Homomorphic Authenticated Encryption Secure Against Chosen-Ciphertext Attack

Chihong Joo and Aaram Yun

Ulsan National Institute of Science and Technology (UNIST)
Republic of Korea
{chihongjoo,aaramyun}@unist.ac.kr

Abstract. We study *homomorphic authenticated encryption*, where privacy and authenticity of data are protected simultaneously. We define homomorphic versions of various security notions for privacy and authenticity, and investigate relations between them. In particular, we show that it is possible to give a natural definition of IND-CCA for homomorphic authenticated encryption, unlike the case of homomorphic encryption. Also, we construct a homomorphic authenticated encryption scheme supporting arithmetic circuits, which is chosen-ciphertext secure both for privacy and authenticity. Our scheme is based on the error-free approximate GCD assumption.

Key words: homomorphic authenticated encryption, homomorphic MAC, homomorphic encryption, approximate GCD assumption

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 [16] 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 [23, 22, 7, 8, 17, 12, 13, 4, 6, 18, 19, 10, 5, 11]. 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 symmetric-key cryptography, the authenticated encryption [21, 2, 14, 20] 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. A HAE is a symmetric-key primitive 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, if there exists a fully homomorphic signature, or even a fully homomorphic MAC, then we may construct a fully homomorphic authenticated encryption scheme by generic composition [2]. Unfortunately, the homomorphic signature scheme closest to

being fully homomorphic is [3], where only low-degree polynomial functions are supported. A fully homomorphic MAC is proposed by Gennaro and Wichs [15], but it supports only a limited number of verification queries, so that the solution is in a sense incomplete. So far, the problem of constructing a fully homomorphic authenticated encryption is still not completely solved.

Our contribution in this paper is twofold. First, we define various security notions for HAE and study relations among them. For privacy, we define homomorphic versions of IND-CPA and 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-CPA, the homomorphic version of the unforgeability when the adversary has access to the encryption oracle. We also consider UF-CCA, where the adversary not only has the encryption oracle but also the decryption oracle. Moreover, we consider strong unforgeability flavors of authenticity and define homomorphic versions accordingly: SUF-CPA and SUF-CCA. We investigate relationship between these notions, and, for example, show that 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 a HAE scheme supporting arithmetic circuits. This scheme is not fully homomorphic, but only somewhat homomorphic, but we show that our scheme is secure and satisfies both IND-CCA and SUF-CCA. Another appeal of our scheme is that it is a simple and natural construction based on the *error-free approximate GCD* (EF-AGCD) assumption. EF-AGCD assumption was used before [23, 12, 13, 10, 11] 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 $Q \in \mathbb{Z}^+$.

2 Related work

Gennaro and Wichs [15] proposed the first construction of the fully homomorphic MAC. 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. Construction of a fully-homomorphic authenticated encryption scheme satisfying chosen ciphertext security is still an interesting open problem.

Catalano and Fiore [9] proposed two somewhat homomorphic MACs supporting arithmetic circuits on $\mathbb{Z}_p$ for prime modulus p. In their construction, a MAC 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 α is equal to randomness determined by the 'label' τ 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 β 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 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

MAC. 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.

Our scheme can also be compared with a homomorphic encryption scheme called IDGHV presented in [10]. 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, symmetric-key variant of IDGHV where $\ell = 1$, and where encryption randomness is pseudorandomly generated from the label. We intentionally omitted encryption of multiple plaintexts for simplicity of exposition, but our construction can naturally be extended in this way.

The privacy of our scheme is proved using the decisional EF-AGCD assumption, which is suggested by Cheon at al. [10]. Recently, Coron et al. proposed a scale-invariant fully homomorphic encryption scheme over integers (to be published in PKC 2014 [11]). In the paper, 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 paper, 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.

3 Preliminary

3.1 Notations

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

$$\begin{aligned}
[a, b] &:= \{x \in \mathbb{Z} \mid a \leq x \leq b\}, \\
(a, b] &:= \{x \in \mathbb{Z} \mid a < x \leq b\}, \\
[a, b) &:= \{x \in \mathbb{Z} \mid a \leq x < b\}, \\
(a, b) &:= \{x \in \mathbb{Z} \mid a < x < b\}.
\end{aligned}$$

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

$$x \bmod n := x - \left\lfloor \frac{x}{n} \right\rceil \cdot n$$

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 $\gcd(n, m) = 1$, $\mathrm{CRT}_{(n,m)}$ is the isomorphism from $\mathbb{Z}_n \times \mathbb{Z}_m$ onto $\mathbb{Z}_{nm}$, satisfying

$$(\mathrm{CRT}_{(n,m)}(a, b) \bmod n, \mathrm{CRT}_{(n,m)}(a, b) \bmod m) = (a, b)$$

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

In this paper, the security parameter is always denoted as λ, and the expression

$$f(\lambda) = negl(\lambda)$$

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.

3.2 Security assumptions

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

Definition 1. *For any positive integers p, q_0, ρ, let us define the following distributions.*

$$\mathcal{D}(p, q_0, \rho) := \{choose\ q \xleftarrow{\$} [0, q_0),\ r \xleftarrow{\$} (-2^\rho, 2^\rho) :\ \ output\ pq + r\},$$

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

Definition 2 (Error-Free Approximate GCD Assumption)**.** *The (computational) (ρ, η, γ)-EF-AGCD assumption is that, for any PPT adversary A, we have*

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

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

The EF-AGCD assumption is suggested by Coron et al. [12] to prove the security of their variant of the DGHV scheme [23].

There is also a decisional version of the EF-AGCD assumption, which is suggested by Cheon at al. [10].

Definition 3 (Decisional Error-Free Approximate GCD Assumption)**.** *The decisional (ρ, η, γ)-EF-AGCD assumption is that, for any PPT distinguisher D, we have*

$$\begin{aligned}&\Big|\Pr\left[D(\rho, \eta, \gamma, y_0, \mathcal{D}(p, q_0, \rho), z) = 1 \mid z \leftarrow \mathcal{D}(p, q_0, \rho)\right] \\ &\qquad - \Pr\left[D(\rho, \eta, \gamma, y_0, \mathcal{D}(p, q_0, \rho), z) = 1 \mid z \xleftarrow{\$} \mathbb{Z}_{y_0}\right]\Big| \\ &= negl(\lambda)\end{aligned}$$

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

Recently, Coron et al. proved the equivalence of the EF-AGCD assumption and the decisional EF-AGCD assumption in [11]. 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.

4 Homomorphic authenticated encryption

In this section, we define the homomorphic authenticated encryption (HAE) and its security. In the following, $\mathcal{M}$ and $\mathcal{C}$ are the *plaintext space* and the *ciphertext space*, respectively, $\mathcal{L}$ is the *label space*, and $\mathcal{F}$ is the *admissible function space*.

