Designs, Codes and Cryptography (2024) 92:1729–1746
https://doi.org/10.1007/s10623-024-01363-4

A conceptually simple and generic construction of plaintext checkable encryption in the standard model

Yu-Chi Chen[1]

Received: 5 April 2023 / Revised: 30 September 2023 / Accepted: 18 January 2024 /
Published online: 24 February 2024

Abstract
Plaintext-checkable encryption (PCE) can support searches over ciphertext by directly using plaintext. The functionality of a search is modeled by a specific check algorithm that takes a pair of target plaintext and ciphertext as input and returns 1 if the correct decryption result of the ciphertext is identical to the target plaintext. A trivial solution is to use an existing scheme (e.g., deterministic RSA) to achieve this, but there is no security guarantee with this method. Previous rigorous works have either relied on some mathematical structures to build PCE that can proven in the standard model or can be generic, as in the random oracle model. Hence, in this work, we aim to construct PCE that can be proven in the standard model by using standard primitives in a modular way in two steps. The first step is to present a warm-up construction of PCE from hash garbling and hash functions whose security is only proven in the random oracle model. The second step is to provide a full-fledged construction based on the warm-up, with slight modifications for achieving security in the standard model. Finally, we show the feasibility of the proposed construction through experiments.

Keywords Cloud storage · Hash garbling · Plaintext checkable encryption · Provable security · Public key encryption

Mathematics Subject Classification 11T71 · 94A60 · 68P25

1 Introduction

Recently, big data analysis techniques have been introduced with cloud aid, as such an overwhelming cost cannot be absorbed by personal computers. Many information technologies and cloud service providers have developed cloud computing applications and platforms. Indeed, the analytics offered by cloud computing from users' data may help predict their

Communicated by K. Matsuura.

✉ Yu-Chi Chen
wycchen@ntut.edu.tw; wycchen@ieee.org

1 Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei 10608, Taiwan

potential activities. However, it is expected to protect some sensitive data with privacy and security concerns, which implies two kinds of issues: privacy and confidentiality. Differential privacy (DP) [8] is a method for quantifying the individual's privacy for queries on a dataset. However, unlike DP, cryptographic solutions aim for confidentiality (e.g., encryption) and usually handle computing over ciphertext.

Plaintext checkable encryption (PCE) can provide simple functionality between ciphertext and plaintext. The concept of PCE was first introduced by [3] and provides a special *check* algorithm (denoted by Check). This algorithm takes target plaintext, ciphertext, and a public key as input, and then outputs *correct* if the ciphertext is an encryption of the target plaintext with the same public key. In real-life applications, the following scenario can be reached by using PCE. Let us take semi-honest[1] cloud storage into account. A user can upload encrypted data associated with some ciphertexts of tags $\mathsf{Enc}(t)$ to a server. A search request is a plain tag t'. Once the server receives t', it can run Check whether the underlying tag of $\mathsf{Enc}(t)$ is identical to the request t'. Returns the corresponding encrypted data whose encrypted tag is $\mathsf{Enc}(t)$ if $\mathsf{Check}(\mathsf{Enc}(t), t')$ returns 1 such that $t = t'$.

PCE was rooted by Canard et al. [3]. Its basic security model is called unlinkable CPA security. In a sense, PCE makes it impossible to meet general CPA security, as the adversary can always access the check algorithm. In unlinkable CPA security, two unlinked adversaries are decoupled to perform the typical CPA game. The follow-up work of Ma et al. [17] formalized the other security notion, $\mathsf{s\text{-}priv1\text{-}cca}$ security, independent of unlinkable CPA security. In addition, they also presented a generic construction of PCE, and applied smooth projective hash as the underlying assumptions. The follow-up works [15, 16] provides a generic construction based on pairing-friendly smooth projective hash functions. They also provide instantiations from k-MDDH and SXDH assumptions, respectively. In particular, [16] is the first scheme offering *verifiably*.

Das et al. [5] modified the framework of PCE for *partially* to achieve CPA security. Their framework only allows the designated checker (who has been delegated check power) to run Check. However, if the adversary is the designed checker, the security is at most unlinkable CPA. Recently, Chen [4] revisited the security notion of PCE and presented a few possible security models and improvements. However, Chen did not aim to construct a pure PCE scheme.

1.1 Contributions

Our motivation comes from underlying assumptions [17] that rely on some mathematical structures (smooth projective hash), while [3] only gave generic constructions in the random oracle model. In this paper, we use standard cryptographic primitives to build a PCE scheme that can be proven in the standard model. To achieve this, we use two phases: the first is building an intermediate notion called the hash garbling (HG) scheme, and the other is building generic constructions of PCE from HG and conventional public key encryption (PKE). The details of our design principles, challenges, and techniques are elaborated as follows.

[1] Semi-honesty means that the cloud server will follow the procedure of the system protocols and algorithms and does not have any malicious behavior, such as tampering.

Table 1 Comparisons

Scheme	Generic	Security	Primitives
CFGL12 [3] scheme 1	Achieve	RO model	Hash function
CFGL12 [3] scheme 2	–	Standard model (claimed)	Bilinear map
MMS18 [17]	–	Standard model (claimed)	Smooth projective hash function
MH19 [15]	Achieve	Standard model	Smooth projective hash function (pairing-friendly)
MHLX19 [16]	Achieve	Standard model	Smooth projective hash functionl-firendly)
Our warm-up	Achieve	RO model	HG
Our construction	Achieve	Standard model	Blind HG

1.1.1 Initial idea for constructing PCE

We want to provide a conceptually simple and generic manner to construct a PCE scheme. Our first attempt keeps the decryption correctness by applying the traditional PKE and then developing an extra, special component (in a ciphertext) that can be used to provide plaintext checkability. Suppose we have a program obfuscation $\mathcal{O}$ [1, 10, 11, 13] that can convert program P into $\widetilde{P}$. $\widetilde{P}$ preserves the functionality of P such that for an input x, $\widetilde{P}(x) = P(x)$ but does not reveal additional information about the code of P. We can prepare a program P that takes a test plaintext M as input and outputs 1 if M is identical to the underlying plaintext of the ciphertext. Our ciphertext is composed of $(\mathsf{Enc}(M), \widetilde{P})$, where $\widetilde{P}$ is as above and Enc is the encryption algorithm of PKE. Unfortunately, obfuscation has been referred to as a non-standard primitive until now, as there is no secure construction based on standard assumptions. This fact forces us to choose the other candidate to realize the special component.

1.1.2 Candidate building block: hash garbling

Recently, hash garbling (HG) [12] has been proposed to provide some properties similar to those of obfuscation. HG is somewhat similar to garbled circuits (GCs). In general, HG consists of a few main algorithms $\mathsf{Hash}, \mathsf{HObf}, \mathsf{HInp}$. Hash is similar to the usual hash function, taking a long input x to return a short output y. HObf is identical to GC, aiming to produce some state information and the GC $\widetilde{P}$ from P. In particular, HInp takes the state and y to output $\widetilde{y}$. However, we need an evaluation that given $x, \widetilde{y}, \widetilde{P}$ returns $P(x)$. Note that HInp does not have any knowledge of the pre-image x, and the evaluation must know x. We present a construction of HG from hash encryption (HE) and GCs, and then prove its simulation security (a.k.a $\langle x, \widetilde{P}, \widetilde{y}\rangle \approx \langle \mathsf{Sim}(x, P(x))\rangle$ informally).

