A CP-ABE Scheme Supporting Arithmetic Span Programs

Chao Ma, Haiying Gao, and Duo Wei

PLA SSF Information Engineering University, Zhengzhou, Henan, China

Correspondence should be addressed to Chao Ma; mac_88@163.com

Received 18 September 2019; Revised 25 January 2020; Accepted 3 February 2020; Published 16 March 2020

Academic Editor: Bruce M. Kapron

Copyright © 2020 Chao Ma et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Attribute-based encryption achieves fine-grained access control, especially in a cloud computing environment. In a ciphertext-policy attribute-based encryption (CP-ABE) scheme, the ciphertexts are associated with the access policies, while the secret keys are determined by the attributes. In recent years, people have tried to find more effective access structures to improve the efficiency of encryption systems. This paper presents a ciphertext-policy attribute-based encryption scheme that supports arithmetic span programs. On the composite-order bilinear group, the security of the scheme is proven by experimental sequence based on the combination of composite-order bilinear entropy expansion lemma and subgroup decision (SD) assumption. And, it is an adaptively secure scheme with constant-size public parameters.

1. Introduction

In the cloud computing environment, the traditional public key encryption system cannot meet the realistic needs due to the feature that it only achieves one-to-one encrypted data sharing. In 2006, Goyal et al. [1] proposed attribute-based encryption (ABE), which can achieve one-to-many encryption, making the sharing of encrypted data more convenient. Besides, the encrypter does no need to know the specific identifying information of the visitors but only needs to use the access structure to complete the access control of the user’s identity on the fine-grained level, which provides a new idea for data sharing. ABE is divided into two types based on ciphertexts or keys being marked as attributes. For example, in a CP-ABE scheme, keys are marked as attributes and the ciphertexts are linked with access policies. Conversely, the key-policy ABE (KP-ABE) means that keys are linked with access policies and the ciphertexts are marked as a series of attributes.

In 2006, Goyal et al. [1] came up with a KP-ABE scheme that supports an access tree. The size of the public parameters is linearly related to the size of the attributes, that is, the size is not constant. In 2008, Katz et al. [2] put forward the first KP-ABE scheme based on the inner product on the composite-order bilinear group. It is a selectively secure scheme, and the length of the ciphertext increases linearly with the vector’s dimension. In 2010, Herranz et al. [3] proposed a CP-ABE scheme with a constant-size ciphertext, but it only supports the threshold access control. In 2011, based on dual pairing vector space, Okamoto and Takashima [4] presented a zero-inner product encryption scheme and a nonzero inner product encryption scheme which are fully secure under the standard model, in which the ciphertext’s length or the key’s length can reach a constant. In 2011, Attrapadung et al. [5] first proposed a KP-ABE scheme that supports the nonmonotonic access control. The scheme has a constant-size ciphertext, but it can only be proved under the selective model. In 2013, Chen et al. [6] gave a general construction method from inner product encryption to ABE and presented an ABE scheme supporting threshold access control based on inner product encryption. This scheme achieves adaptive security with constant-size ciphertext. In 2014, Wee [7] first proposed an ABE scheme supporting the arithmetic span programs [8], but did not give a specific scheme (just a framework). In 2015, Attrapadung et al. [9] proposed a general conversion between the ABE scheme supporting the arithmetic span programs and the KP-ABE scheme when we do not limit the size of the span programs, but the size of the attributes is limited. This scheme achieves adaptive security with a constant-size ciphertext, but the
length of the public parameters is still not constant. In 2017, Chen et al. [10] first proposed a KP-ABE scheme supporting arithmetic span programs via bilinear entropy expansion, and the scheme is adaptive security with constant-size parameters. In particular, Table 1 illustrates the development of ABE about the access structure. Besides, the existing ABE scheme can be converted into a scheme supporting the arithmetic span program. Compared with the ABE scheme achieved by the Boolean circuit, the computational complexity and parameter size of the scheme supporting the arithmetic span program are relatively small. Therefore, based on the fact that the composite-order bilinear group has fewer algorithm components and the algorithm represents simple and clear advantages, it is meaningful to construct a CP-ABE scheme.

“Can we design a CP-ABE scheme that supports arithmetic span program on a bilinear group?”

1.1. Our Contribution. Although CP-ABE and KP-ABE have many similarities in structure, even a dual relationship, the application scenarios are very different. In the CP-ABE scheme, because the policy is embedded in ciphertext, the data owner can set policies to determine which properties can access the ciphertext. That is, encrypted access control for this data can be refined to the attribute level. The application scenario of CP-ABE is usually data encryption storage and fine-grained sharing on the public cloud, while the application scenario of KP-ABE is more inclined to pay videoweb sites, log encryption management, and so on. Inspired by [10], we consider designing an adaptively secure CP-ABE scheme. There are some schemes supporting arithmetic span programs [10, 11], where [10, 11] are KP-ABE schemes. However, considering that the composite-order bilinear group has fewer algorithm components and the algorithm represents simple and clear advantages, it is meaningful to construct a CP-ABE scheme on composite-order groups. Specifically, to reduce the parameter size, we first give the composite-order bilinear entropy expansion lemma, which contains the specific form of public parameters, ciphertext, and the key. In the setup, we use some random numbers as the master secret key and use the master secret key to calculate the master public key. In the Enc, we subtly embed the strategy into certain components of the ciphertext in combination with the public parameters and the bilinear entropy extension vector. In the KeyGen, we combine the attribute vector, the public parameter, and the bilinear entropy extension vector to generate the secret key. In the Dec, the arithmetic span program is used as a standard for decryption and the user can encrypt normally. Finally, based on SD assumption and composite-order bilinear entropy expansion lemma, the scheme is proved to have adaptive security.

1.2. Organization. We first list some relevant knowledge in Section 2. Then, we present the formal definition of our scheme in Section 3.1 and propose the adaptive security model in Section 3.2. Specifically, we present our scheme in Section 3.3 and verify its correctness in Section 3.4. Finally, we prove its adaptive security by a series of experiments in Section 3.5.

2. Preliminaries

Notation. We let $\mathbb{Z}_p^n$ denote a ring of algebraic integers modules a prime number $p$ and $\mathbb{Z}_p^n$ denote an $m$-dimension vector in $\mathbb{Z}_p^n$. $\mathbb{G}_N$ and $e$ represent a group of order $N$ and a bilinear map, respectively. We denote $[n]$ as the set $\{1, 2, \ldots, n\}$ and $n$-dimensional vector as the bold letter $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.

2.1. Bilinear Maps

Definition 1 (see [12, 13] bilinear maps). Let $G_N, H_N,$ and $G_T$ be bilinear groups of order $N = p_1p_2\cdots p_3$, where $p_1, p_2,$ and $p_3$ are primes. Let $g$ be the generator of $G_N$ and $g_1, g_2,$ and $g_3$ be the generators of $G_{p_1}, G_{p_2},$ and $G_{p_3}$, respectively. Let $h$ be the generator of $H_N$, and $h_1, h_2,$ and $h_3$ be the generators of $H_{p_1}, H_{p_2},$ and $H_{p_3}$, respectively.