4.1 Syntax

Labeled programs. First, let us define labeled programs, a concept first introduced in [15].

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 $f : \mathcal{M}^l \to \mathcal{M}$ 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 $P = (f, \tau_1, \dots, \tau_l)$, where $f \in \mathcal{F}$ is an admissible function $f : \mathcal{M}^l \to \mathcal{M}$, and $\tau_i \in \mathcal{L}$ are labels for $i = 1, \dots, l$ for each input of f. The idea is that, if m_i are plaintexts encrypted under the label τ_i, respectively, then the evaluation of the labeled program $P = (f, \tau_1, \dots, \tau_l)$ is $f(m_1, \dots, m_l)$.

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

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

- $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$: given a security parameter λ, $\mathrm{Gen}(1^\lambda)$ outputs a public evaluation key ek and a secret key sk.
- $c \leftarrow \mathrm{Enc}(sk, \tau, m)$: given a secret key sk, a label $\tau \in \mathcal{L}$ and a plaintext $m \in \mathcal{M}$, $\mathrm{Enc}(sk, \tau, m)$ outputs a ciphertext $c \in \mathcal{C}$.
- $\tilde{c} \leftarrow \mathrm{Eval}(ek, f, c_1, \cdots, c_l)$: given an evaluation key ek, an arity-l admissible function $f : \mathcal{M}^l \to \mathcal{M}$ in $\mathcal{F}$ and l ciphertexts $c_1, \cdots, c_l \in \mathcal{C}$, the deterministic algorithm Eval outputs a ciphertext $\tilde{c} \in \mathcal{C}$.
- m or $\perp \leftarrow \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$: given a secret key sk, a labeled program $(f, \tau_1, \cdots, \tau_l)$ and a ciphertext $\hat{c} \in \mathcal{C}$, the deterministic algorithm Dec outputs a message $m \in \mathcal{M}$ or $\perp$.

We assume that evaluation key ek implicitly contains the information about $\mathcal{M}$, $\mathcal{C}$, $\mathcal{L}$, and $\mathcal{F}$. As mentioned above, we assume 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 $\mathrm{Eval}(ek, \dots)$ and $\mathrm{Enc}(sk, \cdot, \cdot)$ are bounded by λ^c for any choice of their input, when $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$. 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
$$m = \mathrm{Dec}(sk, I_\tau, \mathrm{Enc}(sk, \tau, m)),$$
for any $\lambda \in \mathbb{Z}^+$, $\tau \in \mathcal{L}$ and $m \in \mathcal{M}$, when $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$.
- We should have
$$f(m_1, \ldots, m_l) = \mathrm{Dec}(sk, (f, \tau_1, \ldots, \tau_l), c),$$
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 $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$, $c_i \leftarrow \mathrm{Enc}(sk, \tau_i, m_i)$ for $i = 1, \ldots, l$, and $c \leftarrow \mathrm{Eval}(ek, f, c_1, \ldots, c_l)$.

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

4.2 Constant testability

Given a HAE Π, an admissible function $f : \mathcal{M}^l \to \mathcal{M}$ of arity l, a subset I of the index set $\{1, \cdots, l\}$, plaintexts $(m_i)_{i \in I} \in \mathcal{M}^{|I|}$, and their corresponding ciphertexts $(c_i)_{i \in I} \in C^{|I|}$, consider the following functions:
$$\tilde{f}_{(m_i)_{i \in I}} := f(m_i)_{i \in I},$$
$$\tilde{c}_{f,(c_i)_{i \in I}} := \mathrm{Eval}(ek, f, (c_i)_{i \in I}).$$

More explicitly, $\tilde{f}_{(m_i)_{i \in I}}$ is a function from $\mathcal{M}^{l-|I|}$ to $\mathcal{M}$ defined by

$$\tilde{f}_{(m_i)_{i \in I}}(m_j)_{j \notin I} := f(m_1, \cdots, m_l),$$

for any $(m_j)_{j \notin I} \in \mathcal{M}^{l-|I|}$. That is, plaintexts for the index set I are fixed, and plaintexts for the indices in $\{1, \ldots, l\} \setminus I$ are considered as variables. In short, $\tilde{f} = \tilde{f}_{(m_i)_{i \in I}}$ is a partially evaluated admissible function.

Similarly, $\tilde{c}_{f,(c_i)_{i \in I}}$ is a function from $\mathcal{C}^{l-|I|}$ to $\mathcal{C}$ defined by

$$\tilde{c}_{f,(c_i)_{i \in I}}(c_j)_{j \notin I} := \mathrm{Eval}(ek, f, c_1, \cdots, c_l),$$

for any $(c_j)_{j \notin I} \in \mathcal{C}^{l-|I|}$. In particular, $\tilde{f}_{(m_i)_{i \in I}}$ and $\tilde{c}_{f,(c_i)_{i \in I}}$ are constant functions if $I = \{1, \ldots, l\}$.

Sometimes we would like to determine whether such a function $\tilde{f}_{(m_i)_{i \in I}}$ or $\tilde{c}_{f,(c_i)_{i \in I}}$ is constant or not. Therefore we define a property called 'constant testability'. Depending whether we are working on plaintexts or ciphertexts, we define two versions of constant testability accordingly.

Definition 4. *We say that a HAE Π satisfies the* plaintext constant testability (PCT) *if there exists a PPT algorithm that determines if the function $\tilde{f}_{(m_i)_{i \in I}}$ is constant or not with overwhelming probability, for any admissible function $f : \mathcal{M}^l \to \mathcal{M}$ 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 5. *We say that a HAE Π 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 Π.*Gen*, any admissible function $f : \mathcal{M}^l \to \mathcal{M}$ 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 paper, as shown in Theorem 8.

Moreover, we show that if Π is a HAE which does not necessarily satisfy CCT, then there is a simple generic transformation which turns it into another HAE Π' which satisfies CCT, while preserving original security properties satisfied by Π. This will be shown in Theorems 5, 6 and 7.

Therefore, without loss of generality, we assume the CCT property to be an additional requirement for a HAE to satisfy.

4.3 Privacy

Here we define security notions of privacy for HAE. First, let us define IND-CPA.

Indistinguishability under chosen plaintext attack. Our definition of privacy for HAE is a homomorphic version of the IND-CPA security. We use the following security game $\text{IND-CPA}_{\Pi,A}$ between the challenger and the adversary A, which is a natural adaptation of the corresponding security game of the symmetric-key encryption.