1.1.3 Overview of our warm-up construction: techniques and challenges

Our final goal is to use HG (with PKE) to build the PCE construction. We follow our initial idea to replace obfuscation with HG. For a clear presentation, let us focus on producing the special component in the ciphertext. At first, it is necessary to prepare a program P that hardwires the plaintext m and returns 1 if its input is identical m. The

PCE encryption will generate $\mathsf{Enc}(m)$, $\widetilde{P}$, and $\widetilde{y}$, whose condition on $y = \mathsf{Hash}(m)$. The PCE decryption directly runs Dec of PKE, and Check of PCE relies on evaluation with m', $\widetilde{P}$, and $\widetilde{y}$ for some test plaintext m'. The above-mentioned construction provides checkability but faces a challenge in proving its security. By simulating the security of HG, we can obtain that $\langle \mathsf{Enc}(m_\beta), \widetilde{P}_{m_\beta}, \widetilde{y}_{m_\beta} \rangle$ is computationally indistinguishable from $\langle \mathsf{Enc}(m_\beta), \widetilde{P}_{\mathsf{Sim},m_\beta}, \widetilde{y}_{\mathsf{Sim},m_\beta} \rangle$, where $\widetilde{P}_{\mathsf{Sim},m_\beta}, \widetilde{y}_{\mathsf{Sim},m_\beta}$ are simulated. However, we cannot directly use the CPA-security of PKE to switch $\mathsf{Enc}(m_0)$ to $\mathsf{Enc}(m_1)$, as $\widetilde{P}_{\mathsf{Sim},m_\beta}, \widetilde{y}_{\mathsf{Sim},m_\beta}$ depend on m_β. To overcome this dependency, the encryption algorithm of PCE randomly chooses a number r and returns $(\widetilde{y}', r)$ instead of $\widetilde{y}$, where $\widetilde{y}' = H(m||r) \oplus \widetilde{y}$ with the other hash function H. Let us go back the proof. According to the slight modification, it suffices to obtain $\langle \mathsf{Enc}(m_\beta), \widetilde{P}_{\mathsf{Sim},m_\beta}, H(m||r) \oplus \widetilde{y}_{\mathsf{Sim},m_\beta}, r \rangle \approx \langle \mathsf{Enc}(m_\beta), U_{|\widetilde{P}|+|\widetilde{y}|+|r|} \rangle$ in the random oracle model, where the uniform is denoted by U. The security proof can go through by PKE security to switch $\langle \mathsf{Enc}(m_0), U_{|\widetilde{P}+\widetilde{y}|} \rangle$ to $\langle \mathsf{Enc}(m_1), U_{|\widetilde{P}+\widetilde{y}|} \rangle$.

1.1.4 Construction in the standard model

The above solution is generic and modular but still in the random oracle model. In other words, it achieves the same security as the schemes of [3]. However, this suffices to slightly modify the warm-up construction to a full-fledged one that can be proven in the standard model. Before we show the modification, let us introduce another property of HG: so-called *blindness* [12]. In an HG scheme, blindness means $\langle \widetilde{P}_{\mathsf{Sim}}, \widetilde{y}_{\mathsf{Sim}} \rangle \approx \langle U_{|\widetilde{P}+\widetilde{y}|} \rangle$ when $P(x)$ is uniform. This property inspires us to modify the circuit P in the warm-up, and then we avoid using the random oracle to remedy our proof. Our full-fledged construction includes two slight modifications. One is to set P as a pseudorandom generator [2] that can make $P(x)$ close to uniform, and the other is to verify the evaluation of $(x, \widetilde{P}, \widetilde{y})$ in Check. Recall the security proof where $\langle \mathsf{Enc}(m_\beta), \widetilde{P}_{\mathsf{Sim},m_\beta}, \widetilde{y}_{\mathsf{Sim},m_\beta} \rangle \approx \langle \mathsf{Enc}(m_\beta), U_{|\widetilde{P}|+|\widetilde{y}|} \rangle$ can be directly achieved without the random oracle model. Finally, we need to emphasize that this solution involves the non-black-box use of one-way functions, as the HG must know the codes of pseudorandom generators. Questions of how to use the primitives on the black box will be the subject of our future work on PCE proven in the standard model.

To summarize the results, we briefly compare our schemes with the existing ones in Table 1. All of the schemes satisfy the syntax of PCE, which implies that they can be used to reach the same above-mentioned applications. We do not show any comparison on efficiency, as our schemes may be slower than previous ones (depending on the underlying primitives). The value of our proposal is related to the module construction and its security in the standard model. In practical implementations, the underlying HG may require more space and computation cost (proportional to the program complexity) than the use of a bilinear map. However, our construction is generic and offers *flexibility* for using the primitive; for example, it can be made up of post-quantum cryptographic building blocks against quantum computers.

1.2 Organization

The rest of this paper is organized as follows. In Sect. 2, we introduce cryptography tools and their definitions of security, which will be used throughout this paper. In Sect. 3, we first build a hash garbling scheme and then prove its security based on HE and the GC. In Sect. 4, a warm-up and a non-black-box PCE construction are presented with their security analysis. In Sect. 5, we provide the experiments for implementing our constructions. Finally, Sect. 6 provides the conclusion of this paper.

2 Preliminaries

Prior to presenting the preliminaries, we must state a few notations used in this paper. Let n be the security parameter. We say that for a negligible function negl for any polynomial function $P(n)$ that satisfies an n that is sufficiently large, $\mathsf{negl}(n) \leq \frac{1}{P(n)}$ holds. For a probabilistic polynomial time algorithm $\mathcal{D}$ with security parameter n, we define $|Pr\left[\mathcal{D}(X) = 1\right] - Pr\left[\mathcal{D}(X') = 1\right] \leq \mathsf{negl}(n)$ such that X is indistinguishable from X'. This is also denoted by $\langle X \rangle \overset{c}{\approx} \langle X' \rangle$ as indistinguishability on distribution.

2.1 Public key encryption (PKE)

A PKE scheme consists of three polynomial time algorithms: $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{Gen}(1^n)$: It takes as input the security parameter n and outputs a pair of keys (pk, sk).
- $\mathsf{Enc}(pk, m)$: It takes as input a public key pk and a message m, and outputs a ciphertext c.
- $\mathsf{Dec}(sk, c)$: It takes as input a secret key sk and a ciphertext c, and outputs a message or $\perp$.

Definition 1 (*Security of PKE*) If a PKE scheme is a chosen plaintext attack (CPA) that is secure against any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, then we use $Exp^{CPA}_{\mathcal{A},\mathsf{PKE}}(n)$ security to quantify that for every PPT algorithm $\mathcal{A}$, for all n and any equal length of plaintext input, we have

$$\begin{aligned} Pr\left[Exp^{CPA}_{\mathcal{A},\mathsf{PKE}}(n)\right] &= \left|Pr\left[\mathcal{A}(pk, m_0, m_1, C_{m_0}) = 1\right] - Pr\left[\mathcal{A}(pk, m_0, m_1, C_{m_1}) = 1\right]\right| \\ &\leq \mathsf{negl}(n), \end{aligned}$$

where $\{m_0, m_1\} \overset{\$}{\leftarrow} \mathcal{M}$, $pk \leftarrow \mathsf{Gen}(1^n)$, $C_{m_b} \leftarrow \mathsf{Enc}(pk, m_b)$. We define the advantage of any algorithm $\mathcal{A}$ as the difference of probabilities above. The above formulation is also identical to $\langle pk, m_0, m_1, C_{m_0} \rangle \overset{c}{\approx} \langle pk, m_0, m_1, C_{m_1} \rangle$.

2.2 Plaintext checkable encryption (PCE)

A plaintext checkable encryption (PCE) [3, 4] scheme, PCE, is composed of four polynomial time algorithms. Formally, let $\mathsf{PCE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Check})$.

- $\mathsf{Gen}(1^n)$: Given the security parameter n, it returns a pair of keys (pk, sk).
- $\mathsf{Enc}(pk, m)$: Given a public key pk and a message m, it returns a ciphertext c.
- $\mathsf{Dec}(sk, c)$: Given a secret key sk and a ciphertext c, it returns a message or $\perp$.
- $\mathsf{Check}(pk, m, c)$: Given the public key pk, a message m, and a ciphertext c, it returns 1 if c is an encryption of m, or returns 0 if not.

Definition 2 (*Security of PCE*) A PCE scheme is unlinkable CPA secure against any probabilistic polynomial time adversaries $\mathcal{A}$, which was described by [3]. Here, we use $Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)$ security to quantify that for every PPT algorithm $\mathcal{A}$ for all n and any equal length of plaintext input:

$$\begin{aligned} Pr\left[Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)\right] &= \left|Pr\left[\mathcal{A}(pk, C_{m_0}) = 1\right] - Pr\left[\mathcal{A}(pk, C_{m_1}) = 1\right]\right| \\ &\leq \mathsf{negl}(n), \end{aligned}$$

where $\{m_0, m_1\} \xleftarrow{\$} \mathcal{M}$, $pk \leftarrow \mathsf{Gen}(1^n)$, $C_{m_b} \leftarrow \mathsf{Enc}(pk, m_b)$. Note that there is no input m in unlinkable CPA security. We define the advantage of any algorithm $\mathcal{A}$ as the difference of the probabilities above.

2.3 Garble circuit

A projective circuit garbling (GC) [14, 19] scheme consists of three polynomial time algorithms, where $\mathsf{GC} = (\mathsf{Garble}, \mathsf{GarbleInp}, \mathsf{Eval})$.