$e: (G_N \times H_N) \rightarrow G_T$ is a bilinear map, if it satisfies the following three properties:

1. Bilinearity: $e(g_0^a, h_0^b) = e(g_0, h_0)^{ab}$ for all $a, b \in \mathbb{Z}_p, g_0 \in G_N, h_0 \in H_N$.
2. Nondegeneracy: there exists $g_0 \in G_N, h_0 \in H_N$, such that the order of $e(g_0, h_0)$ is $N$.
3. Computability: for all $g_0 \in G_N, h_0 \in H_N$, there is an efficient algorithm to compute $e(g_0, h_0)$.

Also, the composite-order bilinear map satisfies the orthogonality $e(g_i, h_j) = 1$, for all $i, j \in \{1, 2, 3\}, i \neq j$.

2.2. Arithmetic Span Program

Definition 2 (arithmetic span program [8]). An arithmetic span program $(\nu, \rho)$ is a map $\rho: \{l\} \rightarrow [n]$, and a collection of row vectors $\nu = \{y_j, z_j\}: j \in [l], y_j, z_j \in \mathbb{Z}_p^n$, for $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$, $\mathbf{x} \in \mathbb{Z}_p^n$ satisfies $(\nu, \rho)$ iff there exists constants $\omega_1, \ldots, \omega_l \in \mathbb{Z}_p$ such that

$$\sum_{j=1}^l \omega_j (y_j + x_{\rho(j)}z_j) = 1,$$

(1)

where $1 = (1, 0, \ldots, 0) \in \mathbb{Z}_p^l$.

Like in paper [9], we limit $\rho$ to be an identity map and $l = n$.

2.3. Computational Assumptions

Assumption 1 (SD$_{\mathbb{G}_N} \rightarrow_p \mathbb{G}_{p_1}$, [12, 13]). We define the subgroup decision assumption (denoted by SD$_{\mathbb{G}_N} \rightarrow_p \mathbb{G}_{p_1}$) holds if for all probability polynomial time (PPT) adversaries $A$, and the following advantage function is negligible in $\lambda$:
the security of the scheme against selectively chosen plaintext attacks can be proven in the standard model;

Table 1: Development of ABE about access structure.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Access policy</th>
<th>Access structure</th>
<th>Parameter</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>KP</td>
<td>AND and OR gates</td>
<td>Not constant</td>
<td>Selective</td>
</tr>
<tr>
<td>[2]</td>
<td>KP</td>
<td>Inner product</td>
<td>Not constant</td>
<td>Selective</td>
</tr>
<tr>
<td>[3]</td>
<td>CP</td>
<td>Threshold</td>
<td>Constant</td>
<td>Selective</td>
</tr>
<tr>
<td>[5]</td>
<td>KP</td>
<td>Nonmonotonic</td>
<td>Constant</td>
<td>Selective</td>
</tr>
<tr>
<td>[6]</td>
<td>Both</td>
<td>Threshold</td>
<td>Constant</td>
<td>Adaptive</td>
</tr>
<tr>
<td>[7]</td>
<td>KP</td>
<td>Arithmetic span programs</td>
<td>Constant</td>
<td>Adaptive</td>
</tr>
<tr>
<td>[10]</td>
<td>KP</td>
<td>Arithmetic span programs</td>
<td>Constant</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Our</td>
<td>CP</td>
<td>Arithmetic span programs</td>
<td>Constant</td>
<td>Adaptive</td>
</tr>
</tbody>
</table>

①: the security of the scheme against selectively chosen plaintext attacks can be proven in the standard model; ②: this paper did not give a specific scheme (just a framework).

\[ \text{Adv}^{SDH_{\text{Diff}}}_\mathcal{A}(\lambda) = |\Pr[\mathcal{A}(G, D, T_0) = 1] - \Pr[\mathcal{A}(G, D, T_1) = 1]|, \]

where

\[ D = (h_1, h_2, t_2, g_1, g_2, g_3), \]
\[ T_0 = (g_2, g_2^y, g_3^x), \]
\[ T_1 = (g_2^2, g_2^y, g_2^2), \]

Assumption 2. (DDH_{\text{Diff}}^G) The p_{2}-DDH assumption (denoted by DDH_{\text{Diff}}^G), holds if for all probability polynomial time (PPT) adversaries \( \mathcal{A} \), and the following advantage function is negligible in \( \lambda \):

\[ \text{Adv}^{\text{DDH}^G}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(G, D, T_0) = 1] - \Pr[\mathcal{A}(G, D, T_1) = 1]|, \]

2.4. Bilinear Entropy Expansion Lemma. For an adversary \( A \), the advantage of distinguishing the following two distributions in any polynomial time is negligible:

\[
\begin{align*}
\text{Adv}_{\mathcal{A}}^{SDH_{\text{Diff}}}(\lambda) &= |\Pr[\mathcal{A}(G, D, T_0) = 1] - \Pr[\mathcal{A}(G, D, T_1) = 1]|, \\
\end{align*}
\]

2.4. Bilinear Entropy Expansion Lemma. For an adversary \( A \), the advantage of distinguishing the following two distributions in any polynomial time is negligible:

\[
\begin{align*}
\text{Adv}_{\mathcal{A}}^{\text{DDH}^G}(\lambda) &= |\Pr[\mathcal{A}(G, D, T_0) = 1] - \Pr[\mathcal{A}(G, D, T_1) = 1]|, \\
\end{align*}
\]

The p_{2}-DDH assumption (denoted by DDH_{\text{Diff}}^G), holds if for all probability polynomial time (PPT) adversaries \( \mathcal{A} \), and the following advantage function is negligible in \( \lambda \):

\[
\text{Adv}^{\text{DDH}^G}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(G, D, T_0) = 1] - \Pr[\mathcal{A}(G, D, T_1) = 1]|, \\
\]

2.4. Bilinear Entropy Expansion Lemma. For an adversary \( A \), the advantage of distinguishing the following two distributions in any polynomial time is negligible:

\[
\begin{align*}
\text{Adv}_{\mathcal{A}}^{\text{DDH}^G}(\lambda) &= |\Pr[\mathcal{A}(G, D, T_0) = 1] - \Pr[\mathcal{A}(G, D, T_1) = 1]|, \\
\end{align*}
\]

For an adversary \( A \), the advantage of distinguishing the following two distributions in any polynomial time is negligible:

\[
\begin{align*}
\text{Adv}_{\mathcal{A}}^{\text{DDH}^G}(\lambda) &= |\Pr[\mathcal{A}(G, D, T_0) = 1] - \Pr[\mathcal{A}(G, D, T_1) = 1]|, \\
\end{align*}
\]
See Appendix for details about the proof of this lemma.

3. CP-ABE Supporting Arithmetic Span Programs

3.1. Formal Definition of the CP-ABE Scheme Supporting Arithmetic Span Program

Setup($\ell, p$): input security parameters ($\ell, p$) and output the master public key $mpk$ and the master secret key $msk$.

Enc($mpk, v, m$): input access structure $v = \{(y_j, z_j) : j \in [n], y_j, z_j \in Z_p^\ell\}$ and plaintext $m$ and output ciphertext $ct_v$.

KeyGen($mpk, msk, x$): input the vector $x \in Z_p^n$ and output the secret key $sk_x$.

Dec($mpk, sk_x, ct_v$): input $sk_x$ and $ct_v$ and output $m$ if $x, v$ satisfies $\sum_{j=1}^{\ell} \omega_j (y_j + x_j z_j) = 1$.

3.2. Adaptively Security Model for CP-ABE Schemes Supporting Arithmetic Span Programs. We present an adaptive security model of the CP-ABE scheme that supports the arithmetic span program through the games about the challenger $B$ and adversary $A$.

