How to Handle Invalid Queries for Malicious-Private Protocols Based on Homomorphic Encryption

Koji Nuida[1,2]

[1] Institute of Mathematics for Industry (IMI), Kyushu University, Japan
nuida@imi.kyushu-u.ac.jp
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan

Abstract

We consider a setting of two-party computation between a server and a client where every message received by the server is encrypted by a fully homomorphic encryption (FHE) scheme and its decryption key is held by the client only. Akavia and Vald (IACR ePrint Archive, 2021) revisited the privacy problem in such protocols against malicious servers and revealed (as opposed to a naive expectation) that under certain condition, a malicious server can recover the client's input even if the underlying FHE scheme is IND-CPA secure. They also gave some sufficient conditions for the FHE scheme to ensure the privacy against malicious servers. However, their argument did not consider the possibility that a query from a malicious server to a client involves an invalid ciphertext. In this paper, we show, by giving a concrete example, that if such an invalid query is just rejected by the client, then the sufficient conditions in Akavia and Vald's result do not in general ensure the privacy against malicious servers. We also propose another option to handle an invalid query in a way that the client returns a random ciphertext (without explicitly rejecting the query), and show that such a protocol is private against malicious servers if the underlying FHE scheme is IND-CPA secure, sanitizable (in the sense of Ducas and Stehlé, EUROCRYPT 2016), and circular secure.

1 Introduction

Secure multiparty computation (MPC) is currently one of the most active research areas in cryptography. In MPC, a number of parties have their individual inputs, and by executing a protocol, each party will obtain the own desired output while keeping the own input secret against other parties. (We note that some party may have empty input and may obtain no individual output.) Among various settings and underlying cryptographic tools for MPC, in this paper we focus on secure two-party computation based on homomorphic encryption (HE) in a so-called client-server model. In the model, a party called *client* is supposed to send encrypted input as well as the underlying public key to the other party called *server*, where the public key and the corresponding secret key are generated by the client. Then the server performs some computation using the homomorphic functionality of the encryption scheme and may interact with the client. Finally, the server sends resulting ciphertexts to the client, and the client decrypts them to obtain the output.

In MPC, a *semi-honest* adversary is supposed to completely follow the protocol specification, while a *malicious* adversary may deviate from the protocol specification in an arbitrary manner. In the study of HE-based MPC on the client-server model, usually much attention is paid to the security of a protocol against the (either semi-honest or malicious) client who has the secret key, while less attention seems to be paid to the security against the server. The reason is that in the client-server model every message received by the server is encrypted by the client's key; in such a case, it is naively expected that the IND-CPA security of the underlying HE scheme ensures at least privacy of the client's input against the server (though it seems not easy to ensure the correctness of the output against malicious servers). The naive expectation is basically correct when the server has no intermediate interaction with the client. However, protocols in

the client-server model may require intermediate interaction between the server and the client in order to, e.g., realize advanced functionality beyond the power of the underlying additively-HE scheme itself [7] or improve the efficiency of a protocol based on fully homomorphic encryption (FHE) [8]. A recent work by Akavia and Vald [2] revisited this naive expectation for privacy of HE-based protocols against servers with interaction, and revealed that *this expectation may fail* typically for the case of malicious servers and even for some significantly restricted situation with semi-honest servers. They also gave some sufficient conditions for such protocols to have privacy against (semi-honest or malicious) servers, one of which is based on the notion of *sanitization* for ciphertexts of FHE schemes [4] (that is, an operation to make the original FHE scheme circuit-private).

In more detail, the work in [2] focused on the situation that a server can make a query of the form (c, f) (or its multi-input and/or multi-output version) to the client where c is a ciphertext and f is a function on the plaintext space, and the server then obtains a ciphertext for plaintext $f(m)$ as the client's reply where m is the decryption result of c. Such a protocol is called an $(\mathcal{E}, \mathcal{G})$-*aided protocol* in [2]. Then in [2], a concrete example of an $(\mathcal{E}, \mathcal{G})$-aided protocol was given in which the reply to a kind of query of the form above enables the server to fully recover the client's input. Moreover, as a part of the affirmative results in [2], the authors introduced an enhanced variant of IND-CPA security called funcCPA-*security*, and showed that an $(\mathcal{E}, \mathcal{G})$-aided protocol ensures privacy against malicious servers if the underlying HE scheme satisfies funcCPA-security. They also showed that funcCPA-secure schemes can be constructed based on sanitizable FHE schemes in [4].

However, here we point out that the formulation of funcCPA-security as well as the proof of privacy based on funcCPA-security in [2] did not consider the possibility that *a query made by the server involves an invalid ciphertext*. As a malicious server may deviate from the protocol specification in an arbitrary manner, such a possibility of invalid queries must be taken into account when the privacy against malicious servers is discussed. The aim of the present paper is to investigate what happens for the results in [2] when invalid queries are considered.

1.1 Our Contributions

In this paper, we first consider a seemingly natural option of handling an invalid query (a query involving invalid ciphertext) in an $(\mathcal{E}, \mathcal{G})$-aided protocol; the client rejects an invalid query and then even aborts the protocol after such an invalid query (for prohibiting a malicious server from obtaining further information during the protocol). For this option, we show (in Section 4) that an $(\mathcal{E}, \mathcal{G})$-aided protocol is *not* always private against malicious servers even if the underlying HE scheme is funcCPA-secure in the sense of [2]. In fact, the "sanitized" version of the FHE scheme FHEW [3] proposed in [4] is funcCPA-secure as shown in [2], but our malicious server in Section 4 violates the privacy of an $(\mathcal{E}, \mathcal{G})$-aided protocol that rejects invalid queries. Roughly speaking, this is because a certain "shift" of a ciphertext for plaintext 0 in FHEW is still a valid ciphertext (for plaintext 1), while the same shift of a ciphertext for plaintext 1 becomes an invalid ciphertext, therefore those two cases can be distinguished by querying the shifted ciphertext to the client and then observing whether the query is rejected or not.

In more detail, recall that an adversary for funcCPA-security defined in [2] is allowed to make some query corresponding to a query in an $(\mathcal{E}, \mathcal{G})$-aided protocol. First we extend funcCPA-security proposed in [2] in two ways by clarifying how a query involving an invalid ciphertext is handled. In the first option, which we also call funcCPA-security, an invalid query is completely prohibited; more precisely, if an invalid query is made by an adversary then the adversary is immediately regarded as losing the security game. In the second option, which we call funcCPA†-security, if an invalid query is made by an adversary then the query is rejected and the adversary loses the access to the oracle in the sequel. Then we prove (in Theorem 1) that funcCPA†-security implies funcCPA-security, and (in Theorem 3) that any $(\mathcal{E}, \mathcal{G})$-aided protocol based on a funcCPA†-secure HE scheme is private against malicious servers. On the other hand, funcCPA-security in the sense of [2] immediately implies funcCPA-security in our sense, but the funcCPA-security does not in general imply the privacy of the protocol as the aforementioned counterexample in Section 4 shows.

We note that an essential reason behind the aforementioned counterexample and the non-implication of privacy from funcCPA-security is that in an HE scheme, especially for noise-based FHE schemes such

as FHEW, it is not always possible to efficiently distinguish invalid ciphertexts from valid ciphertexts by using public information only. In fact, if invalid ciphertexts can be efficiently distinguished by using public information, then $\mathsf{funcCPA}$-security implies $\mathsf{funcCPA}^{\dagger}$-security (see Theorem 2) and hence implies privacy of the protocol. Therefore, an important problem is how to ensure the privacy when invalid ciphertexts cannot be efficiently distinguished. To resolve the problem in this case, we introduce another option of handling an invalid query in an $(\mathcal{E}, \mathcal{G})$-aided protocol; when the query is invalid, instead of rejecting the query, the client returns a random ciphertext for plaintext 0 to the server (and does not abort the protocol). We call this option *silent abort*. Intuitively, the silent abort is expected to prohibit, owing to the security of the underlying HE scheme, a malicious server from obtaining information using replies to invalid queries.

Towards proving privacy for $(\mathcal{E}, \mathcal{G})$-aided protocols with silent abort based on FHE schemes (as the proof of privacy in [2] based on FHE), an obstacle is that handing invalid ciphertexts directly is beyond the functionality of FHE schemes (where the behavior of homomorphic evaluation is guaranteed only for valid ciphertexts). To resolve the issue, we utilize the idea of bootstrapping in FHE schemes [5]. Roughly speaking, given a valid or an invalid ciphertext c as a query, a simulator encrypts each bit of c to obtain a sequence of ciphertexts $\widetilde{c}_1, \ldots, \widetilde{c}_\ell$. Then the simulator applies to $\widetilde{c}_1, \ldots, \widetilde{c}_\ell$ homomorphic evaluation corresponding to a circuit that distinguishes invalid ciphertexts (by using a secret key), which results in a ciphertext whose plaintext represents whether c is invalid or not. Such an argument enables us to prove the privacy for $(\mathcal{E}, \mathcal{G})$-aided protocols with silent abort based on sanitized FHE schemes, by assuming (as in the circular security) that a public key of the FHE scheme involves encryption of a secret key. More precisely, first we introduce another variant of $\mathsf{funcCPA}$-security corresponding to the case of silent abort, called $\mathsf{funcCPA}^{\dagger\dagger}$-security, and show (in Theorem 5) that $\mathsf{funcCPA}^{\dagger\dagger}$-security implies privacy of $(\mathcal{E}, \mathcal{G})$-aided protocols with silent abort. Then we prove (in Theorem 6) that a sanitized FHE scheme under the assumption above is $\mathsf{funcCPA}^{\dagger\dagger}$-secure, hence implying the privacy. It is an open problem to extend the last result above to the case of FHE schemes without circular security, and also to obtain some similar affirmative result in the case of less powerful HE schemes such as additive HE schemes [6] and two-level HE schemes [1].