- $\mathsf{Garble}(1^n, C)$: This takes as input a security n and a circuit C, and then outputs a GC $\widetilde{C}$ and labels $e_C = \{X_{l,0}, X_{l,1}\}_{l\in[k]}$, where k is the number of input wires of C.
- $\mathsf{GarbleInp}(e_C, x)$: This encodes an $x \in \{0,1\}^k$ with the input labels $e_C = \{X_{l,0}, X_{l,1}\}_{l\in[k]}$ and outputs $\widetilde{x} \leftarrow \{X_{l,X_l}\}_{l\in[k]}$.
- $\mathsf{Eval}(\widetilde{C}, \widetilde{x})$: This takes as input a GC $\widetilde{C}$, and as a garbled input $\widetilde{x}$, and outputs δ.

Definition 3 (*Correctness of GC*) For any circuit C and input $x \in \{0,1\}^k$, the correctness is implied by

$$Pr\left[\mathsf{Eval}(\widetilde{C}, \widetilde{x}) = C(x)\right] = 1,$$

where $(\widetilde{C}, e_c = \{X_{l,0}, X_{l,1}\}) \xleftarrow{\$} \mathsf{Garble}(1^n, C)$ and $\widetilde{x} \leftarrow \mathsf{GarbleInp}(e_C, x)$.

Definition 4 (*Security of GC*) There exists a PPT simulator Sim such that for any circuit C and any input x, we have

$$\langle \widetilde{C}, \widetilde{x} \rangle \overset{c}{\approx} \langle \mathsf{Sim}(1^n, C(x)) \rangle, \tag{1}$$

where $(\widetilde{C}, e_c = \{X_{l,0}, X_{l,1}\}) \xleftarrow{\$} \mathsf{Garble}(1^n, C)$ and $\widetilde{x} \leftarrow \mathsf{GarbleInp}(e_C, x)$.

More generally, we use $Exp^{IND}_{\mathcal{A},\mathsf{GC}}(n)$ security to quantify that for every PPT algorithm $\mathcal{A}$, the Eq. (1) is computationally indistinguishable, so we have

$$\begin{aligned} Pr\left[Exp^{IND}_{\mathcal{A},\mathsf{GC}}(n)\right] &= \left|Pr\left[\mathcal{A}(\widetilde{C}, \widetilde{x}) = 1\right] - Pr\left[\mathcal{A}(\mathsf{Sim}(1^n, C(x))) = 1\right]\right| \\ &\leq \mathsf{negl}(n). \end{aligned}$$

Definition 5 (*Blindness*) The blindness is

$$\langle \mathsf{Sim}(1^n, C(x)) \rangle \overset{c}{\approx} \langle U_{|C(x)|} \rangle.$$

The output of the simulator on a completely uniform output is indistinguishable from a uniform bit string.

2.4 Hash encryption (HE)

An HE [7] scheme consists of four polynomial time algorithms: $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Enc}$, and $\mathsf{Dec})$.

- $\mathsf{Gen}(1^n, m)$: This takes as input the security parameter n and an input length m, and outputs a key k.
- $\mathsf{Hash}(k, x)$: This takes as input a key k and an input $x \in \{0,1\}^m$, and outputs a hash value h of n bits.

- $\mathsf{Enc}(k, (h, i, c), m)$: This takes as input a key k, a hash value h, an index $i \in [m]$, $c \in \{0, 1\}$ and a message $m, \in \{0, 1\}^*$ and outputs a ciphertext ct. We generally assume that the index i and the bit c are included alongside.
- $\mathsf{Dec}(k, x, ct)$: This takes as inputs a key k, an input x, and a ciphertext ct, and outputs a value $m \in \{0, 1\}^*$ or $\perp$.

Definition 6 (*Correctness of HE*) For any input $x \in \{0, 1\}^m$, index $i \in [m]$, the correctness is implied by

$$Pr\left[\mathsf{Dec}(k, x, \mathsf{Enc}(k, (\mathsf{Hash}(k, x), i, x_i), m))\right] \geq 1 - \mathsf{negl}(n),$$

where x_i denotes the ith bit of x, and the randomness is taken over $k \leftarrow \mathsf{Gen}(1^n, m)$.

Definition 7 (*Security of HE*) If an HE is selectively indistinguishable secure against any probabilistic polynomial time algorithm $\mathcal{A}$, then we use $Exp^{IND}_{\mathcal{A},\mathsf{HE}}(n)$ security to quantify that for every PPT algorithm $\mathcal{A}$, for all n, the length of m and any equal length of plaintext input, we have

$$\begin{aligned} Pr\left[Exp^{IND}_{\mathcal{A},\mathsf{HE}}(n)\right] &= \left|Pr\left[\mathcal{A}(k, x, ct_{m_0}) = 1\right] - Pr\left[\mathcal{A}(k, x, ct_{m_1}) = 1\right]\right| \\ &\leq \mathsf{negl}(n), \end{aligned}$$

where $\{m_0, m_1\} \xleftarrow{\$} \mathcal{M}$, $k \leftarrow \mathsf{Gen}(1^n, m)$ and $ct_{m_b} \leftarrow \mathsf{Enc}(k, (\mathsf{Hash}(k, x), i, 1 - x_i), m_b)$. We define the advantage of any algorithm $\mathcal{A}$ as the difference of the probabilities above.

Definition 8 (*Blindness*) The blindness is

$$\langle k, x, \mathsf{Enc}(k, (h, i, c)), m)\rangle \overset{c}{\approx} \langle k, x, U_{|ct|}\rangle,$$

where m is a uniform bit string.

2.5 Hash garbling

An HG [12] scheme consists of five polynomial time algorithms, $\mathsf{HG} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{HObf}, \mathsf{HInp}$, and $\mathsf{Eval})$.[2]

- $\mathsf{Gen}(1^n, k)$: This takes as input the security parameter n and an input length parameter k for $k \leq poly(n)$, and outputs a hash key hk. (Gen runs in poly(n) time.)
- $\mathsf{Hash}(hk, x)$: This takes as input hk and $x \in \{0, 1\}^k$, and outputs a value $y \in \{0, 1\}^n$.
- $\mathsf{HObf}(hk, C)$: This takes as input hk and a circuit C, and outputs a secret state $st \in \{0, 1\}^n$ and a circuit $\widetilde{C}$.
- $\mathsf{HInp}(hk, y, st)$: This takes as input hk, y, st and outputs $\widetilde{y}$.
- $\mathsf{Eval}(\widetilde{C}, \widetilde{y}, x)$: This takes as input a GC $\widetilde{C}$ and the value $\widetilde{y}$ and x, and outputs δ.

Definition 9 (*Correctness of HG*) For all $n, k, hk \leftarrow \mathsf{Gen}(1^n, k)$, circuit C, input $x \in \{0, 1\}^k$, $st \in \{0, 1\}^n$, $\widetilde{C} \leftarrow \mathsf{HObf}(hk, C, st)$ and $\widetilde{y} \leftarrow \mathsf{HInp}(hk, \mathsf{Hash}(hk, x), st)$, we have

$$Pr\left[\mathsf{Eval}(\widetilde{C}, \widetilde{y}, x) = C(x)\right] = 1.$$

[2] Here, we slightly modify the definition of HG proposed by [12] and change the original $\mathsf{HObf}(hk, C, st)$ to $\mathsf{HObf}(hk, C)$. This change does not affect the implementation or correctness of HG, but facilitates the subsequent presentation.

Table 2 Labels e_C

c	i			
	1	2	...	k
0	$X_{1,0}$	$X_{2,0}$	$X_{l,0}$	$X_{k,0}$
1	$X_{1,1}$	$X_{2,1}$	$X_{l,1}$	$X_{k,1}$

Definition 10 (*Security of HG*) There exists a PPT simulator Sim such that for all n, k and PPT (in n) $\mathcal{A}$ we have

$$\langle hk, x, \widetilde{C}, \widetilde{y}\rangle \overset{c}{\approx} \langle hk, x, \mathsf{Sim}(hk, x, 1^{|C|}, C(x))\rangle, \tag{2}$$

where $hk \leftarrow \mathsf{Gen}(1^n, k)$, $(C, x) \leftarrow \mathcal{A}(hk)$, $(\widetilde{C}, st) \leftarrow \mathsf{HObf}(hk, C)$, $st \leftarrow \{0,1\}^n$ and $\widetilde{y} \leftarrow \mathsf{HInp}(hk, \mathsf{Hash}(hk, x), st)$.

