered by the following encryption simulation.

\textbf{Encryption Simulation}

For each authentication query $(\tau, x)$ of $A$, if $\tau$ is new, then choose a random integer $r$ in the set $[0, \lceil 2^{\gamma/Q} \rceil)$ and give $\sigma := rQ + x$ to $A$ as an answer for the query. And then update $H(\tau) := (x, \sigma)$. Otherwise, that is, if $\tau$ is used, then reject the query.

We have to show that the statistical distance between the distributions of a ciphertext $c_0$ in GAME\textsubscript{0} and a ciphertext $c_1$ in GAME\textsubscript{1}, is negligible. The distribution of $c_1$ is $c_1 = rQ + x$, where $r \overset{\$}{\leftarrow} [0, \lceil 2^\gamma/pQ \rceil)$ and the distribution of $c_0$ is

\begin{align*}
c_0 &= rpQ + a \quad ; r \overset{\$}{\leftarrow} [0, \lceil 2^\gamma/pQ \rceil) \\
&= rpQ + r'Q + x \quad ; r' \overset{\$}{\leftarrow} [0, p) \\
&= (rp + r')Q + x \\
&= r''Q + x \quad ; r'' \overset{\$}{\leftarrow} [0, p \cdot \lceil 2^\gamma/pQ \rceil).
\end{align*}

So, $\Delta(c_0, c_1) \leq \frac{pQ}{2^\gamma} \leq \frac{1}{2^{\gamma-2}r}$ and the distance is negligible if $\gamma - 2\lambda = \omega(\log \lambda)$. Thus,

$$\left| \text{Adv}^{\text{GAME}\textsubscript{0}}_A(\lambda, Q) - \text{Adv}^{\text{GAME}\textsubscript{1}}_A(\lambda, Q) \right| \leq q(\lambda) \cdot \text{negl}(\lambda),$$

where $q$ is the number of encryption queries made by $A$, which is polynomially bounded. Note that the above simulation of encryption does not use the secret prime $p$ and we can postpone the choice of $p$ until it is needed.

Finally, define GAME\textsubscript{2} as follows.

\textbf{GAME\textsubscript{2}(1^\lambda, Q):}
Initialization

Given a security parameter $\lambda$ and $Q$, let $\gamma := \gamma(\lambda)$ and $\beta := \beta(\lambda)$. Initialize an authentication history $H$ as $\perp$ and give $pp := (\lambda, \gamma, \beta, F, Q)$ to $A$.

Queries

$A$ can make authentication queries adaptively. For each authentication query $(\tau, x)$ of $A$, if $\tau$ is new, then choose a random integer $r$ in the set $[0, 2^\gamma Q)$ and give $\sigma := rQ + x$ to $A$ as an answer for the query. And then update $H(\tau) := (x, \sigma)$. Otherwise, that is, if $\tau$ is used, then reject the query.

Finalization

For the forgery attempt $(f(\tau_1, \ldots, \tau_l), y, \hat{\sigma})$ of $A$, randomly choose $(c_i)_{i \in I}$ in the set $[0, 2^\gamma)^{|I|}$ and compute $\tilde{\sigma} := \text{Eval}_{f(\tau_1, \ldots, \tau_l)}|_H(c_i)_{i \in I}$. If $\tilde{\sigma} \neq \hat{\sigma}$ and $(\tilde{\sigma} - \hat{\sigma}) \mod p = 0$ for a randomly chosen prime integer $p \in [2^{\lambda-1}, 2^\lambda)$, then output 1. Otherwise, output 0.

The game GAME$_2$ is identical to GAME$_1$ except the finalization phase. The only difference between two games GAME$_1$ and GAME$_2$, is the case that $\text{Eval}_{f(\tau_1, \ldots, \tau_l)}|_H$ is not constant but $\tilde{\sigma} = \hat{\sigma}$ in the finalization phase.

$\Box$
Chapter 5

Homomorphic authenticated encryption

In this section, we define a primitive of a homomorphic authenticated encryption (HAE) and its security. And we propose a concrete scheme that is somewhat homomorphic but fully secure under the EF-AGCD assumption.

5.1 Definition

A HAE is a tuple $\Pi = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ of the following four PPT algorithms.

- $(ek, sk) \leftarrow \text{Gen}(1^\lambda)$: given a security parameter $\lambda$, $\text{Gen}(1^\lambda)$ outputs a public evaluation key $ek$ and a secret key $sk$.

- $c \leftarrow \text{Enc}(sk, \tau, m)$: given a secret key $sk$, a label $\tau \in L$ and a plaintext $m \in M$, $\text{Enc}(sk, \tau, m)$ outputs a ciphertext $c \in C$.

- $\tilde{c} \leftarrow \text{Eval}(ek, f, c_1, \cdots, c_l)$: given an evaluation key $ek$, an arity-$l$ admissible function $f : M^l \rightarrow M$ in $F$ and $l$ ciphertexts $c_1, \cdots, c_l \in C$, $\text{Eval}$ outputs a ciphertext $\tilde{c} \in C$.

- $m$ or $\perp \leftarrow \text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \tilde{c})$: given a secret key $sk$, a labeled program $(f, \tau_1, \cdots, \tau_l)$ and a ciphertext $\tilde{c} \in C$, $\text{Dec}$ outputs a message $m \in M$ or $\perp$.

We assume that evaluation key $ek$ implicitly contains the information about a plaintext space $M$, a ciphertext space $C$, a label space $L$, and an admissible function space $F$. And both $\text{Eval}$ and $\text{Dec}$ are deterministic algorithms.

Compactness.

In order to exclude trivial constructions, we require that there exists some $c > 0$ such that, for any $\lambda \in \mathbb{Z}^+$, the output size of $\text{Eval}(ek, \cdots)$ and $\text{Enc}(sk, \cdot, \cdot)$ are bounded by $\lambda^c$ for any choice
of their input, when \((ek, sk) \leftarrow \text{Gen}(1^\lambda)\). That means that the ciphertext size is independent of the choice of the admissible function \(f\) or the arity of \(f\).

**Correctness.**

A HAE scheme must satisfy the following two correctness properties:

- We should have
  \[ m = \text{Dec}(sk, I_\tau, \text{Enc}(sk, \tau, m)), \]
  for any \(\lambda \in \mathbb{Z}^+\), \(\tau \in \mathcal{L}\) and \(m \in \mathcal{M}\), when \((ek, sk) \leftarrow \text{Gen}(1^\lambda)\).

- We should have
  \[ f(m_1, \ldots, m_l) = \text{Dec}(sk, f, \tau_1, \ldots, \tau_l, c), \]
  for any \(\lambda \in \mathbb{Z}^+\), any \(f \in \mathcal{F}\), any \(\tau_i \in \mathcal{L}\), \(m_i \in \mathcal{M}\) for \(i = 1, \ldots, l\), when \((ek, sk) \leftarrow \text{Gen}(1^\lambda)\), \(c_i \leftarrow \text{Enc}(sk, \tau_i, m_i)\) for \(i = 1, \ldots, l\), and \(c \leftarrow \text{Eval}(ek, f, c_1, \ldots, c_l)\).

In addition, we require that a HAE should satisfy a property we call ciphertext constant testability, which will be explained next.

**Constant testability.**

As done in an HMA, constant testability is defined as follows. Given an HAE \(\Pi\), an admissible function \(f: \mathcal{M}^l \to \mathcal{M}\) of arity \(l\), a subset \(I\) of the index set \(\{1, \ldots, l\}\), plaintexts \((m_i)_{i \in I} \in \mathcal{M}^{|I|}\), and their corresponding ciphertext \((c_i)_{i \in I} \in \mathcal{C}^{|I|}\), consider the following functions:

\[
\tilde{m}_{f,(m_i)_{i \in I}} := f(m_i)_{i \in I}
\]
\[
\tilde{c}_{f,(c_i)_{i \in I}} := \text{Eval}(ek, f, (c_i)_{i \in I})
\]

More explicitly, \(\tilde{m}_{f,(m_i)_{i \in I}}\) is a function from \(\mathcal{M}^{|I|}\) to \(\mathcal{M}\) defined by
\[
\tilde{m}_{f,(m_i)_{i \in I}}(m_j)_{j \notin I} := f(m_1, \ldots, m_l),
\]
for any \((m_j)_{j \notin I} \in \mathcal{M}^{|I|}\). And \(\tilde{c}_{f,(c_i)_{i \in I}}\) is a function from \(\mathcal{C}^{|I|}\) to \(\mathcal{C}\) defined by
\[
\tilde{c}_{f,(c_i)_{i \in I}}(c_j)_{j \notin I} := \text{Eval}(ek, f, c_1, \ldots, c_l),
\]

That is, messages or ciphertexts for indices in \(I\) are fixed, and messages or ciphertexts for indices in \(\{1, \ldots, l\} \setminus I\) are considered as variables. In short, \(\tilde{m} = \tilde{m}_{f,(m_i)_{i \in I}}\) and \(\tilde{c} = \tilde{c}_{f,(c_i)_{i \in I}}\) are partially evaluated functions. In particular, \(\tilde{m}_{f,(m_i)_{i \in I}}\) and \(\tilde{c}_{f,(c_i)_{i \in I}}\) are constant functions if \(I = \{1, \ldots, l\}\).

We may need to determine whether such a function \(\tilde{m} = \tilde{m}_{f,(m_i)_{i \in I}}\) or \(\tilde{c}_{f,(c_i)_{i \in I}}\) is constant or not. So we define a property called ‘constant testability’ as follows. Depending whether we are working on messages or ciphertexts, we define two versions of constant testability accordingly.
Definition 7. We say that a HAE scheme $\Pi$ satisfies the plaintext constant testability (PCT) if there exists a PPT algorithm that determines if the function $\tilde{m} = \tilde{m}_{f,(m_i)_{i \in I}}$ is constant or not with overwhelming probability, for any evaluation key $ek$ generated by $\Pi$.Gen, any admissible function $f : \mathcal{M}^I \rightarrow \mathcal{M}$ of arity $l$, any subset $I$ of the index set $\{1, \cdots, l\}$ and any $(m_i)_{i \in I} \in \mathcal{M}^{|I|}$.

Definition 8. We say that a HAE scheme $\Pi$ satisfies the ciphertext constant testability (CCT) if there exists a PPT algorithm that determines if the function $\tilde{c}_{f,(c_i)_{i \in I}}$ is constant or not with overwhelming probability, for any evaluation key $ek$ generated by $\Pi$.Gen, any admissible function $f : \mathcal{M}^I \rightarrow \mathcal{M}$ of arity $l$, any subset $I$ of the index set $\{1, \cdots, l\}$ and any $(c_i)_{i \in I} \in \mathcal{C}^{|I|}$.

When the set of admissible functions supported by a HAE is simple, both PCT and CCT may be satisfied. But, the plaintext constant testability might be a difficult property to be satisfied in general; for example, if a HAE supports general boolean circuits, then PCT implies that the CIRCUIT-SAT problem can be solved in polynomial time with overwhelming probability, therefore the polynomial hierarchy PH collapses.

On the other hand, we claim that a HAE to satisfy the ciphertext constant testability is a relatively mild requirement: unlike the plaintext space $\mathcal{M}$, often the ciphertext space $\mathcal{C}$ might be a large ring, and $\tilde{c}_{f,(c_i)_{i \in I}}$ is a polynomial on the ring $\mathcal{C}$, in which case we may use the Schwartz-Zippel lemma to perform the polynomial identity testing. This applies to our HAE scheme to be presented in this thesis, as shown in Theorem 16.

Moreover, we show that if $\Pi$ is a HAE which does not necessarily satisfy CCT, then there is a simple generic transformation which turns it into another HAE $\Pi'$ which satisfies CCT, while preserving original security properties satisfied by $\Pi$. This will be shown in Theorems 13, 14 and 15. Without loss of generality, we assume the CCT property to be an additional requirement for a HAE to satisfy.

5.2 Security Notions

The security goals of an HAE scheme is both privacy and authenticity as same as an authenticated encryption. To define a security game for an HAE scheme, we consider two possible attack models; one is the chosen-plaintext attack (CPA) and the other is the chosen-ciphertext attack (CCA). In the CPA, we allow that an adversary adaptively makes encryption queries. In the CCA, we allow that an adversary adaptively makes not only encryption queries but also decryption queries. There is one trivial restriction on encryption queries; a label used to generate a ciphertext for some plaintext can not be used again to generate a ciphertext for another plaintext. In other words, for each label $\tau \in \mathcal{L}$, either $\tau$ is used, that is, there exists a unique plaintext $m \in \mathcal{M}$ such that exactly one ciphertext $c \in \mathcal{C}$ has generated by the encryption algorithm $\text{Enc}(sk, \tau, m)$, or not used. To prevent a used label from being reused in the encryption algorithm $\text{Enc}$, we can maintain a history of encryption queries as follows.
**Encryption History** An encryption history \( H : \mathcal{L} \rightarrow \{ \bot \} \cup (\mathcal{M} \times \mathcal{T}) \) is a function, which is dynamically changed as encryption queries made by an adversary.

- At first, \( H \) is initialized as constantly \( \bot \). That is, \( H(\tau) = \bot \) for all \( \tau \in \mathcal{L} \).
- For each encryption query \( (\tau, m) \in \mathcal{L} \times \mathcal{M} \), if \( \tau \) is new, that is, \( H(\tau) = \bot \), then the query is accepted and we update \( H(\tau) := (m, c) \), where \( c \leftarrow \text{Enc}(sk, \tau, m) \). Otherwise, if \( \tau \) is used, that is, \( H(\tau) \neq \bot \), then the query is rejected.

In the following, we assume that an adversary does not make an encryption query for a used label.

**Privacy**

Here we define two security notions, IND-CPA and IND-CCA for privacy of HAE. Our definition of IND-CPA for HAE is a homomorphic version of the IND-CPA security. We use the following security game IND-CPA\(_{\Pi,A}(1^\lambda)\) between the challenger and the adversary \( A \), which is a natural adaptation of the corresponding security game of the symmetric-key encryption.