Setup 1: challenger $B$ runs the initialization algorithm and sends $mpk$ to adversary $A$.

Stage 1: adversary $A$ chooses $x$ to perform multiple secret key queries. Challenger $B$ runs the KeyGen and sends the secret key to the adversary $A$.

Challenge: adversary $A$ sends two equal-length plaintexts ($m_{0i}$ and $m_{1i}$) and the challenge access structure $v^* = \{(y_j, z_j) : j \in [n], y_j, z_j \in Z_p^\ell\}$ to challenger $B$ (any query vector $x'$ and the challenge access structure $v^* = \{(y_j, z_j) : j \in [n], y_j, z_j \in Z_p^\ell\}$ do not satisfy $\sum_{j=1}^{\ell} \omega_j (y_j + x_j z_j) = 1$). Challenger $B$ chooses $b \in \{0, 1\}$ randomly and computes $ct_{v^*} = Enc(mpk, v^*, m_{bi})$. Then, Challenger $B$ sends the challenge ciphertext $ct_{v^*}$ to the adversary $A$.

Stage 2: same as Stage 1.

Guess: adversary $A$ outputs the guess $b'$ about $b$.

We say adversary $A$ wins this game if $b' = b$, and the advantage of adversary $A$ is $Adv_A(\lambda) = |Pr(b' = b) - 1/2|$.

The encryption scheme is adaptively secure if the advantages of winning the above games are negligible, for all PPT adversaries.

3.3. Our Construction

Setup$(\ell, 1^n)$: input the number of security parameters $\lambda$ and attributes $\ell$ and and select $G = (p = p_1 p_2 p_3, G_N, H_N, e)$ $\longrightarrow G(\ell^\lambda)$. Pick random generators $g_1, h_1$, and $h_{123}$ of $G_{p_1}^\ast, H_{p_1}$, and $H_N$, respectively. Sample $w, w_0, w_1, w', w_0, w_1$, $u_0, u_0, u_0, u_0$ $\longrightarrow Z_N^\ast$ and output the master public key

$$mpk = \left((G_N, H_N, e) : g_{11}, g_1^w, g_1^{u_0}, g_1^{u_0}, g_{11}^w, \right.$$  

$$g_1^w, g_1^{u_0}, g_1^{u_0, u_0}, g_1^w, (g_1, h_{123})^u \right),$$

and the master secret key

$$msk = (h_{123}, h_1, w, w_0, w_1, w', w_0, w_1, u_0).$$

Enc($mpk, v, m$): input the access structure $v = \{(y_j, z_j) : j \in [l], y_j, z_j \in Z_p^\ell\}$ and the message $m \in \{0, 1\}^\lambda$. Select $s, s_j \longrightarrow Z_N^\ast$, $u \longrightarrow Z_N^\ast$ for all $j \in [n].$ Compute and output

$$ct_v = \left\{ \begin{array}{ll} C_0 = g_1^w \\ C_{0, j} = g_1^{s_j, w} \\ C_{1, j} = g_1^{s_j, u_j + j w} \\ C = e(g_1, h_{123})^{u \cdot m} \end{array} \right\}_{j \in [n]}$$

KeyGen($mpk, msk, x$): input the master secret key $msk$ and vector $x = (x_1, x_2, \ldots, x_n) \in Z_p^n$. Select $r, r_j, r_j \longrightarrow Z_N$ for all $j \in [n]$ and output

$$sk_x = \left\{ \begin{array}{ll} K_0 = h_{123}^{a_j} h_1^{u_j}, K_1 = h_1^r \\ K_{1, j} = h_1^{r_j (u_j + j w)} \\ K_{z, j} = h_1^{r_j} \end{array} \right\}_{j \in [n]}$$

Dec($mpk, sk_x, ct_v$): input secret key $sk_x$ and ciphertext $ct_v$. If $(x, v)$ satisfies $\sum_{j=1}^{\ell} \omega_j (y_j + x_j z_j) = 1$, then compute $m = C \cdot C' \cdot c(C_0, K_0)$.
where

\[
C' = \prod_{j=1}^{n} \left( e(C_{0,j} \cdot (C'_{0,j})^{\gamma_j}, K_1) \cdot e(C_{1,j}, K_{1,j})^{-1} \cdot e(C_{2,j}, K_{2,j}) \cdot e((C_{2,j})^{\gamma_j}, K'_{2,j})^{\omega_j} \right)
\]  \quad (12)

3.4. Correctness. For all \((x, v)\) satisfies \(\sum_{j=1}^{l} \omega_j (y_j + x_j z_j) = 1\), we compute

\[
e(C_{0,j} \cdot (C'_{0,j})^{\gamma_j}, K_1) \cdot e(C_{1,j}, K_{1,j})^{-1} \cdot e(C_{2,j}, K_{2,j}) \cdot e((C_{2,j})^{\gamma_j}, K'_{2,j})^{\omega_j}
\]

\[
\left( g_1 \cdot \left( \sum_{k=1}^{n} \omega_k \cdot g_k \right) \right)^{\gamma_j} \cdot h_1^{\omega_j} \cdot h_0^{\omega_j}
\]

\[
C' = \prod_{j=1}^{n} \left( e(g_1, h_1)^{\gamma_j} \cdot e(g_1, h_1)^{\omega_j} \right)
\]

\[
e(g_1, h_1)^{\gamma_j} \cdot e(g_1, h_1)^{\omega_j}
\]

\[
e(g_1, h_1)^{\gamma_j} \cdot e(g_1, h_1)^{\omega_j}
\]

\[
e(g_1, h_1)^{\gamma_j} \cdot e(g_1, h_1)^{\omega_j}
\]
3.5. Security. The proof of the security relies on a series of games that cannot be distinguished. We first define the ciphertext and secret key distributions that are needed in the process of the proof.

3.5.1. Ciphertext Distributions

Standard ciphertext: generated by the encryption algorithm:

\[ C_0 = g_1^i \]

\[ C_{0,j} = g_1 \]

\[ C_{1,j} = g_1^j, C_{2,j} = g_1^{j(w_0+jw)} \]

\[ C = e(g_1, h_{123})^a \cdot m \]

\[ ct_v = \left\lbrace \begin{array}{l}
C_0 = g_1^i \\
C_{0,j} = g_1 \\
C_{1,j} = g_1^j, C_{2,j} = g_1^{j(w_0+jw)} \\
C = e(g_1, h_{123})^a \cdot m
\end{array} \right\} j \in [n] \] (14)

Entropy expansion ciphertext: the difference between it and standard ciphertext is given as follows: \( w \rightarrow v_j \mod p_2, w_0 \rightarrow v'_j \mod p_2, w_j \rightarrow u_j \mod p_2, \)

\[ w_0 + jw \rightarrow u'_j \mod p_2 (\forall j \in [n]), \]

\[ \begin{array}{l}
C_0 = g_1^i g_2^j \\
C_{0,j} = g_1 \\
C_{1,j} = g_1^j, C_{2,j} = g_1^{j(w_0+jw)} \\
C = e(g_1^i g_2^j, h_{123})^a \cdot m
\end{array} \] (15)

