8-2017

Sequence aware functional encryption and its application in searchable encryption

Tran Viet Xuan PHUONG

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Willy SUSILO

Fuchun GUO

Qiong HUANG

Journal of Information Security and Applications 35 (2017) 106–118

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Sequence aware functional encryption and its application in searchable encryption

Tran Viet Xuan Phuong[a,*], Guomin Yang[a], Willy Susilo[a], Fuchun Guo[a], Qiong Huang[b]

[a] School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia
[b] College of Mathematics and Informatics, South China Agricultural University, China

ARTICLE INFO

Article history:
Available online 20 June 2017

Keywords:
Functional encryption
Hamming distance
Privacy preserving
Searchable encryption

ABSTRACT

As a new broad vision of public-key encryption systems, functional encryption provides a promising solution for many challenging security problems such as expressive access control and searching on encrypted data. In this paper, we present two Sequence Aware Function Encryption (SAFE) schemes. Such a scheme is very useful in many forensics applications where the order (or pattern) of the attributes forms an important characteristic of an attribute sequence. Our first scheme supports the matching of two bit strings, while the second scheme can support the matching of general characters. These two schemes are constructed based on the standard Decision Linear and Decision Bilinear Diffie-Hellman assumptions. In addition, we show that our SAFE schemes can also provide the additional feature of attribute-hiding, which is desirable in forensics applications. Moreover, we give an interesting application of SAFE schemes in constructing Sequential Aware Keyword Search (SAKS) schemes.

1. Introduction

Functional encryption, which is a general term for a range of new public-key encryption systems including Identity-based Encryption [1,6,7,19,21], Attribute-Based Encryption [4,5,11], Predicate Encryption [8,13,14,22], and Inner-Product Encryption [3,16,17], gives a new way of viewing encryption. It has provided a promising solution for many challenging security problems such as expressive access control and searchable encryption, which traditional public-key encryption cannot solve.

This paper focuses on a special type of functional encryption named *Sequence Aware Functional Encryption (SAFE)*. The essential property of a SAFE scheme is that a ciphertext CT encrypted under a string V can be decrypted by a secret key corresponding to another string X if and only if V and X have a 'similar' pattern. For example, if V is a DNA sequence 'GACTCAGT', then both "GACTCACC' and "TTCTCAGT" can be considered similar if we allow different symbols to appear in at most two positions. The term "Sequence Aware" comes from that fact that the order of the symbols in the sequences forms a key attribute in measuring the similarity.

SAFE schemes can be defined and constructed based on different similarity metrics such as Hamming Distance, Edit Distance, Longest Common Subsequence, etc. In this paper, we focus on the constructions based on the Hamming Distance and propose two SAFE schemes that can support matching test between binary strings and general character strings, respective. As user privacy is another important issue that must be addressed in many forensics applications, privacy-preserving is another essential requirement in the design of a SAFE scheme. In SAFE, this means if the decryption sequence does not match the encryption sequence, then no information about the encrypted data or the encryption sequence (except the fact of mismatch) should be revealed.

Similar to the "Fuzzy Identity-Based Encryption" proposed by Sahai and Waters [19], a SAFE scheme naturally implies a 'fault-tolerant" IBE, and hence can be used in the application of IBE with biometric-based identity. Since biometric data such as fingerprint or DNA sequence is very sensitive personal information, it is desirable that this information associated with a ciphertext is hidden from any party who cannot successfully decrypt the ciphertext. Therefore, to implement an IBE using biometric-based identity, it is more preferred to use a SAFE.

Our Contribution. Our goal is to define and construct secure SAFE schemes under standard assumptions. More precisely, this paper contributes towards the above goal via the following steps.

- We provide a formal definition and security model for SAFE schemes. In particular, the security model can capture the property of attribute-hiding (i.e., privacy-preserving).

* Corresponding author.
E-mail addresses: tvxp750@uowmail.edu.au (T.V. Xuan Phuong), gyang@uow.edu.au (G. Yang), wsusilo@uow.edu.au (W. Susilo), fuchun@uow.edu.au (F. Guo), csqhuang-c@my.cityu.edu.hk (Q. Huang).

http://dx.doi.org/10.1016/j.jisa.2017.06.002
2214-2126/

- We first present a SAFE construction (named SAFE-1) for binary strings. Then we address the problem of building a SAFE scheme for general character strings. For both schemes, the decryption can be processed successfully if and only if the Hamming Distance of two bit/character strings is equal to a value k associated with a user's key. In order to allow the matching test in our SAFE-2 scheme, we also propose a novel technique for computing the Hamming distance between two DNA strings including four characters {*A, T, G, C*}. Extensively, we show that our scheme can be used the general characters (in this case we use English characters) for the computation of Hamming distance between two strings. We also prove the security of our SAFE-1 and SAFE-2 schemes under two standard assumptions – the Decision Linear assumption and Decision Bilinear Diffie-Hellman assumption.
- In addition, we extend our SAFE-2 scheme to a Sequence Aware Keyword Search (SAKS) scheme, which allows a fuzzy matching test between two (encrypted) DNA strings.

Related Work. A SAFE scheme may look very similar to the "Fuzzy Identity-Based Encryption" proposed by Sahai and Waters [19]. In a Fuzzy IBE, a ciphertext created under an ID ω' can be decrypted by a private key corresponding to another ID ω if the set-overlap between ω and ω' is above a threshold. Sahai and Waters also mentioned that a Fuzzy IBE based on the Hamming Distance can be built by following a similar technique. However, the Sahai and Waters Fuzzy IBE requires the encryption identity to be attached with the ciphertext, and therefore does not satisfy the property of attribute-hiding. Cheung et al. [9] later proposed another Fuzzy IBE scheme with the attribute-hiding property. However, their construction is under the composite-order group. According to the results in [10,12], pairing in composite-order group is much more expensive (between 10 times to 192 times slower) than that in prime-order group. Compared with [9], both of our SAFE schemes are under prime-order group, and hence are more efficient.

Another work that is closely related to ours is the Wildcarded IBE (or WIBE for short) proposed by Abdalla et al. in [1,2]. A wildcarded IBE allows wildcard symbols to appear in an identity used in the encryption process, and the wildcard positions will be ignored when measuring the equality of two identity strings. It is easy to see that a WIBE scheme can also be considered as a SAFE scheme where the similarity metric is defined via an equality test supporting wildcard symbols. However, different from the hamming distance, in WIBE, the non-wildcard symbols at the same position of the two sequences must be identical.