**Indistinguishability under Chosen Plaintext Attack (IND-CPA)**

\[
\text{IND-CPA}_{\Pi,A}(1^\lambda): \\
\text{Initialization.} \ A \text{ key pair } (ek, sk) \leftarrow \text{Gen}(1^\lambda) \text{ is generated. Then } ek \text{ is given to } A. \\
\text{Queries.} \ A \text{ may make encryption queries adaptively. For each encryption query } (\tau, m) \text{ of } A, \text{ the challenger returns an answer } c \leftarrow \text{Enc}(sk, \tau, m) \text{ to } A. \\
\text{Challenge.} \ A \text{ outputs the challenge } (\tau^*, m^*_0, m^*_1). \text{ The challenger flips a coin } b \overset{\$}{\leftarrow} \{0, 1\}, \text{ gives the corresponding challenge ciphertext } c^* \leftarrow \text{Enc}(sk, \tau^*, m^*_b) \text{ to } A. \\
\text{Queries.} \ A \text{ again may make encryption queries adaptively, and such queries are answered precisely as before.} \\
\text{Finalization.} \ A \text{ outputs a bit } b', \text{ and then the challenger returns } 1 \text{ if } b = b', \text{ and } 0 \text{ otherwise.}
\]

The advantage of \( A \) in the game IND-CPA for the scheme \( \Pi \) is defined as

\[
\text{Adv}^{\text{IND-CPA}}_{\Pi,A}(\lambda) := \left| \Pr[\text{IND-CPA}_{\Pi,A}(1^\lambda) = 1] - \frac{1}{2} \right|.
\]

We say that a HAE \( \Pi \) satisfies IND-CPA, if the advantage \( \text{Adv}^{\text{IND-CPA}}_{\Pi,A}(\lambda) \) is negligible for any PPT adversary \( A \).

We also consider a homomorphic version of the IND-CCA security of secret-key encryption. Even though the usual IND-CCA security is not achievable for homomorphic encryption due to the malleability, nevertheless we may define a version of IND-CCA for HAE. It is because
that for HAE, decryption of a ciphertext is done with respect to a labeled program. So, while the ciphertext is still malleable by function evaluation, a decryption query should essentially declare how the ciphertext was produced. This allows a homomorphic version of IND-CCA to be defined naturally as follows.

Homomorphic IND-CCA for a HAE $\Pi = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$ is defined using the following security game $\text{IND-CCA}_{\Pi,A}$, which is also a natural extension of the security game $\text{IND-CCA}$ of a secret-key encryption.

This time, the important difference is on the definition of legality of a decryption query after the Challenge phase. In the IND-CCA game for the symmetric encryption, the only illegal decryption query after Challenge phase is the decryption query for the challenge ciphertext itself. On the other hand, in the HAE case, any decryption query for a ciphertext that was produced by function evaluation which may nontrivially depend on the input $m^*_0$ or $m^*_1$ should be considered illegal, since decryption of that ciphertext could reveal the bit $b$. This is formalized as follows.

To check the legality of a decryption query after Challenge, the security game keeps the encryption history $S$. Then, we say that a decryption query $( (f, \tau_1, \cdots, \tau_l), c)$ after Challenge phase is illegal, if $\tau^* = \tau_{i^*}$ for some $i^* \in I$, and the two functions $\tilde{f}_0$ and $\tilde{f}_1$ are not equal, where

$$I := \{i \in \{1, \cdots, l\} \mid (\tau_i, m_i, c) \in S \text{ for some } m_i \in \mathcal{M}, c \in \mathcal{C}\},$$

$$\tilde{f}_0 := f(m_i)_{i \in I}, \text{ with } m_{i^*} = m^*_0,$$

$$\tilde{f}_1 := f(m_i)_{i \in I}, \text{ with } m_{i^*} = m^*_1.$$

This means that the function value of $f$ depends nontrivially whether $m^*_0$ or $m^*_1$ is used as the $i^*$th plaintext input. In the following security game, it is forbidden for the adversary $A$ to make any illegal decryption query after the Challenge phase.

### Indistinguishability under Chosen Ciphertext Attack (IND-CCA)

**IND-CCA}_{\Pi,A}(1^\lambda):**

**Initialization.** A key pair $(ek, sk) \leftarrow \text{Gen}(1^\lambda)$ is generated. Then $ek$ is given to the adversary $A$.

**Queries.** $A$ may make encryption queries and decryption queries adaptively. For each encryption query $(\tau, m)$ of $A$, the challenger returns an answer $\text{Enc}(sk, \tau, m)$ to $A$. For each decryption query $((f, \tau_1, \cdots, \tau_l), c)$ of $A$, the challenger returns an answer $\text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), c)$.

**Challenge.** $A$ outputs the challenge tuple $(\tau^*, m^*_0, m^*_1)$. The challenger flips a coin $b \overset{\$}{\leftarrow} \{0, 1\}$, gives the corresponding challenge ciphertext $c^* \leftarrow \text{Enc}(sk, \tau^*, m^*_b)$ to $A$.  

34
Queries After Challenge. Again, $A$ may make encryption queries and decryption queries adaptively. This time, it is forbidden for $A$ to make illegal decryption queries. Then, any encryption or decryption query of $A$ is answered precisely as before.

Finalization. $A$ outputs a bit $b'$, and then the challenger returns 1 if $b = b'$, and 0 otherwise.

The advantage of $A$ in the game IND-CCA for the scheme $\Pi$ is defined as

$$\text{Adv}^{\text{IND-CCA}}_{\Pi,A}(\lambda) := \left| \Pr[\text{IND-CCA}_{\Pi,A}(1^\lambda) = 1] - \frac{1}{2} \right|.$$  

We say that a HAE $\Pi$ satisfies IND-CCA, if the advantage $\text{Adv}^{\text{IND-CCA}}_{\Pi,A}(\lambda)$ is negligible for any PPT adversary $A$ which does not make illegal decryption queries.

Authenticity

The goal of an adversary in a security game for authenticity is to make a forgery. We define a forgery as follows.

Forgery Let $((f, \tau_1, \cdots, \tau_l), \hat{c})$ be a forgery attempt given by an adversary. It is a forgery if and only if $\bot \neq \text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$ and satisfies one of the following two conditions.

- $\tilde{m}_{f,(m_i)\in I}$ is not constant; a forgery of type 1.
- $\tilde{m}_{f,(m_i)\in I}$ is constantly $\tilde{m}$ but $\tilde{m} \neq \text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$; a forgery of type 2.

where $I$ is the set of indices of used labels in $\tau_1, \cdots, \tau_l$.

Also, we define a strong forgery to obtain a notion of a stronger security,

Strong Forgery Let $((f, \tau_1, \cdots, \tau_l), \hat{c})$ be a forgery attempt given by an adversary. It is a strong forgery if and only if $\bot \neq \text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$ and satisfies one of the following two conditions.

- Either $\tilde{m}_{f,(m_i)\in I}$ or $\tilde{c}_{f,(c_i)\in I}$ is not constant; a strong forgery of type 1.
- $\tilde{m}_{f,(m_i)\in I}$ is constantly $\tilde{m}$ and $\tilde{c}_{f,(c_i)\in I}$ is constantly $\tilde{c}$ but $(\tilde{m}, \tilde{c}) \neq (\hat{m}, \hat{c})$; a strong forgery of type 2.

where $I$ is the set of indices of used labels in $\tau_1, \cdots, \tau_l$.

Now, we can define four security notions, UF-CPA, SUF-CCA, UF-CPA and SUF-CCA for authenticity of HAE.
Unforgeability under Chosen Plaintext Attack (UF-CPA)

**UF-CPA\_A(1^\lambda):**

**Initialization**
Given a security parameter \( \lambda \), generate a pair \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\). Give \( pp \) to \( A \).

**Queries**
\( A \) can make encryption queries adaptively. For each encryption query \((\tau, m)\) of \( A \), give \( \text{Enc}(sk, \tau, m) \) to \( A \) as an answer for the query.

**Finalization**
For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) of \( A \), if it is indeed a forgery, then output 1. Otherwise, output 0.

The advantage of \( A \) in the game UF-CPA is defined as

\[
\text{Adv}\_A^{\text{UF-CPA}}(1^\lambda) := \Pr\left[\text{UF-CPA}_A(1^\lambda) = 1\right]
\]

We say that an HAE satisfies UF-CPA, if \( \text{Adv}\_A^{\text{UF-CPA}}(1^\lambda) \) is negligible for any PPT adversary \( A \).

**Strong Unforgeability under Chosen Plaintext Attack (SUF-CPA)**

**SUF-CPA\_A(1^\lambda):**

**Initialization**
Given a security parameter \( \lambda \), generate a pair \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\). Give \( pp \) to \( A \).

**Queries**
\( A \) can make encryption queries adaptively. For each encryption query \((\tau, m)\) of \( A \), give \( \text{Enc}(sk, \tau, m) \) to \( A \) as an answer for the query.

**Finalization**
For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) of \( A \), if it is indeed a strong forgery, then output 1. Otherwise, output 0.

The advantage of \( A \) in the game SUF-CPA is defined as

\[
\text{Adv}\_A^{\text{SUF-CPA}}(1^\lambda) := \Pr\left[\text{SUF-CPA}_A(1^\lambda) = 1\right]
\]

We say that an HAE satisfies SUF-CPA, if \( \text{Adv}\_A^{\text{SUF-CPA}}(1^\lambda) \) is negligible for any PPT adversary \( A \).
Unforgeability under Chosen Ciphertext Attack (UF-CCA)

UF-CCA\(_A(1^\lambda)\):

 Initialization
 Given a security parameter \( \lambda \), generate a pair \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\). Give pp to \(A\).

 Queries
 \(A\) can make encryption queries and decryption queries adaptively. For each encryption query \((\tau, m)\) of \(A\), give \(\text{Enc}(sk, \tau, m)\) to \(A\) as an answer for the query. For each decryption query \(((f, \tau_1, \cdots, \tau_l), c)\) of \(A\), give \(\text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), c)\) to \(A\) as an answer for the query.

 Finalization
 For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) of \(A\), if it is indeed a forgery, then output 1. Otherwise, output 0.

 The advantage of \(A\) in the game UF-CCA is defined as

\[
\text{Adv}_{UF-CCA}^A(1^\lambda) := \Pr[\text{UF-CCA}_A(1^\lambda) = 1]
\]

 We say that an HAE satisfies UF-CCA, if \(\text{Adv}_{UF-CCA}^A(1^\lambda)\) is negligible for any PPT adversary \(A\).

Strong Unforgeability under Chosen Ciphertext Attack (SUF-CCA)

SUF-CCA\(_A(1^\lambda)\):

 Initialization
 Given a security parameter \( \lambda \), generate a pair \((pp, sk) \leftarrow \text{Gen}(1^\lambda)\). Give pp to \(A\).

 Queries
 \(A\) can make encryption queries and decryption queries adaptively. For each encryption query \((\tau, m)\) of \(A\), give \(\text{Enc}(sk, \tau, m)\) to \(A\) as an answer for the query. For each decryption query \(((f, \tau_1, \cdots, \tau_l), c)\) of \(A\), give \(\text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), c)\) to \(A\) as an answer for the query.

 Finalization
 For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) of \(A\), if it is indeed a strong forgery, then output 1. Otherwise, output 0.

 The advantage of \(A\) in the game SUF-CCA is defined as

\[
\text{Adv}_{SUF-CCA}^A(1^\lambda) := \Pr[\text{SUF-CCA}_A(1^\lambda) = 1]
\]

 We say that an HAE satisfies SUF-CCA, if \(\text{Adv}_{SUF-CCA}^A(1^\lambda)\) is negligible for any PPT adversary \(A\).

37
5.3 Relations on security notions

In this section, we investigate relations between the six security notions defined in the previous section. First, we have trivial implications from CCA security to CPA security.

**Theorem 9.** UF-CCA implies UF-CPA, SUF-CCA implies SUF-CPA, and IND-CCA implies IND-CPA.

**Proof.** Trivial. □

The following theorem says that the strong unforgeability implies unforgeability.

**Theorem 10.** SUF-CCA implies UF-CCA. And SUF-CPA implies UF-CPA.

**Proof.** It is enough to show that a forgery is also a strong forgery. Let \((f, \tau_1, \cdots, \tau_l, \hat{c})\) be a forgery. If it is a forgery of type 1, then \(\tilde{f} = f(m_i)_{i \in I}\) is not constant. That is, there exist two tuples \((m_1^i)_{j \in I}\) and \((m_2^i)_{j \in I}\) such that \(\tilde{f}(m_1^i)_{j \in I} \neq \tilde{f}(m_2^i)_{j \in I}\). Then there exist two distinct tuples \((c_1^i)_{j \in I}\) and \((c_2^i)_{j \in I}\) such that \(m_1^i = \text{Dec}(sk, I_{\tau_j}, c_1^i)\) and \(m_2^i = \text{Dec}(sk, I_{\tau_j}, c_2^i)\) for each \(j \notin I\). Then we have \(\hat{c}(c_1^i)_{j \notin I} \neq \hat{c}(c_2^i)_{j \notin I}\) by the correctness property, which shows that \(\hat{c} = \text{Eval}(ek, f, (c_i)_{i \in I})\) is nonconstant. So it is a strong forgery of type 1.

If it is a forgery of type 2 but not a strong forgery of type 1, then both \(\tilde{f}\) and \(\hat{c}\) are constants and \(\tilde{f} \neq \text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})\). But \(\tilde{f} = \text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})\), again by the correctness. This means that \(\hat{c} \neq \hat{c}\), and this shows that it is a strong forgery of type 2. □

Bellare et al. [1] showed that, in case of a MAC, strong unforgeability implies strong unforgeability even when the adversary has access to the verification oracle, and in case of an AE, integrity of ciphertexts implies integrity of ciphertexts even when the adversary has access to the verification oracle. The following can be considered as a homomorphic analogue to the result.

**Theorem 11.** SUF-CPA implies SUF-CCA.

**Proof.** We prove the theorem by a hybrid argument to transform the game SUF-CCA into another game that is essentially the same as the game SUF-CPA.

Let \(A\) be any PPT adversary. Without loss of generality, we may assume that \(A\) makes exactly \(q = q(\lambda)\) decryption queries.

For each \(i \in \{0, \ldots, q\}\), define SUF-CCA\(^i\) to be the game that is identical to SUF-CCA except that the first \(i\) decryption queries are answered by the following decryption simulation.