The main difference is that, in case of a HAE, at most one message $m \in \mathcal{M}$ can be encrypted to produce the corresponding ciphertext $c \leftarrow \mathrm{Enc}(sk, \tau, m)$ under each label $\tau \in \mathcal{L}$. In other words, a label used once in an encryption cannot be used again. To prevent from generating two ciphertexts with respect to the same label, an encryption history S will be kept in the game IND-CPA. If a label τ is not in the history S, then we say that the label τ is *new*.

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

Initialization. A key pair $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$ is generated, a set S is initialized as the empty set $\emptyset$. Then ek is given to A.

Queries. A may make encryption queries adaptively. For each encryption query (τ, m) of A, if $(\tau, \cdot, \cdot) \notin S$ (that is, $(\tau, m, c) \notin S$ for any $m \in \mathcal{M}$, $c \in \mathcal{C}$), then the challenger returns the answer $c \leftarrow \mathrm{Enc}(sk, \tau, m)$ to A and updates $S \leftarrow S \cup \{(\tau, m, c)\}$. Otherwise, the query is rejected.

Challenge. A outputs the challenge (τ^*, m_0^*, m_1^*). If $(\tau^*, \cdot, \cdot) \notin S$, then the challenger flips a coin $b \xleftarrow{\$} \{0, 1\}$, gives the corresponding challenge ciphertext $c^* \leftarrow \mathrm{Enc}(sk, \tau^*, m_b^*)$ to A and updates $S \leftarrow S \cup \{(\tau^*, m_b^*, c^*)\}$. Otherwise, the challenge is rejected.

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 Π is defined as

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

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

Indistinguishability under chosen ciphertext attack. We also consider a homomorphic version of the IND-CCA security of symmetric-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 $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Eval}, \mathrm{Dec})$ is defined using the following security game $\text{IND-CCA}_{\Pi,A}$, which is also a natural extension of the security game IND-CCA of a symmetric-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 $((f, \tau_1, \cdots, \tau_l), \hat{c})$ 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.

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

Initialization. A key pair $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$ is generated and a set S is initialized to be the empty set $\emptyset$. Then ek is given to the adversary A.

Queries. A may make encryption queries and decryption queries adaptively. For each encryption query (τ, m) of A, if $(\tau, \cdot, \cdot) \notin S$ then the query is replied with an answer $c \leftarrow \mathrm{Enc}(sk, \tau, m)$ and S is updated as $S \leftarrow S \cup \{(\tau, m, c)\}$. Otherwise, the encryption query is rejected. Each decryption query $((f, \tau_0, \cdots, \tau_l), \hat{c})$ of A is answered by $\mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$.

Challenge. A outputs the challenge tuple (τ^*, m_0^*, m_1^*). If $(\tau^*, \cdot, \cdot) \notin S$, then a coin $b \xleftarrow{\$} \{0, 1\}$ is flipped and the challenge ciphertext $c^* \leftarrow \mathrm{Enc}(sk, \tau^*, m_b^*)$ is given to A, and S is updated as $S \leftarrow S \cup \{(\tau^*, m_b^*, c^*)\}$. Otherwise, the challenge is rejected.

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 Π is defined as

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

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

4.4 Authenticity

Unforgeability under chosen plaintext attack. Our authenticity definition for HAE is an adaptation of the definition given by Catalano and Fiore [9] for homomorphic MACs.

First, we define the forgery of an adversary. Let $((f, \tau_1, \cdots, \tau_l), \hat{c})$ be a forgery attempt of an adversary, and let S be the encryption history maintained in the security game. We say that it is a *forgery*, if the following holds:

1. It is valid, that is, $\perp \neq \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$ and,
2. One of the following holds:
   - Type 1 forgery: $\tilde{f} = f(m_i)_{i \in I}$ is not constant, or,
   - Type 2 forgery: $\tilde{f} = f(m_i)_{i \in I}$ is constant but $\tilde{f} \neq \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$,

   where

$$I = \{i \in \{1, \cdots, l\} \mid (\tau_i, m_i, c) \in S \text{ for some } m_i \in \mathcal{M}, c \in \mathcal{C}\},$$
$$\tilde{f} : \mathcal{M}^{l-|I|} \to \mathcal{M} \quad \text{defined by } \tilde{f}(m_j)_{j \notin I} := f(m_1, \ldots, m_l).$$

We define the *unforgeability under chosen plaintext attack* (UF-CPA) of a HAE Π using the following security game $\text{UF-CPA}_{\Pi,A}$.

$\text{UF-CPA}_{\Pi,A}(1^\lambda)$:

Initialization. A key pair $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$ is generated and a set S is initialized to be the empty set $\emptyset$. Then ek is given to the adversary A.

Queries. A may make encryption queries adaptively. For each encryption query (τ, m) of A, if $(\tau, \cdot, \cdot) \notin S$, then the query is replied with the answer $c \leftarrow \mathrm{Enc}(sk, \tau, m)$, and S is updated with $S \leftarrow S \cup \{(\tau, m, c)\}$. Otherwise, the query is rejected.

Finalization. A outputs a forgery attempt $((f, \tau_1, \cdots, \tau_l), \hat{c})$. The challenger returns 0 if $\perp = \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$. Otherwise, I is initialized as $\emptyset$, and for each $i = 1, \ldots, l$, if $(\tau_i, m, c) \in S$ for some $m \in \mathcal{M}, c \in \mathcal{C}$, then let $I \leftarrow I \cup \{i\}$ and $m_i \leftarrow m$. And then let $\tilde{f} = f(m_i)_{i \in I}$. If $\tilde{f}$ is not constant, or if $\tilde{f}$ is constant but $\tilde{f} \neq \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$, then the challenger returns 1. Otherwise, 0 is returned.

The advantage of A in the game UF-CPA for the scheme Π is defined as

$$\mathbf{Adv}^{\text{UF-CPA}}_{\Pi,A}(\lambda) := \Pr\left[\text{UF-CPA}_{\Pi,A}(1^\lambda) = 1\right].$$

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

Unforgeability under chosen ciphertext attack. It is also natural to consider a stronger variant of unforgeability, in which an adversary is allowed to make decryption queries as well as encryption queries. We call this variant UF-CCA. The only difference of UF-CCA from UF-CPA is the **Queries** phase, which is given below.

Queries. A may make encryption queries and decryption queries adaptively. For each encryption query (τ, m) of A, if $(\tau, \cdot, \cdot) \notin S$, then the query is replied with the answer $c \leftarrow \mathrm{Enc}(sk, \tau, m)$, and S is updated with $S \leftarrow S \cup \{(\tau, m, c)\}$. Otherwise, the query is rejected. Each decryption query $((f, \tau_0, \cdots, \tau_l), \hat{c})$ of A is answered by $\mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$.

The advantage of A in the game UF-CCA for the scheme Π is defined as

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

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

Strong unforgeability under chosen plaintext attack. Sometimes it is useful to consider stronger definition of authenticity. So let us define *strong unforgeability* for HAE.