More generally, we use $Exp^{IND}_{\mathcal{A},\mathsf{HG}}(n)$ security to quantify that for every PPT algorithm $\mathcal{A}$, (2) is computationally indistinguishable, so we have

$$Pr\left[Exp^{IND}_{\mathcal{A},\mathsf{HG}}(n)\right] = |Pr\left[\mathcal{A}(hk, x, \widetilde{C}, \widetilde{y}) = 1\right] - Pr\left[\mathcal{A}(hk, x, \mathsf{Sim}(hk, x, 1^{|C|}, C(x)) = 1\right]| \\ \leq \mathsf{negl}(n).$$

Definition 11 (*Weak security of HG*) There exists a PPT simulator Sim such that for all n, k and PPT (in n) $\mathcal{A}$, we have the following weak notion

$$\langle hk, \widetilde{C}, \widetilde{y}\rangle \overset{c}{\approx} \langle hk, \mathsf{Sim}(hk, x, 1^{|C|}, C(x))\rangle.$$

Note that there is no input x in the weak security. If a scheme meets the security of HG, then it absolutely meets the weak security.

Definition 12 (*Blindness*) The blindness is

$$\langle hk, x, \mathsf{Sim}(hk, x, 1^{|C|}, C(x))\rangle \overset{c}{\approx} \langle hk, x, U_{|\widetilde{C}|+|\widetilde{y}|}\rangle, \tag{3}$$

where the output distribution of $C(x)$ is uniform.

Definition 13 (*Weak blindness*) To undertake Definition 12, we have the following weak notion:

$$\langle hk, \mathsf{Sim}(hk, x, 1^{|C|}, C(x))\rangle \overset{c}{\approx} \langle hk, U_{|\widetilde{C}|+|\widetilde{y}|}\rangle.$$

Note that there is no input x in the weak blindness of HG.

3 The HG scheme

3.1 Construction

Let GC = (Garble, GarbleInp, Eval′) and HE = (Gen″, Hash″, Enc, Dec) be the secure GC and HE. We present the construction of HG, which consists of the following algorithms (Gen, Hash, HObf, HInp, Eval).

- $\mathsf{Gen}(1^n, k)$: This takes as input the security parameter n and the number of input wires k of a circuit C and computes $hk \leftarrow \mathsf{Gen}''(1^n, k)$. It outputs a hash key hk.

– $\mathsf{Hash}(hk, x)$: This takes as input hk and $x \in \{0,1\}^k$ and computes as follows:

$$y \leftarrow \mathsf{Hash}''(hk, x).$$

It outputs a value $y \in \{0,1\}^n$.

– $\mathsf{HObf}(hk, C)$: This takes as input hk and a circuit C and computes as follows:

$$(\widetilde{C}, e_C) \leftarrow \mathsf{Garble}(hk, C).$$

This outputs a GC $\widetilde{C}$ and labels $e_C = \{X_{l,0}, X_{l,1}\}_{l \in [k]}$, where $X_{l,*} \in \{0,1\}^n$.

– $\mathsf{HInp}(hk, y, e_C)$: This takes as input hk, y, e_C and computes the following:

$$\widetilde{y} \leftarrow \mathsf{Enc}(hk, (y, i, c), e_C),$$

where i represents the serial number of $x, c \in \{0,1\}$. Hence, it outputs the value of $\widetilde{y} = (c_{1,*}, \ldots, c_{l,*} \ldots, c_{k,*})$.
According to Table 2, each $c_{l,*}$ is calculated as follows:

$$c_{l,0} \leftarrow \mathsf{Enc}(hk, (y, l, 0), X_{l,0}).$$
$$c_{l,1} \leftarrow \mathsf{Enc}(hk, (y, l, 1), X_{l,1}).$$

– $\mathsf{Eval}(\widetilde{C}, \widetilde{y}, x)$: This takes as input $\widetilde{C}$, $\widetilde{y}$, x and computes as follows:

$$e_C{}' \leftarrow \mathsf{Dec}(hk, x, \widetilde{y}).$$
$$\widetilde{x} \leftarrow \mathsf{GarbleInp}(e_C{}', x).$$
$$\delta \leftarrow \mathsf{Eval}'(\widetilde{C}, \widetilde{x}).$$

From Definition 3, it is equivalent to output $C(x)$.

Correctness: From Definition 6 and Table 2, we know that for

$$ct \leftarrow \mathsf{Enc}(hk, (\mathsf{Hash}''(hk, x), i, 1 - c), X_{i,c}),$$

the adversary $\mathcal{A}$ cannot distinguish $X_{i,c}$. In other words, for the HInp of $x \in \{0,1\}^k$'s i-bit c, Dec can only get the corresponding label $X_{i,c}$. For example, if the value of $x = 1011$, $\mathcal{A}$ is only able to get the labels $X_{1,1}, X_{2,0}, X_{3,1}, X_{4,1}$, but it has no information about other labels.

3.2 Security analysis

Theorem 1 *The HG construction meets simulation security* (*Definition* 10)*, assuming the underlying hash encryption and garbled circuit are secure.*

Proof To show that the HG construct meets the security of Eq. (2), we need to prove that the output of a PPT simulator ($\mathsf{Sim}_{\mathsf{HG}}$), which represents the right side of (2), is indistinguishable from that of HG. Moreover, $\mathsf{View}_{\mathsf{HG}}$ represents the left side of the (2). To do this, we have to define a sequence of hybrids ($\mathsf{Hyb}_0, \mathsf{Hyb}_1, \mathsf{Hyb}_2$) and demonstrate that $\mathsf{Sim}_{\mathsf{HG}}$ is computationally indistinguishable from the output of $\mathsf{View}_{\mathsf{HG}}$ by proving the relationship between the following views:

$$\mathsf{View}_{\mathsf{HG}} \equiv \mathsf{Hyb}_0 \approx \mathsf{Hyb}_1 \approx \mathsf{Hyb}_2 \equiv \mathsf{Sim}_{\mathsf{HG}}.$$

Similar to our presentation of Construction 3.1, here, all views use the same security parameters 1^n and k, and the value of $x \in \{0,1\}^k$ is fixed.

- Hyb_0 (encrypt in real game): We make the output of Hyb_0 the same as $\mathsf{View}_{\mathsf{HG}}$, and obviously, $\mathsf{View}_{\mathsf{HG}} \equiv \mathsf{Hyb}_0$.
- Hyb_1: Based on Hyb_0, we modify the value of $\widetilde{y}$ for the bit corresponding to the x value at the l-th value, assuming that 1, $(1 = c \leftarrow x_l)$, and the generated ciphertext is $c_{l,1}$. We keep the following unchanged:

$$c_{l,1} \leftarrow \mathsf{Enc}(hk, (y, l, 1), X_{l,1}).$$

In contrast, for another label $X_{l,0}$, we set it to 0, namely:

$$c_{l,0} \leftarrow \mathsf{Enc}(hk, (y, l, 0), 0).$$

We do the same encryption process according to the x in $\{0, 1\}^k$ string corresponding to the label $X_{l,*}$ in e_C Table 2. If we assume that the value of x is 1011, the corresponding $\widetilde{y}'$ value should be as follows:

$$\widetilde{y}' = \begin{pmatrix} c'_{1,0} & c_{2,0} & c'_{3,0} & c'_{4,0} \\ c_{1,1} & c'_{2,1} & c_{3,1} & c_{4,1} \end{pmatrix}.$$

- Hyb_2: Based on Hyb_1, we modify the value of $\widetilde{x}$ and $\widetilde{C}$, and for each $l \in [0, k]$, in the x path, we use the form of a simulator to generate $\widetilde{C}_{\mathsf{Sim}}$ and $\widetilde{x}_{\mathsf{Sim}}$ for the corresponding label e_C, namely:

$$(\widetilde{C}_{\mathsf{Sim}}, \widetilde{x}_{\mathsf{Sim}}) \leftarrow \mathsf{Sim}(1^n, C(x)).$$

Note that for the composition of $\widetilde{x}_{\mathsf{Sim}}$, for the bit corresponding to the x value at the l-th value, assuming that 1. We use the label $X_{l,1_{\mathsf{Sim}}}$ to indicate. For example, if the value of x is 1011 (consistent with Hyb_1), the composition of e_C should be as follows:

$$e_c = \begin{pmatrix} X_{1,0} & X_{2,0_{\mathsf{Sim}}} & X_{3,0} & X_{4,0} \\ X_{1,1_{\mathsf{Sim}}} & X_{2,1} & X_{3,1_{\mathsf{Sim}}} & X_{4,1_{\mathsf{Sim}}} \end{pmatrix}.$$