**Decryption Simulation.** For a decryption query \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) made by the adversary \(A\), let \(I \leftarrow \emptyset\) and do the following for \(i = 1, \cdots, l\): If \((\tau_i, m, c) \in S\) for some \(m \in \mathcal{M}\) and \(c \in \mathcal{C}\), then \(I \leftarrow I \cup \{i\}\) and \(m_i = m, c_i = c\). And then let \(\hat{c} = \text{Eval}(ek, f, (c_i)_{i \in I})\) and \(\tilde{f} = f(m_i)_{i \in I}\). If \(\hat{c}\) is constant and \(\hat{c} = \hat{c}\), then return \(\tilde{f}\). Otherwise, return \(\perp\).
In particular, SUF-CCA is equal to SUF-CCA\(^0\). So,
\[
\text{Adv}_{\Pi,A}^{\text{SUF-CCA}}(\lambda) = \text{Adv}_{\Pi,A}^{\text{SUF-CCA}}(\lambda).
\]

Moreover, since the decryption simulation does not use any secret information and is efficiently computable by the CCT property in SUF-CCA\(^0\), the adversary \(A\) does not obtain any useful information by the decryption queries at all in this game. Formally, we can easily construct an adversary \(A'\) which plays SUF-CPA game and makes the same number of encryption queries as \(A\) does, and satisfying
\[
\text{Adv}_{\Pi,A}^{\text{SUF-CCA}}(\lambda) = \text{Adv}_{\Pi,A}^{\text{SUF-CPA}}(\lambda).
\]

For each \(i \in \{1, \ldots, q\}\), the difference between \(\text{Adv}_{\Pi,A}^{\text{SUF-CCA}^i}(\lambda)\) and \(\text{Adv}_{\Pi,A}^{\text{SUF-CCA}^{i-1}}(\lambda)\) is bounded by the probability that the decryption simulation on the \(i\)th decryption query made by \(A\) fails (that is, is different from the real decryption). From the definition of a strong forgery, it is easy to check that the decryption simulation fails if and only if the decryption query made by \(A\) is a strong forgery: we have

\[
decryption \text{ simulation fails} 
\iff \tilde{c} \text{ is constant and } \tilde{c} = \hat{c}, \text{ but } \bot = \text{Dec}(sk, (f, \tau_1, \ldots, \tau_l), \hat{c}), \text{ or,}
\]
\[
\hat{c} \text{ is nonconstant, or } \tilde{c} \text{ is constant but } \hat{c} \neq \tilde{c}, \text{ but } \bot \neq \text{Dec}(sk, (f, \tau_1, \ldots, \tau_l), \hat{c}),
\]

but when \(\tilde{c}\) is constant and \(\tilde{c} = \hat{c}\), by the correctness we should have \(\text{Dec}(sk, (f, \tau_1, \ldots, \tau_l), \hat{c}) = \tilde{f}\), which should also be a constant not equal to \(\bot\), therefore this subcase cannot happen. So,

\[
decryption \text{ simulation fails} 
\iff \tilde{c} \text{ is nonconstant, or } \tilde{c} \text{ is constant but } \hat{c} \neq \tilde{c}, \text{ but } \bot \neq \text{Dec}(sk, (f, \tau_1, \ldots, \tau_l), \hat{c})
\iff ((f, \tau_1, \ldots, \tau_l), \hat{c}) \text{ is a strong forgery}.
\]

Hence, we may construct a PPT adversary \(A''\) for the game SUF-CPA using the \(i\)th decryption query made by \(A\). Specifically, \(A''\) runs the adversary \(A\) until it makes the \(i\)th decryption query, while answering the encryption queries using its own encryption queries and answering the previous decryption queries by the decryption simulation. Then \(A''\) aborts the running of \(A\), and outputs the \(i\)th decryption query of \(A\) as its own forgery attempt.

So,
\[
\left| \text{Adv}_{\Pi,A}^{\text{SUF-CCA}^i}(\lambda) - \text{Adv}_{\Pi,A}^{\text{SUF-CCA}^{i-1}}(\lambda) \right| \leq \text{Adv}_{\Pi,A''}^{\text{SUF-CPA}}(\lambda) = \text{negl}(\lambda).
\]

Therefore,
\[
\text{Adv}_{\Pi,A}^{\text{SUF-CCA}}(\lambda) \leq \text{Adv}_{\Pi,A''}^{\text{SUF-CPA}}(\lambda) + \left| \text{Adv}_{\Pi,A}^{\text{SUF-CCA}}(\lambda) - \text{Adv}_{\Pi,A''}^{\text{SUF-CPA}}(\lambda) \right|
\leq \text{negl}(\lambda) + \sum_{i=1}^{q} \left| \text{Adv}_{\Pi,A}^{\text{SUF-CCA}^i}(\lambda) - \text{Adv}_{\Pi,A}^{\text{SUF-CCA}^{i-1}}(\lambda) \right|
\leq \text{negl}(\lambda) + q \cdot \text{negl}(\lambda)
= \text{negl}(\lambda).
\]
So $\text{Adv}^{\text{SU}_\Pi\text{-CCA}}(\lambda)$ is also negligible for any PPT adversary $A$ and therefore $\Pi$ is SUF-CCA. \hfill \Box

**Theorem 12.** IND-CPA and SUF-CPA together imply IND-CCA.

**Proof.** Proof of this theorem is similar to that of Theorem 11; again we prove this theorem by a hybrid argument to transform the game IND-CCA into another game that is essentially same as the game IND-CPA.

Let $A$ be a PPT adversary engaging in the game IND-CCA. Again, without loss of generality, we assume that $A$ makes exactly $q_b = q_b(\lambda)$ decryption queries before the Challenge phase, and $q_a = q_a(\lambda)$ decryption queries after the Challenge phase.

For each $i \in \{0, \ldots, q_b\}$, define IND-CCA$^{b,i}$ to be the game that is equal to IND-CCA except that the first $i$ decryption queries before the Challenge phase are answered by the same decryption simulation as shown in Theorem 11.

For each $i \in \{0, \ldots, q_a\}$, define IND-CCA$^{a,i}$ to be the game that is equal to IND-CCA except that all decryption queries before the Challenge phase, and the first $i$ decryption queries after the Challenge phase are answered by the same decryption simulation.

By definition, IND-CCA$^{b,0} = \text{IND-CCA}$ and IND-CCA$^{b,q_b} = \text{IND-CCA}^{a,0}$. So,

$$\text{Adv}_{\Pi,A}^{\text{IND-CCA}^{b,0}}(\lambda) = \text{Adv}_{\Pi,A}^{\text{IND-CCA}}(\lambda),$$

$$\text{Adv}_{\Pi,A}^{\text{IND-CCA}^{b,q_b}}(\lambda) = \text{Adv}_{\Pi,A}^{\text{IND-CCA}^{a,0}}(\lambda).$$

Now, we construct a PPT adversary $A'$ for the security game IND-CPA using the adversary $A$. The adversary $A'$ simulates the game IND-CCA$^{b,q_b}$ for the adversary $A$ as follows:

**The adversary $A'(1^\lambda)$:**

**Initialization.** The evaluation key $ek$ is generated and given to $A'$. Then $A'$ initializes $S \leftarrow \emptyset$, and gives $ek$ to the adversary $A$.

**Queries.** When $A$ makes an encryption query $(\tau, m)$, if $(\tau, \cdot, \cdot) \notin S$ then $A'$ makes the same encryption query, receives the ciphertext $c \leftarrow \text{Enc}(sk, \tau, m)$, replies $A$ with the answer $c$, and updates $S$ by $S \leftarrow S \cup \{(\tau, m, c)\}$. Otherwise, the encryption query is rejected.

When $A$ makes a decryption query $((f, \tau_0, \cdots, \tau_l), \hat{c})$, it is answered by the decryption simulation as in Theorem 11.

**Challenge.** $A$ outputs the challenge tuple $(\tau^*, m_0^*, m_1^*)$. If $(\tau^*, \cdot, \cdot) \notin S$, then $A'$ outputs the same challenge tuple, and receives the challenge ciphertext $c^* \leftarrow \text{Enc}(sk, \tau^*, m_0^*)$. $A'$ gives the challenge ciphertext $c^*$ to $A$, and updates $S$ by $S \leftarrow S \cup \{(\tau^*, m_0^*, c^*)\}$. Otherwise, the challenge is rejected.

**Queries After Challenge.** Any encryption query, or any decryption query of $A$ is answered precisely as before.
Finalization. When $A$ outputs a bit $b'$, the adversary $A'$ outputs the same bit $b'$.

In the first Queries phase, the simulation of $A'$ for the game $\text{IND-CCA}^{a,q_0}$ is perfect. But, in the Challenge phase, the history $S$ is updated by $S \leftarrow S \cup \{ (\tau^*, m_0^*, c^*) \}$ because $A'$ does not know the coin $b$, while in the actual game $\text{IND-CCA}^{a,q_0}$, $S$ is updated by $S \leftarrow S \cup \{ (\tau^*, m_0^*, c^*) \}$. We need to show that, despite this the simulation of $A'$ for the game $\text{IND-CCA}^{a,q_0}$ in the ‘Queries After Challenge’ phase is correct.

We see that the decryption simulation might potentially be incorrect only when $b = 1$ and $\tau^* \in I$. So, suppose that a decryption query of $A$ is $((f, \tau_0, \cdots, \tau_l), \hat{c})$ when $b = 1$ and $\tau^* \in I$. Let $\tau^* = \tau_{i^*}$ for some $i^* \in \{1, \ldots, l\}$. Let us compare how this query is answered in the game $\text{IND-CCA}^{a,q_0}$ and in the simulation of $A'$.

In the game $\text{IND-CCA}^{a,q_0}$, $m_i^*$ is encrypted under the label $\tau^*$ to produce the ciphertext $c^*$. So, in the game $\text{IND-CCA}^{a,q_0}$, $\hat{c} = \text{Eval}(ek, f, (c_i)_{i \in I})$ and $\hat{f} = f(m_i)_{i \in I}$ are computed, and the decryption query is answered with $\hat{f}$ if and only if $\hat{c}$ is constant and equal to $\hat{c} \in C$. And in the computation of $\hat{f}$, $m_i^*$ is used for the $i^*$th plaintext input. To emphasize this fact, let us denote this $\hat{f}$ as $\tilde{f}_i$, meaning that $m_i^*$ was used to produce this plaintext.

In the simulation of $A'$, still $m_i^*$ is encrypted under the label $\tau^*$ to produce the ciphertext $c^*$, and $\hat{c} = \text{Eval}(ek, f, (c_i)_{i \in I})$ and $\tilde{f} = f(m_i)_{i \in I}$ are computed, and the decryption query is answered with $\tilde{f}$ if and only if $\hat{c}$ is constant equal to $\hat{c} \in C$. But, this $\tilde{f}$ is computed using $m_0^*$ as the $i^*$th plaintext input. So let us denote this $\tilde{f}$ as $\tilde{f}_0$.

Therefore, in both scenarios, the decryption query is answered by $\bot$ if and only if $\hat{c}$ is nonconstant, or $\hat{c}$ is constant but not equal to $\hat{c}$. Meanwhile, when $\hat{c}$ is constantly equal to $\hat{c}$, then the game $\text{IND-CCA}^{a,q_0}$ will output $\tilde{f}_i$, but the simulation of $A'$ will output $\tilde{f}_0$. Despite this, recall that any decryption query made by $A$ after the Challenge phase is legal by the definition of $\text{IND-CCA}$. Hence, we have $\tilde{f}_0 = \tilde{f}_1$. This shows that $A'$ correctly simulates the game $\text{IND-CCA}^{a,q_0}$, and we conclude that

$$\text{Adv}_{\Pi, A}^{\text{IND-CCA}^{a,q_0}}(\lambda) = \text{Adv}_{\Pi, A'}^{\text{IND-CCA}^{a,q_0}}(\lambda).$$

Now consider the difference of each consecutive two games. Again, for each $i \in \{1, \ldots, q_b\}$, the difference between $\text{Adv}_{\Pi, A}^{\text{IND-CCA}^{b_{\mu-1}}}(\lambda)$ and $\text{Adv}_{\Pi, A}^{\text{SUF-CCA}^{b_{\mu-1}}}(\lambda)$ is not greater than the probability that the decryption simulation on the $i$th decryption query made by $A$ before Challenge fails, and we may use this to construct an adversary $A''$ for the game $\text{SUF-CPA}$ just like in Theorem 11. Note that $A''$ aborts the running of $A$ before it has any chance to output the challenge tuple.

So,

$$\left| \text{Adv}_{\Pi, A}^{\text{IND-CCA}^{b_{\mu-1}}}(\lambda) - \text{Adv}_{\Pi, A}^{\text{IND-CCA}^{b_{\mu-1}}}(\lambda) \right| \leq \text{Adv}_{\Pi, A''}^{\text{SUF-CPA}}(\lambda) = \negl(\lambda).$$

Similarly, for each $i \in \{1, \ldots, q_a\}$, the difference between $\text{Adv}_{\Pi, A}^{\text{IND-CCA}^{a_{\mu-1}}}(\lambda)$ and $\text{Adv}_{\Pi, A}^{\text{SUF-CCA}^{a_{\mu-1}}}(\lambda)$ is not greater than the probability that the decryption simulation on the $i$th decryption query made by $A$ after Challenge fails, and we may use this to construct an adversary $A''$ for the
game SUF-CPA just like in Theorem 11. In this case, the challenge tuple output of $A$ is handled by $A''$. $A''$ flips the coin $b \leftarrow \{0, 1\}$, and obtains the challenge ciphertext via its encryption query $(\tau^*, m^*_b)$. Since $A''$ knows the coin $b$, the correct encryption history is maintained: $(\tau^*, m^*_b, c^*) \in S$.

So,

$$\left| \text{Adv}^{\text{IND-CCA}, i-1}_{\Pi, A} (\lambda) - \text{Adv}^{\text{IND-CCA}, i}_{\Pi, A} (\lambda) \right| \leq \text{Adv}^{\text{SUF-CPA}}_{\Pi, A''} (\lambda) = \text{negl}(\lambda).$$

Hence,

$$\text{Adv}^{\text{IND-CCA}}_{\Pi, A} (\lambda) \leq \text{Adv}^{\text{IND-CPA}}_{\Pi, A'} (\lambda) + \left| \text{Adv}^{\text{IND-CCA}}_{\Pi, A} (\lambda) - \text{Adv}^{\text{IND-CPA}}_{\Pi, A'} (\lambda) \right|$$

$$= \text{negl}(\lambda) + \left| \text{Adv}^{\text{IND-CCA}}_{\Pi, A} (\lambda) - \text{Adv}^{\text{IND-CCA}}_{\Pi, A''} (\lambda) \right|$$

$$\leq \text{negl}(\lambda) + \sum_{i=1}^{q_b} \left| \text{Adv}^{\text{IND-CCA}}_{\Pi, A} (\lambda) - \text{Adv}^{\text{IND-CCA}}_{\Pi, A} (\lambda) \right|$$

$$+ \sum_{i=1}^{q_a} \left| \text{Adv}^{\text{IND-CCA}}_{\Pi, A} (\lambda) - \text{Adv}^{\text{IND-CCA}}_{\Pi, A} (\lambda) \right|$$

$$\leq \text{negl}(\lambda) + (q_b + q_a) \cdot \text{negl}(\lambda) = \text{negl}(\lambda).$$

So, $\text{Adv}^{\text{IND-CCA}}_{\Pi, A} (\lambda)$ is also negligible for any PPT adversary $A$. Therefore $\Pi$ is IND-CCA.

\[\square\]

In conclusion, we see that IND-CPA and SUF-CPA together imply the strongest security notions, IND-CCA and SUF-CCA. When we discuss our construction in Section 5.5, we show that our scheme is IND-CPA and SUF-CPA.