We first define strong forgery of an adversary. Let $((f, \tau_1, \cdots, \tau_l), \hat{c})$ be a forgery attempt of an adversary, and let S be the encryption history maintained in the security game. We say that it is a *strong forgery*, if the following holds:

1. It is valid, that is, $\perp \neq \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$ and,
2. One of the following holds:
   - Type 1 strong forgery: $\tilde{c} = \mathrm{Eval}(ek, f, (c_i)_{i \in I})$ is not constant, or,
   - Type 2 strong forgery: $\tilde{c} = \mathrm{Eval}(ek, f, (c_i)_{i \in I})$ is constant but $\tilde{c} \neq \hat{c}$,

   where

$$I = \{i \in \{1, \cdots, l\} \mid (\tau_i, m, c_i) \in S \text{ for some } m \in \mathcal{M},\ c_i \in \mathcal{C}\},$$
$$\tilde{c} : \mathcal{C}^{l-|I|} \to \mathcal{C} \quad \text{defined by } \tilde{c}(c_j)_{j \notin I} := \mathrm{Eval}(ek, f, c_1, \cdots, c_l).$$

Now, we define the *strong unforgeability under chosen plaintext attack* (SUF-CPA) of a HAE Π using the following security game $\text{SUF-CPA}_{\Pi,A}$.

$\text{SUF-CPA}_{\Pi,A}(1^\lambda)$:

Initialization. A key pair $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$ is generated and a set S is initialized to be the empty set $\emptyset$. Then ek is given to the adversary A

Queries. A may make encryption queries adaptively. For each encryption query (τ, m) of A, if $(\tau, \cdot, \cdot) \notin S$, then the query is replied with the answer $c \leftarrow \mathrm{Enc}(sk, \tau, m)$, and S is updated with $S \leftarrow S \cup \{(\tau, m, c)\}$. Otherwise, the query is rejected.

Finalization. A outputs a forgery attempt $((f, \tau_1, \cdots, \tau_l), \hat{c})$. The challenger returns 0 if $\perp = \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$. Otherwise, I is initialized as $\emptyset$, and for each $i = 1, \ldots, l$, if $(\tau_i, m, c) \in S$ for some m, c, then let $I \leftarrow I \cup \{i\}$ and $c_i \leftarrow c$. And then let $\tilde{c} = \mathrm{Eval}(ek, f, (c_i)_{i \in I})$. If $\tilde{c}$ is not constant, or if $\tilde{c}$ is constant but $\tilde{c} \neq \hat{c}$, then the challenger returns 1. Otherwise, 0 is returned.

The advantage of A in the game SUF-CPA for the scheme Π is defined as

$$\mathbf{Adv}^{\text{SUF-CPA}}_{\Pi,A}(\lambda) := \Pr\left[\text{SUF-CPA}_{\Pi,A}(1^\lambda) = 1\right].$$

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

Strong unforgeability under chosen ciphertext attack. Also for strong unforgeability, we consider security against chosen ciphertext attacks, which we call SUF-CCA. Again, the only difference of SUF-CCA from SUF-CPA is the **Queries** phase, which is given below.

Queries. A may make encryption queries and decryption queries adaptively. For each encryption query (τ, m) of A, if $(\tau, \cdot, \cdot) \notin S$, then the query is replied with the answer $c \leftarrow \mathrm{Enc}(sk, \tau, m)$, and S is updated with $S \leftarrow S \cup \{(\tau, m, c)\}$. Otherwise, the query is rejected. Each decryption query $((f, \tau_0, \cdots, \tau_l), \hat{c})$ of A is answered by $\mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$.

The advantage of A in the game SUF-CCA for the scheme Π is defined as

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

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

4.5 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 1. *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 2. *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 $(m_j^1)_{j \notin I}$ and $(m_j^2)_{j \notin I}$ such that $\tilde{f}(m_j^1)_{j \notin I} \neq \tilde{f}(m_j^2)_{j \notin I}$. Then there exists two distinct tuples $(c_j^1)_{j \notin I}$ and $(c_j^2)_{j \notin I}$ such that $m_j^1 = \mathrm{Dec}(sk, I_{\tau_j}, c_j^1)$ and $m_j^2 = \mathrm{Dec}(sk, I_{\tau_j}, c_j^2)$ for each $j \notin I$. Then we have $\tilde{c}(c_j^1)_{j \notin I} \neq \tilde{c}(c_j^2)_{j \notin I}$ by the correctness property, which shows that $\tilde{c} = \mathrm{Eval}(ek, f, (c_i)_{i \in I})$ 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 $\tilde{c}$ are constants and $\tilde{f} \neq \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$. But $\tilde{f} = \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \tilde{c})$, again by the correctness. This means that $\hat{c} \neq \tilde{c}$, 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 3. *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 $\tilde{c} = \mathrm{Eval}(ek, f, (c_i)_{i \in I})$ and $\tilde{f} = f(m_i)_{i \in I}$. If $\tilde{c}$ is constant and $\tilde{c} = \hat{c}$, then return $\tilde{f}$. Otherwise, return $\perp$.

In particular, SUF-CCA is equal to SUF-CCA^0. So,

$$\mathbf{Adv}^{\text{SUF-CCA}^0}_{\Pi,A}(\lambda) = \mathbf{Adv}^{\text{SUF-CCA}}_{\Pi,A}(\lambda).$$

Moreover, since the decryption simulation does not use any secret information and is efficiently computable by the CCT property in SUF-CCA^q, 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

$$\mathbf{Adv}^{\text{SUF-CCA}^q}_{\Pi,A}(\lambda) = \mathbf{Adv}^{\text{SUF-CPA}}_{\Pi,A'}(\lambda).$$

For each $i \in \{1, \ldots, q\}$, the difference between $\mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}^{i-1}}(\lambda)$ and $\mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}^{i}}(\lambda)$ is bounded by the probability that the decryption simulation on the ith 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

$$\begin{aligned}
&\text{decryption simulation fails} \\
\iff &\tilde{c} \text{ is constant and } \tilde{c} = \hat{c}, \text{ but } \perp = \mathrm{Dec}(sk, (f, \tau_1, \ldots, \tau_l), \hat{c}), \text{ or,} \\
&\tilde{c} \text{ is nonconstant, or } \tilde{c} \text{ is constant but } \tilde{c} \neq \hat{c}, \text{ but } \perp \neq \mathrm{Dec}(sk, (f, \tau_1, \ldots, \tau_l), \hat{c}),
\end{aligned}$$

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

$$\begin{aligned}
&\text{decryption simulation fails} \\
\iff &\tilde{c} \text{ is nonconstant, or } \tilde{c} \text{ is constant but } \tilde{c} \neq \hat{c}, \text{ but } \perp \neq \mathrm{Dec}(sk, (f, \tau_1, \ldots, \tau_l), \hat{c}) \\
\iff &((f, \tau_1, \ldots, \tau_l), \hat{c}) \text{ is a strong forgery.}
\end{aligned}$$

Hence, we may construct a PPT adversary A'' for the game SUF-CPA using the ith decryption query made by A. Specifically, A'' runs the adversary A until it makes the ith 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 ith decryption query of A as its own forgery attempt.