In a predicate encryption system [8,14,13,22] for a (polynomial-time) predicate P, two inputs (besides some public parameters) are required in the encryption process, one is the message M to be encrypted, and the other one is an index string i. A decryption key is generated based on a master secret and a key index k. The decryption key can successfully decrypt a valid encryption of (i, M) if and only if $P(k, i) = 1$. In this sense, a SAFE scheme can also be treated as a predicate encryption scheme where the predicate P is the similarity between i and k.

As one type of predicate encryption, Hidden Vector Encryption (HVE) schemes [15,18,20] can also be treated as SAFE schemes based on equality test supporting wildcard symbols. However, in HVE schemes, wildcard symbols will appear in the attribute string associated with the user secret key rather than that of the ciphertext. Recently, Waters [23] proposed functional encryption for regular languages. Using the terminology of predicate encryption, the key index k is a Deterministic Finite Automata (DFA), while the index key i is an input for the DFA. The message can be recovered if and only if i is accepted by k.

Organization of the paper: In the next section, we present the definition, security models, and some preliminaries that will be used in the rest of the paper. The SAFE-1 scheme for bit string is then presented in Section 3, followed by the SAFE-2 scheme for general character strings in Section 4. We then give the security proofs for both schemes in Section 5. The Sequence Aware Keyword Search (SAKS) scheme is presented in Section 6. We compare and discuss in Section 7. The paper is concluded in Section 8.

2. Preliminaries

2.1. SAFE Scheme

Let $\vec{v}, \vec{x}$ be two vectors over a finite alphabet Σ and have the same length n. We define a predicate $F = \{f_{\vec{v}} | \vec{v} \in \Sigma\}$ based on the Hamming Distance of two vectors such that $f_{\vec{v}}(\vec{x}) = 1$ iff Hamming Distance $(\vec{v}, \vec{x}) = k$ where $0 \leq k \leq n$ is a fixed integer.

Definition 1. A SAFE scheme based on Hamming Distance is a probabilistic polynomial-time algorithms which has four algorithms as follows:

- **Setup**$(1^\lambda, n)$ on input a security parameter 1^λ and the vector length $n = poly(\lambda)$, the algorithm outputs a public key PK and a master secret key MSK.
- **Encrypt**$(M, PK, \vec{v} = (v_1, v_2, \ldots, v_n))$: on input a message M, the public key PK, and a vector $\vec{v} \in \Sigma^n$, it outputs a ciphertext CT.
- **KeyGen**$(MSK, \vec{x} = (x_1, x_2, \ldots, x_n), k)$: on input the master secret key MSK, a vector $\vec{x} \in \Sigma$, and an integer $k \leq n$, the algorithm outputs a secret key SK.
- **Decrypt**(CT, SK): on input a secret key SK (w.r.t. a vector $\vec{x}$) and a ciphertext CT (w.r.t. a vector $\vec{v}$), if $f_{\vec{v}}(\vec{x}) = 1$ (i.e. Hamming Distance $(\vec{v}, \vec{x}) = k$), the algorithm outputs a message M; otherwise, it outputs $\perp$.

2.2. *Security model for a SAFE scheme*

Definition 2. A SAFE scheme based on Hamming Distance is selectively secure if for any PPT adversary $\mathcal{A}$, its advantage defined in the following interactive game with a challenger $\mathcal{B}$ is negligible in the security parameter λ.

1. **Init:** $\mathcal{A}$ outputs two vectors $\vec{v}, \vec{x} \in \Sigma$ and an integer k.
2. **Setup:** The challenger $\mathcal{B}$ runs setup algorithm to generate PK and MSK, then sends PK to $\mathcal{A}$.
3. **Query Phase 1**: $\mathcal{A}$ can adaptively request keys for any vector $\vec{y} \in \Sigma$ with the following constrain
   - Hamming Distance $(\vec{v}, \vec{y}) = k$ if and only if Hamming Distance $(\vec{x}, \vec{y}) = k$.

   $\mathcal{B}$ responds to $\mathcal{A}$ with $SK \leftarrow KeyGen(MSK, \vec{y}, k)$.
4. **Challenge**: $\mathcal{A}$ outputs two messages M_0, M_1 with equal length. If $M_0 \neq M_1$, then it is required that Hamming Distance $(\vec{v}, \vec{y}) \neq k \neq$ Hamming Distance $(\vec{x}, \vec{y})$ for any $\vec{y}$ appeared in Query Phase 1. $\mathcal{B}$ flips a random coin $b \in \{0, 1\}$. If $b = 0$, $\mathcal{B}$ returns $CT \leftarrow Encrypt(PK, \vec{v}), M_0$ to $\mathcal{A}$; otherwise, if $b = 1$, $\mathcal{B}$ returns $CT \leftarrow Encrypt(PK, \vec{x}), M_1$ to $\mathcal{A}$.
5. **Query Phase 2**: Phase 1 is repeatedly.
6. **Guess:** $\mathcal{A}$ outputs a guess bit b' and succeeds if $b' = b$.

The advantage of $\mathcal{A}$ in this game is defined as

$$\mathbf{Adv}_{\mathcal{A}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

Full Security: In the above selective security model, the adversary is required to commit the challenge vectors before seeing the system parameters. In the full security model [16], the adversary can choose the challenge vectors in the Challenge phase, which makes the model stronger.

2.3. Bilinear map and its related assumptions

Let $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ be two multiplicative cyclic groups of same prime order p. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ be a bilinear map with the following properties:

1. Bilinearity : $e(u^a, v^b) = e(u^b, v^a) = e(u, v)^{ab}$. for any $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$.
2. Non-degeneracy : $e(g, g) \neq 1$

Definition 3. The Decisional Bilinear Diffie-Hellman (DBDH) problem in $\mathbb{G}$ is defined as follows: given a tuple $(g, g^a, g^b, g^c, T) \in \mathbb{G}^4 \times \mathbb{G}_T$, decide whether $T = e(g, g)^{abc}$ or $T = e(g, g)^r$ where a, b, c, r are randomly selected from $\mathbb{Z}_p$. An algorithm A has advantage ϵ in solving the DBDH problem in $\mathbb{G}$ if

$$\mathbf{Adv}_A^{\mathrm{DBDH}}(\lambda) = \Pr[A(1^k, g, g^a, g^b, g^c, Z) = 1 | Z = e(g, g)^{abc}] - \Pr[A(1^k, g, g^a, g^b, g^c, Z) = 1 | Z = g^r] \leq \epsilon.$$

We say that the DBDH assumptions holds in $\mathbb{G}$ if ϵ is negligible for any PPT algorithm A.