### 5.4 Generic transformation to CCT

Suppose that $\Pi$ is a HAE which is not necessarily ciphertext constant testable. We describe a generic construction that transforms a HAE $\Pi$ into another HAE $\Pi'$ satisfying CCT while preserving IND-CPA or SUF-CPA of the original scheme $\Pi$. Our construction is based on the Merkle hash tree technique used by Gennaro and Wichs [14]. For concreteness, here we assume that $\Pi$ represents admissible functions as circuits. In such a case, we also assume that Eval algorithm works by evaluating ciphertexts gate by gate; the evaluation of a circuit becomes a circuit of ciphertexts.

Now, using a pseudorandom function $F$ and a family $\mathcal{H}$ of collision-resistant hash functions, we can transform a HAE $\Pi$ to another HAE $\Pi'$ as follows.

**Scheme $\Pi'$** = (Gen', Enc', Eval', Dec'):

- $(ek', sk') \leftarrow \text{Gen}'(1^\lambda)$: Generate keys $(ek, sk) \leftarrow \text{Gen}(1^\lambda)$, $k \leftarrow \{0, 1\}^\lambda$ and $H \leftarrow \mathcal{H}$. Return $ek' = (ek, H)$ and $sk' = (sk, k)$.
- $c' \leftarrow \text{Enc}'(sk', \tau, m)$: Let $h = H(F_k(\tau))$ and $c \leftarrow \text{Enc}(sk, \tau, m)$. Return $(h, c)$.
\begin{itemize}
  \item $\tilde{c}' \leftarrow \text{Eval}'(ek', f, c_1', \ldots, c_i')$: Let $f : \mathcal{M}^I \rightarrow \mathcal{M}$ be a circuit. For each $i = 1, \ldots, l$, parse $c_i' = (h_i, c_i)$. Let $\tilde{h} = f^H(h_1, \ldots, h_i)$ and $\tilde{c} \leftarrow \text{Eval}(ek, f, c_1, \ldots, c_i)$. Return $(\tilde{h}, \tilde{c})$.
  \item $m \leftarrow \text{Dec}'(sk', (f, r_1, \ldots, r_i), \tilde{c}')$: Let $f : \mathcal{M}^I \rightarrow \mathcal{M}$ be a circuit. Parse $\tilde{c}' = (\tilde{h}, \tilde{c})$. For each $i = 1, \ldots, l$, let $h_i = H(F(k(r_i)))$. If $\tilde{h} = f^H(h_1, \ldots, h_i)$, then return $\text{Dec}(sk, (f, r_1, \ldots, r_i), \tilde{c})$. Otherwise, return $\perp$.
\end{itemize}

Above, we assume that $F_k : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ and $H : \{0,1\}^\ast \rightarrow \{0,1\}^\lambda$.

$\Pi'$ and $\Pi$ share the same message space $\mathcal{M}$ and the label space $\mathcal{L}$. The ciphertext space of $\Pi'$ is $\{0,1\}^\lambda \times \mathcal{C}$, where $\mathcal{C}$ is the ciphertext space of $\Pi$.

And the correctness of $\Pi'$ can easily be concluded from that of $\Pi$. Below, we show that the constructed scheme $\Pi'$ satisfies the CCT property, and also this generic transformation preserves both SUF-CPA and IND-CPA.

**Theorem 13.** The HAE scheme $\Pi'$ satisfies CCT.

**Proof.** As in p. 31, let $f : \mathcal{M}^I \rightarrow \mathcal{M}$ be an admissible function, $I$ a subset of the index set $\{1, \ldots, l\}$, and $(m_i)_{i \in I} \in \mathcal{M}^{\lvert I\rvert}$ and $(c'_i = (h_i, c_i))_{i \in I} \in (\{0,1\}^\lambda \times \mathcal{C})^{\lvert I\rvert}$ some plaintexts and their corresponding ciphertexts.

Consider $\tilde{c}' : (\{0,1\}^\lambda \times \mathcal{C})^{\lvert I\rvert} \rightarrow \{0,1\}^\lambda \times \mathcal{C}$ defined as

$$\tilde{c}' := \text{Eval}(ek', f, (c'_i)_{i \in I}) = (f^H(h_i)_{i \in I}, \text{Eval}(ek, f, (c_i)_{i \in I})).$$

The above expression $\tilde{c}'$ is a function of the values for the ‘missing’ indices: $(h_i)_{i \not\in I}$ and $(c_i)_{i \not\in I}$.

If $I = \{1, \ldots, l\}$, then $\tilde{c}'$ is clearly constant. Now, suppose that $I \neq \{1, \ldots, l\}$. Consider the case when the $i$th input of $f$ is unused for all indices $i \not\in I$. In that case, again clearly $\tilde{c}'$ is constant.

Finally, consider the case that the $i$th input of $f$ is actually used for at least one index $i \in I$. Since the underlying hash function $H$ is collision-resistant, the hash tree $f^H(h_1, \ldots, h_i)$ cannot be constant on the variable $h_i$ except with negligible probability. So, $\tilde{c}'$ is not constant except with negligible probability. This means that we can trivially determine if $\tilde{c}'$ is constant or not. Therefore, $\Pi'$ satisfies CCT.

**Theorem 14.** If $\Pi$ is IND-CPA, then $\Pi'$ is also IND-CPA.

**Proof.** We will just provide a sketch of the proof. Let $A'$ be any PPT adversary for the game IND-CPA$_{\Pi', A'}$. We construct a PPT adversary $A$ for the game IND-CPA$_{\Pi, A}$ that simulates the game IND-CPA$_{\Pi', A'}$ for the adversary $A'$.

Most of the simulation is trivial message-passing, but for the challenge $(\tau^*, m_0^*, m_1^*)$ made by $A'$, $A$ returns the challenge ciphertext $(H(r), c^*)$ to $A'$, where $r \leftarrow \{0,1\}^\lambda$ and $c^*$ is the challenge ciphertext given to $A$ in the security game IND-CPA$_{\Pi, A}$.

This simulation does work because $\tau^*$ is a new label and $F$ is a PRF. Moreover, $H(r)$ in the challenge ciphertext given to $A'$ does not contain any information that may help $A'$ to distinguish
between \( m_0^* \) and \( m_1^* \). So the advantage of \( A' \) entirely comes from \( c^* \). Therefore, the difference

of advantages of \( A \) and \( A' \) is negligible. Formally, we have

\[
\text{Adv}_{\Pi',A'}^\text{IND-CPA}(\lambda) \leq \text{Adv}_{\Pi,A}^\text{IND-CPA}(\lambda) + \text{Adv}_{F,A'}^\text{PRF}(\lambda).
\]

\( \square \)

**Theorem 15.** If \( \Pi \) is SUF-CPA, then \( \Pi' \) is also SUF-CPA.

**Proof.** Let \( A' \) be a PPT adversary engaged in the security game SUF-CPA\(_{\Pi',A'}\). Using \( A' \), we construct a PPT adversary \( A \) for the security game SUF-CPA\(_{\Pi,A}\) which simulates the game SUF-CPA\(_{\Pi',A'}\) for the adversary \( A' \).

**Adversary** \( A(1^\lambda) \):

**Initialization.** A set \( S \) is initialized to be the empty set \( \emptyset \). Receiving the evaluation key \( ek \), the adversary \( A \) picks \( H \leftarrow \mathcal{H} \), \( k \leftarrow \{0,1\}^\lambda \), and gives the evaluation key \( ek' := (ek, H) \) to \( A' \), and keeps the PRF key \( k \) by himself.

**Queries.** Whenever \( A' \) makes an encryption query \((\tau,m)\), if \( (\tau,\cdot) \notin S \), then \( A \) makes the same encryption query. Receiving the answer \( c \leftarrow \text{Enc}(sk, \tau, m) \), the adversary \( A \) computes \( h := H(F_k(\tau)) \), answers the encryption query of \( A' \) by \( c' := (h,c) \), and updates \( S \) by \( S \leftarrow S \cup \{(\tau,m,c')\} \). Otherwise, the query is rejected.

**Forgery.** \( A' \) outputs a forgery attempt \(((f,\tau_1,\cdots,\tau_l),\hat{c}')\). Parse \( \hat{c}' = (\hat{h},\hat{c}) \). Then the adversary \( A \) outputs \(((f,\tau_1,\cdots,\tau_l),\hat{c})\).

Now, let us show that if the forgery attempt \(((f,\tau_1,\cdots,\tau_l),\hat{c}' = (\hat{h},\hat{c}))\) of \( A' \) is a strong forgery for SUF-CPA\(_{\Pi',A'}\), then \(((f,\tau_1,\cdots,\tau_l),\hat{c})\) is also a strong forgery for SUF-CPA\(_{\Pi,A}\), except with negligible probability. Let \( I \) be the set of indices \( i \in \{1,\ldots,l\} \) such that \((\tau_i,m_i,c_i) \in S \).

Since \(((f,\tau_1,\cdots,\tau_l),\hat{c}')\) is valid in \( \Pi' \), \(((f,\tau_1,\cdots,\tau_l),\hat{c})\) is also valid in \( \Pi \). Also, we have

\[
f^H(H(F_k(\tau_1)),\cdots,H(F_k(\tau_l))) = \hat{h}.
\]

Suppose \( I \neq \{1,\ldots,l\} \), and assume there exists at least an index \( j \notin I \) which is used in the circuit \( f \). In this case, we claim that \( f^H(H(F_k(\tau_1)),\cdots,H(F_k(\tau_l))) = \hat{h} \) only with negligible probability. Since \( j \) is new, \( F_k(\tau_j) \) is computationally indistinguishable to a random number \( r_j \leftarrow \{0,1\}^\lambda \). Due to the collision resistance of \( H \), the probability

\[
\Pr\left[f^H(H(F_k(\tau_1)),\cdots,r_j,\cdots,H(F_k(\tau_l))) = \hat{h} \mid r_j \leftarrow \{0,1\}^\lambda\right]
\]

should be negligible. So, \( f^H(H(F_k(\tau_1)),\cdots,H(F_k(\tau_l))) = \hat{h} \) holds only with negligible probability. Therefore, since we already have \( f^H(H(F_k(\tau_1)),\cdots,H(F_k(\tau_l))) = \hat{h} \), we may assume with negligible exception that any \( i \notin I \) is unused in the circuit \( f \). In this case, both \( \hat{c}' \) and \( \hat{c} \) are constants, where

\[
\hat{c}' = (\hat{h},\hat{c}),
\]
Hence $\Pi$ for each $k$ values of plaintext inputs in $Z$ discussions on the correctness property.

For a polynomial circuits, that is, circuits consisting of $+$ gates and $\times$ gates. Such a circuit $f$ of arity $l$ determines a polynomial $f : Z^l \rightarrow Z$ with integral coefficients. We use such a circuit to compute function values of plaintext inputs in $Z_Q$, and also to homomorphically evaluate ciphertexts in $Z_{q_0}$. The precise description of the admissible function space will be given in the next, together with discussions on the correctness property.

5.5 Construction

In this section, we describe our HAE scheme $\Pi$ given in [20] and show that it satisfies correctness and CCT. All of the parameters $\rho, \eta, \gamma, d$ of the scheme are polynomials in $\lambda$. The specific choices of these parameters are given after the description of the scheme.

We use a pseudorandom function $F$ in our construction. We assume that $F_k : \{0, 1\}^\lambda \rightarrow Z_{q_0}$ for each $k \in \{0, 1\}^\lambda$. The message space and the ciphertext space of our scheme is $Z_Q$ and $Z_{q_0}$, respectively, and the label space is $\{0, 1\}^\lambda$. To represent admissible functions we use arithmetic circuits, that is, circuits consisting of $+$ gates and $\times$ gates. Such a circuit $f$ of arity $l$ determines a polynomial $f : Z^l \rightarrow Z$ with integral coefficients. We use such a circuit to compute function values of plaintext inputs in $Z_Q$, and also to homomorphically evaluate ciphertexts in $Z_{q_0}$. The precise description of the admissible function space will be given in the next, together with discussions on the correctness property.

SCHEME. $\Pi = (Gen, Enc, Eval, Dec)$

- $(ek, sk) \leftarrow Gen(1^\lambda, Q)$: Given security parameter $\lambda$ and any modulus $Q \in [2, 2^\lambda]$, choose $p \leftarrow [2^{\lambda - 1}, 2^\lambda) \cap \text{PRIME}$ and $q_0 \leftarrow [0, \frac{2^\lambda}{p}) \cap \text{ROUGH}(2^\lambda)$. Let $y_0 = pq_0$. Choose a PRF key $k \leftarrow \{0, 1\}^\lambda$. Return $(ek, sk)$, where $ek = (Q, y_0)$, and $sk = (p, q_0, Q, k)$.
- $c \leftarrow Enc(sk, \tau, m)$: Given the secret key $sk$, a label $\tau \in \{0, 1\}^\lambda$ and a plaintext $m \in Z_Q$, choose $r \leftarrow (-2^p, 2^p)$. Let $a = rQ + m$ and $b = F_k(\tau)$. Return $c = CRT(p, q_0)(a, b)$.
- $\tilde{c} \leftarrow Eval(ek, f, c_1, \ldots, c_l)$: Given the evaluation key $ek$, an arithmetic circuit $f$ of arity $l$ and ciphertexts $c_1, \ldots, c_l$, return $f(c_1, \ldots, c_l) \mod y_0$
- $m \leftarrow Dec(sk, (f, \tau_1, \ldots, \tau_l), \tilde{c})$: For $i = 1$ to $l$, compute $b_i \leftarrow F_k(\tau_i)$ and $b = f(b_1, \ldots, b_l) \mod q_0$. Return $m = (\tilde{c} \mod p) \mod Q$, if $b = \tilde{c} \mod q_0$. Otherwise, return $\bot$.  