2 Preliminaries

We write $a \xleftarrow{\$} S$ to mean that an element a is chosen from a set S uniformly at random. We let "PPT" stand for "probabilistic polynomial-time", and consider (unless otherwise noticed) the notion of "polynomial-time" with respect to a security parameter λ. We say that a non-negative function $f(\lambda)$ in a security parameter λ is *negligible* if for each integer $k > 0$, there exists an integer $N > 0$ satisfying that for any $\lambda \geq N$ we have $f(\lambda) \leq 1/\lambda^k$. We let $\{0,1\}^* = \bigcup_{n \geq 0} \{0,1\}^n$. For two probability distributions X, Y over a finite set S, their statistical distance $\Delta(X, Y)$ is defined by

$$\Delta(X, Y) = \frac{1}{2} \sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]| \ .$$

2.1 Public Key Encryption

We clarify the notion of public key encryption (PKE) adopted in this paper. In order to concern later whether or not the set of ciphertexts is efficiently recognizable, here we suppose that any ciphertext is encoded as a bit sequence.

Definition 1 (Public Key Encryption)**.** We say that a tuple of PPT algorithms Gen, Enc and a deterministic (i.e., non-probabilistic) polynomial-time algorithm Dec is a *public key encryption* (*PKE*) *scheme* if it follows the syntax below and satisfies the correctness condition below, where PT is the plaintext space (implicitly depending on the security parameter, the public key, and the secret key), CT_m for $m \in \mathsf{PT}$ are disjoint subsets of $\{0,1\}^*$ (implicitly depending on the security parameter, the public key, and the secret key), $\mathsf{CT} = \bigcup_{m \in \mathsf{PT}} \mathsf{CT}_m$ is the ciphertext space, and negl is some negligible function:

- The key generation algorithm Gen is given a security parameter λ as input and outputs a pair of a public key pk and a secret key sk; $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.

- The encryption algorithm Enc is given a public key pk and a plaintext $m \in \mathsf{PT}$ as input and outputs an encryption result c; $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$.
- The decryption algorithm Dec is given a secret key sk and $c \in \{0,1\}^*$ as input and outputs a decryption result m'; $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. This Dec is supposed to satisfy that $\mathsf{Dec}(\mathsf{sk}, c) = m$ for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, $m \in \mathsf{PT}$, and $c \in \mathsf{CT}_m$.
- **(Correctness for Enc)** With probability at least $1 - \mathsf{negl}(\lambda)$ for the choice of $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ the following holds: For any $m \in \mathsf{PT}$, we have

$$\Pr[\mathsf{Enc}(\mathsf{pk}, m) \in \mathsf{CT}_m] \geq 1 - \mathsf{negl}(\lambda) \ ,$$

  where the latter probability is taken over the internal randomness for Enc.

We say that a PKE scheme as above is

- *with privately recognizable ciphertexts* if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ satisfying that for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $c \in \{0,1\}^*$, we have $\mathcal{R}(\mathsf{pk}, \mathsf{sk}, c) = 1$ if $c \in \mathsf{CT}$ and $\mathcal{R}(\mathsf{pk}, \mathsf{sk}, c) = 0$ if $c \notin \mathsf{CT}$;
- *with publicly recognizable ciphertexts* if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ satisfying that for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $c \in \{0,1\}^*$, we have $\mathcal{R}(\mathsf{pk}, c) = 1$ if $c \in \mathsf{CT}$ and $\mathcal{R}(\mathsf{pk}, c) = 0$ if $c \notin \mathsf{CT}$.

For simplifying our argument, we suppose for any PKE scheme that every element of PT has the same bit length.

2.2 Homomorphic Encryption

We clarify the notion of homomorphic encryption (HE) adopted in this paper, where the plaintext space is fixed to be $\{0,1\}$ for simplifying the argument.

Definition 2 (Homomorphic Encryption)**.** Let $\mathcal{C} = (\mathcal{C}_\lambda)_{\lambda \geq 1}$ be a family of sets of circuits where, for each circuit in each set $\mathcal{C}_\lambda$, its input is a bit sequence and its output is a single bit. We say that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is a *$\mathcal{C}$-homomorphic encryption ($\mathcal{C}$-HE) scheme* (with plaintext space $\{0,1\}$) if $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a PKE scheme with $\mathsf{PT} = \{0,1\}$, Eval is a PPT algorithm, and it also follows the syntax below and satisfies the correctness and compactness conditions below, where negl is some negligible function and ρ is some positive polynomial:

- Let $C \in \mathcal{C}_\lambda$ and let ℓ be the input length of the circuit C. Then the evaluation algorithm Eval is given a public key pk, the circuit C, and $c_1, \ldots, c_\ell \in \{0,1\}^*$ as input and outputs an evaluation result c'; $c' \leftarrow \mathsf{Eval}(\mathsf{pk}, C, c_1, \ldots, c_\ell)$.
- **(Correctness for Eval)** With probability at least $1 - \mathsf{negl}(\lambda)$ for the choice of $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ the following holds. Let $C \in \mathcal{C}_\lambda$ and let ℓ be the input length of the circuit C. Then for any $m_i \in \{0,1\}$ $(i = 1, \ldots, \ell)$ and $c_i \in \mathsf{CT}_{m_i}$ $(i = 1, \ldots, \ell)$, we have

$$\Pr[\mathsf{Eval}(\mathsf{pk}, C, c_1, \ldots, c_\ell) \in \mathsf{CT}_{C(m_1, \ldots, m_\ell)}] \geq 1 - \mathsf{negl}(\lambda) \ ,$$

  where the latter probability is taken over the internal randomness for Eval.
- **(Compactness)** Any ciphertext in CT has bit length at most $\rho(\lambda)$.

In the context above, a fully homomorphic encryption (FHE) scheme may mean a $\mathcal{C}$-HE scheme with the set $\mathcal{C}_\lambda$ containing any practically feasible circuits. But here we do not formalize the meaning of "practically feasible" circuits, as our result below is formulated without using the notion of FHE schemes.

2.3 Sanitization and Sanitized HE Schemes

The following notion was introduced in [4] (with only slight differences).

Definition 3 (Sanitization Algorithm)**.** Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme. We say that a PPT algorithm $\mathsf{Sanitize}$ is a (*statistical*) *sanitization algorithm* if the following conditions hold, where negl is some negligible function:

- **(Message-Preserving)** With probability at least $1 - \mathsf{negl}(\lambda)$ for the choice of $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ the following holds: For any $m \in \mathsf{PT}$ and $c \in \mathsf{CT}_m$, we have

$$\Pr[\mathsf{Sanitize}(\mathsf{pk}, c) \in \mathsf{CT}_m] \geq 1 - \mathsf{negl}(\lambda) \ .$$

- **(Sanitizing)** With probability at least $1 - 2^{-\lambda}$ for the choice of $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ the following holds: For any $m \in \mathsf{PT}$ and $c_1, c_2 \in \mathsf{CT}_m$, we have

$$\Delta(\mathsf{Sanitize}(\mathsf{pk}, c_1), \mathsf{Sanitize}(\mathsf{pk}, c_2)) \leq 2^{-\lambda} \ .$$

Based on the notion of sanitization, a "sanitized" version of HE schemes was introduced in [2]. We explain the construction as follows.

Definition 4 (Sanitized HE Scheme)**.** Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a $\mathcal{C}$-HE scheme having a sanitization algorithm $\mathsf{Sanitize}$. Then we define the *sanitized scheme* $\mathcal{E}^{\mathsf{santz}} = (\mathsf{Gen}, \mathsf{Enc}^{\mathsf{santz}}, \mathsf{Dec}, \mathsf{Eval}^{\mathsf{santz}})$ as follows, where Gen and Dec are the same as $\mathcal{E}$:

- $\mathsf{Enc}^{\mathsf{santz}}(\mathsf{pk}, m)$ outputs $\mathsf{Sanitize}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m))$.
- $\mathsf{Eval}^{\mathsf{santz}}(\mathsf{pk}, C, c_1, \ldots, c_\ell)$ computes $c'_i \leftarrow \mathsf{Sanitize}(\mathsf{pk}, c_i)$ for $i = 1, \ldots, \ell$ and then outputs

$$\mathsf{Sanitize}(\mathsf{pk}, \mathsf{Eval}(\mathsf{pk}, C, c'_1, \ldots, c'_\ell)) \ .$$

It is mentioned in [2] that if $\mathcal{E}$ is a $\mathcal{C}$-HE scheme, then $\mathcal{E}^{\mathsf{santz}}$ is also a $\mathcal{C}$-HE scheme.
We recall the following example of sanitization algorithms proposed in [4].