Definition 4. The Decisional Linear (DLIN) problem in $\mathbb{G}$ defined as follows: given a tuple $(g, g^a, g^b, g^{ac}, g^d, Z) \in \mathbb{G}^5 \times \mathbb{G}_T$, decide whether $T = g^{b(c+d)}$ or Z in random in $\mathbb{G}$. An algorithm A has advantage ϵ in solving the DLIN problem in $\mathbb{G}$ if

$$\mathbf{Adv}_A^{\mathrm{DLIN}}(\lambda) = \Pr[A(1^k, g, g^a, g^b, g^{ac}, g^d, Z) = 1 | Z = g^{b(c+d)}] - \Pr[A(1^k, g, g^a, g^b, g^{ac}, g^d, Z) = 1 | Z = g^r] \leq \epsilon$$

where $a, b, c, d, r \in_R \mathbb{Z}_p$. We say that the DLIN assumptions holds in $\mathbb{G}$ if ϵ is negligible for any PPT algorithm A.

2.4. Hamming distance

Lemma 1. *Let v and x be two bit strings of equal length n, then Hamming Distance $(v, x) = \sum_{i=1}^{n} v_i(1 - 2x_i) + 1 \times \sum x_i$ where $x = (x_1, \ldots, x_n), v = (v_1, \ldots, v_n)$.*

Proof. Since

$$a \oplus b = a + b - 2ab = a(1 - 2b) + b, \forall a, b \in [0, 1] \quad (1)$$

we can compute the Hamming Distance (HD) between two bit vectors v and x as follows:

$$\mathrm{HD}(v, x) = \sum_{i=1}^{n} v_i \oplus x_i = \sum_{i=1}^{n} v_i(1 - 2x_i) + \left(\sum_{i=1}^{n} x_i\right).$$

□

3. SAFE-1 For bit string

We present our SAFE-1 scheme for bit strings in this section. The new scheme achieves the attribute hiding property, and can be proven secure under standard assumptions. Our scheme works as follows.

- **Setup**$(1^k, n)$: The setup algorithm first randomly generates $(g, \mathbb{G}, \mathbb{G}_T, p, e)$. It then chooses randomly $\gamma_1, \gamma_2, \theta_1, \theta_2, u_1, \{u_{1,i}\}_{i=1}{}^n, t_1, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}{}^n, t_2, w_1, \{w_{1,i}\}_{i=1}^n, z_1\{z_{1,i}\}_{i=1}{}^n, \{z_{2,i}\}_{i=1}{}^n, z_2$ in $\mathbb{Z}_p$ and g_2 in $\mathbb{G}$. Then it selects a random $\Delta \in \mathbb{Z}_p$ and obtains $\{u_{2,i}\}^n{}_{i=1}, \{w_{2,i}\}^n{}_{i=1}, w_2, u_2$ under the condition:

  $\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}, \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i},$
  $\Delta = \gamma_1 u_2 - \gamma_2 u_1, \Delta = \theta_1 w_2 - \theta_2 w_1.$

  For i from 1 to n, it creates:

  $U_{1,i} = g^{u_{1,i}}, U_{2,i} = g^{u_{2,i}}, U_1 = g^{u_1}, U_2 = g^{u_2},$
  $W_{1,i} = g^{w_{1,i}}, W_{2,i} = g^{w_{2,i}}, W_1 = g^{w_1}, W_2 = g^{w_2},$
  $T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}}, T_1 = g^{t_1}, T_2 = g^{t_2},$
  $Z_{1,i} = g^{z_{1,i}}, Z_{2,i} = g^{z_{2,i}}, Z_1 = g^{z_1}, Z_2 = g^{z_2},$
  $V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}.$

  Next it sets $g_1 = g^{\Delta}, Y = e(g, g_2)$, and the public key PK and master key MSK as

  $$PK = (g, \mathbb{G}, \mathbb{G}_T, p, e, g_1, Y, \{U_{1,i}, U_{2,i}, T_{1,i}, T_{2,i}, W_{1,i}, W_{2,i}, Z_{1,i}, Z_{2,i}\}_{i=1}^n, \{U_i, T_i, V_i, X_i\}_{i=1}{}^2)$$
  $$MSK = (g_2, \{u_{1,i}, u_{2,i}, t_{1,i}, t_{2,i}, w_{1,i}, w_{2,i}, z_{1,i}, z_{2,i}\}^n{}_{i=1}, \{u_i, t_i, w_i, z_i, \gamma_i, \theta_i\}^2{}_{i=1}).$$

- **Encryption**$(PK, M, \vec{v} = (v_1, v_2, \ldots, v_n))$: The encryption algorithm chooses random $s_1, s_2, \alpha, \beta \in \mathbb{Z}_p$ and creates the ciphertext as follows:

  $$C_m = M \cdot Y^{s_2}, C_A = g^{s_2}, C_B = g_1^{s_1},$$
  $$\{C_{1,i}, C_{2,i}\} = \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i \alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i \alpha}\},$$
  $$\{C_1, C_2\} = \{U_1^{s_1} T_1^{s_2} V_1^{\alpha}, U_2^{s_1} T_2^{s_2} V_2^{\alpha}\},$$
  $$\{C_{3,i}, C_{4,i}\} = \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{v_i \beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{v_i \beta}\},$$
  $$\{C_3, C_4\} = \{W_1^{s_1} Z_1^{s_2} X_1^{\beta}, W_2^{s_1} Z_2^{s_2} X_2^{\beta}\}.$$

  Then ciphertext CT is set as:

  $$CT = (C_m, C_A, C_B, \{C_{1,i}, C_{2,i}, C_{3,i}, C_{4,i}\}_{i=1}^n, \{C_1, C_2, C_3, C_4\}).$$