$\square$
Correctness

To show the correctness of the scheme, let \((ek, sk) \leftarrow \text{Gen}(1^\lambda, Q)\) for any \(\lambda \in \mathbb{Z}^+\) and any modulus \(Q \in [2, 2^\lambda]\). Let \(c_i \leftarrow \text{Enc}(sk, \tau_i, m_i)\) for each \(i = 1, \cdots, l\). And \(\tilde{c} \leftarrow \text{Eval}(ek, f, c_1, \cdots, c_l)\) for any arithmetic circuit \(f\) of arity \(l\). We identify an arithmetic circuit \(f\) of arity \(l\) with the \(l\)-variate integral polynomial determined by \(f\). Let \(d := \deg(f)\), then

\[
\tilde{c} \mod p = (f(c_1, \cdots, c_l) \mod y_0) \mod p = f(c_1, \cdots, c_l) \mod p = f(c_1 \mod p, \cdots, c_l \mod p) \mod p = f(r_1Q + m_1, \cdots, r_lQ + m_l) \mod p = f(r_1Q + m_1, \cdots, r_lQ + m_l)
\]

The last equality in the above equations holds if

\[
|f(r_1Q + m_1, \cdots, r_lQ + m_l)| \leq \frac{p}{2}.
\]

And so, in this case,

\[
(\tilde{c} \mod p) \mod Q = f(r_1Q + m_1, \cdots, r_lQ + m_l) \mod Q = f(m_1, \cdots, m_l) \mod Q
\]

Since \(|f(r_1Q + m_1, \cdots, r_lQ + m_l)| \leq \|f\|_1 \cdot 2^{d(\rho + \lambda)}\) and \(2^{\eta - 2} \leq p/2\), the correctness is guaranteed if

\[
\|f\|_1 \cdot 2^{d(\rho + \lambda)} \leq 2^{\eta - 2},
\]

equally,

\[
d \leq \frac{\eta - 2 - \log \|f\|_1}{\rho + \lambda}.
\]

If \(\|f\|_1 \leq 2^d\), we have

\[
d \leq \frac{\eta - 2}{\rho + \lambda + 1}.
\]

Let \(\tilde{d} = \left\lfloor \frac{\eta - 2}{\rho + \lambda + 1} \right\rfloor\). Then an admissible function in our scheme is an arithmetic circuit \(f\) such that \(\deg f \leq \tilde{d}\) and \(\|f\|_1 \leq 2^d\) as a polynomial over \(\mathbb{Z}_Q\).

Parameter selection.

In the scheme, the parameters \(\rho, \eta, \gamma\) are given as follows.

- \(\rho = \omega(\lg \lambda)\) to resist the brute force attack on the EF-AGCD problem.

- \(\eta \geq \tilde{d}(\rho + \lambda + 1) + 2\) for the upper bound \(\tilde{d}\) on degrees of admissible functions. This is a consequence of discussions about correctness property. If we choose \(\tilde{d} = \mathcal{O}(\lambda)\) and \(\rho = \mathcal{O}(\lambda)\), then \(\eta = \mathcal{O}(\lambda^2)\).
\begin{itemize}
    \item $\gamma = \eta^2 \omega(\lg \lambda)$ to resist known attacks on the EF-AGCD problem as explained in [25, 9].
\end{itemize}

If we choose $\eta = \mathcal{O}(\lambda^2)$, then $\gamma = \mathcal{O}(\lambda^5)$

**Theorem 16.** The scheme II satisfies CCT.

**Proof.** Let $ek$ be an evaluation key generated by Gen$(1^\lambda, Q)$ for some $\lambda \in \mathbb{Z}^+$ and a modulus $Q$, $f$ any admissible arity-$l$ arithmetic circuit for some $l \in \mathbb{Z}^+$ and $(c_i)_{i \in I}$ any element in $\mathbb{Z}_{y_0}^{|I|}$ for some subset $I$ of the index set $\{1, \cdots, l\}$. We construct an algorithm ALG-CCT that determines if $\tilde{c} = \text{Eval}(ek, f, (c_i)_{i \in I})$ is constant or not with overwhelming probability, as follows.

\begin{algorithm}
\textbf{procedure} ALG-CCT($ek, f, (c_i)_{i \in I}$):
\begin{algorithmic}
    \STATE \textbf{if} $I = \{1, \cdots, l\}$ \textbf{then}
    \STATE \hspace{1em} \textbf{return} 1
    \ELSE
    \STATE $(c_j^0)_{j \notin I}, (c_j^1)_{j \notin I} \overset{\$}{\leftarrow} (\mathbb{Z}_{y_0})^{l-|I|}$
    \STATE \textbf{if} $\tilde{c}(c_j^0)_{j \notin I} \equiv \tilde{c}(c_j^1)_{j \notin I} \text{ mod } y_0$ \textbf{then}
    \STATE \hspace{1em} \textbf{return} 1
    \ELSE
    \STATE \hspace{1em} \textbf{return} 0
    \ENDIF
\end{algorithmic}
\end{algorithm}

The algorithm ALG-CCT is essentially the usual probabilistic polynomial identity testing. In the scheme II, $\tilde{c}$ can be considered as an $(l - |I|)$-variate polynomial over $\mathbb{Z}_{y_0}$ of degree not greater than $\text{deg } f$. We have

$$\tilde{c} = f(c_i)_{i \in I} \text{ mod } y_0 : \mathbb{Z}_{y_0}^{|l-|I|} \rightarrow \mathbb{Z}_{y_0}$$

In case $I = \{1, \cdots, l\}$, $\tilde{c}$ is clearly constant and the algorithm outputs 1 correctly. In case $I \subseteq \{1, \cdots, l\}$, consider the function $\tilde{c}' := \tilde{c} - \tilde{c}(c_j^0)_{j \notin I} \text{ mod } y_0$ for any $(c_j^0)_{j \notin I} \in (\mathbb{Z}_{y_0})^{l-|I|}$. If $\tilde{c}$ is constant, then $\tilde{c}'$ is constantly zero and $\tilde{c}'(c_j^1)_{j \notin I} = \tilde{c}(c_j^1)_{j \notin I} - \tilde{c}(c_j^0)_{j \notin I} \equiv 0 \text{ mod } y_0$ for any $(c_j^1)_{j \notin I} \in (\mathbb{Z}_{y_0})^{l-|I|}$. So, $\tilde{c}(c_j^0)_{j \notin I} \equiv \tilde{c}(c_j^1)_{j \notin I} \text{ mod } y_0$ and the algorithm outputs 1 correctly. If $\tilde{c}$ is not constant, then $\tilde{c}'$ is not constantly zero and the algorithm outputs the incorrect answer 1 when $\tilde{c}(c_j^0)_{j \notin I} \equiv \tilde{c}(c_j^1)_{j \notin I} \text{ mod } y_0$, that is, $\tilde{c}'(c_j^1)_{j \notin I} \equiv 0 \text{ mod } y_0$. This is the only case that the algorithm outputs an incorrect answer. So the error probability of the algorithm is

$$\Pr \left[ \tilde{c}'(c_j^1)_{j \notin I} \equiv 0 \text{ mod } y_0 \mid (c_j^1)_{j \notin I} \overset{\$}{\leftarrow} (\mathbb{Z}_{y_0})^{l-|I|} \right],$$

when $\tilde{c}'$ is not constantly zero.

To find an upper bound on the error probability of the algorithm using Schwartz-Zippel lemma, we need the following fact. A $2^\lambda$-rough random integer $y_0$ is square-free with overwhelming probability and there exists some prime factor $p'$ of $y_0$ such that $\tilde{c}' \text{ mod } p'$ is not
constantly zero and \( p' \geq 2^\lambda \). From these, we have
\[
\Pr \left[ \hat{c} \left( c_j^1 \right)_{j \notin I} \equiv 0 \mod y_0 \mid (c_j^1)_{j \notin I} \xleftarrow{\$} (\mathbb{Z}_{y_0})^{\lvert I \rvert} \right] \\
\leq \Pr \left[ \hat{c} \left( c_j^1 \right)_{j \notin I} \equiv 0 \mod p' \mid (c_j^1)_{j \notin I} \xleftarrow{\$} (\mathbb{Z}_{y_0})^{\lvert I \rvert} \right] \\
= \Pr \left[ \hat{c} \left( c_j^1 \right)_{j \notin I} \equiv 0 \mod p' \mid (c_j^1)_{j \notin I} \xleftarrow{\$} (\mathbb{Z}_{p'})^{\lvert I \rvert} \right] \\
\leq \frac{\deg f}{p'} \leq \frac{\bar{d}}{2^\lambda} = \text{negl}(\lambda)
\]

where \( \bar{d} \) is the upper bound on degrees of the admissible functions in our scheme, which is polynomially bounded. Therefore, the error probability of the algorithm is negligible and we can efficiently determine if \( \tilde{c} \) is constant or not with overwhelming probability.

5.6 Security Proof

In this section, we prove our HAE scheme satisfies both IND-CPA and SUF-CPA. From this, we conclude that \( \Pi \) is actually IND-CCA and SUF-CCA by Theorem 11 and Theorem 12. For simplicity, we consider the scheme \( \Pi \) as an ideal scheme which is obtained by replacing the pseudorandom function \( F \) in our scheme with a random function from \( \{0, 1\}^\lambda \) into \( \mathbb{Z}_{q_0} \). If \( F \) is pseudorandom, then the security of this ideal scheme \( \Pi \) implies that of the real scheme.

Privacy

The privacy of the scheme \( \Pi \) is stated in the following theorem.

As we mentioned in Sec. 3.2, Coron et al. proved the equivalence of the EF-AGCD assumption and the decisional EF-AGCD assumption in [10]. So, this theorem actually shows that \( \Pi \) is IND-CPA under the computational \((\rho, \eta, \gamma)\)-EF-AGCD assumption.

Theorem 17. The scheme \( \Pi \) is IND-CPA under the decisional \((\rho, \eta, \gamma)\)-EF-AGCD assumption.

Proof. Suppose there exists a PPT adversary \( A \) for the IND-CPA security game of the scheme \( \Pi \) such that
\[
\Pr \left[ \text{IND-CPA}_{\Pi, A}(1^\lambda, Q) = 1 \right] \geq 1/2 + \epsilon(\lambda)
\]
for some modulus \( Q \in [2, 2^\lambda] \) and some non-negligible function \( \epsilon \). Then, we can construct a PPT distinguisher \( D \) for the decisional \((\rho, \eta, \gamma)\)-EF-AGCD problem, by simulating the game IND-CPA_{\Pi, A} as follows.

Distinguisher \( D(\rho, \eta, \gamma, y_0, D(p, q_0, \rho), z) \):

Initialization. Initialize a set \( S \leftarrow \emptyset \). Give \( ek = (Q, y_0) \) to \( A \).

Queries. For each encryption query \((\tau, m) \in \{0, 1\}^\lambda \times \mathbb{Z}_Q \) of \( A \), if \((\tau, \cdot, \cdot) \notin S \), then sample \( x \leftarrow D(p, q_0, \rho) \), compute \( c := (xQ + m) \mod y_0 \), return \( c \) to \( A \), and update \( S \) by \( S \leftarrow S \cup \{ (\tau, m, c) \} \). Otherwise, reject the query.
Challenge. For the challenge \((\tau^*, m_0^*, m_1^*)\) of \(A\), if \((\tau^*, \cdot, \cdot) \not\in S\), then flip a coin \(b \leftarrow \{0, 1\}\), compute the challenge ciphertext \(c^* := (zQ + m_b^*) \mod y_0\), return \(c^*\) to \(A\), and update \(S\) by \(S \leftarrow S \cup \{(\tau^*, m_b^*, c^*)\}\). Otherwise, reject the challenge.

Queries. Again \(A\) may make encryption queries adaptively, and such a query is answered exactly as before.

Finalization. For the output \(b'\) of \(A\), return 1 if \(b = b'\). Otherwise, return 0.

Note that \(\gcd(y_0, Q) = 1\) since \(y_0\) has no prime factors less than \(2^\lambda\) and \(Q\) is not greater than \(2^\lambda\). Let us consider the distribution of \(c\) produced by \(D\) in the Queries phase. Since \(c = xQ + m \mod y_0\) for \(x \leftarrow D(p, q_0, \rho)\), we have \(c = pqQ + rQ + m \mod y_0\) for some \(q \leftarrow [0, q_0)\) and \(r \leftarrow (-2^\rho, 2^\rho)\). Therefore, \(c \equiv rQ + m \pmod p\). Also, since \(q\) is uniform random on \([0, q_0)\), \(c \mod q_0\) is also uniform random on \(\mathbb{Z}_{q_0}\).

So, the distribution of \(c\) is identical to that of a real ciphertext and the encryption oracle can be simulated using \(D(p, q_0, \rho)\).

Now, consider the distribution of \(c^*\) in the Challenge phase. If \(z \leftarrow D(p, q_0, \rho)\), then the distribution of \(c^*\) is identical to that of original security game by the same reason as above. But if \(z \leftarrow Z_{y_0}\), then \(c^*\) is also uniformly distributed over \(\mathbb{Z}_{y_0}\) regardless of a random bit \(b\). Thus, \(c^*\) does not contain any information on the challenge plaintext \(m_b\). So

\[
\Pr[D(\rho, \eta, \gamma, y_0, D(p, q_0, \rho), z) = 1 \mid z \leftarrow D(p, q_0, \rho)] = \Pr[\text{IND-CPA}_{\Pi, A}(1^\lambda, Q) = 1] \geq \frac{1}{2} + \epsilon(\lambda)
\]

and

\[
\Pr[D(\rho, \eta, \gamma, y_0, D(p, q_0, \rho), z) = 1 \mid z \leftarrow Z_{y_0}] = \frac{1}{2}
\]

Therefore, the advantage of \(D\) is at least non-negligible \(\epsilon\), and this completes the proof.

Authenticity

The authenticity of the scheme \(\Pi\) is stated in the following theorem.

**Theorem 18.** If the \((\rho, \eta, \gamma)\)-EF-AGCD assumption holds, then the scheme \(\Pi\) is SUF-CPA.

**Proof.** Suppose there exists a PPT adversary \(A\) for the game SUF-CPA such that

\[
\Pr[\text{SUF-CPA}_{\Pi, A}(1^\lambda, Q) = 1] \geq \epsilon(\lambda)
\]

for some modulus \(Q \in [2, 2^\lambda]\) and some non-negligible function \(\epsilon\). Then, we can construct a PPT algorithm \(B\) for the \((\rho, \eta, \gamma)\)-EF-AGCD problem, by simulating the game SUF-CPA as follows.

**Algorithm** \(B(\rho, \eta, \gamma, y_0, D(p, q_0, \rho))\):

**Initialization.** Initialize \(S \leftarrow \emptyset\). Give \(ek = (Q, y_0)\) to \(A\).
Queries. For each encryption query \((\tau, m) \in \{0, 1\}^\lambda \times \mathbb{Z}_Q\) of \(A\), if \((\tau, \cdot, \cdot) \notin S\), then sample \(x \leftarrow D(p, q_0, \rho)\), compute \(c := (xQ + m) \mod y_0\), return \(c\) to \(A\), and update \(S\) by \(S \leftarrow S \cup \{(\tau, m, c)\}\). Otherwise, reject the query.

Finalization. Let \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) be the forgery attempt output by \(A\). For each \(i \in \{1, \cdots, l\}\), set the value of \(c_i\) as follows. Let \(c_i = c\) if \((\tau_i, m, c) \in S\) for some \(m \in \mathcal{M}\) and \(c \in \mathcal{C}\). Otherwise, choose \(c_i \leftarrow \mathbb{Z}_{y_0}\). And then, compute \(\hat{c} = f(c_1, \cdots, c_l) \mod y_0\). Output \(y_0/\gcd(y_0, \hat{c} - \hat{c})\).