Example 1. We consider a sanitization algorithm proposed in Section 4.3 of [4], which is applied to an FHE scheme FHEW given in [3]. Let $q > 0$ be an integer modulus, and let n be a parameter for dimension. For a secret key $\vec{s} \in (\mathbb{Z}/q\mathbb{Z})^n$, a plaintext $\mu \in \{0, 1\}$, and some parameters η and t, we define

$$\mathrm{LWE}^{t:q}_{\vec{s}}(\mu, \eta) = \{(\vec{a}, \langle \vec{a}, \vec{s} \rangle + \mu \cdot \lfloor q/t \rfloor + e) \in (\mathbb{Z}/q\mathbb{Z})^{n+1} \colon |e| < \eta q\} \ ,$$

where $\vec{a} \in (\mathbb{Z}/q\mathbb{Z})^n$ and an element of $\mathbb{Z}/q\mathbb{Z}$ is identified with an integer in interval $[-q/2, q/2)$. The ciphertext space for plaintext μ is set to $\mathsf{CT}_\mu = \mathrm{LWE}^{4:q}_{\vec{s}}(\mu, 1/16)$; see [3] and Section 4.3 of [4] for concrete choices of parameters n and q. An outline of the homomorphic operation is as follows. Given two ciphertexts $c_1 \in \mathsf{CT}_{\mu_1}$ and $c_2 \in \mathsf{CT}_{\mu_2}$, the computation part of the operation yields an element of $\mathrm{LWE}^{2:q}_{\vec{s}}(\mu, 1/4)$ with $\mu = \mathsf{NAND}(\mu_1, \mu_2)$, and then the refreshing part converts it back into a ciphertext in CT_μ:

$$\begin{aligned} \mathsf{NAND} \colon \mathrm{LWE}^{4:q}_{\vec{s}}(\mu_1, 1/16) &\times \mathrm{LWE}^{4:q}_{\vec{s}}(\mu_2, 1/16) \\ &\to \mathrm{LWE}^{2:q}_{\vec{s}}(\mu, 1/4) \overset{\mathsf{Refresh}}{\to} \mathrm{LWE}^{4:q}_{\vec{s}}(\mu, 1/16) \ . \end{aligned}$$

On the other hand, a "washing cycle" for elements of $\mathrm{LWE}^{2:q}_{\vec{s}}(\mu, 1/4)$ was introduced in [4]. An outline is as follows (see Section 4.3 of [4] for the details). It begins with a procedure called homomorphic accumulator to obtain an element of $\mathrm{LWE}^{2:Q}_{\vec{z}}(\mu, \eta)$ for another key $\vec{z}$ and certain parameters Q and η. After this, a rerandomization is performed, by adding to it a random sum of ciphertexts for plaintext 0 and a random noise term. This rerandomization realizes that given two input distributions, the two output distributions have statistical distance that is reduced by a certain factor in comparison to the input distributions. Then

the element in $\mathrm{LWE}^{2:Q}_{\vec{z}}(\mu,\eta)$ is converted back into an element of $\mathrm{LWE}^{2:q}_{\vec{s}}(\mu,1/4)$ by a successive application of key switching and modulus switching. This is the process of a single washing cycle:

$$\mathsf{Wash}\colon \mathrm{LWE}^{2:q}_{\vec{s}}(\mu,1/4) \to \mathrm{LWE}^{2:Q}_{\vec{z}}(\mu,\eta) \to \mathrm{LWE}^{2:q}_{\vec{s}}(\mu,1/4)\ .$$

Based on this, the sanitization algorithm for ciphertexts in CT_μ is constructed by inserting a sufficient number of washing cycles into the cycle $\mathrm{LWE}^{4:q}_{\vec{s}}(\mu,1/16) \to \mathrm{LWE}^{2:q}_{\vec{s}}(\mu,1/4) \to \mathrm{LWE}^{4:q}_{\vec{s}}(\mu_1,1/16)$ where the first map is given by $(\vec{a},b) \mapsto (2\vec{a},2b)$ and the second map is the refreshing procedure used in the construction of NAND:

$$\begin{aligned}\mathsf{Sanitize}\colon \mathrm{LWE}^{4:q}_{\vec{s}}(\mu,1/16) \overset{\times 2}{\to} \mathrm{LWE}^{2:q}_{\vec{s}}(\mu,1/4) \overset{\mathsf{Wash}}{\to} \mathrm{LWE}^{2:q}_{\vec{s}}(\mu,1/4) \overset{\mathsf{Wash}}{\to} \cdots \\ \cdots \overset{\mathsf{Wash}}{\to} \mathrm{LWE}^{2:q}_{\vec{s}}(\mu,1/4) \overset{\mathsf{Refresh}}{\to} \mathrm{LWE}^{4:q}_{\vec{s}}(\mu,1/16)\ .\end{aligned}$$

2.4 Two-Party Protocols

In this paper, we consider two-party protocols π between a client Clnt and a server Srv where the server has no output, and focus on privacy against the server only. Given input x for the client Clnt and input y for a possibly malicious server Srv^*, let $\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(x,y)$ denote the random variable of the view (consisting of input, randomness, and received messages during the protocol) for Srv^* during the protocol π executed with those inputs. Then the privacy of π against possibly malicious servers is defined as follows:

Definition 5 (Privacy against Malicious Servers)**.** In the setting above, we say that π is *private against malicious servers* if for any PPT (possibly malicious) server Srv^*, there exists a PPT simulator $\mathcal{S}^*$ satisfying the following: for any PPT non-uniform distinguisher $\mathcal{D}$, there exists a negligible function negl satisfying that

$$\left|\Pr[\mathcal{D}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(x,y))=1] - \Pr[\mathcal{D}(\mathcal{S}^*(1^\lambda,y))=1]\right| \le \mathsf{negl}(\lambda)$$

for any inputs x and y.

3 Function-Chosen-Plaintext Attack Security, Revisited

In [2], a strengthened variant of CPA security for PKE schemes, called security against function-chosen-plaintext attack ($\mathsf{funcCPA}$-security, in short), was introduced. Let $\mathcal{E} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ be a PKE scheme, and let $\mathcal{G}$ be a family of functions whose input is a number of plaintexts and whose output is a single plaintext. (We note that here we slightly extend the original definition in [2] to allow a multi-input function, though the essence of the definition is not changed.) In order to define the security game, the following decrypt-function-encrypt oracle $\mathsf{Enc}_{\mathsf{pk}}(\mathcal{G}(\mathsf{Dec}_{\mathsf{sk}}(\cdot)))$ was introduced:

- A query to the oracle $\mathsf{Enc}_{\mathsf{pk}}(\mathcal{G}(\mathsf{Dec}_{\mathsf{sk}}(\cdot)))$ consists of a function $g \in \mathcal{G}$ and ciphertexts $c_1,\ldots,c_\ell \in \mathsf{CT}$, where ℓ is the number of input plaintexts for g. Then the oracle computes $m_i \leftarrow \mathsf{Dec}(\mathsf{sk},c_i)$ for $i=1,\ldots,\ell$, $m' \leftarrow g(m_1,\ldots,m_\ell)$, and $c' \leftarrow \mathsf{Enc}(\mathsf{pk},m')$, and return c'.

Roughly speaking, $\mathcal{E}$ is defined to be $\mathsf{funcCPA}$-secure with respect to $\mathcal{G}$ if $\mathcal{E}$ is CPA-secure even when the adversary is given access to the oracle above.

Regarding the definition of $\mathsf{funcCPA}$-security in [2], here we emphasize that the definition in [2] does not clearly state what happens if an invalid ciphertext is queried to the oracle. By focusing on this point, we introduce the following two variants of $\mathsf{funcCPA}$-security with seemingly natural treatment of invalid queries. Intuitively, the first option is that the adversary is simply not allowed to make a query with invalid ciphertext (the adversary loses the security game immediately when such a query is made). The second option is that when a query with invalid ciphertext is made, the oracle aborts and becomes not available in the sequel. Concretely, our security experiment $\mathsf{EXP}^{\mathsf{fCPA}}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda)$ with an adversary $\mathcal{A}$ is defined as follows:

1. The challenger generates a key pair $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sets $\mathsf{abort} = \mathsf{no}$.

2. The adversary $\mathcal{A}$ is given pk and access to the oracle $\mathsf{Enc}_{\mathsf{pk}}(\mathcal{G}(\mathsf{Dec}_{\mathsf{sk}}(\cdot)))$. Here, when at least one ciphertext queried to the oracle is not in CT,
   - (Case 1) the challenger sets $\mathsf{abort} = \mathsf{yes}$ and the oracle returns $\perp$;
   - (Case 2) the oracle returns $\perp$, and after this, $\mathcal{A}$ is not allowed to access to the oracle.
3. $\mathcal{A}$ chooses challenge plaintexts $m_0, m_1 \in \mathsf{PT}$ and sends them to the challenger.
4. The challenger chooses a challenge bit $b^* \xleftarrow{\$} \{0,1\}$, generates a challenge ciphertext $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{b^*})$, and sends c^* to $\mathcal{A}$.
5. $\mathcal{A}$ still has access to the oracle $\mathsf{Enc}_{\mathsf{pk}}(\mathcal{G}(\mathsf{Dec}_{\mathsf{sk}}(\cdot)))$ (unless the oracle access is lost as described above), and finally outputs a bit b. The output of the experiment is defined to be 1 if $b = b^*$ and $\mathsf{abort} = \mathsf{no}$, and 0 otherwise.

Definition 6 ($\mathsf{funcCPA}$-security)**.** In the setting above, we say that $\mathcal{E}$ is $\mathsf{funcCPA}$-secure (respectively, $\mathsf{funcCPA}^\dagger$-secure) with respect to $\mathcal{G}$ if for any PPT non-uniform adversary $\mathcal{A}$, there exists a negligible function negl satisfying

$$\Pr[\mathsf{EXP}^{\mathsf{fCPA}}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \ ,$$

where Case 1 (respectively, Case 2) for the behavior with invalid query is adopted.

We note that the $\mathsf{funcCPA}$-security and $\mathsf{funcCPA}^\dagger$-security become the same as the CPA-security against non-uniform adversaries when $\mathcal{G} = \emptyset$.

From now on, we let $\mathsf{EXP}^{\mathsf{fCPA}\dagger}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda)$ denote the experiment above with Case 2, while $\mathsf{EXP}^{\mathsf{fCPA}}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda)$ denotes the experiment with Case 1. We have the following relation for the two versions of security (the converse does not hold in general, as discussed later).

Theorem 1. *In the setting above, if $\mathcal{E}$ is $\mathsf{funcCPA}^\dagger$-secure with respect to $\mathcal{G}$, then $\mathcal{E}$ is $\mathsf{funcCPA}$-secure with respect to $\mathcal{G}$.*

Roughly speaking, this theorem holds because if an adversary wins the security game for $\mathsf{funcCPA}$, then an invalid query is not made during the game, in which case the games for $\mathsf{funcCPA}$ and $\mathsf{funcCPA}^\dagger$ are equivalent. A formal proof is given below.