3.5.2. Secret Key Distributions

Standard secret key: it is generated by the secret key generation algorithm:

\[ \text{sk}_x = \left\lbrace \begin{array}{l}
K_0 = h_{123}^a h_{16}^{r'}, K_1 = h_1^r \\
K_{1,j} = h_1^{r'(w+jw)} r_j (w_0+jw) + x_j, r'_j (w_0+jw), K_{2,j} = h_1^{r'j}, j \in [n]
\end{array} \right\} \] (16)

Entropy expansion secret key: compared to the standard secret key, we make a copy of \( \left\lbrace h_1^r, h_1^{r'(w+jw)} r_j (w_0+jw) + x_j, r'_j (w_0+jw), h_1^{r'}, j \in [n] \right\} \) in \( H_{F_1} \):

\[ \text{sk}_x = \left\lbrace \begin{array}{l}
K_0 = h_{123}^a h_{16}^{r'}, K_1 = h_1^r \\
K_{1,j} = h_1^{r'(w+jw)} r_j (w_0+jw) + x_j, r'_j (w_0+jw), h_1^{r'}, j \in [n]
\end{array} \right\} \] (17)
Pseudostandard secret key: compared to the entropy expansion secret key, we make a copy of 
\[ \left\{ h_i, h_i^{r(v+x,p_j)} r_{j, u_i+x, p_j}, h_i^r \right\} \] in \( H_p \):

\[
\text{sk}_a = \left\{ \begin{array}{l}
K_0 = h_1^{v(x, w)} h_1, K_1 = h_i h_i^{r(v+x, p_j)} r_{j, u_i+x, p_j}, h_i^r \end{array} \right\}.
\]

(18)

Pseudo-semi-functional secret key: compared to the pseudostandard secret key, we sample \( a \leftarrow H_p \):

\[
\text{sk}_a = \left\{ \begin{array}{l}
K_0 = h_1^{v(x, w)} h_1, K_1 = h_i h_i^{r(v+x, p_j)} r_{j, u_i+x, p_j}, h_i^r \end{array} \right\}.
\]

(19)

Semifunctional secret key: compared to the pseudo-semi-functional secret key, we remove \( \left\{ h_i, h_i^{r(v+x, p_j)} r_{j, u_i+x, p_j}, h_i^r \right\} \):

\[
\text{sk}_a = \left\{ \begin{array}{l}
K_0 = h_1^{v(x, w)} h_1, K_1 = h_i h_i^{r(v+x, p_j)} r_{j, u_i+x, p_j}, h_i^r \end{array} \right\}.
\]

(20)

3.5.3. Games. Assume that an adversary \( A \) makes at most \( Q \) secret key queries. Let the advantage of \( A \) in Game\(_{exx} \) be denoted by \( \text{Adv}_{exx}(\lambda) \). In the following, we describe in detail the specific details of the games, and the comparison of Game\(_{exx} \) is given in Table 2.

<table>
<thead>
<tr>
<th>Game</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game(_{i})</td>
<td>the challenge ciphertext and secret keys are generated by Enc and KeyGen, respectively.</td>
</tr>
<tr>
<td>Game(_{i})</td>
<td>compared to Game(_{i-1}), the first ( i ) secret keys are semifunctional and the last ( Q - i + 1 ) are entropy expansion.</td>
</tr>
<tr>
<td>Game(_{i})</td>
<td>compared to Game(_{i-1}), the first ( i - 1 ) secret keys are semifunctional and the last ( Q - i + 1 ) are entropy expansion.</td>
</tr>
<tr>
<td>Game(_{i})</td>
<td>compared to Game(_{i}), modify the ( i' ) key to the pseudostandard key.</td>
</tr>
</tbody>
</table>

Game\(_{i}\): compared to Game\(_{i}\), modify the \( i' \) key to the pseudo-semifunctional key.

Game\(_{i}\): compared to Game\(_{i}\), modify the \( i' \) key to the semifunctional key.

Game\(_{final}\): challenge ciphertext is the entropy expansion ciphertext about a random message, while the secret keys are semifunctional.

**Lemma 1** (Game\(_{i}\) vs Game\(_{i}\)). There exists a challenger \( B_0 \) who can distinguish the left and right distributions in the bilinear entropy expansion lemma with a non-negligible advantage if \( |\text{Adv}_{0}(\lambda) - \text{Adv}_{0}(\lambda)| > \varepsilon \), that is, \( \text{Time}(B_0) \approx \text{Time}(A) \).
Table 2: Games for proving the adaptive security of our scheme.

<table>
<thead>
<tr>
<th>Game</th>
<th>CT</th>
<th>SK</th>
<th>$\kappa &lt; i$</th>
<th>$\kappa = i$</th>
<th>$\kappa &gt; i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Standard entropy expansion</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0'</td>
<td>Standard entropy expansion</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>Semifunctional</td>
<td>Entropy expansion</td>
<td>Pseudostandard</td>
<td>Pseudo-semifunctional</td>
<td>Semifunctional</td>
</tr>
<tr>
<td>i,1</td>
<td>—</td>
<td>Entropy expansion</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>i,2</td>
<td>—</td>
<td>Entropy expansion</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>i,3</td>
<td>—</td>
<td>Entropy expansion</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Final</td>
<td>Random message</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Proof. Challenger $B_0$ obtains the following distribution:

\[
\begin{align*}
\text{aux}: & \left(g_1, g_1^{\mu}, g_1^{w_0}, g_1^{w_1}, g_1^{\mu'}, g_1^{w_1'}, g_1^{\mu''}, g_1^{w_2}, g_1^{\mu'''}ight) \\
\text{ct}: & \left(C_{0,j}, C_{0,j'}, C_{1,j}, C_{1,j'}, C_{2,j}, C_{2,j'}\right)_{j \in [n]} \\
\text{sk}: & \left(K_{1,j}, K_{1,j'}, K_{2,j}, K_{2,j'}\right)_{j \in [n]}
\end{align*}
\]

(21)

Challenger $B_0$ simulates the secret key generation algorithm and picks $\tilde{r}_j \longleftarrow \mathbb{Z}_N^*$ for all $j \in [n]$. Output

\[
\text{sk}_{x_j} = \left(\tilde{r}_j, x_j, \tilde{r}_j', C_{1,j}, C_{1,j'}, C_{2,j}, C_{2,j'}\right)_{j \in [n]}
\]

(23)

Challenge: adversary $A$ sends two equal-length plaintexts $(m_0, m_1)$ and the challenge access structure $v' = \{(y_j, z_j) : j \in [n], y_j, z_j \in \mathbb{Z}_N^*\}$ to challenger $B_0$ (any query vector $x'$ in Phase 1 and the challenge access structure $v' = \{(y_j, z_j) : j \in [n], y_j, z_j \in \mathbb{Z}_N^*\}$ do not satisfy $\sum_{j=1}^n \omega_j(y_j + x_j \cdot z_j) = 1$). Challenger $B_0$ picks $b \in \{0, 1\}$ and $u_j \longleftarrow \mathbb{Z}_N^{-1}$ and outputs the challenge ciphertext:

\[
\begin{align*}
\text{ct}_{x_j} = & \left(C_0\left(u_0, x_0\right), C_{0,j}\left(u_j, x_j\right), C_{1,j}\left(u_j, x_j\right), C_{2,j}\left(u_j, x_j\right)\right)_{j \in [n]} \\
& \left(C_{0,j}, C_{0,j'}, C_{1,j}, C_{1,j'}, C_{2,j}, C_{2,j'}\right)_{j \in [n]} \\
& \left(C_{1,j}, C_{2,j}, C_{2,j'}, C_{2,j'}, C_{0,j}, C_{0,j'}\right)_{j \in [n]} \\
& \left(C = e\left(C_{0}, h_{123}^{x_j}\right) \cdot m_b\right)
\end{align*}
\]

(24)

Stage 2: same as Stage 1.