- **KeyGen**$(MSK, \vec{x} = (x_1, x_2, \ldots, x_n), k)$: The key generation algorithm chooses randomly $r_{i,1}$, $r_{i,2}$ for $i = 1$ to n, and $f_1, f_2, r_1, r_2 \in \mathbb{Z}_p$, and then creates the secret key as follows:

  $$\{K_{1,i}, K_{2,i}\} = \{g^{-\gamma_2 r_{1,i}} g^{f_1(1-2x_i)u_{2,i}}, g^{\gamma_1 r_{1,i}} g^{-f_1(1-2x_i)u_{1,i}}\},$$
  $$\{K_1, K_2\} = \{g^{-\gamma_2 r_1} g^{f_1(\sum x_i - k)u_2}, g^{\gamma_1 r_1} g^{-f_1(\sum x_i - k)u_1}\},$$
  $$\{K_{3,i}, K_{4,i}\} = \{g^{-\theta_2 r_{2,i}} g^{f_2(1-2x_i)w_{2,i}}, g^{\theta_1 r_{2,i}} g^{-f_2(1-2x_i)w_{1,i}}\},$$
  $$\{K_3, K_4\} = \{g^{-\theta_2 r_2} g^{f_2(\sum x_i - k)w_2}, g^{\theta_1 r_2} g^{-f_2(\sum x_i - k)w_1}\},$$
  $$K_A = g_2 \cdot K_1^{-t_1} K_2^{-t_2} K_3^{-z_1} K_4^{-z_2} \prod_{i=1}^{n} K_{1,i}^{-t_{1,i}} K_{2,i}^{-t_{2,i}} K_{3,i}^{-z_{1,i}} K_{4,i}^{-z_{2,i}},$$
  $$K_B = g^{-(r_1 + r_2)} \prod_{i=1}^{n} g^{-(r_{1,i} + r_{2,i})}.$$

  The secret key is set as:

  $$SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^n, \{K_1, K_2, K_3, K_4\}).$$

- **Decrypt**(SK, CT): Given a ciphertext CT associated with a string $\vec{v} = (v_1, \ldots, v_n)$ and a secret key SK associated with a string $\vec{x} = (x_1, \ldots, x_n)$ and hamming distance k, the decryption algorithm returns

  $$\frac{C_m}{e(C_A, K_A) \cdot e(C_B, K_B) \cdot \prod_{j=1}^{4} e(C_j, K_j) \prod_{i=1}^{n} e(C_{j,i}, K_{j,i})}.$$

  Correctness: This correctness follows to the SAFE-1 scheme.

  $$e(C_{1,i}, K_{1,i}) = e(U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i \alpha}, g^{-\gamma_2 r_{1,i}} g^{f_1(1-2x_i)u_{2,i}})$$
  $$= e(g, g)^{r_{1,i} s_1 (-u_{1,i} \gamma_2)} \cdot e(g, g)^{-r_{1,i} v_i \alpha \gamma_1 \gamma_2} \cdot e(g, K_{1,i})^{t_{1,i} s_2} \cdot e(g, g)^{f_1(1-2x_i) u_{1,i} u_{2,i} s_1}$$

$$
\begin{aligned}
&\cdot e(g,g)^{f_1 v_i(1-2x_i)\alpha\gamma_1 u_{2,i}}.\\
&e(C_{2,i},K_{2,i})\\
&= e(U_{2,i}^{s_1}T_{2,i}^{s_2}V_2^{v_i\alpha}, g^{\gamma_1 r_{1,i}}g^{-f_1(1-2x_i)u_{1,i}})\\
&= e(g,g)^{r_{1,i}s_1u_{2,i}\gamma_1}\cdot e(g,g)^{r_{1,i}v_i\alpha\gamma_1\gamma_2}\\
&\cdot e(g,K_{2,i})^{t_{2,i}s_2}\cdot e(g,g)^{-f_1(1-2x_i)u_{1,i}u_{2,i}s_1}\\
&\cdot e(g,g)^{-f_1v_i(1-2x_i)\alpha\gamma_2u_{1,i}}.\\
&\prod_{j=1}^{2}\prod_{i=1}^{n}e(C_{j,i},K_{j,i})\\
&= \prod_{i=1}^{n}e(g,g)^{r_{1,i}s_1\Delta}\cdot e(g,g)^{f_1v_i(1-2x_i)\alpha\Delta}\\
&\cdot e(g,K_{1,i})^{t_{1,i}s_2}e(g,K_{2,i})^{t_{2,i}s_2}.\\
&e(C_1,K_1)\\
&= e(U_1^{s_1}T_1^{s_2}V_1^{\alpha}, g^{-\gamma_2r_1}g^{f_1(\sum x_i-k)u_2})\\
&= e(g,g)^{r_1s_1(-u_1\gamma_2)}\cdot e(g,g)^{f_1(\sum x_i-k)u_1u_2s_1}\\
&\cdot e(g,g)^{-r_1\alpha\gamma_1\gamma_2}\cdot e(g,g)^{f_1(\sum x_i-k)\alpha\gamma_1u_2}.\\
&e(g,K_1)^{t_1s_2}.\\
&e(C_2,K_2)\\
&= e(U_2^{s_1}T_2^{s_2}V_2^{\alpha}, g^{\gamma_1r_1}g^{-f_1(\sum x_i-k)u_1})\\
&= e(g,g)^{r_1s_1u_2\gamma_1}\cdot e(g,g)^{-f_1(\sum x_i-k)u_1u_2s_1}\\
&\cdot e(g,g)^{r_1\alpha\gamma_1\gamma_2}\cdot e(g,g)^{-f_1(\sum x_i-k)\alpha\gamma_2u_1}\\
&\cdot e(g,K_2)^{t_2s_2}.\\
&\prod_{j=1}^{2}e(C_j,K_j)\\
&= e(g,g)^{r_1s_1\Delta}\cdot e(g,g)^{f_1(\sum x_i-k)\alpha\Delta}\\
&\cdot e(g,K_1)^{t_1s_2}e(g,K_2)^{t_2s_2}.
\end{aligned}
$$

Similarly:

$$
\begin{aligned}
\prod_{j=3}^{4}\prod_{i=1}^{n}e(C_{j,i},K_{j,i}) &= \prod_{i=1}^{n}e(g,g)^{r_{2,i}s_1\Delta}\cdot e(g,g)^{f_2v_i(1-2x_i)\beta\Delta}\\
&\cdot e(g,K_{3,i})^{z_{1,i}s_2}e(g,K_{4,i})^{z_{2,i}s_2}.\\
&\prod_{j=3}^{4}e(C_j,K_j)\\
&= e(g,g)^{r_2s_1\Delta}\cdot e(g,g)^{f_2(\sum x_i-k)\beta\Delta}\\
&\cdot e(g,K_3)^{z_1s_2}e(g,K_4)^{z_2s_2}.
\end{aligned}
$$