Proof of Theorem 1. Let $\mathcal{A}$ be any PPT (non-uniform) adversary for $\mathsf{funcCPA}$ experiment. We define an adversary $\mathcal{A}^\dagger$ for $\mathsf{funcCPA}^\dagger$ experiment as follows: $\mathcal{A}^\dagger$ internally executes $\mathcal{A}$. Here, when $\mathcal{A}$ generates challenge plaintexts, $\mathcal{A}^\dagger$ also use them as challenge plaintexts. When $\mathcal{A}$ produces the final output bit, $\mathcal{A}^\dagger$ also use it as the output bit. Moreover, when $\mathcal{A}$ makes an oracle query (in $\mathsf{funcCPA}$ experiment), $\mathcal{A}^\dagger$ forwards the query to the oracle (in $\mathsf{funcCPA}^\dagger$ experiment) and obtains the reply. If this reply is not $\perp$, then $\mathcal{A}^\dagger$ forwards it to $\mathcal{A}$. On the other hand, if this reply is $\perp$, then $\mathcal{A}^\dagger$ stops the execution of $\mathcal{A}$, sends any two challenge plaintexts to the challenger if $\mathcal{A}^\dagger$ has not done it, and then outputs a random bit b.

By definition, $\mathcal{A}^\dagger$ (as well as $\mathcal{A}$) is PPT and we have

$$\Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger}_{\mathcal{A}^\dagger,\mathcal{E},\mathcal{G}}(\lambda) = 1] \geq \Pr[\mathsf{EXP}^{\mathsf{fCPA}}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda) = 1] \ .$$

As $\mathcal{E}$ is $\mathsf{funcCPA}^\dagger$-secure, there exists a negligible function $\mathsf{negl}(\lambda)$ satisfying

$$\Pr[\mathsf{EXP}^{\mathsf{fCPA}}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda) = 1] \leq \Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger}_{\mathcal{A}^\dagger,\mathcal{E},\mathcal{G}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \ .$$

This implies that $\mathcal{E}$ is $\mathsf{funcCPA}$-secure, as desired. □

The converse of Theorem 1 holds under an additional condition:

Theorem 2. *In the setting above, if $\mathcal{E}$ is* $\mathsf{funcCPA}$*-secure with respect to $\mathcal{G}$ and is with publicly recognizable ciphertexts, then $\mathcal{E}$ is* $\mathsf{funcCPA}^\dagger$*-secure with respect to $\mathcal{G}$.*

Roughly speaking, this theorem holds because if $\mathcal{E}$ is with publicly recognizable ciphertexts, then an adversary can avoid making an invalid query to the oracle, in which case the games for $\mathsf{funcCPA}$ and $\mathsf{funcCPA}^\dagger$ are equivalent. A formal proof is given below.

Proof of Theorem 2. Let $\mathcal{R}$ be the algorithm implied by the "publicly recognizable ciphertexts" property for $\mathcal{E}$. Let $\mathcal{A}^\dagger$ be any PPT (non-uniform) adversary for $\mathsf{funcCPA}^\dagger$ experiment. We define an adversary $\mathcal{A}$ for $\mathsf{funcCPA}$ experiment as follows: $\mathcal{A}$ internally executes $\mathcal{A}^\dagger$, where the challenge plaintexts and the output bits are common for $\mathcal{A}$ and $\mathcal{A}^\dagger$. Here, when $\mathcal{A}^\dagger$ makes a query $(g, c_1, \ldots, c_\ell)$ to the oracle (in $\mathsf{funcCPA}^\dagger$ experiment), $\mathcal{A}$ first checks if all c_i are in CT by using the algorithm $\mathcal{R}(\mathsf{pk}, \cdot)$. If all c_i are in CT, then $\mathcal{A}$ forwards the query to the oracle in $\mathsf{funcCPA}$ experiment, and sends the reply back to $\mathcal{A}^\dagger$. On the other hand, if at least one c_i is not in CT, then $\mathcal{A}$ sends $\perp$ back to $\mathcal{A}^\dagger$ and in the sequel, rejects any oracle query from $\mathcal{A}^\dagger$.

By definition, $\mathcal{A}$ (as well as $\mathcal{A}^\dagger$) is PPT and we have

$$\Pr[\mathsf{EXP}^{\mathsf{fCPA}}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda) = 1] = \Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger}_{\mathcal{A}^\dagger,\mathcal{E},\mathcal{G}}(\lambda) = 1] \ .$$

As $\mathcal{E}$ is $\mathsf{funcCPA}$-secure, there exists a negligible function $\mathsf{negl}(\lambda)$ satisfying

$$\Pr[\mathsf{EXP}^{\mathsf{fCPA}}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda) = 1] = \Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger}_{\mathcal{A}^\dagger,\mathcal{E},\mathcal{G}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \ .$$

This implies that $\mathcal{E}$ is $\mathsf{funcCPA}^\dagger$-secure, as desired. □

3.1 Relation to Two-Party Protocols with Evaluation Queries

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and $\mathcal{G}$ be as above. We consider the class of two-party protocols between a client Clnt and a server Srv as follows, called $(\mathcal{E}, \mathcal{G})$*-aided protocols*. Here we emphasize that the definition below is almost the same as the one in [2], but in [2], it was not clearly stated how the client should respond when a server's query involves an invalid ciphertext, which should be clarified as now we are considering malicious servers deviating from the protocol specification. Below we adopt a seemingly natural way of handling such invalid queries, that is to abort the protocol once the client receives an invalid query (to do this efficiently, we assume that the PKE scheme is with privately recognizable ciphertexts in the sense of Definition 1). We also note that the definition in [2] considered only such protocols without server's input. A protocol in this class is described as follows:

1. **(Client's input outsourcing phase)** Given client's input $\vec{x} = (x_1, \ldots, x_{\eta(\lambda)})$, the client generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $e_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i)$ for $i = 1, \ldots, \eta(\lambda)$, and sends pk and $\vec{e} = (e_1, \ldots, e_{\eta(\lambda)})$ to the server. (We suppose that the number $\eta(\lambda)$ of inputs is polynomial-time computable.)

2. **(Server's computation phase)** The server may interact with the client by sending a query of the form $(g, \vec{c})$, where $g \in \mathcal{G}$ is an ℓ-input function and $\vec{c} = (c_1, \ldots, c_\ell)$. For such a query,

   - if $c_1, \ldots, c_\ell \in \mathsf{CT}$, then the client computes

     $$\mathsf{Enc}(\mathsf{pk}, g(\mathsf{Dec}(\mathsf{sk}, c_1), \ldots, \mathsf{Dec}(\mathsf{sk}, c_\ell)))$$

     and sends it back to the server;
   - otherwise, the client aborts the protocol (the server receives an aborting signal).

   Finally, the server sends a message to the client.

3. **(Client's output phase)** The client generates an output.

Now basically the same argument as Theorem 7 of [2] implies the following:

Theorem 3. *Let π be a PPT $(\mathcal{E},\mathcal{G})$-aided protocol as above. If $\mathcal{E}$ is with privately recognizable ciphertexts and is $\mathsf{funcCPA}^{\dagger}$-secure with respect to $\mathcal{G}$, then π is private against malicious servers.*

Proof. Let $\mathcal{R}$ be the algorithm in the definition of "with privately recognizable ciphertexts" property for $\mathcal{E}$. Let Srv^* be any PPT malicious server. Then we define a simulator $\mathcal{S}^*$ with server's input y as follows:

1. $\mathcal{S}^*$ generates $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, chooses $x_i^0 \xleftarrow{\$} \mathsf{PT}$ and generates $e_i^0 \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i^0)$ for $i = 1,\ldots,\eta(\lambda)$, and sets $\vec{e}^{\,0} = (e_1^0,\ldots,e_{\eta(\lambda)}^0)$.

2. $\mathcal{S}^*$ internally executes Srv^* with input y and randomness r chosen randomly by $\mathcal{S}^*$, where the list msg is initialized to be empty. Here $\mathcal{S}^*$ sends the pk and $\vec{e}^{\,0}$ to Srv^* in Step 1 of the protocol π. When a query $(g,\vec{c})$ is made by Srv^* in Step 2 of π, $\mathcal{S}^*$ checks if all $c_1,\ldots,c_\ell$ are in CT by using the algorithm $\mathcal{R}(\mathsf{pk},\mathsf{sk},\cdot)$. If $c_1,\ldots,c_\ell \in \mathsf{CT}$, then $\mathcal{S}^*$ computes $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, g(\mathsf{Dec}(\mathsf{sk},c_1),\ldots,\mathsf{Dec}(\mathsf{sk},c_\ell)))$, sends c' to Srv^*, and adds c' to the list msg. On the other hand, if some c_i is not in CT, then $\mathcal{S}^*$ stops the execution of Srv^* and adds the aborting signal to the list msg.

3. Finally, $\mathcal{S}^*$ outputs (y,r,msg) as the simulated view for Srv^*.

We note that $\mathcal{S}^*$ is a PPT algorithm by the construction.

We assume for the contrary that this simulator does not satisfy the privacy condition. That is, there exists a PPT non-uniform distinguisher $\mathcal{D}^{(z_\lambda)}$ with advice z_λ satisfying that for any negligible function negl, we have

$$\left|\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x},y))=1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda,y))=1]\right| > \mathsf{negl}(\lambda)$$

for some λ and some input pair $(\vec{x},y)$. In particular, by letting $(\vec{x}^{\,\lambda},y^\lambda)$ be the pair maximizing the left-hand side above for a fixed λ, it follows that the quantity

$$\left|\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda},y^\lambda))=1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda,y^\lambda))=1]\right| \tag{1}$$

is non-negligible.