Guess: adversary $A$ outputs the guess $b'$ about $b$.

Note: the output is the standard secret key and the standard challenge ciphertext if $B_0$ obtains the left distribution. Conversely, the output is the entropy expansion secret key and the entropy expansion challenge ciphertext if $B_0$ obtains the right distribution. Challenger $B_0$ also distinguishes the left and right distributions of the entropy expansion lemma with a non-negligible advantage if $\left|\text{Adv}_{\text{B}_0}(\lambda) - \text{Adv}_{\text{B}_1}(\lambda)\right| \geq \varepsilon$. Game$_0$ and Game$_1$ cannot be distinguished due to the indistinguishability of the left and right distributions.

Lemma 3 (Game$_0 \equiv$ Game$_1$). We know it in Table 2 easily.

Proof. Firstly, from the SD$_{H_N}^{\mu}$, $p \longleftarrow \mathbb{Z}_N$, assumption, sample $\tilde{r}_j \tilde{r}_j' \longleftarrow \mathbb{Z}_N^*$ for all $j \in [n]$. Then, $B_1$ needs to...
distinguish whether \( \{T, T', T''\} \) is the left distribution or right.

Setup: pick random generator \( h_{123} \) of \( H_N \). Sample \( w, w_0, w_1, w', w_0, w_1, \alpha, u_0, u_1 \sim R_Z \) and output
\[
\text{mpk} = \left( g_1, g_1^w, g_1^{w_0}, g_1^{w_1}, g_1, g_1^{w'}, g_1^{w_0}, g_1^{w_1}, e(g_1, h_{123})^\alpha \right).
\]  

(25)

Stage 1: adversary \( A \) queries the secret key corresponding to vector \( x = (x_1, x_2, \ldots, x_n) \). Challenger \( B_1 \) simulates the secret key generation algorithm and samples \( \tilde{r}_j, \tilde{T}_j \sim _R Z_N \) for all \( j \in [n] \) and outputs
\[
\text{ct}_{x'} := \left\{ \begin{array}{ll}
C_0 = g_1^2 \cdot g_2^2 \\
C_{0,j} = g_1 \cdot u_0^{s_y} \cdot g_2^{s_y} \\
C_{1,j} = g_1^{s_y} \cdot g_2^{s_y} \cdot u_0^{s_y} \\
C = e(C_0, h_{123}) \cdot m_b
\end{array} \right\}.
\]  

(27)

Stage 2: same as Stage 1.

Guess: adversary \( A \) outputs the guess \( b' \) about \( b \).

Note: the output is the entropy secret key if \( B_1 \) obtains the left distribution, which is \( \{T = h_2^r, T_j = h_2^{r_j}, T' = h_2^{r'_j}\} \). The output is the entropy pseudostandard secret key if \( B_1 \) obtains the right distribution, which is \( \{T = h_2^r, T_j = h_2^{r_j}, T'_j = h_2^{r'_j}\} \). Challenger \( B_1 \) also solves SD\( \overline{H}_N \rightarrow p \cdot P_3 \) with a non-negligible advantage if \( |\text{Adv}_i(\lambda) - \text{Adv}_{i+1}(\lambda)| > \epsilon \). Therefore, Game\( i_1 \) and Game\( i_2 \) cannot be distinguished due to SD\( \overline{H}_N \rightarrow p \cdot P_3 \).

\begin{lemma}
(Game\( i_1 \equiv \text{Game}_{i+1} \)). The advantage in Game\( i_1 \) and Game\( i_2 \) satisfies \( |\text{Adv}_{i_1}(\lambda) - \text{Adv}_{i+2}(\lambda)| > \epsilon \) for any adversaries \( A \).
\end{lemma}

\begin{proof}
Challenger \( B_2 \) samples \( u_j, v_j, u'_j, v'_j, \) and \( \tilde{\alpha} \) for all \( j \in [n] \). The difference between Game\( i_1 \) and Game\( i_2 \) is only
the \( i \)th secret key query. The following shows that the challenger \( B_2 \) cannot distinguish these two games.

Setup: pick random generator \( h_{123} \) of \( H_N \). Sample \( w, w_0, w_1, w', w'_0, w'_1, \alpha, u_0 \leftarrow R Z_N \) and output

\[
mpk = \left( g_1, g'_1, g_1, g'_1, g_1, g'_1, g_1, g'_1, e \left( g_1, h_{123} \right)^\alpha \right).
\]  

(28)

Stage 1, 2: adversary \( A \) queries the secret key corresponding to vector \( x' = (x'_1, x'_2, \ldots, x'_n) \). Challenger \( B_2 \) simulates the secret key generation algorithm and samples \( \overline{r}, \overline{r}_j, \overline{r}'_j \leftarrow R Z_N \) for all \( j \in [n] \). Game_{i,1} outputs

\[
\begin{align*}
\text{sk}_{x'} \xleftarrow{i} & \left\{ \begin{array}{l}
K_0 = h_1^{\alpha} h_{123}^{w_0} \overline{r}, K_1 = h_2^{\alpha} h_3^{w_0} \overline{r} \\
K_{1,j} = h_1^{w_j + w'_j + x'_j} (w_j + w'_j) \overline{r}_j (w_j + w'_j) h_2^{r'_j} (r'_j + r'_j) T_j U_j T_j' + x'_j T_j X_j \overline{r}_j \overline{r}'_j h_3^{r'_j} (r'_j + r'_j) T_j U_j T_j' + x'_j T_j X_j \overline{r} \\
K_{2,j} = h_1^{r'_j} h_2^{r'_j} h_3^{r'_j}, K_{2,j}' = h_1^{r'_j} h_2^{r'_j} h_3^{r'_j} \end{array} \right\}, & j \in [n] \end{align*}
\]

(29)

and Game_{i,2} outputs

\[
\begin{align*}
\text{sk}_{x'} \xleftarrow{i} & \left\{ \begin{array}{l}
K_0 = h_1^{\alpha} h_{123}^{w_0} h_1^{w_0} \overline{r}, K_1 = h_2^{\alpha} h_2^{w_0} h_3^{w_0} \overline{r} \\
K_{1,j} = h_1^{w_j + w'_j + x'_j} (w_j + w'_j) \overline{r}_j (w_j + w'_j) h_2^{r'_j} (r'_j + r'_j) T_j U_j T_j' + x'_j T_j X_j \overline{r}_j \overline{r}'_j h_3^{r'_j} (r'_j + r'_j) T_j U_j T_j' + x'_j T_j X_j \overline{r} \\
K_{2,j} = h_1^{r'_j} h_2^{r'_j} h_3^{r'_j}, K_{2,j}' = h_1^{r'_j} h_2^{r'_j} h_3^{r'_j} \end{array} \right\}, & j \in [n] \end{align*}
\]

(30)

Lemma 5 (Game_{i,3} = Game_{i,3}'). There exists a challenger \( B_3 \) who can solve \( SD_{P_2} \) with a non-negligible advantage if \(|Adv_{i,2} (\lambda) - Adv_{i,3} (\lambda)| > \epsilon. \) That is, \( Time(B_3) \approx Time(A). \)