So,

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

Therefore,

$$\begin{aligned}
\mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}}(\lambda) &\leq \mathbf{Adv}_{\Pi,A'}^{\text{SUF-CPA}}(\lambda) + \left| \mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}}(\lambda) - \mathbf{Adv}_{\Pi,A'}^{\text{SUF-CPA}}(\lambda) \right| \\
&= \mathit{negl}(\lambda) + \left| \mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}^{0}}(\lambda) - \mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}^{q}}(\lambda) \right| \\
&\leq \mathit{negl}(\lambda) + \sum_{i=1}^{q} \left| \mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}^{i}}(\lambda) - \mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}^{i-1}}(\lambda) \right| \\
&\leq \mathit{negl}(\lambda) + q \cdot \mathit{negl}(\lambda) \\
&= \mathit{negl}(\lambda).
\end{aligned}$$

So $\mathbf{Adv}_{\Pi,A}^{\text{SUF-CCA}}(\lambda)$ is also negiligible for any PPT adversary A and therefore Π is SUF-CCA. □

Theorem 4. *IND-CPA and SUF-CPA together imply IND-CCA.*

Proof. Proof of this theorem is similar to that of Theorem 3; 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 $\text{IND-CCA}^{\mathsf{b},i}$ 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 3.

For each $i \in \{0, \ldots, q_a\}$, define IND-CCA$^{\mathsf{a},i}$ 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$^{\mathsf{b},0}$ = IND-CCA and IND-CCA$^{\mathsf{b},q_b}$ = IND-CCA$^{\mathsf{a},0}$. So,

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

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$^{\mathsf{a},q_a}$ 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 (τ, 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 3.

Challenge. A outputs the challenge tuple (τ^*, m_0^*, m_1^*). If $(\tau^*, \cdot, \cdot) \notin S$, then A' outputs the same challenge tuple, and receives the challenge ciphertext $c^* \leftarrow \text{Enc}(sk, \tau^*, m_b^*)$. 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 IND-CCA$^{\mathsf{a},q_a}$ 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 IND-CCA$^{\mathsf{a},q_a}$, S is updated by $S \leftarrow S \cup \{(\tau^*, m_b^*, c^*)\}$. We need to show that, despite this the simulation of A' for the game IND-CCA$^{\mathsf{a},q_a}$ 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 IND-CCA$^{\mathsf{a},q_a}$ and in the simulation of A'.

In the game IND-CCA$^{\mathsf{a},q_a}$, m_1^* is encrypted under the label τ^* to produce the ciphertext c^*. So, in the game IND-CCA$^{\mathsf{a},q_a}$, $\tilde{c} = \text{Eval}(ek, f, (c_i)_{i \in I})$ and $\tilde{f} = f(m_i)_{i \in I}$ are computed, and the decryption query is answered with $\tilde{f}$ if and only if $\tilde{c}$ is constantly equal to $\hat{c} \in \mathcal{C}$. And in the computation of $\tilde{f}$, m_1^* is used for the i^*th plaintext input. To emphasize this fact, let us denote this $\tilde{f}$ as $\tilde{f}_1$, meaning that m_1^* was used to produce this plaintext.

In the simulation of A', still m_1^* is encrypted under the label τ^* to produce the ciphertext c^*, and $\tilde{c} = \text{Eval}(ek, f, (c_i)_{i \in I})$ and $\tilde{f} = f(m_i)_{i \in I}$ are computed, and the decryption query is answered with $\tilde{f}$ if and only if $\tilde{c}$ is constantly equal to $\hat{c} \in \mathcal{C}$. 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 $\perp$ if and only if $\tilde{c}$ is nonconstant, or $\tilde{c}$ is constant but not equal to $\hat{c}$. Meanwhile, when $\tilde{c}$ is constantly equal to $\hat{c}$, then the game IND-CCA$^{\mathsf{a},q_a}$ will output $\tilde{f}_1$, 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 IND-CCA. Hence, we have $\tilde{f}_0 = \tilde{f}_1$. This shows that A' correctly simulates the game IND-CCA$^{\mathsf{a},q_a}$, and we conclude that

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

Now consider the difference of each consecutive two games. Again, for each $i \in \{1,\ldots,q_b\}$, the difference between $\mathbf{Adv}^{\text{IND-CCA}^{\mathsf{b},i-1}}_{\Pi,A}(\lambda)$ and $\mathbf{Adv}^{\text{SUF-CCA}^{\mathsf{b},i}}_{\Pi,A}(\lambda)$ is not greater than the probability that the decryption simulation on the ith decryption query made by A before Challenge fails, and we may use this to construct an adversary A'' for the game SUF-CPA just like in Theorem 3. Note that A'' aborts the running of A before it has any chance to output the challenge tuple.

So,

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

Similarly, for each $i \in \{1,\ldots,q_a\}$, the difference between $\mathbf{Adv}^{\text{IND-CCA}^{\mathsf{a},i-1}}_{\Pi,A}(\lambda)$ and $\mathbf{Adv}^{\text{SUF-CCA}^{\mathsf{a},i}}_{\Pi,A}(\lambda)$ is not greater than the probability that the decryption simulation on the ith 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 3. In this case, the challenge tuple output of A is handled by A''; A'' flips the coin $b \xleftarrow{\$} \{0,1\}$, and obtains the challenge ciphertext via its encryption query (τ^*, m^*_b). Since A'' knows the coin b, the correct encryption history is maintained: $(\tau^*, m^*_b, c^*) \in S$.