For the same reason as in Theorem 17, the simulation of the encryption oracle by \(B\) is exact.

Consider the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) made by \(A\) in the Finalization phase. In case \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) is a strong forgery of type 1, \(\hat{c} = f(c_i)_{i \in I}\) is not constant, where \(I\) is the set of indices \(i\) such that \(\tau_i\) is not new with respect to \(S\). So we can apply the probabilistic polynomial identity test as in Theorem 16:

\[
\Pr \left[ \hat{c}(c_j)_{j \notin I} \equiv \hat{c} \mod y_0 \mid (c_j)_{j \notin I} \Stackrel{\$}{\leftarrow} (\mathbb{Z}_{y_0})^{l-|I|} \right] \leq \frac{d}{2^\lambda}
\]

where \(d\) is an upper bound on degrees of admissible functions in our scheme and is polynomially bounded. This means that \(\hat{c}(c_j)_{j \notin I} \neq \hat{c} \mod y_0\) with overwhelming probability. In case \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) is a strong forgery of type 2, \(\hat{c}(c_j)_{j \notin I} = \hat{c} \neq \hat{c} \mod y_0\).

Hence in both cases, we have \(\hat{c} \neq \hat{c} \mod y_0\), but also \(\hat{c} \equiv \hat{c} \mod q_0\), since any strong forgery is valid. Therefore, \(\gcd(y_0, \hat{c} - \hat{c}) = q_0\) and the output of the algorithm \(A'\) is exactly \(p\) with overwhelming probability if the challenge made by \(A\) is a strong forgery. Since \(A\) makes a strong forgery with non-negligible probability, \(A'\) outputs the correct answer \(p\) with non-negligible probability. \(\square\)
Chapter 6

Generic Compositions of HAE

It is natural that we think about a general method to combine a homomorphic secret-key encryption (HSE) for privacy and a homomorphic message authentication (HMA) for authenticity so that we get a homomorphic authenticated encryption (HAE) for both privacy and authenticity. This topic can be said to be the generic composition of a HAE. For the generic composition of a classical primitive of an authenticated encryption, there is the work [2] by Bellare and Namprempre. The following table is a summary of their results.

Along the line of their research, the same method will be applied to the homomorphic version of an authenticated encryption, in this chapter. For a given homomorphic secret-key encryption scheme \( \text{HSE} = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec}) \) and a given homomorphic message authentication scheme \( \text{HMA} = (\text{Gen}, \text{Auth}, \text{Eval}, \text{Verify}) \), we consider three ways of the generic composition of an HAE: Encrypt and Authenticate (E&A), Authenticate then Encrypt (AtE), Encrypt then Authenticate (EtA). In E&A composition, the final ciphertext is the concatenation of the ciphertext made by the encryption of HSE and the tag made by the authentication of HMA for a given plaintext. In AtE composition, a tag is made by the authentication of HMA for a given plaintext, then the final ciphertext is made by the encryption of HSE for the plaintext and the tag. In EtA composition, a ciphertext is made by the encryption of HSE for a given plaintext, then the final ciphertext is the concatenation of the ciphertext and a tag made by the authentication of HMA for the ciphertext.

In this chapter, three ways of generic composition of a homomorphic authenticated encryption are described and analyzed on their security. We consider the cases that HSE is IND-CPA and HMA is either UF-CMA or UF-CTA or SUF-CMA.

6.1 Encrypt and Authenticate (E&A)

The E&A composition of a homomorphic secret-key encryption scheme \( \text{HSE} = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec}) \) and a homomorphic message authentication scheme \( \text{HMA} = (\text{Gen}, \text{Auth}, \text{Eval}, \text{Verify}) \) is a homomorphic authenticated encryption scheme \( \text{HAE} = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec}) \), which is defined as follows. For simplicity and compatibility, we assume that \( \text{HSE}.\mathcal{M} = \text{HMA}.\mathcal{M}, \text{HSE}.\mathcal{F} = \text{HMA}.\mathcal{F} \).
SCHEME. $\text{HAE}_{E\&A} = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})$

- $(ek, sk) \leftarrow \text{Gen}(1^\lambda)$: Given a security parameter $\lambda$, generate key pairs $(\text{HSE}.ek, \text{HSE}.sk) \leftarrow \text{HSE}.\text{Gen}(1^\lambda)$ and $(\text{HMA}.ek, \text{HMA}.sk) \leftarrow \text{HMA}.\text{Gen}(1^\lambda)$. Return $(ek, sk)$, where $ek := (\text{HSE}.ek, \text{HMA}.ek)$ and $sk := (\text{HSE}.sk, \text{HMA}.sk)$.

- $c \leftarrow \text{Enc}(sk, \tau, m)$: Given the secret key $sk = (\text{HSE}.sk, \text{HMA}.sk)$, a label $\tau \in \mathcal{L}$ and a plaintext $m \in \mathcal{M}$, encrypt $c' \leftarrow \text{HSE}.\text{Enc}(\text{HSE}.sk, m)$ and authenticate $\sigma := \text{HMA}.\text{Auth}(\text{HMA}.sk, \tau, m)$. Return $c := (c', \sigma)$.

- $c \leftarrow \text{Eval}(ek, f, c_1, \ldots, c_l)$: Given the evaluation key $ek = (\text{HSE}.ek, \text{HMA}.ek)$, an arity-$l$ admissible function $f : \mathcal{M}^l \rightarrow \mathcal{M}$ and $l$ ciphertexts $c_1, \ldots, c_l$, where $c_i = (c_i', \sigma_i)$ for each $i = 1, \ldots, l$, evaluate $c' \leftarrow \text{HSE}.\text{Eval}(\text{HSE}.ek, f, c_1', \ldots, c_l')$ and $\sigma \leftarrow \text{HMA}.\text{Eval}(\text{HMA}.ek, f, \sigma_1, \ldots, \sigma_l)$. Return $c := (c', \sigma)$.

- $m$ or $\perp \leftarrow \text{Dec}(sk, (f, \tau_1, \ldots, \tau_l), c)$: Given the secret key $sk = (\text{HSE}.sk, \text{HMA}.sk)$, a labeled program $(f, \tau_1, \ldots, \tau_l)$, a ciphertext $c = (c', \sigma)$, decrypt $m \leftarrow \text{HSE}.\text{Dec}(\text{HSE}.sk, c')$ and then verify $b := \text{HMA}.\text{Verify}(\text{HMA}.sk, (f, \tau_1, \ldots, \tau_l), m, \sigma)$. If $b = 1$, then return $m$. Otherwise, return $\perp$.

Note that $\mathcal{M} := \text{HSE}.\mathcal{M} = \text{HMA}.\mathcal{M}$, $\mathcal{F} := \text{HSE}.\mathcal{F} = \text{HMA}.\mathcal{F}$, $\mathcal{L} := \text{HMA}.\mathcal{L}$ and $\mathcal{C} := \text{HSE}.\mathcal{C} \times \text{HMA}.\mathcal{C}$. The correctness and the compactness of the scheme are straightforward.

Security Analysis

Now, let us consider the security of the E&A composition $\text{HAE}_{E\&A}$.

**Theorem 19.** The E&A composition does not preserve IND-CPA of HSE.

**Proof.** Let a homomorphic message authentication scheme $\text{HMA} = (\text{Gen}, \text{Auth}, \text{Eval}, \text{Verify})$ be given. Then we can construct the following trivial scheme $\text{HMA}' = (\text{Gen'}, \text{Auth'}, \text{Eval'}, \text{Verify'})$, which transparently reveals a message.

SCHEME. $\text{HMA}' = (\text{Gen'}, \text{Auth'}, \text{Eval'}, \text{Verify'})$

- $(ek, sk) \leftarrow \text{Gen'}(1^\lambda)$: Given a security parameter $\lambda$, return $(ek, sk)$.

- $\sigma' \leftarrow \text{Auth'}(sk, \tau, m)$: Given the secret key $sk$, a label $\tau \in \mathcal{L}$ and a message $m \in \mathcal{M}$, return $\sigma' := (m, \sigma)$, where $\sigma \leftarrow \text{Auth}(sk, \tau, m)$.

- $\sigma' \leftarrow \text{Eval'}(ek, f, \sigma_1', \ldots, \sigma_l')$: Given the evaluation key $ek$, an arity-$l$ admissible function $f : \mathcal{M}^l \rightarrow \mathcal{M}$ and $l$ tags $\sigma_1', \ldots, \sigma_l'$, where $\sigma_i' = (m_i, \sigma_i)$ for each $i = 1, \ldots, l$, return $\sigma' := (f(m_1, \ldots, m_l), \text{Eval}(ek, f, \sigma_1, \ldots, \sigma_l))$. 52
\textbullet{} \ b \leftarrow \text{Verify}(sk, (f, \tau_1, \cdots, \tau_t), m', \sigma')$: Given the secret key \( sk \), a labeled program \((f, \tau_1, \cdots, \tau_t)\), a message \( m' \in \mathcal{M} \) and a tag \( \sigma' = (m, \sigma) \), return \( b \leftarrow \text{Verify}(sk, (f, \tau_1, \cdots, \tau_t), m, \sigma) \) if \( m = m' \). Otherwise, return 0.

Clearly, \( HMA' \) is as secure as \( HMA \). And an HAE scheme constructed by E&A composition of \( HSE \) and \( HMA' \) produces a ciphertext which transparently contains a plaintext. Thus, it can not satisfy IND-CPA even though \( HSE \) is IND-CPA.

The following theorem shows that the E&A composition preserves the unforgeability of \( HMA \).

**Theorem 20.** If \( HMA \) is UF-CMA or UF-CTA, then \( HAE_{E&A} \) is UF-CPA or UF-CCA, respectively.

**Proof.** Let \( A \) be a PPT adversary for the game UF-CPA\(_{HAE} \) with non-negligible advantage. We construct an adversary \( A' \) for the game UF-CMAC\(_{HMA} \) with non-negligible advantage, which simulates the game UF-CPA\(_{HAE,A} \) as follows.

\[ A'(1^\lambda): \]

**Initialization.** For a given key \( HMA.ek \), generate a pair of keys \((HSE.ek, HSE.sk) \leftarrow HSE.Gen(1^\lambda) \). Then \( ek := (HSE.ek, HMA.ek) \) is given to \( A \).

**Queries.** For each encryption query \((\tau, m)\) of \( A \), encrypt \( c' \leftarrow HSE.\text{Enc}(HSE.sk, m) \) and get an answer \( \sigma \) for the query \((\tau, m)\) from the authentication oracle \( HMA.\text{Auth.} \). Return \( c := (c', \sigma) \) to \( A \) as an answer for the query.

**Finalization.** For the forgery attempt \(((f, \tau_1, \cdots, \tau_t), (\hat{c}', \hat{\sigma})) \) of \( A \), output \(((f, \tau_1, \cdots, \tau_t), \hat{m}, \hat{\sigma}) \), where \( \hat{m} \leftarrow HSE.\text{Dec}(HSE.sk, \hat{c}') \).

Clearly, \( A' \) exactly simulates the game UF-CPA\(_{HAE,A} \). Let \( t_{\text{HAE}} := ((f, \tau_1, \cdots, \tau_t), (\hat{c}', \hat{\sigma})) \) and \( t_{\text{HMA}} := ((f, \tau_1, \cdots, \tau_t), \hat{m}, \hat{\sigma}) \) where \( \hat{m} \leftarrow HSE.\text{Dec}(HSE.sk, \hat{c}') \). Suppose that \( t_{\text{HAE}} \) is a forgery in \( HAE \). Then \( t_{\text{HMA}} \) is valid, that is, \( 1 \leftarrow HMA.\text{Verify}(HMA.sk, t_{\text{HMA}}) \) since \( \bot \neq \text{Dec}(sk, t_{\text{HAE}}) \). If \( t_{\text{HAE}} \) is a forgery of type 1, then \( t_{\text{HMA}} \) is also a forgery of type 1 since both schemes \( HAE \) and \( HMA \) use the same admissible function \( f \). And if \( t_{\text{HAE}} \) is a forgery of type 2, then \( t_{\text{HMA}} \) is also a forgery of type 2 since \( \hat{m} \neq \hat{m} = \text{Dec}(sk, t_{\text{HAE}}) \). Therefore, we can conclude that a forgery attempt \(((f, \tau_1, \cdots, \tau_t), \hat{m}, \hat{\sigma}) \) of \( A' \) is a forgery in \( HMA \), if a forgery attempt \(((f, \tau_1, \cdots, \tau_t), (\hat{c}', \hat{\sigma})) \) of \( A \) is a forgery in \( HAE \). So,

\[ \text{Adv}\text{_{HMA,A'}}^{\text{UF-CMA}}(\lambda) \geq \text{Adv}\text{_{HAE,A}}^{\text{UF-CPA}}(\lambda) \]

Therefore, \( \text{Adv}\text{_{HMA,A'}}^{\text{UF-CMA}}(\lambda) \) is non-negligible if \( \text{Adv}\text{_{HAE,A}}^{\text{UF-CPA}}(\lambda) \) is non-negligible.

Now, we consider the case that \( A \) is an PPT adversary for the game UF-CTA\(_{HAE} \) with non-negligible advantage. We construct an adversary \( A' \) for the game UF-CTA\(_{HMA} \) with non-negligible advantage, which simulates the game UF-CTA\(_{HAE,A} \) as follows. We only describe the query phase, since the other phases are equal to the above.
\(A'(1^\lambda)\):

**Queries.** For each encryption query \((\tau, m)\) of \(A\), encrypt \(c' \leftarrow \text{HSE.Enc}(\text{HSE.sk}, m)\) and get an answer \(\sigma\) for the query \((\tau, m)\) from the authentication oracle \(\text{HMA.Auth.}\). Return \(c := (c', \sigma)\) to \(A\) as an answer for the query. For each decryption query \(((f, \tau_1, \cdots, \tau_l), (\hat{c}', \hat{\sigma}))\) of \(A\), decrypt \(m \leftarrow \text{HSE.Dec}(\text{HSE.sk}, \hat{c}')\) and get an answer \(b\) for the query \(((f, \tau_1, \cdots, \tau_l), m, \sigma)\) from the verification oracle \(\text{HMA.Verify}\). If \(b = 1\), return \(m\). Otherwise, return \(\bot\).