Proof. Same as Lemma 3, challenger \( B_3 \) samples \( u_j, v_j, u'_j, v'_j, \overline{a} \) for all \( j \in [n] \) and obtains \( \{T, T_j, T_j'\} \) with \( g_1, y_2, \) and \( h_1 \). Then, \( B_3 \) needs to distinguish whether \( \{T, T_j, T_j'\} \) is the left distribution or right. And challenger \( B_3 \) outputs

\[
\begin{align*}
\text{sk}_{x'} \xleftarrow{i} & \left\{ \begin{array}{l}
K_0 = h_1^{\alpha} h_{123}^{w_0} h_1^{w_0} \overline{r}, K_1 = h_2^{\alpha} T_j \\
K_{1,j} = h_1^{w_j + w'_j + x'_j} (w_j + w'_j) \overline{r}_j (w_j + w'_j) h_2^{r'_j} (r'_j + r'_j) T_j U_j T_j' + x'_j T_j X_j \overline{r}_j \overline{r}'_j h_3^{r'_j} (r'_j + r'_j) T_j U_j T_j' + x'_j T_j X_j \overline{r} \\
K_{2,j} = h_1^{r'_j} T_j, K_{2,j}' = h_1^{r'_j} T_j' \end{array} \right\}. & j \in [n] \end{align*}
\]

(31)

The output is a pseudo-semifunctional secret key if \( B_3 \) obtains the left distribution, which is

\[
\left\{ T = h_1^{r'_j} h_3^{r'_j}, T_j = h_1^{r'_j} h_3^{r'_j}, T_j' = h_1^{r'_j} h_3^{r'_j} \right\}.
\]

The output is an entropy semifunctional secret key if \( B_3 \) obtains the right distribution, which is

\[
\left\{ T = h_2^{r'_j}, T_j = h_2^{r'_j}, T_j' = h_2^{r'_j} \right\}.
\]

Challenger \( B_3 \) also solves \( SD_{P_3} \) with a non-negligible
advantage if \(|\text{Adv}_i(\lambda) - \text{Adv}_{i,1}(\lambda)| > \varepsilon\). Therefore, Game_{i,2} and Game_{i,3} cannot be distinguished due to SD^\text{\text{DP}}_{P_2} \Rightarrow P_2P_3.

**Lemma 6** (Game_i \equiv Game_{i-1,3}). We know it in Table 2 easily (in fact, they have the same secret key and challenge ciphertext).

**Lemma 7** (Game_{Q+1} \equiv Game_{\text{Final}})

*Proof.* Challenger B_4 samples \(u_j, v_j, u'_j, v'_j\) for all \(j \in [n]\). The difference between these two games is the challenge ciphertext. In Game_{Q+1}, the challenge ciphertext is obtained by \(m\), while the challenge ciphertext in Game_{\text{final}} is obtained by a random message. Let us prove that the two games are indistinguishable. Pick random generator \(h_{123}\) and \(h_3\) of \(H_N\) and \(H_{P_3}\) respectively. Select \(\bar{a}, \bar{a} \leftarrow \mathcal{R} Z_N\) and define \(h_{123}^\alpha := h_{123}^\alpha / h_3^\alpha\). We simulate Game_{Q+1} as follows:

- **Setup:** pick random generator \(h_{123}\) of \(H_N\). Sample \(w, w_0, w_1, w', w'_0, w'_1, \alpha, \alpha_0 \leftarrow \mathcal{R} Z_N\) and output \(\text{mpk} = (g_1, g_1^w, \ldots, g_1^{w_0}, g_1^{w_1}, g_1^{w'}, g_1^{w'_0}, g_1^{w'_1}, e(g_1, h_{123}^\alpha))\).

We can remove \(h_{123}^\alpha\) because \(e(g_1, h_{123}^\alpha) = 1\).

- **Stage 1:** adversary A queries the secret key corresponding to vector \(x' = (x'_1, x'_2, \ldots, x'_n)\). Challenger B_4 simulates the secret key generation algorithm and picks \(r, r_j, r'_j \leftarrow \mathcal{R} Z_N\) for all \(j \in [n]\). Output \(\bar{a} \leftarrow \mathcal{R} Z_N\) to adversary A.

**Challenge:** adversary A sends two equal-length plaintexts \((m_0, m_1)\) and the challenge access structure \(\nu^* = \{(y_j, z_j) : j \in [n], y_j, z_j \in Z_N\}\) to challenger B_4 (any query vector \(x\) in Phase 1 and the challenge access structure \(\nu^* = \{(y_j, z_j) : j \in [n], y_j, z_j \in Z_N\}\) do not satisfy \(\sum_{j=1}^n \omega_j(y_j + x'_j \cdot z_j) = 1\)). Challenger B_4 picks \(b \in \{0, 1\}\) and \(u \leftarrow \mathcal{R} Z_N^{-1}\), and the outputs challenge the ciphertext:

\[
\text{ct}_{\nu^*} := \begin{cases} 
C_0 = g_1^b g_2^\alpha & \\
C_{0,j} = g_1^{s_j} g_2^{\nu_j} & C_{0,j} = g_1^{s_j} g_2^{\nu_j} \\
C_{1,j} = g_1^{s_j} g_1^{s_j} C_{2,j} = g_1^{s_j} g_1^{s_j} C_{2,j} = g_1^{s_j} g_1^{s_j} & C = e(g_1^b g_2^\alpha, e(g_1^b g_2^\alpha, h_{123}^\alpha) \cdot m_b
\end{cases}
\]

**Guess:** adversary A outputs the guess \(b'\) about \(b\).

We have \(e(g_1^b g_2^\alpha, \text{ct}_{\nu^*}) \cdot e(g_1^b g_2^\alpha, h_{123}^\alpha) = e(g_1^b g_2^\alpha, h_{123}^\alpha) \cdot e(g_1^b g_2^\alpha, h_{123}^\alpha)\) in the entropy expansion challenge ciphertext. The distribution of \(e(g_1^b g_2^\alpha, h_{123}^\alpha)\) is \(G_\alpha\) is a uniform distribution due to the random number \(\bar{a}\), that is, the ciphertext which encrypted from a random number and the ciphertext which encrypted from \(m\) have the same distribution.

Therefore, adversary A cannot distinguish these two entropy expansion ciphertexts.