So,

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

Hence,

$$\begin{aligned}
\mathbf{Adv}^{\text{IND-CCA}}_{\Pi,A}(\lambda) &\leq \mathbf{Adv}^{\text{IND-CPA}}_{\Pi,A'}(\lambda) + \left| \mathbf{Adv}^{\text{IND-CCA}}_{\Pi,A}(\lambda) - \mathbf{Adv}^{\text{IND-CPA}}_{\Pi,A'}(\lambda) \right| \\
&= negl(\lambda) + \left| \mathbf{Adv}^{\text{IND-CCA}^{\mathsf{b},0}}_{\Pi,A}(\lambda) - \mathbf{Adv}^{\text{IND-CCA}^{\mathsf{a},q_a}}_{\Pi,A}(\lambda) \right| \\
&\leq negl(\lambda) + \sum_{i=1}^{q_b} \left| \mathbf{Adv}^{\text{IND-CCA}^{\mathsf{b},i}}_{\Pi,A}(\lambda) - \mathbf{Adv}^{\text{IND-CCA}^{\mathsf{b},i-1}}_{\Pi,A}(\lambda) \right| \\
&\qquad + \sum_{i=1}^{q_a} \left| \mathbf{Adv}^{\text{IND-CCA}^{\mathsf{a},i}}_{\Pi,A}(\lambda) - \mathbf{Adv}^{\text{IND-CCA}^{\mathsf{a},i-1}}_{\Pi,A}(\lambda) \right| \\
&\leq negl(\lambda) + (q_b + q_a) \cdot negl(\lambda) = negl(\lambda).
\end{aligned}$$

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

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, we show that our scheme is IND-CPA and SUF-CPA.

Generic transformation for ciphertext constant testability Suppose that Π is a HAE which is *not* necessarily ciphertext constant testable. We describe a generic construction that transforms a HAE Π into another HAE Π' satisfying CCT while preserving IND-CPA or SUF-CPA of the original scheme Π. Our construction is based on the Merkle hash tree technique used by Gennaro and Wichs [15]. For concreteness, here we assume that Π represents admissible functions as circuits[1]. 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.

[1] In general case, the construction is even simpler: instead of the Merkle hash tree, we may simply use a hash function. The hash tree is only needed to support composition of circuit representation of functions.

Let $H : \{0,1\}^* \to \{0,1\}^n$ be a collision-resistant hash function. We define the hash tree as in [15]. The hash tree f^H of a 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:

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

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

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

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

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

Remark 1. In this paper, 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 $\pi_i : \mathcal{M}^l \to \mathcal{M}$, for example. But, under this definition, some input wires may be unused.

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

Scheme $\Pi' = (\mathrm{Gen}', \mathrm{Enc}', \mathrm{Eval}', \mathrm{Dec}')$:

- $(ek', sk') \leftarrow \mathrm{Gen}'(1^\lambda)$: Generate keys $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$, $k \leftarrow \{0,1\}^\lambda$ and $H \leftarrow \mathcal{H}$. Return $ek' = (ek, H)$ and $sk' = (sk, k)$.
- $c' \leftarrow \mathrm{Enc}'(sk', \tau, m)$: Let $h = H(F_k(\tau))$ and $c \leftarrow \mathrm{Enc}(sk, \tau, m)$. Return (h, c).
- $\tilde{c}' \leftarrow \mathrm{Eval}'(ek', f, c'_1, \cdots, c'_l)$: Let $f : \mathcal{M}^l \to \mathcal{M}$ be a circuit. For each $i = 1, \cdots, l$, parse $c'_i = (h_i, c_i)$. Let $\tilde{h} = f^H(h_1, \cdots, h_l)$ and $\tilde{c} \leftarrow \mathrm{Eval}(ek, f, c_1, \cdots, c_l)$. Return $(\tilde{h}, \tilde{c})$.
- $m \leftarrow \mathrm{Dec}'(sk', (f, \tau_1, \cdots, \tau_l), \tilde{c}')$: Let $f : \mathcal{M}^l \to \mathcal{M}$ be a circuit. Parse $\tilde{c}' = (\tilde{h}, \tilde{c})$. For each $i = 1, \cdots, l$, let $h_i = H(F_k(\tau_i))$. If $\tilde{h} = f^H(h_1, \cdots, h_l)$, then return $\mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \tilde{c})$. Otherwise, return $\perp$.

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

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

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

Theorem 5. *The HAE scheme Π' satisfies CCT.*

Proof. As in p. 6, let $f : \mathcal{M}^l \to \mathcal{M}$ be an arity-l admissible function, I a subset of the index set $\{1, \ldots, l\}$, and $(m_i)_{i \in I} \in \mathcal{M}^{|I|}$ and $(c'_i = (h_i, c_i))_{i \in I} \in (\{0,1\}^\lambda \times \mathcal{C})^{|I|}$ some plaintexts and their corresponding ciphertexts.

Consider $\tilde{c}' : (\{0,1\}^\lambda \times \mathcal{C})^{l-|I|} \to \{0,1\}^\lambda \times \mathcal{C}$ defined as

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

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

If $I = \{1, \ldots, l\}$, then $\tilde{c}'$ is clearly constant. Now, suppose that $I \neq \{1, \ldots, l\}$. Consider the case when the ith input of f is unused for all indices $i \notin I$. In that case, again clearly $\tilde{c}'$ is constant.

Finally, consider the case that the ith input of f is actually used for at least one index $i \notin I$. Since the underlying hash function H is collision-resistant, the hash tree $f^H(h_1, \ldots, h_l)$ 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, Π' satisfies CCT. □

Theorem 6. *If Π is IND-CPA, then Π' is also IND-CPA.*

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

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

This simulation does work because τ^* 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

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

□

Theorem 7. *If Π is SUF-CPA, then Π' is also SUF-CPA.*

Proof. Let A' be a PPT adversary engaged in the security game $\text{SUF-CPA}_{\Pi', A'}$. Using A', we construct a PPT adversary A for the security game $\text{SUF-CPA}_{\Pi, A}$ which simulates the game $\text{SUF-CPA}_{\Pi', A'}$ 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 (τ, m), if $(\tau, \cdot, \cdot) \notin S$, then A makes the same encryption query. Receiving the answer $c \leftarrow \mathrm{Enc}(sk, \tau, m)$, 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 Π', $((f, \tau_1, \cdots, \tau_l), \hat{c})$ is also valid in Π. 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 $r_j \xleftarrow{\$} \{0,1\}^\lambda$. Due to the collision resistance of H, the probability

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

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 $\tilde{c}'$ and $\tilde{c}$ are constants, where

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

with

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

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

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

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

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

Hence Π' is SUF-CPA. □

5 Construction

In this section, we describe our HAE Π and show that it satisfies correctness and CCT. All of the parameters $\rho, \eta, \gamma, \bar{d}$ of the scheme are polynomials in λ. 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 $F_k : \{0,1\}^\lambda \to \mathbb{Z}_{q_0}$ for each $k \in \{0,1\}^\lambda$. The message space and the ciphertext space of our scheme is $\mathbb{Z}_Q$ and $\mathbb{Z}_{y_0}$, respectively, and the label space is $\{0,1\}^\lambda$. To represent admissible functions we use arithmetic circuits, that is, circuits consisting of + gates and × gates. Such a circuit f of arity l determines a polynomial $f : \mathbb{Z}^l \to \mathbb{Z}$ with integral coefficients. We use such a circuit to compute function values of plaintext inputs in $\mathbb{Z}_Q$, and also to homomorphically evaluate ciphertexts in $\mathbb{Z}_{y_0}$. The precise description of the admissible function space will be given in the next, together with discussions on the correctness property.