Clearly, \(A'\) exactly simulates the game \(\text{UF-CCA}_{\text{HAE}, A}\). In the same reason as the above, a forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) of \(A'\) is a forgery in \(\text{HMA}\), if a forgery attempt \(((f, \tau_1, \cdots, \tau_l), (\hat{c}', \hat{\sigma}))\) of \(A\) is a forgery in \(\text{HAE}\). So,

\[
\text{Adv}_{\text{HMA}, A'}^{\text{UF-CTA}}(\lambda) \geq \text{Adv}_{\text{HAE}, A}^{\text{UF-CCA}}(\lambda)
\]

Therefore, \(\text{Adv}_{\text{HMA}, A'}^{\text{UF-CTA}}(\lambda)\) is non-negligible if \(\text{Adv}_{\text{HAE}, A}^{\text{UF-CCA}}(\lambda)\) is non-negligible.

**Theorem 21.** The E&A composition does not preserve the strong unforgeability of an HMA.

**Proof.** In general, HSE has the following property due to its homomorphic property.

For a given ciphertext \(c'\) in a scheme HSE, we can easily produce another ciphertext \(c''\) such that \(\text{HSE.Dec}(\text{HSE.sk}, c') = \text{HSE.Dec}(\text{HSE.sk}, c'')\).

This means a strong forgery \((c'', \sigma)\) in \(\text{HAE}_{E&A}\) can be easily made, for a given normal ciphertext \((c', \sigma)\).

From the above theorem, we can conclude that this parallel application of HSE and HMA fails to combine the privacy of HSE and the authenticity of HMA. This means that the E&A composition is useless to construct a HAE scheme, at least in generic ways.

### 6.2 Authenticate then Encrypt (AtE)

The AtE composition of a homomorphic secret-key encryption scheme \(\text{HSE} = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})\) and a homomorphic message authentication scheme \(\text{HMA} = (\text{Gen}, \text{Auth}, \text{Eval}, \text{Verify})\) is a homomorphic authenticated encryption scheme scheme \(\text{HAE}_{ALE} = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})\), which is defined as follows. For simplicity and compatibility, we assume that \(\text{HSE.M} = \text{HMA.M} \times \text{HMA.T}\).

**SCHEME.** \(\text{HAE}_{ALE} = (\text{Gen}, \text{Enc}, \text{Eval}, \text{Dec})\)

- \((ek, sk) \leftarrow \text{Gen}(1^\lambda)\): Given a security parameter \(\lambda\), generate key pairs \((\text{HSE.ek}, \text{HSE.sk}) \leftarrow \text{HSE.Gen}(1^\lambda)\) and \((\text{HMA.ek}, \text{HMA.sk}) \leftarrow \text{HMA.Gen}(1^\lambda)\). Return \((ek, sk)\), where \(ek := (\text{HSE.ek}, \text{HMA.ek})\) and \(sk := (\text{HSE.sk}, \text{HMA.sk})\).
Now, let us consider the security of the AtE composition $\text{HAE}$. The correctness and the compactness of the scheme are straightforward.

Let $\text{HSE}$ be a PPT adversary for the game IND-CPA if $f$ is admissible in the scheme $\text{HAE}_{\text{AE}}$ if $f$ is admissible in the HMA and $\bar{f}$ is admissible in the HSE.

The correctness and the compactness of the scheme are straightforward.

**Security Analysis**

Now, let us consider the security of the AtE composition $\text{HAE}_{\text{AE}}$. The following theorem shows that the AtE composition preserves privacy of HSE.

**Theorem 22.** If HSE is IND-CPA, then $\text{HAE}_{\text{AE}}$ is IND-CPA.

**Proof.** Let $A$ be a PPT adversary for the game IND-CPA_{HAE} with non-negligible advantage. We construct an adversary $A'$ for the game IND-CPA_{HSE} with non-negligible advantage, which simulates the game IND-CPA_{HAE,A} as follows.

$A'(1^\lambda):$

**Initialization.** For a given key $\text{HSE}.ek$, generate a pair of keys $(\text{HMA}.ek, \text{HMA}.sk) \leftarrow \text{HMA}.\text{Gen}(1^\lambda)$. Then $ek := (\text{HSE}.ek, \text{HMA}.ek)$ is given to $A$.

**Queries.** For each encryption query $(\tau, m)$ of $A$, authenticate $\sigma \leftarrow \text{HMA}.\text{Auth}(\text{HMA}.sk, \tau, m)$ and get an answer $c$ for the query $(m, \sigma)$ from the encryption oracle $\text{HSE}.\text{Enc}$. Return $c$ to $A$ as an answer for the query.

**Challenge.** For the challenge $(\tau^\ast, m_0^\ast, m_1^\ast)$ of $A$, $\sigma_0^\ast \leftarrow \text{HMA}.\text{Auth}(\text{HMA}.sk, \tau^\ast, m_0^\ast)$ for each $b \in \{0, 1\}$. Get the challenge ciphertext $c^\ast$ for the challenge $((m_0^\ast, \sigma_0^\ast), (m_1^\ast, \sigma_1^\ast))$ from the encryption oracle $\text{HSE}.\text{Enc}$. Return $c^\ast$ to $A$. 

\begin{itemize}
  
  \item $c \leftarrow \text{Enc}(sk, \tau, m)$: Given the secret key $sk = (\text{HSE}.sk, \text{HMA}.sk)$, a label $\tau \in \mathcal{L}$ and a plaintext $m \in \mathcal{M}$, authenticate $\sigma \leftarrow \text{HMA}.\text{Auth}(\text{HMA}.sk, \tau, m)$ then encrypt $c \leftarrow \text{HSE}.\text{Enc}(\text{HSE}.sk, (m, \sigma))$. Return $c$.
  
  \item $c \leftarrow \text{Eval}(ek, f, c_1, \cdots, c_l)$: Given the evaluation key $ek = (\text{HSE}.ek, \text{HMA}.ek)$, an arity-$l$ admissible function $f : \mathcal{M}^l \rightarrow \mathcal{M}$ and $l$ ciphertexts $c_1, \cdots, c_l$, evaluate $c \leftarrow \text{HSE}.\text{Eval}(\text{HSE}.ek, \bar{f}, c_1, \cdots, c_l)$ where $\bar{f} : \text{HSE} \cdot \mathcal{M}^l \rightarrow \text{HSE} \cdot \mathcal{M}$ is defined as below.

$$\bar{f}((m_1, \sigma_1), \cdots, (m_l, \sigma_l)) = (f(m_1, \cdots, m_l), \text{HMA}.\text{Eval}(\text{HSE}.ek, f, \sigma_1, \cdots, \sigma_l))$$

Return $c$.
  
  \item $m$ or $\perp \leftarrow \text{Dec}(sk, (f, \tau_1, \cdots, \tau_l), c)$: Given the secret key $sk = (\text{HSE}.sk, \text{HMA}.sk)$, a labeled program $(f, \tau_1, \cdots, \tau_l)$, a ciphertext $c \in \mathcal{C}$, decrypt $(m, \sigma) \leftarrow \text{HSE}.\text{Dec}(\text{HSE}.sk, c)$, then verify $b \leftarrow \text{HMA}.\text{Verify}(\text{HMA}.sk, (f, \tau_1, \cdots, \tau_l), m, \sigma)$. If $b = 1$, then return $m$. Otherwise, return $\perp$.
\end{itemize}
Queries. For each encryption query \((\tau, m)\) of \(A\), answer for the query precisely as before.

Finalization. For the output bit \(b'\) of \(A\), return \(b'\)

Clearly, \(A'\) exactly simulates the game IND-CPA_{H, A} and

\[
\text{Adv}^{\text{IND-CPA}}_{\text{HSE, } A'}(\lambda) = \text{Adv}^{\text{IND-CPA}}_{\text{HAE, } A}(\lambda)
\]

Therefore, \(\text{Adv}^{\text{IND-CPA}}_{\text{HSE, } A'}(\lambda)\) is also non-negligible if \(\text{Adv}^{\text{IND-CPA}}_{\text{HAE, } A}(\lambda)\) is non-negligible.

The following theorem shows that the AtE composition preserves the unforgeability of HMA.

**Theorem 23.** If HMA is UF-CMA or UF-CTA, then HAE_{AtE} is UF-CPA or UF-CCA, respectively.

**Proof.** Let \(A\) be a PPT adversary for the game UF-CPA_{HAE} with non-negligible advantage. We construct an adversary \(A'\) for the game UF-CMA_{HMA} with non-negligible advantage, which simulates the game UF-CPA_{HAE, A} as follows.

\(A'(1^\lambda)\):

**Initialization.** For a given key HMA.\(ek\), generate a pair of keys \((\text{HSE.}\!ek, \text{HSE.}\!sk) \leftarrow \text{HSE.Gen}(1^\lambda)\). Then \(ek := (\text{HSE.}\!ek, \text{HMA.}\!ek)\) is given to \(A\).

**Queries.** For each encryption query \((\tau, m)\) of \(A\), get an answer \(\sigma\) for the query \((\tau, m)\) from the authentication oracle HMA.Auth and encrypt \(c \leftarrow \text{HSE.Enc(HSE.sk, } (m, \sigma))\). Return \(c\) to \(A\) as an answer for the query.

**Finalization.** For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) of \(A\), output \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\), where \((\hat{m}, \hat{\sigma}) \leftarrow \text{HSE.Dec(HSE.sk, } \hat{c})\).

Clearly, \(A'\) exactly simulates the game UF-CPA_{HSE, A}. And a forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) of \(A'\) is a forgery in HMA, if a forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) of \(A\) is a forgery in HAE. So,

\[
\text{Adv}^{\text{UF-CMA}}_{\text{HMA, } A'}(\lambda) \geq \text{Adv}^{\text{UF-CPA}}_{\text{HAE, } A}(\lambda)
\]

Therefore, \(\text{Adv}^{\text{UF-CMA}}_{\text{HMA, } A'}(\lambda)\) is also non-negligible if is non-negligible.

In case that \(A\) is a PPT adversary for the game UF-CCA_{HAE} with non-negligible advantage. We construct an adversary \(A'\) for the game UF-CTA_{HMA} with non-negligible advantage, which simulates the game UF-CCA_{HAE, A} as follows. We only describe the query phase, since the other phases are equal to the above.

\(A'(1^\lambda)\):
Queries. For each encryption query \((\tau, m)\) of \(A\), get an answer \(\sigma\) for the query \((\tau, m)\) from the authentication oracle \(HMA\). Auth and encrypt \(c \leftarrow HSE.Enc(HSE.sk, (m, \sigma))\). Return \(c\) to \(A\) as an answer for the query. For each decryption query \(((f, \tau_1, \cdots, \tau_l), c)\) of \(A\), decrypt \((m, \sigma) \leftarrow HSE.Dec(HSE.sk, c)\) and get an answer \(b\) for the query \(((f, \tau_1, \cdots, \tau_l), m, \sigma)\) from the verification oracle \(HMA\). Verify. If \(b = 1\), return \(m\). Otherwise, return \(\bot\).

Clearly, \(A'\) exactly simulates the game \(UF-CCA_{HAE,A}\). And a forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{m}, \hat{\sigma})\) of \(A'\) is a forgery in \(HMA\), if a forgery attempt \(((f, \tau_1, \cdots, \tau_l), \hat{c})\) of \(A\) is a forgery in \(HAE\). So,

\[
Adv_{HMA,A'}^{UF-CTA}(\lambda) \geq Adv_{HAE,A}^{UF-CCA}(\lambda)
\]

Therefore, \(Adv_{HMA,A'}^{UF-CTA}(\lambda)\) is also non-negligible.

\(\Box\)

**Theorem 24.** The AtE composition does not preserve the strong unforgeability of an HMA.

**Proof.** In general, HSE has the following property due to its homomorphic property.

For a given ciphertext \(c\) in a scheme HSE, we can easily produce another ciphertext \(c'\) such that \(HSE.Dec(HSE.sk, c) = HSE.Dec(HSE.sk, c')\).

This means a strong forgery \(c'\) in \(HAE_{E\&A}\) can be easily made, for a given normal ciphertext \(c\).

\(\Box\)

### 6.3 Encrypt then Authenticate (EtA)

The EtA composition of a homomorphic secret-key encryption scheme \(HSE = (Gen, Enc, Eval, Dec)\) and a homomorphic message authentication scheme \(HMA = (Gen, Auth, Eval, Verify)\) is a homomorphic authenticated encryption scheme scheme \(HAE_{EtA} = (Gen, Enc, Eval, Dec)\), which is defined as follows. For simplicity and compatibility, we assume that \(HMA.M = HSE.C\).

**SCHEME.** \(HAE_{EtA} = (Gen, Enc, Eval, Dec)\)

- \((ek, sk) \leftarrow Gen(1^\lambda)\): Given a security parameter \(\lambda\), generate key pairs \((HSE.ek, HSE.sk) \leftarrow HSE.Gen(1^\lambda)\) and \((HMA.ek, HMA.sk) \leftarrow HMA.Gen(1^\lambda)\). Return \((ek, sk)\), where \(ek := (HSE.ek, HMA.ek)\) and \(sk := (HSE.sk, HMA.sk)\).
- \(c \leftarrow Enc(sk, \tau, m)\): Given the secret key \(sk = (HSE.sk, HMA.sk)\), a label \(\tau \in HMA.L\) and a plaintext \(m \in HSE.M\), encrypt \(c' \leftarrow HSE.Enc(HSE.sk, m)\) then authenticate \(\sigma \leftarrow HMA.Auth(HMA.sk, \tau, c')\). Return \(c := (c', \sigma)\).
Now, let us consider the security of the EtA composition

Security Analysis

and the compactness of the scheme are straightforward.

Proof. Let $A$ be a PPT adversary for the game IND-CPA. We construct an adversary $A'$ that simulates the game IND-CPA.

Initialization. For a given key $(HSE.ek, HMA.ek)$, an arity-$l$ admissible function $f : (HSE.M)^l \rightarrow HSE.M$ and $l$ ciphertexts $c_1, \ldots, c_l$, where $c_i = (\sigma_i, \tau_i)$ for each $i = 1, \ldots, l$, evaluate $\tilde{c} \leftarrow HSE.Eval(HSE.ek, f, c_1, \ldots, c_l)$ and $\tilde{\sigma} \leftarrow HMA.Eval(HMA.ek, f, \sigma_1, \ldots, \sigma_l)$, where $f : (HSE.M)^l \rightarrow HMA.M$.

Note that $M := HSE.M$, $L := HMA.L$ and $C := HSE.C \times HMA.T$. A function $f$ is admissible in the scheme $HAE_{E\!L\!A}$ if $f$ is admissible in $HSE$ and $\tilde{f}$ is admissible in the HMA. The correctness and the compactness of the scheme are straightforward.

Theorem 25. If $HSE$ is IND-CPA, then $HAE_{E\!L\!A}$ is IND-CPA.

Proof. Let $A$ be a PPT adversary for the game IND-CPA of $HSE$ as the EtA composition does.

Proof. Let $A$ be a PPT adversary for the game IND-CPA of $HSE$ as the EtA composition does.
Queries. For each encryption query \((\tau, m)\) of \(A\), answer for the query precisely as before.