The label $X_{l,*_{\mathsf{Sim}}}$ is generated by the simulator Sim, and the rest is generated by the $\mathsf{GarbleInp}$ algorithm (which is consistent with Hyb_1). Therefore, $\widetilde{x}_{\mathsf{Sim}}$ is composed as follows:

$$\widetilde{x}_{\mathsf{Sim}} = \begin{pmatrix} \widetilde{x}_{1,0} & \widetilde{x}_{2,0_{\mathsf{Sim}}} & \widetilde{x}_{3,0} & \widetilde{x}_{4,0} \\ \widetilde{x}_{1,1_{\mathsf{Sim}}} & \widetilde{x}_{2,1} & \widetilde{x}_{3,1_{\mathsf{Sim}}} & \widetilde{x}_{4,1_{\mathsf{Sim}}} \end{pmatrix}.$$

Hence, the value of the corresponding $\widetilde{y}_{\mathsf{Sim}}$ is expressed as follows:

$$\widetilde{y}_{\mathsf{Sim}} = \begin{pmatrix} c'_{1,0} & c_{2,0_{\mathsf{Sim}}} & c'_{3,0} & c'_{4,0} \\ c_{1,1_{\mathsf{Sim}}} & c'_{2,1} & c_{3,1_{\mathsf{Sim}}} & c_{4,1_{\mathsf{Sim}}} \end{pmatrix}.$$

Clearly, $\mathsf{Hyb}_2 \equiv \mathsf{Sim}_{\mathsf{HG}}$.

Lemma 1 *Assuming that HE is selectively indistinguishable secure (Definition 7), hybrid views Hyb_0 and Hyb_1 are computationally indistinguishable.*

Proof To prove $\mathsf{Hyb}_0 \overset{c}{\approx} \mathsf{Hyb}_1$, we use the output of Hyb_0 and Hyb_1 as follows:

$$\langle hk, x, \widetilde{C}, \widetilde{y} \rangle \overset{c}{\approx} \langle hk, x, \widetilde{C}, \widetilde{y}' \rangle. \tag{4}$$

The difference between the two sides of the (4) is $\widetilde{y}$ and $\widetilde{y}'$. More specifically, the difference between the two sides of (4) should be information that is not encrypted in the x path $(1-x_{l,*})$, such as $c'_{1,0}$ and $c_{1,0}$, to determine whether the encrypted information is labeled $X_{l,0}$ or 0. In other words, a PPT algorithm $\mathcal{A}$ needs to distinguish the following equation:

$$ct \leftarrow \mathsf{Enc}(hk, (y, i, 1 - x_{l,*}), m_b). \tag{5}$$

Equation (5) translates to prove $Exp^{IND}_{\mathcal{A},\mathsf{HE}}$ security. From Definition 7, we know that the advantage of $\mathcal{A}$ is $\mathsf{negl}(n)$, and the proof is completed. □

Lemma 2 *Assuming that GC meets simulation security (Definition 4), hybrid views Hyb_1 and Hyb_2 are computationally indistinguishable.*

Proof To prove $\mathsf{Hyb}_1 \overset{c}{\approx} \mathsf{Hyb}_2$, we use the output of Hyb_1 and Hyb_2 as follows:

$$\langle hk, x, \widetilde{C}, \widetilde{y}' \rangle \overset{c}{\approx} \langle hk, x, \widetilde{C}_{\mathsf{Sim}}, \widetilde{y}_{\mathsf{Sim}} \rangle. \tag{6}$$

The difference between the two sides of (6) is $(\widetilde{C}, \widetilde{y}')$ and $(\widetilde{C}_{\mathsf{Sim}}, \widetilde{y}_{\mathsf{Sim}})$. More specifically, the difference between the two sides of (6) is the difference in the way $\widetilde{x}$ is generated, that is, whether $\widetilde{x}$ is generated by $\mathsf{GarbleInp}$ or generated by the simulator on the x path $(x_{l,*})$. In other words, a PPT algorithm $\mathcal{A}$ needs to distinguish the following equation:

$$\langle \widetilde{C}, \widetilde{x} \rangle \overset{c}{\approx} \langle \widetilde{C}_{\mathsf{Sim}}, \widetilde{x}_{\mathsf{Sim}} \rangle. \tag{7}$$

Equation (7) translates to prove $Exp^{IND}_{\mathcal{A},\mathsf{GC}}$ security. From Definition 4, we know that the advantage of $\mathcal{A}$ is $\mathsf{negl}(n)$, and the proof is completed. □

This proof is done by proving Lemmas 1 and 2. □

Theorem 2 *The HG construction meets blind security (Definition 12), assuming that the underlying HE and GC are blind security.*

Proof To show that the HG construction meets the security of (3), we use View_{BHG} to represent the right side of the (3) and Sim_{BHG} to represent the left side of (3). For the simulator Sim described above, consider the distribution of $\mathsf{Sim}(hk, x, 1^{|C|}, C(x))$ for a uniformly generated output. By the security of a blind GC (Definition 5), we know the GC simulator $\langle \mathsf{Sim}(1^n, C(x)) \rangle \overset{c}{\approx} \langle U_{|C(x)|} \rangle$. Hence, for each $l \in [k]$, $X_{l,*} \overset{c}{\approx} U$. Thus, by the security of blind HE (Definition 8), $\widetilde{y} = \mathsf{Enc}(hk, (h, i, c), X_{l,*})$, we have $\langle hk, x, \widetilde{y} \rangle \overset{c}{\approx} \langle hk, x, U_{|\widetilde{y}|} \rangle$. Hence, it follows that $\mathsf{Sim}_{BHG} \overset{c}{\approx} \mathsf{View}_{BHG}$. □

4 The proposed PCE scheme

4.1 Warm-up construction

Let $\mathsf{PKE} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ and $\mathsf{HG} = (\mathsf{Gen}'', \mathsf{HObf}, \mathsf{Hash}, \mathsf{HInp}, \mathsf{Eval})$ be the secure public key encryption and HG. The proposed construction of PCE is composed of the following algorithms.

– $\mathsf{Gen}(1^\lambda)$: It takes as input the security parameter λ, which contains two parameters, $(1^n, k)$, and computes the following:

$$(pk', sk') \leftarrow \mathsf{Gen}'(1^n).$$
$$hk \leftarrow \mathsf{Gen}''(1^n, k).$$

Finally, it outputs the public/private key pair (pk, sk), where $pk = (pk', hk)$, $sk = sk'$.

– $\mathsf{Enc}(pk, m)$: This takes as input pk and a message $m \in \{0, 1\}^*$, and computes the following:

$$\begin{aligned} &\mathcal{C} \leftarrow \mathsf{Enc}'(pk', m) \\ &(\widetilde{P}, e_C) \leftarrow \mathsf{HObf}(hk, P) \\ &y \leftarrow \mathsf{Hash}(hk, m). \\ &\widetilde{y_m} \leftarrow \mathsf{HInp}(hk, y, e_C) \\ &t = \widetilde{y_m} \oplus H(m||r) \\ &\widetilde{y} = (t, r), \end{aligned}$$

where r is a random number, $H(\cdot)$ is a cryptographic hash function that supports input of any length, P is explained below, and the generation of a secret state $e_C \in \{0, 1\}^n$ can be seen in Construction 3.1.

The program P is defined below:

– **Hardwired**: m
– **Input**: x
1. if $x = m$, output 1.
2. if $x \neq m$, output $\perp$.

Finally, it outputs ciphertext $C = (\mathcal{C}, \widetilde{P}, \widetilde{y})$.

– $\mathsf{Dec}(sk, C)$: It takes as input the secret key sk and $\mathcal{C}$, and computes the following:

$$m = \mathsf{Dec}'(sk', \mathcal{C}).$$

– $\mathsf{Check}(pk, C, m)$: In order to check the correctness of m, this is calculated as follows:

$$\begin{aligned} &\widetilde{y_m} \leftarrow t \oplus H(m||r) \\ &b \leftarrow \mathsf{Eval}(\widetilde{P}, \widetilde{y_m}, m), \end{aligned}$$

where $b \in \{\perp, 1\}$ ($b = 1$ if C is an encryption of m, or $b = \perp$ if not).

4.1.1 Correctness

In Dec, it is held by $m = \mathsf{Dec}'(sk', \mathsf{Enc}'(pk', m))$, the correctness of the underlying decryption of PKE.

In Check, taking (t, r) from C and the message m^* and computing $H(m^*||r)$, it is held $\widetilde{y_m^*} = t \oplus H(m^*||r)$. With the hash input encoding $\widetilde{y_m^*}$, we can obtain $\widetilde{y_m^*} \leftarrow \mathsf{HInp}(hk, \mathsf{Hash}(hk, m^*), e_C)$. If $\widetilde{P}$ with hardcoded m and e_C are generated by $\mathsf{HObf}(hk, P)$, $\mathsf{Eval}(\widetilde{P}, \widetilde{y_m^*}, m^*)$ will output 1 if and only if m^* is the same with the hardcoded m in P.