Now we define a non-uniform adversary $\mathcal{A}^+$ against $\mathsf{funcCPA}^{\dagger}$-security for $\mathcal{E}$ with respect to $\mathcal{G}$ as follows. The advice for $\mathcal{A}^+$ is $(z_\lambda,\vec{x}^{\,\lambda},y^\lambda)$. Given a public key pk, $\mathcal{A}^+$ chooses $I \xleftarrow{\$} \{1,\ldots,\eta(\lambda)\}$ and sets the challenge plaintexts to be $m_0 = x_I^\lambda$ and $m_1 \xleftarrow{\$} \mathsf{PT}$. Given a challenge ciphertext c^*, $\mathcal{A}^+$ internally executes Srv^* with input y^λ and randomness r chosen randomly by $\mathcal{A}^+$, where msg is initialized to be empty. Here $\mathcal{A}^+$ sends pk and $\vec{e} = (e_1,\ldots,e_{\eta(\lambda)})$ to Srv^* in Step 1 of π, where $x_i^0 \xleftarrow{\$} \mathsf{PT}$ and $e_i \leftarrow \mathsf{Enc}(\mathsf{pk},x_i^0)$ for $i = 1,\ldots,I-1$, $e_I = c^*$, and $e_i \leftarrow \mathsf{Enc}(\mathsf{pk},x_i^\lambda)$ for $i = I+1,\ldots,\eta(\lambda)$. When a query $(g,\vec{c})$ is made by Srv^* in Step 2 of π, $\mathcal{A}^+$ forwards it to the oracle in the $\mathsf{funcCPA}^{\dagger}$ game. If the reply is not $\perp$, then $\mathcal{A}^+$ adds the reply to msg and sends the reply back to Srv^*. If the reply is $\perp$, then $\mathcal{A}^+$ stops the execution of Srv^* and adds the aborting signal to msg. Finally, $\mathcal{A}^+$ executes $\mathcal{D}^{(z_\lambda)}(y^\lambda,r,\mathsf{msg})$ and outputs the output bit b of the $\mathcal{D}^{(z_\lambda)}$. We note that $\mathcal{A}^+$ is PPT.

Let b^* denote the challenge bit in the $\mathsf{funcCPA}^{\dagger}$ game. Then the distribution of interaction between $\mathcal{A}^+$ and Srv^* conditioned on index $I=1$ and $b^*=0$ (respectively, $I=\eta(\lambda)$ and $b^*=1$) is identical to the protocol π with malicious server Srv^* and input pair $(\vec{x}^{\,\lambda},y^\lambda)$ (respectively, interaction between the simulator $\mathcal{S}^*$ and Srv^* with input y^λ). Hence we have

$$\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda},y^\lambda))=1] = \Pr[b=1 \mid (I,b^*)=(1,0)]\ ,$$

$$\Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda,y^\lambda))=1] = \Pr[b=1 \mid (I,b^*)=(\eta(\lambda),1)]\ .$$

On the other hand, for each $2 \le I_0 \le \eta(\lambda)$, the distribution of interaction between $\mathcal{A}^+$ and Srv^* conditioned on $I=I_0$ and $b^*=0$ is identical to that conditioned on $I=I_0-1$ and $b^*=1$. Hence we have

$$\Pr[b=1 \mid (I,b^*)=(I_0,0)] = \Pr[b=1 \mid (I,b^*)=(I_0-1,1)]\ .$$

Therefore, we have

$$\begin{aligned}
&\Pr[b=1 \mid b^*=0] - \Pr[b=1 \mid b^*=1] \\
&= \sum_{I_0=1}^{\eta(\lambda)} \frac{1}{\eta(\lambda)} \Pr[b=1 \mid (I,b^*)=(I_0,0)] - \sum_{I_0=1}^{\eta(\lambda)} \frac{1}{\eta(\lambda)} \Pr[b=1 \mid (I,b^*)=(I_0,1)] \\
&= \frac{1}{\eta(\lambda)} \left(\Pr[b=1 \mid (I,b^*)=(1,0)] - \Pr[b=1 \mid (I,b^*)=(\eta(\lambda),1)]\right) \\
&= \frac{1}{\eta(\lambda)} \left(\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1]\right)
\end{aligned}$$

and

$$\begin{aligned}
&\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1] \\
&= \eta(\lambda) \left(\Pr[b=1 \mid b^*=0] - \Pr[b=1 \mid b^*=1]\right) \\
&= \eta(\lambda) \left(1 - \Pr[b=0 \mid b^*=0] - \Pr[b=1 \mid b^*=1]\right) \\
&= \eta(\lambda) \left(1 - 2\Pr[b=0 \wedge b^*=0] - 2\Pr[b=1 \wedge b^*=1]\right) \\
&= 2\eta(\lambda) \left(\frac{1}{2} - \Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger}_{\mathcal{A}^+,\mathcal{E},\mathcal{G}}(\lambda) = 1]\right) \ .
\end{aligned}$$

As $\eta(\lambda)$ is polynomially bounded (because the client is PPT) and $\mathcal{E}$ is $\mathsf{funcCPA}^\dagger$-secure, it follows that there exists a negligible function negl_1 satisfying that the quantity above is at least $-\mathsf{negl}_1(\lambda)$.

We also define a PPT adversary $\mathcal{A}^-$ for $\mathcal{E}$ in a way that it outputs $b' = 1 - b$ when $\mathcal{A}^+$ outputs b. Then by the argument above, we have

$$\begin{aligned}
&\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1] \\
&= \eta(\lambda) \left(\Pr[b'=0 \mid b^*=0] - \Pr[b'=0 \mid b^*=1]\right) \\
&= \eta(\lambda) \left(\Pr[b'=0 \mid b^*=0] - (1 - \Pr[b'=1 \mid b^*=1])\right) \\
&= \eta(\lambda) \left(2\Pr[b'=0 \wedge b^*=0] + 2\Pr[b'=1 \wedge b^*=1] - 1\right) \\
&= 2\eta(\lambda) \left(\Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger}_{\mathcal{A}^-,\mathcal{E},\mathcal{G}}(\lambda) = 1] - \frac{1}{2}\right) \ .
\end{aligned}$$

Again, as $\mathcal{E}$ is $\mathsf{funcCPA}^\dagger$-secure, there exists a negligible function negl_2 satisfying that the quantity above is at most $\mathsf{negl}_2(\lambda)$. Summarizing, we have

$$-\mathsf{negl}_1(\lambda) \leq \Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1] \leq \mathsf{negl}_2(\lambda)$$

and the quantity in Eq.(1) is at most the negligible value $\max\{\mathsf{negl}_1(\lambda), \mathsf{negl}_2(\lambda)\}$, a contradiction. Hence the simulator $\mathcal{S}^*$ satisfies the condition showing that the protocol π is private against malicious servers, concluding the proof of Theorem 3. □

In contrast to Theorem 3, the $\mathsf{funcCPA}$-security of $\mathcal{E}$ does in general not imply privacy against malicious servers. Intuitively, this is because $\mathsf{funcCPA}$-security completely prohibits an adversary from gaining information by using a query with invalid ciphertexts, while an $(\mathcal{E}, \mathcal{G})$-aided protocol as formulated above allows the server to make an invalid query at most once, getting the information that some queried ciphertext is invalid. We provide a concrete example in Section 4.

3.2 Relation with Sanitized Schemes

It was shown in Theorem 8 of [2] that if $\mathcal{E}$ is a CPA-secure $\mathcal{C}$-HE scheme with a sanitization algorithm, then the sanitized scheme $\mathcal{E}^{\mathsf{santz}}$ is $\mathsf{funcCPA}$-secure with respect to any function family $\mathcal{G}$ satisfying that $\mathcal{G} \subseteq \mathcal{C}_\lambda$

for any λ (where we identify each circuit in $\mathcal{C}_\lambda$ with the function that determines). Here we note that the definition of $\mathsf{funcCPA}$-security in [2] did not consider the possibility that an invalid ciphertext is queried to the oracle. This point becomes not a problem by choosing an option of not allowing invalid queries, i.e., $\mathsf{funcCPA}$-security (rather than $\mathsf{funcCPA}^\dagger$-security) in our sense. Summarizing, we have the following result in terms of our terminology:

Theorem 4. *Let $\mathcal{E}$ be a CPA-secure $\mathcal{C}$-HE scheme with a sanitization algorithm. Let $\mathcal{G}$ be a function family satisfying that $\mathcal{G} \subseteq \mathcal{C}_\lambda$ for any λ. Then the sanitized scheme $\mathcal{E}^{\mathsf{santz}}$ is $\mathsf{funcCPA}$-secure with respect to $\mathcal{G}$.*

We note that, by combining Theorem 4 with Theorem 2, it follows that if the scheme $\mathcal{E}$ in Theorem 4 is moreover with publicly recognizable ciphertexts, then $\mathcal{E}^{\mathsf{santz}}$ is $\mathsf{funcCPA}^\dagger$-secure. But in general, $\mathcal{E}^{\mathsf{santz}}$ in Theorem 4 is not $\mathsf{funcCPA}^\dagger$-secure, as shown in Section 4.

4 funcCPA-Security Does Not Imply Privacy

In this section, we give a concrete example showing that for an $(\mathcal{E}, \mathcal{G})$-aided protocol π as in Section 3.1, the assumption for $\mathcal{E}$ being (with privately recognizable ciphertexts and) $\mathsf{funcCPA}$-secure with respect to $\mathcal{G}$ does in general not imply that π is private against malicious servers.

We consider an FHE scheme $\mathcal{E}$ with sanitization algorithm described in Example 1 of Section 2.3. Its ciphertext space for plaintext $\mu \in \{0,1\}$ is $\mathsf{CT}_\mu = \mathrm{LWE}_{\vec{s}}^{4:q}(\mu, 1/16)$ as defined in Section 2.3. This scheme is with privately recognizable ciphertexts; indeed, given an element $c = (\vec{a}, b) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$ and a candidate of plaintext μ, it is efficient to compute (by using the secret key $\vec{s}$) the term e satisfying $b = \langle \vec{a}, \vec{s} \rangle + \mu \cdot \lfloor q/t \rfloor + e$ and then check if $|e| < q/16$ or not.