Then we have:

$$
\begin{aligned}
&\prod_{j=1}^{4}e(C_j,K_j)\prod_{i=1}^{n}e(C_{j,i},K_{j,i})\\
&= e(g,K_1)^{t_1s_2}e(g,K_2)^{t_2s_2}e(g,K_3)^{z_1s_2}e(g,K_4)^{z_2s_2}\\
&\quad e(g,g)^{r_1s_1\Delta}e(g,g)^{r_2s_1\Delta}\\
&\quad e(g,g)^{(\sum v_i(1-2x_i)+\sum x_i-k)f_1\alpha\Delta}\\
&\quad e(g,g)^{(\sum v_i(1-2x_i)+\sum x_i-k)f_2\beta\Delta}\\
&\quad \prod_{i=1}^{n}e(g,K_{1,i})^{t_{1,i}s_2}e(g,K_{2,i})^{t_{2,i}s_2}\\
&\quad e(g,K_{3,i})^{z_{1,i}s_2}e(g,K_{4,i})^{z_{2,i}s_2}\\
&\quad e(g,g)^{r_{1,i}s_1\Delta}e(g,g)^{r_{2,i}s_1\Delta}.
\end{aligned}
$$

Also, since

$$
e(C_A,K_A) = e\left(g^{s_2}, g_2\cdot K_1^{-t_1}K_2^{-t_2}K_3^{-z_1}K_4^{-z_2}\prod_{i=1}^{n}K_{1,i}^{-t_{1,i}}K_{2,i}^{-t_{2,i}}K_{3,i}^{-z_{1,i}}K_{4,i}^{-z_{2,i}}\right)
$$

$$
e(C_B,K_B) = e\left(g^{s_1\Delta}, g^{-(r_1+r_2)}\prod_{i=1}^{n}g^{-(r_{1,i}+r_{2,i})}\right)
$$

Fig. 1 we have

$$
\begin{aligned}
&\frac{C_m}{e(C_A,K_A)\cdot e(C_B,K_B)\cdot\prod_{j=1}^{4}e(C_j,K_j)\prod_{i=1}^{n}e(C_{j,i},K_{j,i})}\\
&= \frac{M}{e(g,g)^{(\sum v_i(1-2x_i)+\sum x_i-k)(f_2\beta+f_1\alpha)\Delta}}.
\end{aligned}
$$

Therefore, the message M will be returned iff Hamming Distance $(v,x)=k$ meaning $\sum v_i(1-2x_i)+\sum x_i-k=0$.

4. SAFE-2 For character string

In this section, we present our SAFE-2 scheme based on the Hamming distance of two character (instead of bit) strings. The major challenge here is that we cannot directly apply the Hamming distance evaluation formula given in Section II-D for character strings. In order to address this problem, we propose a new technique for computing the hamming distance of two characters.

4.1. Hamming distance of character string

For simplicity, we demonstrate our idea using DNA strings where the alphabet is {*A, G, T, C*}. Then each character in a DNA string can be encoded as

$$
\begin{cases}
A = 00\\
G = 01\\
T = 10\\
C = 11.
\end{cases}
$$

We should note that the idea can be easily extended to strings consisting of characters from a different alphabet.

Let X and Y denote two characters that are located at the same position in two character strings. We first calculate the Hamming distance between X and Y by applying the XOR operation between the two 2-bit strings corresponding to X and Y, and then obtain the Hamming distance between two character strings based on the Hamming distance of individual characters. Below we demonstrate our technique using a concrete example.

Suppose that we have two character strings:

$$
\begin{cases}
\vec{v} = (A,T,C)\\
\vec{x} = (C,G,A).
\end{cases}
$$

First, each character is converted to a bit string:

$$
\begin{cases}
\vec{v_1} = A = (0,0)\\
\vec{v_2} = T = (1,0)\\
\vec{v_3} = C = (1,1)\\
\vec{x_1} = G = (0,1)\\
\vec{x_2} = T = (1,0)\\
\vec{x_3} = A = (0,0)
\end{cases}
$$

Then, the Hamming distance between individual characters is calculated as follows:

$$
\begin{cases}
1-(A\boxplus G) = 1-[1-(0\oplus 0)]\times[1-(0\oplus 1)]\\
\qquad\qquad = 1-0=1\\
1-(T\boxplus T) = 1-[1-(1\oplus 1)]\times[1-(0\oplus 0)]\\
\qquad\qquad = 1-1=0\\
1-(C\boxplus A) = 1-[1-(1\oplus 0)]\times[1-(1\oplus 0)]\\
\qquad\qquad = 1-0=1
\end{cases}
$$

$$\begin{aligned}\prod_{i=1}^{2}(1-x_i-y_i+2x_iy_i) &= (1-x_1-y_1+2x_1y_1)\cdot(1-x_2-y_2+2x_2y_2)\\ &= 1-x_1-y_1-x_2-y_2-2x_2y_2+x_1x_2+x_1y_2-2x_1x_2y_2\\ &\quad +x_2y_1+y_1y_2-2x_1y_1y_2-2x_1x_2y_1-2x_1y_1y_2+4x_1x_2y_1y_2\end{aligned} \tag{2}$$

Hence, we can construct two vectors as :

$$\begin{aligned}\vec{x_A} &= (1, \quad -x_1,-1,-2x_1,-x_2,-1,-2x_2,x_1x_2,x_1,-2x_1x_2,x_2,1,-2x_1,-2x_1x_2,-2x_1,4x_1x_2)\\ \vec{x_B} &= (1, \quad 1,y_1,y_1,1,y_2,y_2,y_2,1,y_2,y_2,y_1,y_1y_2,y_2,y_1,y_1y_2,y_1y_2).\end{aligned} \tag{3}$$

Fig. 1. The algorithm to create two vectors to check the hamming distance between two characters.

Therefore, we have $HammingDistance(\vec{v},\vec{x}) = 2$. From the above simply example, we can get the following Lemma.