4.1.2 Security proof

The security of the warm-up is stated with the following theorem.

Theorem 3 *The above generic construction of PCE satisfies unlinkable CPA security in the random oracle model if the underlying public key encryption is CPA secure and the security of HG meets Definition 11.*

Proof At a high level, we assume that there is a PPT adversary $\mathcal{A}$, which breaks the $Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)$ security, and then we can create the PPT algorithm $\mathcal{A}$, which also breaks $Exp^{CPA}_{\mathcal{A},\mathsf{PKE}}(n)$ or $Exp^{IND}_{\mathcal{A},\mathsf{HG}}(n)$ security. However, for completing the proof, we start by defining an event HIT where the adversary $\mathcal{A}_2$ exactly accesses m_0 or m_1 to Check. This suffices to obtain the following result:

$$\begin{aligned} Pr\left[Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)=1\right] &= Pr\left[Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)=1\wedge \mathsf{HIT}\right] \\ &\quad + Pr\left[Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)=1\wedge \overline{\mathsf{HIT}}\right] \\ &\le Pr\left[\mathsf{HIT}\right] + Pr\left[Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)=1\wedge \overline{\mathsf{HIT}}\right]. \end{aligned}$$

We claim $Pr[\mathsf{HIT}]=\frac{2}{2^{\ell(n)}}\le \mathsf{negl}(n)$, as m_0, m_1 are $\ell(n)$-bit, where ℓ is some polynomial.

The rest of the proof focuses on $Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)=1\wedge\overline{\mathsf{HIT}}$. We abuse the notation $\widetilde{y}_{(m_\beta)}$ as $\widetilde{y}$ with $t=\widetilde{y_{m_\beta}}\oplus H(m_\beta||r)$. Our goal is to show that $\langle pk', \mathsf{Enc}'(m_0), hk, \widetilde{P}_{m_0}, \widetilde{y}_{(m_0)}\rangle$ is indistinguishable to $\langle pk', \mathsf{Enc}'(m_1), hk, \widetilde{P}_{m_1}, \widetilde{y}_{(m_1)}\rangle$, where $\widetilde{P}_{m_\beta}, \widetilde{y}_{m_\beta}$ denotes the underlying garbled encoding for plaintext m_β. For short, denote by $\langle \mathsf{Enc}'(m_0), \widetilde{P}_{m_0}, \widetilde{y}_{(m_0)}\rangle \approx \langle \mathsf{Enc}'(m_1), \widetilde{P}_{m_1}, \widetilde{y}_{(m_1)}\rangle$ with public pair of (pk', hk). We apply the standard hybrid arguments to complete the proof. At the beginning, start by defining two top-level hybrids, Hyb_0 and Hyb_1,[3].

- Hyb_β: This experiment is identical to $Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n)$, except the challenge ciphertext is $\langle \mathsf{Enc}'(m_\beta), \widetilde{P}_{m_\beta}, \widetilde{y}_{m_\beta}\rangle$.

Note that Hyb_β is identical to $\langle pk', \mathsf{Enc}'(m_\beta), hk, \widetilde{P}_{m_\beta}, \widetilde{y}_{m_\beta}\rangle$. In addition, we further define a few hybrids $\mathsf{Hyb}_{\beta,0}$, $\mathsf{Hyb}_{\beta,1}$ as follows (with $\beta\in\{0,1\}$):

- $\mathsf{Hyb}_{\beta,1}$: This is similar to Hyb_β, except $\widetilde{P}_{m_\beta}, \widetilde{y}_{m_\beta}$ is replaced with $\mathsf{Sim}(hk, m_\beta, 1^{|P_{m_\beta}|}, P_{m_\beta}(m_\beta))$.
- $\mathsf{Hyb}_{\beta,2}$: This is similar to $\mathsf{Hyb}_{\beta,1}$, except the simulated part is replaced with $U_{|\widetilde{P}|+|\widetilde{y}|}$.

To achieve $\langle pk', \mathsf{Enc}'(m_0), hk, \widetilde{P}_{m_0}, \widetilde{y}_{m_0}\rangle \overset{c}{\approx} \langle pk', \mathsf{Enc}'(m_1), hk, \widetilde{P}_{m_1}, \widetilde{y}_{m_1}\rangle$, a sequence of hybrids is denoted by

$$\mathsf{Hyb}_0\approx\mathsf{Hyb}_{0,1}\approx\mathsf{Hyb}_{0,2}\approx\mathsf{Hyb}_{1,2}\approx\mathsf{Hyb}_{1,1}\approx\mathsf{Hyb}_1.$$

For each neighboring hybrid, we can use an assumption of security to complete the reduction. We directly state the following lemmas, and refer to the missing proofs of Lemmas 3 and 5 in Appendix.

Lemma 3 *If the underlying hash garbling scheme meets weak security (Definition 11), then no poly-time adversary can distinguish with non-negligible probability between* Hyb_0 *and* $\mathsf{Hyb}_{0,1}$.

Lemma 4 *No poly-time adversary can distinguish with non-negligible probability between* $\mathsf{Hyb}_{0,1}$ *and* $\mathsf{Hyb}_{0,2}$ *in the random oracle model.*

The distributions of $\mathsf{Hyb}_{0,1}$ and $\mathsf{Hyb}_{0,2}$ in the random oracle model are identical. Details are shown below. Trivially, we have $\langle \widetilde{P}, r, \widetilde{y_m}\oplus H(m||r)\rangle \equiv \langle \widetilde{P}, r, U_{|\widetilde{y_m}|}\rangle$ in the random

[3] Hyb_β is $\langle \mathsf{Enc}'(m_\beta), \widetilde{P}_{m_\beta}\widetilde{y}_{(m_\beta)}\rangle$

oracle. It is easy to obtain $\langle \widetilde{P}, r, U_{|y_m|} \rangle \equiv \langle \widetilde{P}, U_{|\widetilde{y}|} \rangle$ with identical distribution. Finally, after cutting the correlation from $\widetilde{P}$ to $\widetilde{y_m}$, we conclude that $\widetilde{P}$ is not evaluable, and further obtain $\langle \widetilde{P}, U_{|\widetilde{y}|} \rangle \equiv \langle U_{|\widetilde{P}+\widetilde{y}|} \rangle$.

Lemma 5 *If our public key encryption scheme is CPA-secure (Definition 1), then no poly-time adversary can distinguish with non-negligible probability between* $\mathsf{Hyb}_{0,2}$ *and* $\mathsf{Hyb}_{1,2}$.

These lemmas can be used to show $\mathsf{Hyb}_{1,2} \approx \mathsf{Hyb}_{1,1} \approx \mathsf{Hyb}_1$ with the same manner. Finally, we conclude that $Pr[Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n) = 1 \wedge \overline{\mathsf{HIT}}] \leq \mathsf{negl}(n)$, which implies $Pr[Exp^{unlink}_{\mathcal{A},\mathsf{PCE}}(n) = 1] \leq \mathsf{negl}(n)$, completing the proof.

Here, we sketch the idea of using the random oracle for Lemma 4. Let us focus on $\langle \widetilde{P}_{\mathsf{Sim},m_\beta}, H(m_\beta||r) \oplus \widetilde{y}_{\mathsf{Sim},m_\beta}, r \rangle$; it cannot be flipped to uniform, as $\widetilde{y}_{\mathsf{Sim},m_\beta}$ depends on $\widetilde{P}_{\mathsf{Sim},m_\beta}$ and $H(m_\beta||r)$ on r. The main technique is to model H as a random oracle for cutting such a relationship. In the random oracle model, we directly have $\langle \widetilde{P}_{\mathsf{Sim},m_\beta}, U_{|\widetilde{y}|} \oplus \widetilde{y}_{\mathsf{Sim},m_\beta}, r \rangle$, as for each input $m||r$, the random oracle H uniformly determines a random number. It is clear to obtain $\langle \widetilde{P}_{\mathsf{Sim},m_\beta}, U_{|\widetilde{y}|}, r \rangle$. Finally, without existing $\widetilde{y}$, we can claim $\widetilde{P}_{\mathsf{Sim},m_\beta}$ as uniform $U_{|\widetilde{P}|}$. The proof sketch is done with $|r| \geq |\widetilde{y}|$, which guarantees uniform r can span all possible values on $\widetilde{y}$. □

4.2 Non-black-box PCE construction in the standard model

Let $\mathsf{PKE} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ and $\mathsf{HG} = (\mathsf{Gen}'', \mathsf{HObf}, \mathsf{Hash}, \mathsf{HInp}, \mathsf{Eval})$ be the secure PKE and HG. The final construction of PCE is almost identical to the warm-up. In the following, only the modifications are shown.