Let the function family $\mathcal{G}$ be consisting of the identity function id only: $\mathcal{G} = \{\mathsf{id}\}$. Then by Theorem 4, the sanitized scheme $\mathcal{E}^{\mathsf{santz}}$ is $\mathsf{funcCPA}$-secure with respect to $\mathcal{G}$. Now we consider any $(\mathcal{E}^{\mathsf{santz}}, \mathcal{G})$-aided protocol π in which the server is allowed (by the protocol specification) to make at least one query of the form $(\mathsf{id}, *)$. For the protocol π and the underlying scheme $\mathcal{E}^{\mathsf{santz}}$, we consider a malicious server Srv^* that behaves as follows:

- Given a public key and a ciphertext c of (a part of) the client's input x, Srv^* computes $c' = c + (0^n, \lfloor q/4 \rfloor) \in (\mathbb{Z}/q\mathbb{Z})^{n+1}$ where $0^n = (0, \ldots, 0)^{\mathrm{T}} \in (\mathbb{Z}/q\mathbb{Z})^n$ and then makes a query (id, c') to the client.

In the situation above, if $c \in \mathsf{CT}_0$, i.e., $c = (\vec{a}, b)$ and $b = \langle \vec{a}, \vec{s} \rangle + e$ with $|e| < q/16$, then $c' = (\vec{a}, b')$ with $b' = \langle \vec{a}, \vec{s} \rangle + \lfloor q/4 \rfloor + e$, therefore $c' \in \mathsf{CT}_1$. In this case, the client on receiving the query above does not abort the protocol. On the other hand, if $c \in \mathsf{CT}_1$, i.e., $c = (\vec{a}, b)$ and $b = \langle \vec{a}, \vec{s} \rangle + \lfloor q/4 \rfloor + e$ with $|e| < q/16$, then $c' = (\vec{a}, b')$ with $b' = \langle \vec{a}, \vec{s} \rangle + 2 \cdot \lfloor q/4 \rfloor + e$. This c' is not an element of $\mathsf{CT} = \mathsf{CT}_0 \cup \mathsf{CT}_1$, therefore in this case, the client on receiving the query above aborts the protocol. Hence the malicious server can distinguish these two cases, i.e., can know (a part of) the client's input x, by observing whether the client aborts the protocol or not. This breaks the privacy of π against malicious servers, despite that $\mathcal{E}^{\mathsf{santz}}$ is (with privately recognizable ciphertexts and) $\mathsf{funcCPA}$-secure with respect to $\mathcal{G}$.

We also note that in this example, the sanitized scheme $\mathcal{E}^{\mathsf{santz}}$ is not $\mathsf{funcCPA}^\dagger$-secure (as otherwise Theorem 3 would imply that π is private against malicious servers). This also gives a counterexample mentioned at the end of Section 3.2.

5 Achieving Privacy against Malicious Servers

We have seen above that in the seemingly natural style of aborting the protocol when receiving an invalid query, the privacy of an $(\mathcal{E}, \mathcal{G})$-aided protocol against malicious servers is not always achieved even if the underlying scheme $\mathcal{E}$ is $\mathsf{funcCPA}$-secure and with privately recognizable ciphertexts. In this section, we propose a countermeasure to achieve the privacy against malicious servers with such an underlying scheme $\mathcal{E}$. The countermeasure here is to change the client's behavior when receiving an invalid query as explained

below. Concretely, we consider a class of protocols described as follows, which we call an $(\mathcal{E}, \mathcal{G})$-*aided protocol with silent abort*:

1. **(Client's input outsourcing phase)** Given client's input $\vec{x} = (x_1, \ldots, x_{\eta(\lambda)})$, the client generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $e_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i)$ for $i = 1, \ldots, \eta(\lambda)$, and sends pk and $\vec{e} = (e_1, \ldots, e_{\eta(\lambda)})$ to the server. (We suppose that the number $\eta(\lambda)$ of inputs is polynomial-time computable.)

2. **(Server's computation phase)** The server may interact with the client by sending a query of the form $(g, \vec{c})$, where $g \in \mathcal{G}$ is an ℓ-input function and $\vec{c} = (c_1, \ldots, c_\ell)$. For such a query,

   - if $c_1, \ldots, c_\ell \in \mathsf{CT}$, then the client computes

     $$\mathsf{Enc}(\mathsf{pk}, g(\mathsf{Dec}(\mathsf{sk}, c_1), \ldots, \mathsf{Dec}(\mathsf{sk}, c_\ell)))$$

     and sends it back to the server;
   - otherwise, the client computes $\mathsf{Enc}(\mathsf{pk}, 0)$ (where 0 denotes a fixed and polynomial-time computable element of PT) and sends it back to the server.

   Finally, the server sends a message to the client.

3. **(Client's output phase)** The client generates an output.

In order to study the privacy of such a protocol, we introduce yet another variant of $\mathsf{funcCPA}$-security by also modifying the behavior of the security game when a query is invalid. Let $\mathcal{E}$ and $\mathcal{G}$ be as in Section 3. Our security experiment $\mathsf{EXP}^{\mathsf{fCPA}\dagger\dagger}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda)$ with an adversary $\mathcal{A}$ is defined as follows:

1. The challenger generates a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.

2. The adversary $\mathcal{A}$ is given pk and access to the oracle $\mathsf{Enc}_{\mathsf{pk}}(\mathcal{G}(\mathsf{Dec}_{\mathsf{sk}}(\cdot)))$. Here, when at least one ciphertext queried to the oracle is not in CT, the oracle returns $\mathsf{Enc}(\mathsf{pk}, 0)$ to the adversary (where 0 denotes a fixed and polynomial-time computable element of PT).

3. $\mathcal{A}$ chooses challenge plaintexts $m_0, m_1 \in \mathsf{PT}$ and sends them to the challenger.

4. The challenger chooses a challenge bit $b^* \xleftarrow{\$} \{0, 1\}$, generates a challenge ciphertext $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{b^*})$, and sends c^* to $\mathcal{A}$.

5. $\mathcal{A}$ still has access to the oracle $\mathsf{Enc}_{\mathsf{pk}}(\mathcal{G}(\mathsf{Dec}_{\mathsf{sk}}(\cdot)))$, and finally outputs a bit b. The output of the experiment is defined to be 1 if $b = b^*$, and 0 otherwise.

Definition 7 ($\mathsf{funcCPA}^{\dagger\dagger}$-security)**.** In the setting above, we say that $\mathcal{E}$ is $\mathsf{funcCPA}^{\dagger\dagger}$-secure with respect to $\mathcal{G}$ if for any PPT non-uniform adversary $\mathcal{A}$, there exists a negligible function negl satisfying

$$\Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger\dagger}_{\mathcal{A},\mathcal{E},\mathcal{G}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda) \ .$$

Then we give a result analogous to Theorem 3, as follows:

Theorem 5. *Let π be a PPT $(\mathcal{E}, \mathcal{G})$-aided protocol with silent abort as above. If $\mathcal{E}$ is with privately recognizable ciphertexts and is $\mathsf{funcCPA}^{\dagger\dagger}$-secure with respect to $\mathcal{G}$, then π is private against malicious servers.*

Proof. Let $\mathcal{R}$ be the algorithm in the definition of "with privately recognizable ciphertexts" property for $\mathcal{E}$. Let Srv^* be any PPT malicious server. Then we define a simulator $\mathcal{S}^*$ with server's input y as follows:

1. $\mathcal{S}^*$ generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, chooses $x_i^0 \xleftarrow{\$} \mathsf{PT}$ and generates $e_i^0 \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i^0)$ for $i = 1, \ldots, \eta(\lambda)$, and sets $\vec{e}^{\,0} = (e_1^0, \ldots, e_{\eta(\lambda)}^0)$.

2. $\mathcal{S}^*$ internally executes Srv^* with input y and randomness r chosen randomly by $\mathcal{S}^*$, where the list msg is initialized to be empty. Here $\mathcal{S}^*$ sends the pk and $\vec{e}^0$ to Srv^* in Step 1 of the protocol π. When a query $(g, \vec{c})$ is made by Srv^* in Step 2 of π, $\mathcal{S}^*$ checks if all $c_1, \ldots, c_\ell$ are in CT by using the algorithm $\mathcal{R}(\mathsf{pk}, \mathsf{sk}, \cdot)$. If $c_1, \ldots, c_\ell \in \mathsf{CT}$, then $\mathcal{S}^*$ computes $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, g(\mathsf{Dec}(\mathsf{sk}, c_1), \ldots, \mathsf{Dec}(\mathsf{sk}, c_\ell)))$, sends c' to Srv^*, and adds c' to the list msg. On the other hand, if some c_i is not in CT, then $\mathcal{S}^*$ computes $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, 0)$, sends c' to Srv^*, and adds c' to the list msg.

3. Finally, $\mathcal{S}^*$ outputs (y, r, msg) as the simulated view for Srv^*.

We note that $\mathcal{S}^*$ is a PPT algorithm by the construction.

We assume for the contrary that this simulator does not satisfy the privacy condition. That is, there exists a PPT non-uniform distinguisher $\mathcal{D}^{(z_\lambda)}$ with advice z_λ satisfying that for any negligible function negl, we have

$$\left|\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^\pi_{\mathsf{Srv}^*}(\vec{x}, y)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y)) = 1]\right| > \mathsf{negl}(\lambda)$$

for some λ and some input pair $(\vec{x}, y)$. In particular, by letting $(\vec{x}^{\,\lambda}, y^\lambda)$ be the pair maximizing the left-hand side above for a fixed λ, it follows that the quantity

$$\left|\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^\pi_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1]\right| \tag{2}$$

is non-negligible.