Obtained from the above analysis, we have \(|\text{Adv}_{Q+1}(\lambda) - \text{Adv}_{\text{Final}}(\lambda)| = 0\).
Theorem 1. Our CP-ABE scheme supporting arithmetic span programs is adaptively secure under the entropy expansion lemma and subgroup decision assumption decision. Also,

$$\max\{\text{Time}(B_0), \text{Time}(B_1), \text{Time}(B_2)\} \approx \text{Time}(A). \quad (35)$$

Proof. The advantage of adversary $A$ in our scheme is equivalent to the advantage in Game$_0$ under the adaptively secure model. By Lemmas 1–7, we obtain

$$\text{Adv}_0(\lambda) = \text{Adv} (\lambda) - \text{Adv}_0 (\lambda) + \text{Adv}_0' (\lambda) - \text{Adv}_1 (\lambda)$$

$$+ \cdots + \text{Adv}_Q (\lambda) - \text{Adv}_{Q+1}(\lambda)$$

$$+ \text{Adv}_{Q+1}(\lambda) - \text{Adv}_{\text{Final}}(\lambda) + \text{Adv}_{\text{Final}}(\lambda)$$

$$\leq |\text{Adv}_0 (\lambda) - \text{Adv}_0' (\lambda)| + |\text{Adv}_0' (\lambda) - \text{Adv}_1 (\lambda)|$$

$$+ \cdots + |\text{Adv}_0 (\lambda) - \text{Adv}_{Q+1}(\lambda)|$$

$$+ |\text{Adv}_{Q+1}(\lambda) - \text{Adv}_{\text{Final}}(\lambda)| + \text{Adv}_{\text{Final}}(\lambda).$$

By Lemma 1, we know

$$|\text{Adv}_0 (\lambda) - \text{Adv}_0' (\lambda)| \leq \varepsilon. \quad (37)$$

By Lemma 2, we know

$$|\text{Adv}_0' (\lambda) - \text{Adv}_1 (\lambda)| = 0. \quad (38)$$

The indistinguishability between Game$_i$ and Game$_{i+1}$ is due to

$$\text{Game}_i - \text{Game}_{i+1} = (\text{Game}_i - \text{Game}_{i+1}) + (\text{Game}_{i+1} - \text{Game}_{i+2})$$

$$+ (\text{Game}_{i+2} - \text{Game}_{i+3}) + (\text{Game}_{i+3} - \text{Game}_{i+4}).$$

By Lemma 3–6, we know

$$|\text{Adv}_i (\lambda) - \text{Adv}_{i+1} (\lambda)|$$

$$\leq |\text{Adv}_i (\lambda) - \text{Adv}_{i+1} (\lambda)|$$

$$+ |\text{Adv}_{i+1} (\lambda) - \text{Adv}_{i+2} (\lambda)|$$

$$+ |\text{Adv}_{i+2} (\lambda) - \text{Adv}_{i+3} (\lambda)|$$

$$+ |\text{Adv}_{i+3} (\lambda) - \text{Adv}_{i+4} (\lambda)|$$

$$\leq 2\varepsilon.$$
Lemma 8

\[ GN \approx \frac{1}{p} HN. \]

Lemma 9
The following highlights the proof of Lemmas A.1 and A.2.

**Lemma A.1.** Under the DDH$_{P_1}$, DDH$_{P_2}$, SD$_{P_1}$, SD$_{P_2}$, and SD$_{P_1}$ → $p$, $p'$, and SD$_{P_2}^{|n|}$ → $p$, $p'$, assumptions, we have

\[
\begin{align*}
\text{aux: } & g_1, g_2, g_1^{w_i}, g_1^{w_i'}, g_1^{u_i}, g_1^{u_i'} \\
\text{ct: } & \{ g_1^{s_j} g_1^{s_j(u_i+jw_i)} g_1^{u_i'}, g_1^{s_j(w_i'+jw_i')} \}_j \{ n \} \\
\text{sk: } & \{ h_1^{r_j(u_i+jw_i)} h_2^{r_j(w_i'+jw_i')} h_2^{r_j}, h_2^{r_j'}, h_2^{r_j} \}_j \{ n \}.
\end{align*}
\]

(A.2)

**Proof.** The proof is similar to Lemma 3 in paper [10], and we first modify the game sequence in Lemma 3.

Game$_0$: it is the same as the left distribution in Lemma A.1:

\[
\begin{align*}
\text{aux: } & g_1, g_2, g_1^{w_i}, g_1^{w_i'}, g_1^{u_i}, g_1^{u_i'} \\
\text{ct: } & \{ g_1^{s_j} g_1^{s_j(u_i+jw_i)} g_1^{u_i'}, g_1^{s_j(w_i'+jw_i')} \}_j \{ n \} \\
\text{sk: } & \{ h_1^{r_j(u_i+jw_i)} h_2^{r_j(w_i'+jw_i')} h_2^{r_j}, h_2^{r_j'}, h_2^{r_j} \}_j \{ n \}.
\end{align*}
\]

(A.3)

Game$_0$: modify sk as follows:

\[
\begin{align*}
\text{sk: } & \{ h_1^{r_j(u_i+jw_i)} h_2^{r_j(w_i'+jw_i')} h_2^{r_j}, h_2^{r_j'}, h_2^{r_j} \}_j \{ n \} \\
\end{align*}
\]

(A.4)

Now, we briefly explain that Game$_0$ ≈ Game$_d$. Under the DDH$_{P_1}$ assumption, we have

\[
\begin{align*}
\{ h_2^{r_j(u_i+jw_i)}, h_2^{r_j(w_i'+jw_i')}, h_2^{r_j}, h_2^{r_j'}, h_2^{r_j} \}_j \{ n \}
\end{align*}
\]

(A.5)

where $v_j, v_j' \leftarrow R_{Z_N}$, and set $u_j = v_j + jw_i$, $u_j' = v_j' + jw_i'$.

Game$_i$ ($i = 1, 2, \ldots, n+1$): modify ct as follows:

\[
\begin{align*}
\text{ct: } & \{ g_1^{s_j} g_1^{s_j(u_i+jw_i)} g_1^{s_j(w_i'+jw_i')} g_1^{s_j}, g_1^{s_j} \}_j \{ n \} \\
\end{align*}
\]

(A.6)

It is easy to know that Game$_0$ ≈ Game$_d$. Then, we will prove that Game$_d$ ≈ Game$_{r_1}$ through the following game sequence.

Game$_{r_1}$: modify ct as follows:

\[
\begin{align*}
\text{ct: } & \{ g_1^{s_j} g_1^{s_j(u_i+jw_i)} g_1^{s_j(w_i'+jw_i')} g_1^{s_j}, g_1^{s_j} \}_j \{ n \}.
\end{align*}
\]

(A.7)

Now, we briefly explain that Game$_d$ ≈ Game$_{r_1}$. Under the SD$_{P_1}$ → $p, p'$ assumption, we have

\[
\begin{align*}
& g_i^{s_j} g_i^{s_j}, \text{ given } g_1, g_2, h_1, h_2.
\end{align*}
\]

(A.8)

Game$_{r_2}$: modify sk as follows:

\[
\begin{align*}
\text{sk: } & \{ h_1^{r_j(u_i+jw_i)} h_2^{r_j(w_i'+jw_i')} h_2^{r_j}, h_2^{r_j'}, h_2^{r_j} \}_j \{ n \}.
\end{align*}
\]

(A.9)

Now, we briefly explain that Game$_{r_2}$ ≈ Game$_{r_2}$. Under the SD$_{P_1}$ assumption, we have

\[
\begin{align*}
& g_i^{s_j} g_i^{s_j}, \text{ given } g_1, g_2, h_1, h_2.
\end{align*}
\]

(A.8)
\[
\{ h_3^t, h_2^t, h_1^t \}_{t \in [n]} \\
\approx \{ h_3^t, h_2^t, h_1^t \}_{t \in [n]}, \quad \text{given } g_1, g_2, g_3, h_1, h_2, h_3, \quad (A.10)
\]

where \( \nu_j, \nu_j^t \longmapsto \mathbb{Z}_N \), and set \( u_j = \tilde{u}_0 + (j-i)\nu_j \), \( u_j^t = \tilde{u}_0^t + (j-i)\nu_j^t \).