– $\mathsf{Enc}(pk, m)$: It takes as input pk and a message $m \in \{0, 1\}^*$, and computes the following:

$$\begin{aligned} &\mathcal{C} \leftarrow \mathsf{Enc}'(pk', m) \\ &(\widetilde{P}, e_C) \leftarrow \mathsf{HObf}(hk, P) \\ &y \leftarrow \mathsf{Hash}(hk, m) \\ &\widetilde{y} \leftarrow \mathsf{HInp}(hk, y, e_C). \end{aligned}$$

The P is explained below, and the generation of a secret state $e_C \in \{0, 1\}^n$ can be seen in Construction 3.1.
The program P is defined below:

– **Hardwired**: m, r
– **Input**: x
1. if $x = m$, output $\mathsf{PRG}(x)$.
2. if $x \neq m$, output $\mathsf{PRG}(x \oplus r)$,

where r is a random number, and PRG is a pseudorandom generator [2]. Finally, it outputs ciphertext $C = (\mathcal{C}, \widetilde{P}, \widetilde{y})$.

– $\mathsf{Check}(pk, C, m)$: In order to check the correctness of m, it is calculated as follows:

$$\mathsf{PRG}(m) \stackrel{?}{=} \mathsf{Eval}(\widetilde{P}, \widetilde{y}, m). \tag{8}$$

If C is encrypted by m, (8) holds, and vice versa.

We say that the above modifications can lift the construction to be secure in the standard model. However, the algorithm HObf must know the code of PRG precisely, which implies

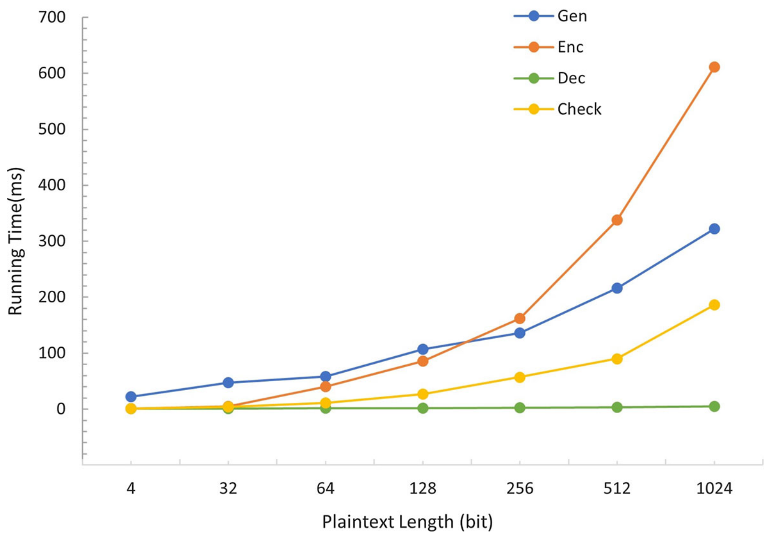

Fig. 1 Implementation result of performance of PCE algorithms

the construction is of non-black-box use for one-way functions. In the following proof, we omit to repeat the steps of the proof of Theorem 3. Here, we briefly describe the security argument by stating a lemma that is different from Lemma 4.

Lemma 6 *Assuming that HG meets week blindness (Definition 13), then no poly-time adversary can distinguish with non-negligible probability between* $\mathsf{Hyb}_{0,1}$ *and* $\mathsf{Hyb}_{0,2}$.

Proof In a nutshell, we need to prove $\langle pk', \mathsf{Enc}'(m_0), hk, \mathsf{Sim}(hk, m_0, 1^{|P_{m_0}|}, P_{m_0}(m_0))\rangle \overset{c}{\approx} \langle pk', \mathsf{Enc}'(m_0), hk, U_{|\widetilde{p}|+|\widetilde{y}|}\rangle$, from the proposed construction. Recall that the result of $P_{m_0}(x)$ is close to uniform, as the output of PRG is close to uniform. Thus, $\mathsf{Hyb}_{0,1} \overset{c}{\approx} \mathsf{Hyb}_{0,2}$ can be easily achieved as long as the HG meets weak blindness security without any random oracle. □

5 Experiments

In this section, we present the experimental evaluation of our PCE construction, which mainly consists of two cryptographic primitives. The first one is PKE, so we use the ElGamal algorithm [9] based on the decision Diffie–Hellman assumption [18]. The second is HG, which consists of cryptographic tools such as HE and GC. We choose Chameleon Encryption [6] based on the computation Diffie-Hellman assumption as the HE, and use AES or SHA to implement the GC.

We implement experiments by using C++ programming under an Intel(R) Core(TM) i5-3427U CPU of 1.80 GHz and 4 GB of memory, running in Ubuntu−18.04.1. In addition, we rely on the OpenSSL library for the hash function, the PBC library for group operations, and point exponentiation calculations. Group elements for $\mathbb{G}$ are set with 1024-bit. As mentioned in Construction 4.1, it is clear that the main influencing factor of the performance of the PCE instance is the plaintext length. The fixed plaintext length is 4, 32, 64, 128, 256, 512, and

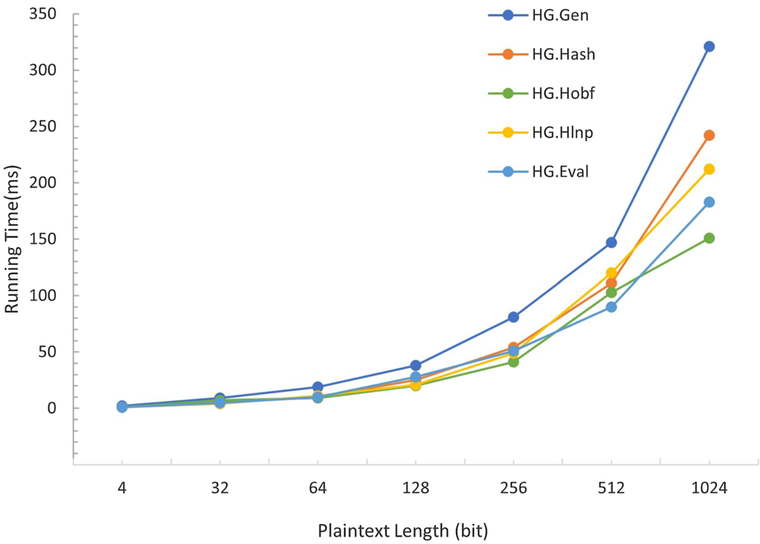

Fig. 2 Implementation result of performance of HG algorithms

1024 bits. The main metrics of performance include the PCE and HG algorithms (see Figs. 1 and 2).

For the performance, in Fig. 1, with the exception of the Dec algorithm, the remaining algorithms experience a climbing trend with the increase of the length of plaintext. Moreover, from Fig. 2, it can be seen that the HG algorithm occupies the main performance of PCE. In fact, for the length of the plaintext, the time of execution of the PKE algorithm is also constant, as can be seen by comparing Figs. 1 and 2.

6 Conclusions

In this paper, we have shown a construction of HG based on hash encryption and garbled circuits. Then, we have built a warm-up solution to realize the construction of plaintext checkable encryption from HG, but its security has been proven in the random oracle. With slight modifications from our warm-up, the full-fledged construction of PCE has proved to be secure in the standard model. Finally, the experiments for implementing our warm-up PCE have shown effectiveness for real-life applications.

Acknowledgements We highly appreciate the insightful and helpful comments of the reviewers on the presentation of this paper. This work was supported by National Science and Technology Council of Taiwan (Nos. 112-2218-E-A49-023, 112-2634-F-027-001-MBK, and 112-2221-E-027-069).

Appendix A: Missing proofs

Proof of Lemma 3 Suppose $\mathcal{A}$ is the adversary with non-negligible probability to distinguish Hyb_0 and $\mathsf{Hyb}_{0,1}$, then we can create another algorithm $\mathcal{B}$ which runs $\mathcal{A}$ as a subroutine to break the weak security of HG.