Now we define a non-uniform adversary $\mathcal{A}^+$ against $\mathsf{funcCPA}^{\dagger\dagger}$-security for $\mathcal{E}$ with respect to $\mathcal{G}$ as follows. The advice for $\mathcal{A}^+$ is $(z_\lambda, \vec{x}^{\,\lambda}, y^\lambda)$. Given a public key pk, $\mathcal{A}^+$ chooses $I \xleftarrow{\$} \{1, \ldots, \eta(\lambda)\}$ and sets the challenge plaintexts to be $m_0 = x_I^\lambda$ and $m_1 \xleftarrow{\$} \mathsf{PT}$. Given a challenge ciphertext c^*, $\mathcal{A}^+$ internally executes Srv^* with input y^λ and randomness r chosen randomly by $\mathcal{A}^+$, where msg is initialized to be empty. Here $\mathcal{A}^+$ sends pk and $\vec{e} = (e_1, \ldots, e_{\eta(\lambda)})$ to Srv^* in Step 1 of π, where $x_i^0 \xleftarrow{\$} \mathsf{PT}$ and $e_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i^0)$ for $i = 1, \ldots, I-1$, $e_I = c^*$, and $e_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i^\lambda)$ for $i = I+1, \ldots, \eta(\lambda)$. When a query $(g, \vec{c})$ is made by Srv^* in Step 2 of π, $\mathcal{A}^+$ forwards it to the oracle in the $\mathsf{funcCPA}^{\dagger\dagger}$ game, adds the oracle's reply to msg, and sends the reply back to Srv^*. Finally, $\mathcal{A}^+$ executes $\mathcal{D}^{(z_\lambda)}(y^\lambda, r, \mathsf{msg})$ and outputs the output bit b of the $\mathcal{D}^{(z_\lambda)}$. We note that $\mathcal{A}^+$ is PPT.

Let b^* denote the challenge bit in the $\mathsf{funcCPA}^{\dagger\dagger}$ game. Then the distribution of interaction between $\mathcal{A}^+$ and Srv^* conditioned on index $I = 1$ and $b^* = 0$ (respectively, $I = \eta(\lambda)$ and $b^* = 1$) is identical to the protocol π with malicious server Srv^* and input pair $(\vec{x}^{\,\lambda}, y^\lambda)$ (respectively, interaction between the simulator $\mathcal{S}^*$ and Srv^* with input y^λ). Hence we have

$$\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^\pi_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] = \Pr[b = 1 \mid (I, b^*) = (1, 0)] \ ,$$

$$\Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1] = \Pr[b = 1 \mid (I, b^*) = (\eta(\lambda), 1)] \ .$$

On the other hand, for each $2 \le I_0 \le \eta(\lambda)$, the distribution of interaction between $\mathcal{A}^+$ and Srv^* conditioned on $I = I_0$ and $b^* = 0$ is identical to that conditioned on $I = I_0 - 1$ and $b^* = 1$. Hence we have

$$\Pr[b = 1 \mid (I, b^*) = (I_0, 0)] = \Pr[b = 1 \mid (I, b^*) = (I_0 - 1, 1)] \ .$$

Therefore, we have

$$\begin{aligned}
&\Pr[b = 1 \mid b^* = 0] - \Pr[b = 1 \mid b^* = 1] \\
&= \sum_{I_0=1}^{\eta(\lambda)} \frac{1}{\eta(\lambda)} \Pr[b = 1 \mid (I, b^*) = (I_0, 0)] - \sum_{I_0=1}^{\eta(\lambda)} \frac{1}{\eta(\lambda)} \Pr[b = 1 \mid (I, b^*) = (I_0, 1)] \\
&= \frac{1}{\eta(\lambda)} \left(\Pr[b = 1 \mid (I, b^*) = (1, 0)] - \Pr[b = 1 \mid (I, b^*) = (\eta(\lambda), 1)]\right) \\
&= \frac{1}{\eta(\lambda)} \left(\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^\pi_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1]\right)
\end{aligned}$$

and

$$\begin{aligned}
&\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1] \\
&= \eta(\lambda)\,(\Pr[b = 1 \mid b^* = 0] - \Pr[b = 1 \mid b^* = 1]) \\
&= \eta(\lambda)\,(1 - \Pr[b = 0 \mid b^* = 0] - \Pr[b = 1 \mid b^* = 1]) \\
&= \eta(\lambda)\,(1 - 2\Pr[b = 0 \wedge b^* = 0] - 2\Pr[b = 1 \wedge b^* = 1]) \\
&= 2\eta(\lambda)\left(\frac{1}{2} - \Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger\dagger}_{\mathcal{A}^+,\mathcal{E},\mathcal{G}}(\lambda) = 1]\right) \ .
\end{aligned}$$

As $\eta(\lambda)$ is polynomially bounded (because the client is PPT) and $\mathcal{E}$ is $\mathsf{funcCPA}^{\dagger\dagger}$-secure, it follows that there exists a negligible function negl_1 satisfying that the quantity above is at least $-\mathsf{negl}_1(\lambda)$.

We also define a PPT adversary $\mathcal{A}^-$ for $\mathcal{E}$ in a way that it outputs $b' = 1 - b$ when $\mathcal{A}^+$ outputs b. Then by the argument above, we have

$$\begin{aligned}
&\Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1] \\
&= \eta(\lambda)\,(\Pr[b' = 0 \mid b^* = 0] - \Pr[b' = 0 \mid b^* = 1]) \\
&= \eta(\lambda)\,(\Pr[b' = 0 \mid b^* = 0] - (1 - \Pr[b' = 1 \mid b^* = 1])) \\
&= \eta(\lambda)\,(2\Pr[b' = 0 \wedge b^* = 0] + 2\Pr[b' = 1 \wedge b^* = 1] - 1) \\
&= 2\eta(\lambda)\left(\Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger\dagger}_{\mathcal{A}^-,\mathcal{E},\mathcal{G}}(\lambda) = 1] - \frac{1}{2}\right) \ .
\end{aligned}$$

Again, as $\mathcal{E}$ is $\mathsf{funcCPA}^{\dagger\dagger}$-secure, there exists a negligible function negl_2 satisfying that the quantity above is at most $\mathsf{negl}_2(\lambda)$. Summarizing, we have

$$-\mathsf{negl}_1(\lambda) \leq \Pr[\mathcal{D}^{(z_\lambda)}(\mathsf{view}^{\pi}_{\mathsf{Srv}^*}(\vec{x}^{\,\lambda}, y^\lambda)) = 1] - \Pr[\mathcal{D}^{(z_\lambda)}(\mathcal{S}^*(1^\lambda, y^\lambda)) = 1] \leq \mathsf{negl}_2(\lambda)$$

and the quantity in Eq.(2) is at most the negligible value $\max\{\mathsf{negl}_1(\lambda), \mathsf{negl}_2(\lambda)\}$, a contradiction. Hence the simulator $\mathcal{S}^*$ satisfies the condition showing that the protocol π is private against malicious servers, concluding the proof of Theorem 5. □

From now, we aim at extending Theorem 4 to the case of $\mathsf{funcCPA}^{\dagger\dagger}$-security (and hence implying the privacy against malicious servers for the protocols), by putting some additional assumptions on the underlying scheme $\mathcal{E}$. Let $\mathcal{E}$ be a $\mathcal{C}$-HE scheme with plaintext space $\mathsf{PT} = \{0, 1\}$. For simplifying the argument, we suppose that for each fixed security parameter λ, any secret key sk has a fixed bit length $\nu(\lambda)$, and any ciphertext in CT has a fixed bit length $\rho(\lambda)$. Then for each function g in function family $\mathcal{G}$ that has ℓ inputs from PT and has output in PT, we consider the function $\widetilde{g}_\lambda$ with $(\nu(\lambda)+\ell\cdot\rho(\lambda))$-bit input and one-bit output as follows:

- Given input $(\widetilde{s}, \widetilde{c}_1, \ldots, \widetilde{c}_\ell)$ with $\widetilde{s} \in \{0,1\}^{\nu(\lambda)}$ and $\widetilde{c}_i \in \{0,1\}^{\rho(\lambda)}$, the function $\widetilde{g}_\lambda$ first checks, by using $\widetilde{s}$ regarded as a secret key, if each $\widetilde{c}_i$ belongs to CT or not;
  - if $\widetilde{c}_1, \ldots, \widetilde{c}_\ell \in \mathsf{CT}$, then $\widetilde{g}_\lambda$ decrypts each $\widetilde{c}_i$ by using $\widetilde{s}$ as a secret key to obtain plaintext $\widetilde{m}_i$, and finally outputs $g(\widetilde{m}_1, \ldots, \widetilde{m}_\ell)$;
  - otherwise, $\widetilde{g}_\lambda$ outputs 0.

Let $\widetilde{\mathcal{G}}_\lambda = \{\widetilde{g}_\lambda \mid g \in \mathcal{G}\}$. Based on these definitions, we have the following result:

Theorem 6. *Let $\mathcal{E}$ be a CPA-secure $\mathcal{C}$-HE scheme as above with a sanitization algorithm. Let $\mathcal{G}$ be a function family. Suppose that for each $1 \leq i \leq \nu(\lambda)$, any public key pk for $\mathcal{E}$ involves an element of $\mathsf{CT}_{\mathsf{sk}[i]}$, where $\mathsf{sk}[i]$ denotes the i-th bit of the secret key sk. Suppose moreover that $\widetilde{\mathcal{G}}_\lambda \subseteq \mathcal{C}_\lambda$ for any λ. Then the sanitized scheme $\mathcal{E}^{\mathsf{santz}}$ is $\mathsf{funcCPA}^{\dagger\dagger}$-secure with respect to $\mathcal{G}$.*

Proof. Let $\widetilde{\mathcal{A}}$ be any PPT adversary for the $\mathsf{funcCPA}^{\dagger\dagger}$-security game of $\mathcal{E}^{\mathsf{santz}}$. Then we define an adversary $\mathcal{A}$ for the CPA-security game of $\mathcal{E}$ as follows, where $\widetilde{\mathsf{sk}_i}$ denotes the element of $\mathsf{CT}_{\mathsf{sk}[i]}$ involved in a public key:

- Given a public key pk for $\mathcal{E}$, $\mathcal{A}$ internally executes $\widetilde{\mathcal{A}}$ with input pk.
- When $\widetilde{A}$ makes a query $(g, \vec{c})$ with $\vec{c} = (c_1, \ldots, c_\ell)$ in the $\mathsf{funcCPA}^{\dagger\dagger}$-security game,
  - if some c_i is not a $\rho(\lambda)$-bit sequence, $\mathcal{A}$ sends $\mathsf{Sanitize}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 0))$ back to $\widetilde{\mathcal{A}}$;
  - otherwise, $\mathcal{A}$ generates $\widetilde{c_{i,j}} \leftarrow \mathsf{Enc}(\mathsf{pk}, c_i[j])$ for each $1 \le i \le \ell$ and $1 \le j \le \rho(\lambda)$ where $c_i[j]$ denotes the j-th bit of c_i; and $\mathcal{A}$ computes
$$\mathsf{Sanitize}(\mathsf{pk}, \mathsf{Eval}(\mathsf{pk}, \widetilde{g}_\lambda, \widetilde{\mathsf{sk}_1}, \ldots, \widetilde{\mathsf{sk}_{\nu(\lambda)}}, \widetilde{c_{1,1}}, \ldots, \widetilde{c_{1,\rho(\lambda)}}, \ldots, \widetilde{c_{\ell,1}}, \ldots, \widetilde{c_{\ell,\rho(\lambda)}}))$$
    and sends it back to $\widetilde{\mathcal{A}}$.
- When $\widetilde{\mathcal{A}}$ generates two challenge plaintexts m_0, m_1, $\mathcal{A}$ sends them to the challenger in the CPA-security game and receives a challenge ciphertext c^*; and $\mathcal{A}$ computes $\mathsf{Sanitize}(\mathsf{pk}, c^*)$ and sends it back to $\widetilde{\mathcal{A}}$.
- Finally, when $\widetilde{\mathcal{A}}$ outputs a bit, $\mathcal{A}$ also outputs the same bit.

Note that $\mathcal{A}$ is PPT as well as $\widetilde{\mathcal{A}}$.

For each query $(g, \vec{c})$ made by $\widetilde{\mathcal{A}}$, we compare the distribution of the object returned by $\mathcal{A}$ with the distribution of the reply by the oracle in the true $\mathsf{funcCPA}^{\dagger\dagger}$-security game. Let E denote the event that the key pair $(\mathsf{pk}, \mathsf{sk})$ chosen in the CPA-security game for $\mathcal{E}$ does not satisfy one of the "Sanitizing" condition of Definition 3, the "Correctness" condition for Enc of Definition 1, and the "Correctness" condition for Eval of Definition 2; note that E occurs with probability at most $2^{-\lambda} + \mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda)$ where negl_1 and negl_2 are negligible functions implied by the correctness of Enc and Eval, respectively. From now, we suppose that the event E has not occurred. If some c_i is not a $\rho(\lambda)$-bit sequence, then $c_i \notin \mathsf{CT}$ and the query is invalid. In this case, $\widetilde{\mathcal{A}}$ in the true $\mathsf{funcCPA}^{\dagger\dagger}$-security game receives $\mathsf{Enc}^{\mathsf{santz}}(\mathsf{pk}, 0) = \mathsf{Sanitize}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 0))$, which is identical to the case of the game simulated by $\mathcal{A}$.

From now, we focus on the other case where all $c_1, \ldots, c_\ell$ are $\rho(\lambda)$-bit sequences. We also focus on the case that $\widetilde{c_{i,j}} \in \mathsf{CT}_{c_i[j]}$ for every $1 \le i \le \ell$ and $1 \le j \le \rho(\lambda)$, which occurs with probability at least $1 - \ell \cdot \rho(\lambda) \cdot \mathsf{negl}_1(\lambda)$ due to the correctness of Enc. Now suppose that the oracle returns $\mathsf{Enc}^{\mathsf{santz}}(\mathsf{pk}, \mu)$ in the true $\mathsf{funcCPA}^{\dagger\dagger}$-security game. We divide the situation into the two cases below:

- If all $c_1, \ldots, c_\ell$ are elements of CT, then $\mu = g(\mathsf{Dec}(\mathsf{sk}, c_1), \ldots, \mathsf{Dec}(\mathsf{sk}, c_\ell))$, which is also the same as $\widetilde{g}_\lambda(\mathsf{sk}, c_1, \ldots, c_\ell)$ in this case.
- Otherwise, $\mu = 0$, which is also the same as $\widetilde{g}_\lambda(\mathsf{sk}, c_1, \ldots, c_\ell)$ in this case.

In any case, both $\mathsf{Enc}(\mathsf{pk}, \mu)$ and
$$\mathsf{Eval}(\mathsf{pk}, \widetilde{g}_\lambda, \widetilde{\mathsf{sk}_1}, \ldots, \widetilde{\mathsf{sk}_{\nu(\lambda)}}, \widetilde{c_{1,1}}, \ldots, \widetilde{c_{1,\rho(\lambda)}}, \ldots, \widetilde{c_{\ell,1}}, \ldots, \widetilde{c_{\ell,\rho(\lambda)}})$$
belong to CT_μ with probability at least $1 - \mathsf{negl}_1(\lambda) - \mathsf{negl}_2(\lambda)$; conditioned on this case, the distributions of $\mathsf{Sanitize}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \mu)) = \mathsf{Enc}^{\mathsf{santz}}(\mathsf{pk}, \mu)$ and
$$\mathsf{Sanitize}(\mathsf{pk}, \mathsf{Eval}(\mathsf{pk}, \widetilde{g}_\lambda, \widetilde{\mathsf{sk}_1}, \ldots, \widetilde{\mathsf{sk}_{\nu(\lambda)}}, \widetilde{c_{1,1}}, \ldots, \widetilde{c_{1,\rho(\lambda)}}, \ldots, \widetilde{c_{\ell,1}}, \ldots, \widetilde{c_{\ell,\rho(\lambda)}}))$$
have statistical distance at most $2^{-\lambda}$.

Summarizing, for each query, the distributions of the objects received by $\widetilde{\mathcal{A}}$ in the true and the simulated games have statistical distance at most
$$(2^{-\lambda} + \mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda)) + \ell \cdot \rho(\lambda) \cdot \mathsf{negl}_1(\lambda) + (\mathsf{negl}_1(\lambda) + \mathsf{negl}_2(\lambda)) + 2^{-\lambda}$$

which is negligible; we write it as $\mathsf{negl}_3(\lambda)$. Moreover, note that $\mathcal{A}$ wins the CPA-security game if and only if $\widetilde{\mathcal{A}}$ wins the $\mathsf{funcCPA}^{\dagger\dagger}$-security game simulated by $\mathcal{A}$. Now by putting the (polynomially bounded) number of queries made by $\widetilde{\mathcal{A}}$ as $Q(\lambda)$, it follows that the output distributions of $\widetilde{\mathcal{A}}$ in the true and the simulated games have statistical distance at most $Q(\lambda) \cdot \mathsf{negl}_3(\lambda)$. Hence we have

$$\Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger\dagger}_{\widetilde{\mathcal{A}},\mathcal{E}^{\mathsf{santz}},\mathcal{G}}(\lambda) = 1] \leq \Pr[\mathsf{EXP}^{\mathsf{CPA}}_{\mathcal{A},\mathcal{E}}(\lambda) = 1] + Q(\lambda) \cdot \mathsf{negl}_3(\lambda)$$

where $\mathsf{EXP}^{\mathsf{CPA}}_{\mathcal{A},\mathcal{E}}$ denotes the CPA-security experiment for $\mathcal{E}$ with adversary $\mathcal{A}$. As $\mathcal{E}$ is CPA-secure, we have

$$\Pr[\mathsf{EXP}^{\mathsf{CPA}}_{\mathcal{A},\mathcal{E}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}_4(\lambda)$$

for some negligible function negl_4. Hence

$$\Pr[\mathsf{EXP}^{\mathsf{fCPA}\dagger\dagger}_{\widetilde{\mathcal{A}},\mathcal{E}^{\mathsf{santz}},\mathcal{G}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}_4(\lambda) + Q(\lambda) \cdot \mathsf{negl}_3(\lambda)$$

and $\mathsf{negl}_4(\lambda) + Q(\lambda) \cdot \mathsf{negl}_3(\lambda)$ is negligible. This implies that $\mathcal{E}^{\mathsf{santz}}$ is $\mathsf{funcCPA}^{\dagger\dagger}$-secure, concluding the proof of Theorem 6. □

Acknowledgements This work was supported by JST CREST JPMJCR19F6 and by JSPS KAKENHI Grant Number 19H01109.

References

[1] N. Attrapadung, G. Hanaoka, S. Mitsunari, Y. Sakai, K. Shimizu, and T. Teruya, "Efficient Two-level Homomorphic Encryption in Prime-order Bilinear Groups and A Fast Implementation in WebAssembly", in: Proceedings of ACM ASIACCS 2018, pp.685–697, 2018

[2] A. Akavia and M. Vald, "On the Privacy of Protocols based on CPA-Secure Homomorphic Encryption", IACR Cryptology ePrint Archive, Report 2021/803, 2021, Version 20210616:092806 (accessed on October 20, 2021)

[3] L. Ducas and D. Micciancio, "FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second", in: Proceedings of EUROCRYPT 2015 (Part I), LNCS 9056, pp.617–640, 2015

[4] L. Ducas and D. Stehlé, "Sanitization of FHE Ciphertexts", in: Proceedings of EUROCRYPT 2016 (Part I), LNCS 9665, pp.294–310, 2016

[5] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices", in: Proceedings of ACM STOC 2009, pp.169–178, 2009

[6] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes", in: Proceedings of EUROCRYPT 1999, pp.223–238, 1999

[7] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy Preserving Error Resilient DNA Searching through Oblivious Automata", in: Proceedings of ACM CCS 2007, pp.519–528, 2007

[8] W. Wang, Y. Jiang, Q. Shen, W. Huang, H. Chen, S. Wang, X. Wang, H. Tang, K. Chen, K. Lauter, D. Lin, "Toward Scalable Fully Homomorphic Encryption Through Light Trusted Computing Assistance", arXiv:1905.07766v1, 2019 (accessed on October 20, 2021)