Finalization. For the output bit \(b'\) of \(A\), return \(b'\).

Clearly, if \(c^* \leftarrow \text{HSE.Enc}(\text{HSE.sk}, m^*_b)\), then \(c^* \leftarrow \text{Enc}(sk, \tau^*, m^*_b)\). So, \(A'\) exactly simulates the game \(\text{IND-CPA}_{\text{HAE}, A}\) and

\[
\text{Adv}^{\text{IND-CPA}}_{\text{HSE}, A'}(\lambda) = \text{Adv}^{\text{IND-CPA}}_{\text{HAE}, A}(\lambda)
\]

Therefore, \(\text{Adv}^{\text{IND-CPA}}_{\text{HSE}, A'}(\lambda)\) is also non-negligible.

The following theorem shows that the EtA composition preserves the unforgeability of \(\text{HMA}\).

**Theorem 26.** If \(\text{HMA}\) is \(\text{UF-CMA}\) or \(\text{UF-CTA}\), then \(\text{HAE}_{\text{EtA}}\) is \(\text{UF-CPA}\) or \(\text{UF-CCA}\), respectively.

**Proof.** Let \(A\) be a PPT adversary for the game \(\text{UF-CPA}_{\text{HAE}}\) with non-negligible advantage. We construct an adversary \(A'\) for the game \(\text{UF-CMA}_{\text{HMA}}\) with non-negligible advantage, which simulates the game \(\text{UF-CPA}_{\text{HAE}, A}\) as follows.

\(A'(1^\lambda)\):

**Initialization.** For a given key \(\text{HMA.ek}\), generate a pair of keys \((\text{HSE.ek}, \text{HSE.sk}) \leftarrow \text{HSE.Gen}(1^\lambda)\) and initialize a set \(S\) to be empty. Then \(ek := (\text{HSE.ek}, \text{HMA.ek})\) is given to \(A\).

**Queries.** For each encryption query \((\tau, m)\) of \(A\), if \((\tau, \cdot, \cdot) \in S\), then reject the query of \(A\). Otherwise, encrypt \(c \leftarrow \text{HSE.Enc}(\text{HSE.sk}, m)\) and get an answer \(\sigma\) for the query \((\tau, c)\) from the authentication oracle \(\text{HMA.Auth}\). Return \((c, \sigma)\) to \(A\) as an answer for the query.

**Finalization.** For the forgery attempt \(((f, \tau_1, \cdots, \tau_l), (\hat{c}, \hat{\sigma}))\) of \(A\), output \(((\bar{f}, \tau_1, \cdots, \tau_l), \hat{c}, \hat{\sigma})\), where \(\bar{f} : (\text{HMA.}\mathcal{M})^l \to \text{HMA.}\mathcal{M}\) is defined as below:

\[
\bar{f}(c_1, \cdots, c_l) := \text{HSE.Eval}(\text{HSE.ek}, f, c_1, \cdots, c_l)
\]

In case that \(A\) is an PPT adversary for the game \(\text{UF-CCA}_{\text{HAE}}\) with non-negligible advantage, we can also construct an adversary \(A''\) for the game \(\text{UF-CTA}_{\text{HMA}}\) with non-negligible advantage, which simulates the game \(\text{UF-CCA}_{\text{HAE}, A}\) as follows. We only describe the query phase, since the other phases are equal to the above.

\(A''(1^\lambda)\):
Queries. For each encryption query \((\tau, m)\) of \(A\), if \((\tau, \cdot, \cdot) \in S\), then reject the query of \(A\). Otherwise, encrypt \(c \leftarrow \text{HSE.Enc(HSE.sk, } m)\) and get an answer \(\sigma\) for the query \((\tau, c)\) from the authentication oracle HMA.Auth. Return \((c, \sigma)\) to \(A\) as an answer for the query. For each decryption query \(((\bar{f}, \tau_1, \cdots, \tau_l), (c, \sigma))\) of \(A\), get an answer \(b\) for the query \(((\bar{f}, \tau_1, \cdots, \tau_l), c, \sigma)\) from the verification oracle HMA.Verify. If \(b = 1\), return \(m \leftarrow \text{HSE.Dec(HSE.sk, } c)\). Otherwise, return \(\perp\).

Clearly, \(A'\) and \(A''\) exactly simulate the game UF-CPA\(_{\text{HAE, } A}\) and UF-CCA\(_{\text{HAE, } A}\), respectively.

Now we only need to show that if \(t := ((f, \tau_1, \cdots, \tau_l), (\hat{c}, \hat{\sigma}))\) is a forgery in HAE, then \(t' := ((\bar{f}, \tau_1, \cdots, \tau_l), \hat{c}, \hat{\sigma})\) is a forgery in HMA. If so,

\[
\text{Adv}^{\text{UF-CMA}}_{\text{HMA, } A'}(\lambda) \geq \text{Adv}^{\text{UF-CPA}}_{\text{HAE, } A}(\lambda)
\]

and

\[
\text{Adv}^{\text{UF-CTA}}_{\text{HMA, } A''}(\lambda) \geq \text{Adv}^{\text{UF-CCA}}_{\text{HAE, } A}(\lambda).
\]

Thus, we can conclude that \(\text{Adv}^{\text{UF-CMA}}_{\text{HMA, } A'}(\lambda)\) and \(\text{Adv}^{\text{UF-CTA}}_{\text{HMA, } A''}(\lambda)\) is non-negligible if \(\text{Adv}^{\text{UF-CPA}}_{\text{HAE, } A}(\lambda)\) and \(\text{Adv}^{\text{UF-CCA}}_{\text{HAE, } A}(\lambda)\) is non-negligible, respectively.

Suppose that \(t\) is a forgery in the scheme HAE. Firstly, \(1 \leftarrow \text{HMA.Verify(HMA.sk, } t')\), that is, \(t'\) is valid in HMA since \(t\) is valid in HAE. In case that \(t\) is a forgery of type 1, \(f(m_i)_{i \in I}\) is nonconstant and so \(\bar{f}(c_i)_{i \in I}\) is also nonconstant. This means that \(t'\) is a forgery of type 1 in HMA. In case that \(t\) is a forgery of type 2, \(\bar{m} := f(m_i)_{i \in I}\) is constant and \(\bar{m} \neq \text{HSE.Dec(HSE.sk, } \hat{c})\). In this case, \(t'\) is clearly a forgery of type 1 in HMA if \(\bar{f}(c_i)_{i \in I}\) is nonconstant. If \(\hat{c} := f(c_i)_{i \in I}\) is constant, then \(\hat{c} \neq \hat{c}\) since \(\bar{m} \leftarrow \text{HSE.Dec(HSE.sk, } \hat{c})\). This means that \(t'\) is a forgery of type 2 in HMA. Therefore, if \(t\) is a forgery in HAE, then \(t'\) is a forgery in HMA. This completes the proof. \(\square\)

The following theorem shows that the EtA composition also preserves the strong unforgeability of HMA.

**Theorem 27.** If HMA is SUF-CMA or SUF-CTA, then HAE\(_{\text{HAE}}\) is SUF-CPA or SUF-CCA, respectively.

**Proof.** Let \(A\) be a PPT adversary for the game SUF-CPA\(_{\text{HAE}}\) with non-negligible advantage. We construct an adversary \(A'\) for the game SUF-CMA\(_{\text{HMA}}\) with non-negligible advantage, which simulates the game SUF-CPA\(_{\text{HAE, } A}\) in the same way as the game \(A'\) described in the proof of the previous theorem. In case that \(A\) is a PPT adversary for the game SUF-CCA\(_{\text{HAE}}\) with non-negligible advantage, we can also construct an adversary \(A''\) for the game SUF-CTA\(_{\text{HMA}}\) with non-negligible advantage, which simulates the game SUF-CCA\(_{\text{HAE, } A}\) in the same way as the game \(A''\) described in the proof of the previous theorem. Clearly, \(A'\) and \(A''\) exactly simulate the game SUF-CPA\(_{\text{HAE, } A}\) and SUF-CCA\(_{\text{HAE, } A}\), respectively.
Now we only need to show that if $t := ((f, \tau_1, \cdots, \tau_l), (\hat{c}, \hat{\sigma}))$ is a strong forgery in HAE, then $t' := ((\hat{f}, \tau_1, \cdots, \tau_l), \hat{c}, \hat{\sigma})$ is a strong forgery in HMA. If so,

$$\text{Adv}_{\text{HMA}, A'}^{\text{SUFCMA}}(\lambda) \geq \text{Adv}_{\text{HAE}, A}^{\text{SUFCPA}}(\lambda)$$

and

$$\text{Adv}_{\text{HMA}, A'}^{\text{SUFCFTA}}(\lambda) \geq \text{Adv}_{\text{HAE}, A}^{\text{SUFCCCA}}(\lambda).$$

Thus, we can conclude that $\text{Adv}_{\text{HMA}, A'}^{\text{SUFCMA}}(\lambda)$ and $\text{Adv}_{\text{HMA}, A'}^{\text{SUFCFTA}}(\lambda)$ is non-negligible if $\text{Adv}_{\text{HAE}, A}^{\text{SUFCPA}}(\lambda)$ and $\text{Adv}_{\text{HAE}, A}^{\text{SUFCCCA}}(\lambda)$ is non-negligible, respectively.

Suppose that $t$ is a strong forgery in the scheme HAE. Firstly, $1 \leftarrow \text{HAE}.\text{Verify}(\text{HAE}.sk, t')$, that is, $t'$ is valid in HMA since $t$ is valid in HAE. In case that $t$ is a strong forgery of type 1, either $f(m_i)_{i \in I}$ or $\text{Eval}(ek, f, (c_i, \sigma_i)_{i \in I})$ is nonconstant. If $f(m_i)_{i \in I}$ is nonconstant, then $f(c_i)_{i \in I}$ is also nonconstant and so $t'$ is a strong forgery of type 1 in HMA. If $\text{Eval}(ek, f, (c_i, \sigma_i)_{i \in I})$ is nonconstant, then either $f(c_i)_{i \in I}$ or HMA.$\text{Eval}(\text{HMA}.ek, f, (\sigma_i)_{i \in I})$ is nonconstant since

$$\text{Eval}(ek, f, (c_i, \sigma_i)_{i \in I}) = (f(c_i)_{i \in I}, \text{HMA}.\text{Eval}(\text{HMA}.ek, f, (\sigma_i)_{i \in I})).$$

This means that $t'$ is a strong forgery of type 1 in HMA. In case that $t$ is a forgery of type 2, $(\hat{c}, \hat{\sigma}) := \text{Eval}(ek, f, (c_i, \sigma_i)_{i \in I})$ is constant and $(\hat{c}, \hat{\sigma}) \neq (\hat{c}, \hat{\sigma})$. In this case, $t'$ is clearly a forgery of type 2 in HMA. Therefore, if $t$ is a strong forgery in HAE, then $t'$ is a strong forgery in HMA.

This completes the proof.

\[\square\]

**Theorem 28.** If HSE is IND-CPA and HMA is UF-CTA, then HAE$_{E1A}$ is IND-CCA.

**Proof.** Let $A$ be a PPT adversary for the game IND-CCA$_{HAE}$ with non-negligible advantage. We construct an adversary $A'$ for the game IND-CPA$_{HSE}$ with non-negligible advantage, which simulates the game IND-CCA$_{HAE,A}$ as follows.

$A'(1^\lambda)$:

**Initialization.** Let $ek := (\text{HSE}.ek, \text{HMA}.ek)$ and then $ek$ is given to $A$.

**Queries.** For each encryption query $(\tau, m)$ of $A$, if $(\tau, \cdot, \cdot) \in S$, then reject the query of $A$. Otherwise, get an answer $c$ for the query $m$ from the encryption oracle HSE.Enc and then get an answer $\sigma$ for the query $(\tau, c)$ from the authentication oracle HMA.Auth. Return $(c, \sigma)$ to $A$ as an answer for the query. For each decryption query $((f, \tau_1, \cdots, \tau_l), (c, \sigma))$ of $A$, get an answer $b$ for the query $((\hat{f}, \tau_1, \cdots, \tau_l), c, \sigma)$ from the verification oracle HMA.Verify. If $b = 1$, return $m = f(m_1, \cdots, m_l)$ as an answer for the query, where $m_i \notin \mathcal{M}$ for each $i$ with $(\tau_i, \cdot, \cdot) \notin S$. Otherwise, return $\bot$. 

61
**Challenge.** For the challenge \((\tau^*, m_0^*, m_1^*)\) of \(A\), if \((\tau^*, \cdot, \cdot) \in S\), then reject the challenge of \(A\). Otherwise, get the challenge ciphertext \(c^*\) for challenge \((m_0^*, m_1^*)\) from HSE. Enc. Then, get an answer \(\sigma^*\) for the query \((\tau^*, c^*)\) from the authentication oracle HMA. Auth. Then return \((c^*, t^*)\) to \(A\).

**Queries.** For each encryption or decryption query of \(A\), answer for the query precisely as before.

**Finalization.** For the output bit \(b'\) of \(A\), return \(b'\)

This game simulates the original security game IND-CCA_{HAE,A} except that the decryption query \((f, \tau_1, \cdots, \tau_l, (c, \sigma))\) of \(A\) is a forgery in HAE, that is,

\[
1 \leftarrow \text{HMA.} \text{Verify}(\text{HMA.} sk, (\bar{f}, \tau_1, \cdots, \tau_l), c, \sigma) \\
\bar{f}(m_1, \cdots, m_l) \neq \text{HSE.} \text{Dec}(\text{HSE.} sk, c).
\]

As we have shown before, it means that \(((\bar{f}, \tau_1, \cdots, \tau_l), c, t)\) is a forgery in HMA. Thus

\[
\text{Adv}^{\text{IND-CCA}_{\text{HAE}}}(\lambda) \leq \text{Adv}^{\text{IND-CPA}_{\text{HSE}}}(\lambda) + q \cdot \text{Adv}^{\text{UF-CTA}_{\text{HMA}}}(\lambda)
\]

where \(q \leq poly(\lambda)\) is the number of the decryption queries of \(A\). This proves that \(\text{Adv}^{\text{IND-CPA}_{\text{HSE}}}(\lambda)\) and \(\text{Adv}^{\text{UF-CTA}_{\text{HMA}}}(\lambda)\) are negligible, then \(\text{Adv}^{\text{IND-CCA}_{\text{HAE}}}(\lambda)\) is also negligible.

**Corollary 1.** If HSE is IND-CPA and HMA is SUF-CMA, then HAE_{ELA} is IND-CCA and SUF-CCA.

**Corollary 2.** If HSE is IND-CPA and HMA is UF-CTA, then HAE_{ELA} is IND-CCA and UF-CCA.
Bibliography