Lemma 2. *Let x and y be two characters from an alphabet where each character can be represented using 2 bits, then Hamming Distance $(x,y) = \prod_{i=1}^{2}(1-x_i-y_i+2x_iy_i)$.*

Proof. Since

$$a \oplus b = a + b - 2ab \forall a, b \in [0,1] \tag{4}$$

we can compute the Hamming Distance (HD) between two bit vectors x and y as follows:

$$\mathrm{HD}(x,y) = \prod_{i=1}^{2}(1-x_i\oplus y_i) = \prod_{i=1}^{2}(1-x_i-y_i+2x_iy_i).$$

Hence, we can derive a general formula to compute the Hamming distance of two characters as follows. Given

$\vec{A} = (a_1, a_2)$
$\vec{B} = (b_1, b_2)$

we first construct two vectors following (3) and (4)

$$\begin{aligned}\vec{x_A} &= (1,-a_1,-1,-2a_1,-a_2,-1,-2a_2,a_1a_2,a_1,\\ &\quad -2a_1a_2,a_2,1,-2a_1,-2a_1a_2,-2a_1,4a_1a_2) \qquad (i)\\ \vec{x_B} &= (1,1,b_1,b_1,1,b_2,b_2,b_2,1,b_2,b_2,b_1,b_1b_2,b_2,\\ &\quad b_1,b_1b_2,b_1b_2) \qquad (ii).\end{aligned}$$

Then

$$\begin{aligned}<\vec{x_A},\vec{x_B}> &= \prod_{i=1}^{2}(1-a_i-b_i+2a_ib_i)\\ &= (1-(a_1\oplus b_1))\times(1-(a_2\oplus b_2))\\ &= HammingDistance(\vec{A},\vec{B}).\end{aligned} \tag{5}$$

□

4.2. Construction of SAFE2-scheme:

Based on the method for calculating the hamming distance of two characters, we present our SAFE-2 scheme below.

- **Setup**$(1^k, \Sigma, n)$: The setup algorithm first randomly generates $(g, \mathbb{G}, \mathbb{G}_T, p, e)$. It then chooses randomly $\gamma_1, \gamma_2, \theta_1, \theta_2, u_1, \{u_{1,i}\}_{i=1}^n, t_1, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, t_2, w_1, \{w_{1,i}\}_{i=1}^n$, $z_1\{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n, z_2$ in $\mathbb{Z}_p$ and g_2 in $\mathbb{G}$. Then it selects a random $\Delta \in \mathbb{Z}_p$ and obtains $\{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, u_2$ under the condition:

  $$\Delta = \gamma_1u_{2,i} - \gamma_2u_{1,i}, \Delta = \theta_1w_{2,i} - \theta_2w_{1,i},$$
  $$\Delta = \gamma_1u_2 - \gamma_2u_1, \Delta = \theta_1w_2 - \theta_2w_1.$$

  For i from 1 to n, it creates:

  $$U_{1,i} = g^{u_{1,i}}, U_{2,i} = g^{u_{2,i}}, U_1 = g^{u_1}, U_2 = g^{u_2},$$
  $$W_{1,i} = g^{w_{1,i}}, W_{2,i} = g^{w_{2,i}}, W_1 = g^{w_1}, W_2 = g^{w_2},$$
  $$T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}}, T_1 = g^{t_1}, T_2 = g^{t_2},$$
  $$Z_{1,i} = g^{z_{1,i}}, Z_{2,i} = g^{z_{2,i}}, Z_1 = g^{z_1}, Z_2 = g^{z_2},$$
  $$V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}.$$

  Next it sets $g_1 = g^{\Delta}, Y = e(g, g_2)$, and the public key PK and master key MSK as

  $$\begin{aligned}PK &= (g, \mathbb{G}, \mathbb{G}_T, p, e, g_1, Y, \{U_{1,i}, U_{2,i}, T_{1,i}, T_{2,i},\\ &\quad W_{1,i}, W_{2,i}, Z_{1,i}, Z_{2,i}\}^n{}_{i=1}, \{U_i, T_i, V_i, X_i\}_{i=1}{}^2)\\ MSK &= (g_2, \{u_{1,i}, u_{2,i}, t_{1,i}, t_{2,i}, w_{1,i}, w_{2,i}, z_{1,i}, z_{2,i}\}^n{}_{i=1},\\ &\quad \{u_i, t_i, w_i, z_i, \gamma_i, \theta_i\}^2{}_{i=1}).\end{aligned}$$

- **Encryption**$(PK, M, \vec{v'} = (v'_1, v'_2, \ldots, v'_n) \in \Sigma^n)$: The encryption algorithm chooses random $s_1, s_2, \alpha, \beta \in \mathbb{Z}_p$. In $\vec{v'}$, each v'_i is converted to a 2-bit string (a_1, a_2). Then it computes as in (i):

  $$v_{i_0} = 1, v_{i_1} = -a_1, v_{i_2} = -1, \ldots, \quad v_{i_{14}} = -2a_1, \quad v_{i_{15}} = 4a_1a_2,$$

  Then a vector $\vec{v}$ for $\vec{v'}$ is created as

  $$\vec{v} = (v_{1_0}, \ldots, v_{1_{15}}, v_{2_0}, \ldots, v_{2_{15}}, \ldots, v_{n_0}, \ldots, v_{n_{15}}).$$

  Next, the ciphertext is created as follows

  $$\begin{aligned}C_m &= M\cdot Y^{s_2}, C_A = g^{s_2}, C_B = g_1^{s_1},\\ \{C_{1,i,j}, C_{2,i,j}\} &= \{U_{1,i}^{s_1}T_{1,i}^{s_2}V_1^{v_{i_j}\alpha}, U_{2,i}^{s_1}T_{2,i}^{s_2}V_2^{v_{i_j}\alpha}\},\\ \{C_1, C_2\} &= \{U_1^{s_1}T_1^{s_2}V_1^{\alpha}, U_2^{s_1}T_2^{s_2}V_2^{\alpha}\},\\ \{C_{3,i,j}, C_{4,i,j}\} &= \{W_{1,i}^{s_1}Z_{1,i}^{s_2}X_1^{v_{i_j}\beta}, W_{2,i}^{s_1}Z_{2,i}^{s_2}X_2^{v_{i_j}\beta}\},\\ \{C_3, C_4\} &= \{W_1^{s_1}Z_1^{s_2}X_1^{\beta}, W_2^{s_1}Z_2^{s_2}X_2^{\beta}\}.\end{aligned}$$

  The final ciphertext CT is set as

  $$\begin{aligned}CT &= (C_m, C_A, C_B, C_1, C_2, C_3, C_4,\\ &\quad \{\{C_{1,i,j}, C_{2,i,j}, C_{3,i,j}, C_{4,i,j}\}_{j=0}^{15}\}_{i=1}^n).\end{aligned}$$