Following the hybrid, $\mathcal{A}$ receives a challenge as $(\mathsf{Enc}'(pk', m_0), \widetilde{\mathsf{P}}, \widetilde{\mathsf{y}})$. $\widetilde{\mathsf{y}}$ can be parsed as (t, r) such that $\widetilde{\mathsf{y}_{m_0}} = t \oplus H(m_0||r)$. The pair of $(\widetilde{\mathsf{P}}, \widetilde{\mathsf{y}_{m_0}})$ is $(\widetilde{P}_{m_0}, \widetilde{y}_{m_0})$ or $(\widetilde{P}_{\mathsf{Sim}}, \widetilde{y}_{\mathsf{Sim}})$ where $\widetilde{P}_{\mathsf{Sim}}, \widetilde{y}_{\mathsf{Sim}}$ is generated by $\mathsf{Sim}(hk, m_0, 1^{|P_{m_0}|}, \perp)$. Note that $\perp$ by the previous argument for HIT. Accordingly, $\mathcal{A}$ can transform parts of the challenge into $\mathcal{B}$'s input. $\mathcal{A}$ directly sets $\widetilde{\mathsf{P}}, \widetilde{\mathsf{y}_{m_0}}$ as the input of $\mathcal{B}$, and waits for the output of $\mathcal{B}$. Finally, $\mathcal{A}$'s output (one bit) is set to be identical to $\mathcal{B}$'s, and thus it implies that $\Pr[\mathcal{A}(\mathsf{Enc}'(pk', m_0), \widetilde{\mathsf{P}}, \widetilde{\mathsf{y}}) = 1] = \Pr[\mathcal{B}(\widetilde{\mathsf{P}}, \widetilde{\mathsf{y}_{m_0}})] = 1$. Weak security of HG offers $|\Pr[\mathcal{B}(\widetilde{P}_{m_0}, \widetilde{y}_{m_0}) = 1] - \Pr[\mathcal{B}(\widetilde{P}_{\mathsf{Sim}}, \widetilde{y}_{\mathsf{Sim}}) = 1]| \leq \mathsf{negl}(n)$, so we obtain

$$\begin{aligned}&|\Pr[\mathcal{A}(\mathsf{Enc}'(pk', m_0), \widetilde{P}_{m_0}, (\widetilde{y}_{m_0} \oplus H(m_0||r), r)) = 1]\\&- \Pr[\mathcal{A}(\mathsf{Enc}'(pk', m_0), \widetilde{P}_{\mathsf{Sim}}, (\widetilde{y}_{\mathsf{Sim}} \oplus H(m_0||r), r)) = 1]| \leq \mathsf{negl}(n)\end{aligned}$$

as well as Hyb_0 and $\mathsf{Hyb}_{0,1}$ are computationally indistinguishable. The proof of this lemma is done. □

Proof of Lemma 5 Before we prove the lemma, we quickly remark the proof intuition of the main theorem. Our final goal is from Hyb_0 to Hyb_1 to replace $\mathsf{Enc}(pk', m_0)$ with $\mathsf{Enc}(pk', m_1)$. However, it cannot be directly replaced, since $(\widetilde{\mathsf{P}}, \widetilde{\mathsf{y}})$ in Hyb_0 includes the information of m_0. However, Lemma 3 is used to eliminate the underlying m_0 for $(\widetilde{P}_{m_0}, \widetilde{y}_{m_0})$ by the power of the random oracle.

Let go back to this proof. Suppose $\mathcal{A}$ is the adversary with non-negligible probability to distinguish $\mathsf{Hyb}_{0,2}$ and $\mathsf{Hyb}_{1,2}$, then we can create another algorithm $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine to break the CPA security of public key encryption. Following $\mathsf{Hyb}_{\beta,2}$, $\mathcal{A}$ receives a challenge as $(\mathsf{Enc}'(pk', m_\beta), \widetilde{\mathsf{P}}, \widetilde{\mathsf{y}})$ where $\widetilde{\mathsf{P}}, \widetilde{\mathsf{y}} \leftarrow U_{|\widetilde{P}|+|\widetilde{y}|}$. Consequently, $\mathcal{A}$ can transform parts of the challenge into $\mathcal{B}$'s input. $\mathcal{A}$ directly sets $\mathsf{Enc}'(pk', m_\beta)$ as the input of $\mathcal{B}$, and waits for the output of $\mathcal{B}$. Similarly to the proof of the above lemma, it implies that $\Pr[\mathcal{A}(\mathsf{Enc}'(pk', m_\beta), \widetilde{\mathsf{P}}, \widetilde{\mathsf{y}})] = \Pr[\mathcal{B}(\mathsf{Enc}'(pk', m_\beta))]$. The CPA security says $|\Pr[\mathcal{B}(\mathsf{Enc}'(pk', m_0)) = 1] - \Pr[\mathcal{B}(\mathsf{Enc}'(pk', m_1)) = 1]| \leq \mathsf{negl}(n)$. We finally obtain

$$|\Pr[\mathcal{A}(\mathsf{Enc}'(pk', m_0), \widetilde{\mathsf{P}}, \widetilde{\mathsf{y}}) = 1] - \Pr[\mathcal{A}(\mathsf{Enc}'(pk', m_1), \widetilde{\mathsf{P}}, \widetilde{\mathsf{y}}) = 1]| \leq \mathsf{negl}(n)$$

as well as $\mathsf{Hyb}_{0,2}$ and $\mathsf{Hyb}_{1,2}$ are computationally indistinguishable. The proof of this lemma is done. □

References

1. Barak B., Goldreich O., Impagliazzo R., Rudich S., Sahai A., Vadhan S., Yang K.: On the (im) possibility of obfuscating programs. In: Annual International Cryptology Conference, pp. 1–18. Springer, Berlin (2001).
2. Blum M., Micali S.: How to generate cryptographically strong sequences of pseudorandom bits. SIAM J. Comput. **13**(4), 850–864 (1984).
3. Canard S., Fuchsbauer G., Gouget A., Laguillaumie F.: Plaintext-checkable encryption. In: Cryptographers' Track at the RSA Conference, pp. 332–348. Springer, Berlin (2012).
4. Chen Y.-C.: Plaintext checkable encryption with check delegation revisited. Int. J. Ad Hoc Ubiquitous Comput. **34**(2), 102–110 (2020).
5. Das A., Adhikari A., Sakurai K.: Plaintext checkable encryption with designated checker. Adv. Math. Commun. **9**(1), 37–53 (2015).
6. Döttling N., Garg S.: Identity-based encryption from the Diffie–Hellman assumption. In: Annual International Cryptology Conference, pp. 537–569. Springer, Cham (2017).
7. Döttling N., Garg S., Hajiabadi M., Masny D.: New constructions of identity-based and key-dependent message secure encryption schemes. In: IACR International Workshop on Public Key Cryptography, pp. 3–31. Springer, Cham (2018).

8. Dwork C., McSherry F., Nissim K., Smith A.: Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference, pp. 265–284. Springer, Berlin (2006).
9. ElGamal T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985).
10. Garg S., Gentry C., Halevi S., Raykova M., Sahai A., Waters B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, pp. 40–49. IEEE Computer Society, Washington (2013).
11. Garg S., Gentry C., Halevi S., Raykova M., Sahai A., Waters B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM J. Comput. **45**(3), 882–929 (2016).
12. Garg S., Hajiabadi M., Mahmoody M., Rahimi A.: Registration-based encryption: removing private-key generator from IBE. In: Theory of Cryptography Conference, pp. 689–718. Springer, Cham (2018).
13. Hada S.: Zero-knowledge and code obfuscation. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 443–457. Springer, Berlin (2000).
14. Lindell Y., Pinkas B.: A proof of security of Yao's protocol for two-party computation. J. Cryptol. **22**(2), 161–188 (2009).
15. Ma S., Huang Q.: Plaintext-checkable encryption with unlink-CCA security in the standard model. In: Information Security Practice and Experience: 15th International Conference, ISPEC 2019, Kuala Lumpur, Malaysia, 26–28 November 2019, Proceedings, vol. 15, pp. 3–19. Springer, Cham (2019).
16. Ma S., Huang Q., Li X., Xiao M.: Plaintext-verifiably-checkable encryption. In: International Conference on Provable Security, pp. 149–166. Springer, Cham (2019).
17. Ma S., Yi M., Susilo W.: A generic scheme of plaintext-checkable database encryption. Inf. Sci. **429**, 88–101 (2018).
18. Tsiounis Y., Yung M.: On the security of Elgamal based encryption. In: International Workshop on Public Key Cryptography, pp. 117–134. Springer, Berlin (1998).
19. Yao A.C.-C.: Protocols for secure computations. In: FOCS, vol. 82, pp. 160–164 (1982).