\[
\begin{align*}
\{ u_0 = \tilde{u}_0 \mod p_1 p_2, & w_0 = \tilde{w}_0 - iw_0 \mod p_3, \\
\{ u_0^t = \tilde{u}_0^t \mod p_1 p_2, & w_0^t = \tilde{w}_0^t - iw_0^t \mod p_3. \}
\end{align*}
\quad (A.11)
\]

\textbf{Game}_{\text{r,3}}: modify sk and ct as follows:

\[
\begin{align*}
\text{ct}_i & : \{ g_1^t g_2^t g_3^t (w_0 + iw_0) g_2^t g_3^t (w_0 + iw_0) \}, \\
\text{sk}_i & : \{ h_1^t (w_0 + iw_0) h_2^t u_i h_3^t, h_1^t (w_0 + iw_0) h_2^t u_i h_3^t \}.
\end{align*}
\quad (A.12)
\]

It is easy to know that \( \text{Game}_{\text{r,2}} \approx \text{Game}_{\text{r,3}} \) based on the fact \( u_0 + iw_0 = u \mod p_3 \) and \( u_0^t + iw_0^t = u_i \mod p_3 \).

\textbf{Game}_{\text{r,4}}: modify ct as follows:

\[
\begin{align*}
\text{ct}_i & : \{ g_1^t g_2^t g_3^t (w_0 + iw_0) g_2^t g_3^t (w_0 + iw_0) \}.
\end{align*}
\quad (A.13)
\]

Now, we briefly explain that \( \text{Game}_{\text{r,3}} \approx \text{Game}_{\text{r,4}} \). Under the SDP assumption, we have

\[
g_2^t \approx g_2^t g_2^t, \quad \text{given } g_1, g_2, h_1, h_2, h_3. \quad (A.14)
\]

\textbf{Game}_{\text{r,5}}: modify sk and ct as follows:

\[
\begin{align*}
\text{ct}_i & : \{ g_1^t g_2^t g_3^t (w_0 + iw_0) g_2^t g_3^t (w_0 + iw_0) \}, \\
\text{sk}_i & : \{ h_1^t (w_0 + iw_0) h_2^t u_i h_3^t, h_1^t (w_0 + iw_0) h_2^t u_i h_3^t \}.
\end{align*}
\quad (A.15)
\]

It is easy to know that \( \text{Game}_{\text{r,4}} \approx \text{Game}_{\text{r,5}} \) based on the fact \( \text{Game}_{\text{r,2}} \approx \text{Game}_{\text{r,3}} \).

\textbf{Game}_{\text{r,6}}: modify sk as follows:

\[
\begin{align*}
\text{sk}_i & : \{ h_1^t (w_0 + iw_0) h_2^t u_i h_3^t, h_1^t (w_0 + iw_0) h_2^t u_i h_3^t \}.
\end{align*}
\quad (A.16)
\]

It is easy to know that \( \text{Game}_{\text{r,5}} \approx \text{Game}_{\text{r,6}} \) based on the fact \( \text{Game}_{\text{r,1}} \approx \text{Game}_{\text{r,2}} \).

\textbf{Game}_{\text{r,7}}: modify ct as follows:

\[
\begin{align*}
\text{ct}_i & : \{ g_1^t g_2^t g_3^t (w_0 + iw_0) g_2^t g_3^t (w_0 + iw_0) \}.
\end{align*}
\quad (A.17)
\]

It is easy to know that \( \text{Game}_{\text{r,6}} \approx \text{Game}_{\text{r,7}} \) based on the fact \( \text{Game}_{\text{r,1}} \approx \text{Game}_{\text{r,2}} \).

\textbf{Lemma A.2}. Sample \( \omega, u_j, v_j, s, s_j \longmapsto \mathbb{Z}_N \), and we have

\[
\text{aux} : g_1, g_2, h_1, g_1^t, g_2^t, h_1^t \\
\text{ct} : \{ g_1^t g_2^t g_3^t (w_0 + iw_0) g_2^t g_3^t (w_0 + iw_0) \} \\
\text{sk} : \{ h_1^t (w_0 + iw_0) h_2^t u_i h_3^t \}
\quad (A.18)
\]

\[
\begin{align*}
\text{aux} : g_1, g_2, h_1, g_1^t, g_2^t, h_1^t \\
\text{ct} : \{ g_1^t g_2^t g_3^t (w_0 + iw_0) g_2^t g_3^t (w_0 + iw_0) \} \\
\text{sk} : \{ h_1^t (w_0 + iw_0) h_2^t u_i h_3^t \}
\end{align*}
\quad (A.19)
\]
Proof. Under the $\text{DDH}_{F_{\mathcal{P}}}^{\text{GN}}$ assumption, we have
\[
\{g_j^x, g_j^s, \bar{g}_j^\omega\}_{j \in [n]} \approx \{g_j^x, g_j^{s'v}, \bar{g}_j^{\omega'}\}_{j \in [n]}, \tag{A.20}
\]
Suppose the adversary $A$ inputs $\{g_j^x, T_j, T'_j\}_{j \in [n]}$ and sets $u_j = r_j \cdot (\bar{u}_j - rw)$, $u'_j = (r'_j)^{-1} \cdot (\bar{u}_j - rw)$, then the system outputs
\[
\{\text{aux: } g_1, g_2, h_1, g_1^w, g_2^w, \bar{h}_2^w, \bar{h}_2^{\omega'}, \bar{h}_1^{\omega}\}_{j \in [n]},
\]
\[
\{\text{ct: } T_j, T'_j, g_2^v, g_2^s, (\bar{u}_j^v, r_j), T_j, g_2^{s'v}, (\bar{u}_j^{v'}, r'_j)\}_{j \in [n]},
\]
\[
\{\text{sk: } h_2^r, \bar{h}_2^r, \bar{h}_1^r, h_2^w, h_2^{\omega'}, h_2^{\omega'}\}_{j \in [n]}.
\] (A.21)

Now, we observe the above output and use this to illustrate the correctness of Lemma A.2.

(1) If $T_j = g_2^w$ and $T'_j = g_2^{\omega'}$ and we write $s, u_j = (s_j, r_j) \cdot (\bar{u}_j - rw)$ and $s, u_j' = (s_j, r_j') \cdot (\bar{u}_j - rw')$, we get $\bar{u}_j = u_j r_j + rw$, $\bar{u}_j' = u_j' r_j + rw'$ and the left distribution.

(2) If $T_j = g_2^{v'}$ and $T'_j = g_2^{v'}$ and we write $s, u_j = (s_j, r_j) \cdot (\bar{u}_j - rv_j)$ and $s, u_j' = (s_j, r_j') \cdot (\bar{u}_j - rv_j')$, we get $\bar{u}_j = u_j r_j + rv_j$, $\bar{u}_j' = u_j' r_j + rv_j'$ and the right distribution.

That is, if we can determine $\{T_j, T'_j\}_{j \in [n]}$, then the $\text{DDH}_{F_{\mathcal{P}}}^{\text{GN}}$ problem will be solved. \hfill $\sqcup$

Data Availability
The data used to support the findings of this study are included within the article.

Conflicts of Interest
The authors declare that they have no conflicts of interest.

Acknowledgments
This research was supported by the National Natural Science Foundation of China (nos. 61702548 and 61601515) and the Fundamental and Frontier Technology Research of Henan Province (no. 162300410192).

References