- **KeyGen**$(MSK, \vec{x'} = (x'_0, x'_1, \ldots, x'_n) \in \Sigma^n, k)$: Choose randomly $r_{i,1}$, $r_{i,2}$ for $i = 1$ to n, and $f_1, f_2, r_1, r_2 \in \mathbb{Z}_p$. In $\vec{x'}$, each x'_i is first converted to a 2-bit string (b_1, b_2). Then it computes as in (ii):

  $$x_{i_0} = 1, x_{i_1} = 1, x_{i_2} = b_1, \ldots, \quad x_{i_{14}} = b_1b_2, \quad x_{i_{15}} = b_1b_2$$

  Then a vector $\vec{x}$ is created as follows

  $$\vec{x} = (x_{1_0}, \ldots, x_{1_{15}}, x_{2_0}, \ldots, x_{2_{15}}, \ldots, x_{n_0}, \ldots, x_{n_{15}}).$$

  Next, compute

  $$\begin{aligned}\{K_{1,i,j}, K_{2,i,j}\} &= \{g^{-\gamma_2r_{1,i}}g^{f_1x_{i_j}u_{2,i}}, g^{\gamma_1r_{1,i}}g^{f_1x_{i_j}u_{1,i}}\},\\ \{K_1, K_2\} &= \{g^{-\gamma_2r_1}g^{f_1(-k)u_2}, g^{\gamma_1r_1}g^{f_1(-k)u_1}\},\\ \{K_{3,i}, K_{4,i}\} &= \{g^{-\theta_2r_{2,i}}g^{f_2x_{i_j}w_{2,i}}, g^{\theta_1r_{2,i}}g^{f_2x_{i_j}w_{1,i}}\},\end{aligned}$$

$$\{K_3, K_4\} = \{g^{-\theta_2 r_2} g^{f_2(-k)w_2}, g^{\theta_1 r_2} g^{-f_2(-k)w_1}\},$$
$$K_A = g_2 \cdot K_1^{-t_1} K_2^{-t_2} K_3^{-z_1} K_4^{-z_2} \prod_{i=1}^{n} K_{1,i,j}^{-t_{1,i}} K_{2,i,j}^{-t_{2,i}} K_{3,i,j}^{-z_{1,i}} K_{4,i,j}^{-z_{2,i}},$$
$$K_B = g^{-(r_1+r_2)} \prod_{i=1}^{n} g^{-(r_{1,i}+r_{2,i})}.$$

The secret key is set as

$$SK = (K_A, K_B, K_1, K_2, K_3, K_4, \{\{K_{1,i,j}, K_{2,i,j}, K_{3,i,j}, K_{4,i,j}\}_{j=0}^{15}\}_{i=1}^{n}).$$

- **Decrypt**(*SK, CT*): Given a ciphertext *CT* associated with a string $v' = (v'_1, \ldots, v'_n)$ and a secret key *SK* associated with another string $x' = (x'_1, \ldots, x'_n)$, the decryption algorithm works as follows.

$$Z = e(C_A, K_A)^{16} \cdot e(C_B, K_B)^{16} \cdot \prod_{t=1}^{4} e(C_t, K_t)^{16} \prod_{i=1}^{n} \prod_{j=0}^{15} e(C_{t,i,j}, K_{t,i,j});$$
$$M = \frac{C_m}{Z}.$$

If two characters $v'_i = x'_i$, then the inner product $< \vec{v}_i, \vec{x}_i >= 0$. Otherwise, if two characters $v'_i \neq x'_i$, then $< \vec{v}_i, \vec{x}_i >= 1$. Therefore, the message M will be returned iff Hamming Distance $(v', x') = k$ meaning $\sum_{i=1}^{n}(\sum_{j=0}^{15} v_{i_j} x_{i_j}) - k = 0$.

5. Security proof of SAFE-1 and SAFE-2

Theorem 1. *Assume the Decision Bilinear Diffie-Hellman assumption and Decisional Linear Assumption hold in group* $\mathbb{G}$*, then our SAFE-1 scheme is secure.*

To prove the security of SAFE-1, we consider two cases $M_0 = M_1$ and $M_0 \neq M_1$.

In the case $M_0 = M_1$, we only consider the following game sequence from **Game**$_1$ to **Game**$_5$. In this case, we only prove the property of attribute hiding. For the other case $M_0 \neq M_1$, we need to consider the whole proof from **Game**$_0$ to **Game**$_6$. Below we first give a high level description of each game. In each game, we separate the vector used to generate ($C_{1,i}$, $C_{2,i}$, C_1, C_2) from the vector for ($C_{3,i}$, $C_{4,i}$, C_3, C_3). However, the same vector is used for both parts in **Game**$_0$ and **Game**$_6$.

Game$_0$: The challenge ciphertext $\mathbf{CT}_0$ is generated under $(\vec{v}, \vec{v})$ and M_0. The ciphertext $\mathbf{CT}_0$ is computed as follows:

$$(M_0 \cdot Y^{-s_2}, g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i\alpha}\}_{i=1}^{n}, \{U_1^{s_1} T_1^{s_2} V_1^{\alpha}, U_2^{s_1} T_2^{s_2} V_2^{\alpha}\}, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{v_i\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{v_i\beta}\}_{i=1}^{n}, \{W_1^{s_1} Z_1^{s_2} X_1^{\beta}, W_2^{s_1} Z_2^{s_2} X_2^{\beta}\})$$

Game$_1$: The challenge ciphertext $\mathbf{CT}_1$ is generated under $(\vec{v}, \vec{v})$ and a random message $R \in \mathbb{G}_T$. The ciphertext $\mathbf{CT}_1$ is computed as follows:

$$(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i\alpha}\}_{i=1}^{n}, \{U_1^{s_1} T_1^{s_2} V_1^{\alpha}, U_2^{s_1} T_2^{s_2} V_2^{\alpha}\}, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{v_i\beta} W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{v_i\beta}\}_{i=1}^{n}, \{W_1^{s_1} Z_1^{s_2} X_1^{\beta}, W_2^{s_1} Z_2^{s_2} X_2^{\beta}\})$$

Game$_2$: The challenge ciphertext $\mathbf{CT}_2$ is generated under $(\vec{v}, \vec{0})$ and a random message $R \in \mathbb{G}_T$. The ciphertext $\mathbf{CT}_2$ is computed as follows:

$$(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i\alpha}\}_{i=1}^{n}, \{U_1^{s_1} T_1^{s_2} V_1^{\alpha}, U_2^{s_1} T_2^{s_2} V_2^{\alpha}\}, \{W_{1,i}^{s_1} Z_{1,i}^{s_2}, W_{2,i}^{s_1} Z_{2,i}^{s_2}\}_{i=1}^{n}, \{W_1^{s_1} Z_1^{s_2}, W_2^{s_1} Z_2^{s_2}\})$$