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

- $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda, Q)$: Given security parameter λ and any modulus $Q \in [2, 2^\lambda]$, choose $p \xleftarrow{\$} [2^{\eta-1}, 2^\eta) \cap \mathrm{PRIME}$ and $q_0 \xleftarrow{\$} [0, \frac{2^\gamma}{p}) \cap \mathrm{ROUGH}(2^\lambda)$. Let $y_0 = pq_0$. Choose a PRF key $k \leftarrow \{0,1\}^\lambda$. Return (ek, sk), where $ek = (Q, y_0)$, and $sk = (p, q_0, Q, k)$.

- $c \leftarrow \mathrm{Enc}(sk, \tau, m)$: Given the secret key sk, a label $\tau \in \{0,1\}^\lambda$ and a plaintext $m \in \mathbb{Z}_Q$, choose $r \xleftarrow{\$} (-2^\rho, 2^\rho)$. Let $a = rQ + m$ and $b = F_k(\tau)$. Return $c = \mathrm{CRT}_{(p,q_0)}(a, b)$.
- $\tilde{c} \leftarrow \mathrm{Eval}(ek, f, c_1, \cdots, c_l)$: Given the evaluation key ek, an arithmetic circuit f of arity l and ciphertexts $c_1, \cdots, c_l$, return $f(c_1, \cdots, c_l) \bmod y_0$
- $m \leftarrow \mathrm{Dec}(sk, (f, \tau_1, \cdots, \tau_l), \hat{c})$: For $i = 1$ to l, compute $b_i \leftarrow F_k(\tau_i)$ and $b = f(b_1, \cdots, b_l) \bmod q_0$. Return $m = (\hat{c} \bmod p) \bmod Q$, if $b = \hat{c} \bmod q_0$. Otherwise, return $\perp$.

Correctness To show the correctness of the scheme, let $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda, Q)$ for any $\lambda \in \mathbb{Z}^+$ and any modulus $Q \in [2, 2^\lambda]$. Let $c_i \leftarrow \mathrm{Enc}(sk, \tau_i, m_i)$ for each $i = 1, \cdots, l$. And $\tilde{c} \leftarrow \mathrm{Eval}(ek, f, c_1, \cdots, c_l)$ 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

$$\begin{aligned}
\tilde{c} \bmod p &= (f(c_1, \cdots, c_l) \bmod y_0) \bmod p \\
&= f(c_1, \cdots, c_l) \bmod p \\
&= f(c_1 \bmod p, \cdots, c_l \bmod p) \bmod p \\
&= f(r_1 Q + m_1, \cdots, r_l Q + m_l) \bmod p \\
&= f(r_1 Q + m_1, \cdots, r_l Q + m_l)
\end{aligned}$$

The last equality in the above equations holds if

$$| f(r_1 Q + m_1, \cdots, r_l Q + m_l) | \leq \frac{p}{2}.$$

And so, in this case,

$$\begin{aligned}
(\tilde{c} \bmod p) \bmod Q &= f(r_1 Q + m_1, \cdots, r_l Q + m_l) \bmod Q \\
&= f(m_1, \cdots, m_l) \bmod Q
\end{aligned}$$

Since $| f(r_1 Q + m_1, \cdots, r_l Q + m_l) | \leq \|f\|_1 \cdot 2^{d(\rho+\lambda)}$ and $2^{\eta-2} \leq p/2$, the correctness is guaranteed if

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

equally,

$$d \leq \frac{\eta - 2 - \lg \|f\|_1}{\rho + \lambda}.$$

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

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

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

Parameter selection. In the scheme, the parameters ρ, η, γ are given as follows.

- $\rho = \omega(\lg \lambda)$ to resist the brute force attack on the EF-AGCD problem.
- $\eta \geq \bar{d}(\rho + \lambda + 1) + 2$ for the upper bound $\bar{d}$ on degrees of admissible functions. This is a consequence of discussions about correctness property. If we choose $\bar{d} = \mathcal{O}(\lambda)$ and $\rho = \mathcal{O}(\lambda)$, then $\eta = \mathcal{O}(\lambda^2)$.

– $\gamma = \eta^2\omega(\lg\lambda)$ to resist known attacks on the EF-AGCD problem as explained in [23, 10] . If we choose $\eta = \mathcal{O}(\lambda^2)$, then $\gamma = \mathcal{O}(\lambda^5)$

Theorem 8. *The scheme Π satisfies CCT.*

Proof. Let ek be an evaluation key generated by $\mathrm{Gen}(1^\lambda, Q)$ 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 $\tilde{c} = \mathrm{Eval}(ek, f, (c_i)_{i\in I})$ is constant or not with overwhelming probability, as follows.

procedure ALG-CCT(ek, f, $(c_i)_{i\in I}$):
  if $I = \{1, \cdots, l\}$ **then**
    return 1
  else
    $(c_j^0)_{j\notin I}$, $(c_j^1)_{j\notin I} \xleftarrow{\$} (\mathbb{Z}_{y_0})^{l-|I|}$
    if $\tilde{c}(c_j^0)_{j\notin I} \equiv \tilde{c}(c_j^1)_{j\notin I} \bmod y_0$ **then**
      return 1
    else
      return 0

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

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

In case $I = \{1, \cdots, l\}$, $\tilde{c}$ is clearly constant and the algorithm outputs 1 correctly. In case $I \subsetneq \{1, \cdots, l\}$, consider the function $\tilde{c}' := \tilde{c} - \tilde{c}(c_j^0)_{j\notin I} \bmod y_0$ 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 $\tilde{c}'(c_j^1)_{j\notin I} = \tilde{c}(c_j^1)_{j\notin I} - \tilde{c}(c_j^0)_{j\notin I} \equiv 0 \bmod y_0$ for any $(c_j^1)_{j\notin I} \in (\mathbb{Z}_{y_0})^{l-|I|}$. So, $\tilde{c}(c_j^0)_{j\notin I} \equiv \tilde{c}(c_j^1)_{j\notin I} \bmod y_0$ 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 $\tilde{c}(c_j^0)_{j\notin I} \equiv \tilde{c}(c_j^1)_{j\notin I} \bmod y_0$, that is, $\tilde{c}'(c_j^1)_{j\notin I} \equiv 0 \bmod y_0$. This is the only case that the algorithm outputs an incorrect answer. So the error probability of the algorithm is

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

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^λ-rough random integer y_0 is square-free with overwhelming probability and there exists some prime factor p' of y_0 such that $\tilde{c}' \bmod p'$ is not constantly zero and $p' \geq 2^\lambda$. From these, we have

$$\begin{aligned}
&\Pr\left[\tilde{c}'(c_j^1)_{j\notin I} \equiv 0 \bmod y_0 \mid (c_j^1)_{j\notin I} \xleftarrow{\$} (\mathbb{Z}_{y_0})^{l-|I|}\right] \\
&\leq \Pr\left[\tilde{c}'(c_j^1)_{j\notin I} \equiv 0 \bmod p' \mid (c_j^1)_{j\notin I} \xleftarrow{\$} (\mathbb{Z}_{y_0})^{l-|I|}\right] \\
&= \Pr\left[\tilde{c}'(c_j^1)_{j\notin I} \equiv 0 \bmod p' \mid (c_j^1)_{j\notin I} \xleftarrow{\$} (\mathbb{Z}_{p'})^{l-|I|}\right] \\
&\leq \frac{\deg f}{p'} \leq \frac{\bar{d}}{2^\lambda} = negl(\lambda)
\end{aligned}$$

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. □

6 Security

In this section, we prove our HAE scheme satisfies both IND-CPA and SUF-CPA. From this, we conclude that Π is actually IND-CCA and SUF-CCA by Theorem 3 and Theorem 4. For simplicity, we consider the scheme Π 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 Π implies that of the real scheme.

6.1 Privacy

The privacy of the scheme Π 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 [11]. So, this theorem actually shows that Π is IND-CPA under the computational (ρ,η,γ)-EF-AGCD assumption.

Theorem 9. *The scheme Π is IND-CPA under the decisional (ρ,η,γ)-EF-AGCD assumption.*

Proof. Suppose there exists a PPT adversary A for the IND-CPA security game of the scheme Π 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 ϵ. Then, we can construct a PPT distinguisher D for the decisional (ρ,η,γ)-EF-AGCD problem, by simulating the game $\text{IND-CPA}_{\Pi,A}$ as follows.

Distinguisher $D(\rho,\eta,\gamma,y_0,\mathcal{D}(p,q_0,\rho),z)$:

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 $x \leftarrow \mathcal{D}(p,q_0,\rho)$, compute $c := (xQ + m) \bmod y_0$, return c to A, and update S by $S \leftarrow S \cup \{(\tau, m, c)\}$. Otherwise, reject the query.

Challenge. For the challenge (τ^*, m_0^*, m_1^*) of A, if $(\tau^*,\cdot,\cdot) \notin S$, then flip a coin $b \xleftarrow{\$} \{0,1\}$, compute the challenge ciphertext $c^* := (zQ + m_b^*) \bmod y_0$, 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^λ and Q is not greater than 2^λ. Let us consider the distribution of c produced by D in the Queries phase. Since $c = xQ + m \bmod y_0$ for $x \leftarrow \mathcal{D}(p,q_0,\rho)$, we have $c = pqQ + rQ + m \bmod y_0$ for some $q \xleftarrow{\$} [0, q_0)$ and $r \xleftarrow{\$} (-2^\rho, 2^\rho)$. Therefore, $c \equiv rQ + m \pmod p$. Also, since q is uniform random on $[0, q_0)$, $c \bmod q_0$ 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 $\mathcal{D}(p,q_0,\rho)$.

Now, consider the distribution of c^* in the Challenge phase. If $z \leftarrow \mathcal{D}(p,q_0,\rho)$, then the distribution of c^* is identical to that of original security game by the same reason as above. But if $z \xleftarrow{\$} \mathbb{Z}_{y_0}$, 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

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

and

$$\Pr\left[D(\rho,\eta,\gamma,y_0,\mathcal{D}(p,q_0,\rho),z)=1 \mid z \xleftarrow{\$} \mathbb{Z}_{y_0}\right]=\frac{1}{2}$$

Therefore, the advantage of D is at least non-negligible ϵ, and this completes the proof. □

6.2 Authenticity

The authenticity of the scheme Π is stated in the following theorem.

Theorem 10. *If the (ρ,η,γ)-EF-AGCD assumption holds, then the scheme Π is SUF-CPA.*

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

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

for some modulus $Q\in[2,2^\lambda]$ and some non-negligible function ϵ. Then, we can construct a PPT algorithm B for the (ρ,η,γ)-EF-AGCD problem, by simulating the game $\text{SUF-CPA}_{\Pi,A}$ as follows.

Algorithm $B(\rho,\eta,\gamma,y_0,\mathcal{D}(p,q_0,\rho))$:

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 $x\leftarrow\mathcal{D}(p,q_0,\rho)$, compute $c:=(xQ+m) \bmod y_0$, 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 $c_i\xleftarrow{\$}\mathbb{Z}_{y_0}$. And then, compute $\tilde{c}=f(c_1,\cdots,c_l) \bmod y_0$. Output $y_0/\gcd(y_0,\tilde{c}-\hat{c})$.

For the same reason as in Theorem 9, 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, $\tilde{c}=f(c_i)_{i\in I}$ is not constant, where I is the set of indices i such that τ_i is not new with respect to S. So we can apply the probabilistic polynomial identity test as in Theorem 8:

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

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

Hence in both cases, we have $\tilde{c}\neq\hat{c} \bmod y_0$, but also $\tilde{c}\equiv\hat{c} \bmod q_0$, since any strong forgery is valid. Therefore, $\gcd(y_0,\hat{c}-\tilde{c})=q_0$ 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. □

Acknowledgments. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. 2011-0025127) and was also supported by the year of 2010 Research Fund of the UNIST (Ulsan National Institute of Science and Technology).

References

1. Mihir Bellare, Oded Goldreich, and Anton Mityagin. The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309, 2004.
2. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptol.*, 21(4):469–491, 2008.
3. Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *Advances in Cryptology — EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168. Springer, 2011.
4. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology — CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886, Berlin, Heidelberg, 2012. Springer.
5. Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography — PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2013.
6. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS 2012, pages 309–325. ACM, 2012.
7. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 97–106. IEEE Computer Society CPS, 2011.
8. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology — CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.
9. Dario Catalano and Dario Fiore. Practical homomorphic MACs for arithmetic circuits. In *Advances in Cryptology — EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 336–352. Springer, 2013.
10. Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology — EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335, Berlin, Heidelberg, 2013. Springer.
11. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2014/032, 2014. (To appear in PKC 2014).
12. Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology — CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 487–504. Springer, 2011.
13. Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology — EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer, 2012.
14. Morris Dworkin. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) for confidentiality and authentication. Special Publication 800-38D, NIST, November 2007.
15. Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 301–320. Springer, 2013.
16. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178. ACM, 2009.
17. Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology — EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.
18. Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology — EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.
19. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology — CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.
20. Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In *Fast Software Encryption*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.
21. Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, August 2003.
22. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography — PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

23. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology — EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.