Game$_3$: The challenge ciphertext $\mathbf{CT}_3$ is generated under $(\vec{v}, \vec{x})$ and a random message $R \in \mathbb{G}_T$. The ciphertext $\mathbf{CT}_3$ is computed as follows:

$$(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i\alpha}\}_{i=1}^{n}, \{U_1^{s_1} T_1^{s_2} V_1^{\alpha}, U_2^{s_1} T_2^{s_2} V_2^{\alpha}\}, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{x_i\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{x_i\beta}\}_{i=1}^{n}, \{W_1^{s_1} Z_1^{s_2} X_1^{\beta}, W_2^{s_1} Z_2^{s_2} X_2^{\beta}\})$$

Game$_4$: The challenge ciphertext $\mathbf{CT}_4$ is generated under $(\vec{0}, \vec{x})$ and a random message $R \in \mathbb{G}_T$. The ciphertext $\mathbf{CT}_4$ is computed as follows:

$$(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2}, U_{2,i}^{s_1} T_{2,i}^{s_2}\}_{i=1}^{n}, \{U_1^{s_1} T_1^{s_2}, U_2^{s_1} T_2^{s_2}\}, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{x_i\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{x_i\beta}\}_{i=1}^{n}, \{W_1^{s_1} Z_1^{s_2} X_1^{\beta}, W_2^{s_1} Z_2^{s_2} X_2^{\beta}\})$$

Game$_5$: The challenge ciphertext $\mathbf{CT}_5$ is generated under $(\vec{x}, \vec{x})$ and a random message $R \in \mathbb{G}_T$. The ciphertext $\mathbf{CT}_5$ is computed as follows:

$$(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{x_i\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{x_i\alpha}\}_{i=1}^{n}, \{U_1^{s_1} T_1^{s_2} V_1^{\alpha}, U_2^{s_1} T_2^{s_2} V_2^{\alpha}\}, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{x_i\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{x_i\beta}\}_{i=1}^{n}, \{W_1^{s_1} Z_1^{s_2} X_1^{\beta}, W_2^{s_1} Z_2^{s_2} X_2^{\beta}\})$$

Game$_6$: The challenge ciphertext $\mathbf{CT}_6$ is generated under $(\vec{x}, \vec{x})$ and message $M_1 \in \mathbb{G}_T$. The ciphertext $\mathbf{CT}_6$ is computed as follows:

$$(M_1 \cdot Y^{-s_2}, g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{x_i\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{x_i\alpha}\}_{i=1}^{n}, \{U_1^{s_1} T_1^{s_2} V_1^{\alpha}, U_2^{s_1} T_2^{s_2} V_2^{\alpha}\}, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{x_i\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{x_i\beta}\}_{i=1}^{n}, \{W_1^{s_1} Z_1^{s_2} X_1^{\beta}, W_2^{s_1} Z_2^{s_2} X_2^{\beta}\})$$

5.1. Indistinguishability between Game$_0$ and Game$_1$

Suppose that there exists an adversary $\mathcal{A}$ which can distinguish the two games with a non-negligible advantage ϵ, we construct another algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve the Decision Bilinear Diffie-Hellman problem also with advantage ϵ. On input $(g, A = g^a, B = g^b, C = g^c, Z) \in \mathbb{G}_4$, $\mathcal{B}$ simulates the game for $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ selects random elements γ_1, γ_2, θ_1, θ_2, λ, u_1, $\{u_{1,i}\}_{i=1}^{n}$, t_1, $\{t_{1,i}\}_{i=1}^{n}$, $\{t_{2,i}\}_{i=1}^{n}$, t_2, w_1, $\{w_{1,i}\}_{i=1}^{n}$, $z_1\{z_{1,i}\}_{i=1}^{n}$, $\{z_{2,i}\}_{i=1}^{n}$, z_2 in $\mathbb{Z}_p$. Then it selects a random $\Delta \in \mathbb{Z}_p$ to obtain $\{u_{2,i}\}_{i=1}^{n}$, $\{w_{2,i}\}_{i=1}^{n}$, w_2, u_2 under the condition:

$$\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}\Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}.$$
$$\Delta = \gamma_1 u_2 - \gamma_2 u_1 \Delta = \theta_1 w_2 - \theta_2 w_1.$$

Then for $i = 1$ to n, $\mathcal{B}$ sets:

$$U_{1,i} = g^{u_{1,i}}, U_1 = g^{u_1}, U_{2,i} = g^{u_{2,i}}, U_2 = g^{u_2},$$
$$T_{1,i} = (g^b)^{v_i\gamma_1} g^{t_{1,i}}, T_1 = (g^b)^{\gamma_1} g^{t_1},$$
$$T_{2,i} = (g^b)^{v_i\gamma_2} g^{t_{2,i}}, T_2 = (g^b)^{\gamma_2} g^{t_2},$$
$$W_{1,i} = g^{w_{1,i}}, W_1 = g^{w_1}, W_{2,i} = g^{w_{2,i}}, W_2 = g^{w_2},$$
$$Z_{1,i} = (g^b)^{v_i\theta_1} g^{z_{1,i}}, Z_1 = (g^b)^{\theta_1} g^{z_1}$$
$$Z_{2,i} = (g^b)^{v_i\theta_1} g^{z_{2,i}}, Z_2 (g^b)^{\theta_1} g^{z_2}.$$

and

$$V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_1}, X_2 = g^{\theta_2}$$
$$g_1 = g^{\Delta}, Y = e(g^a, g^b)^{-\Delta} \cdot e(g, g)^{\lambda}.$$

Each public key component is distributed properly following the random exponents:

$$\bar{t_{1,i}} = v_i\gamma_1 b + t_{1,i}, \bar{t_{2,i}} = v_i\gamma_2 b + t_{2,i},$$

$z_{1,i}^{-} = v_i\theta_1 b + z_{1,i}, z_{2,i}^{-} = v_i\theta_2 b + z_{2,i},$
$t_1^{-} = \gamma_1 b + t_1, t_2^{-} = \gamma_2 b + t_2, z_1^{-} = \theta_1 b + z_1, z_2^{-} = \theta_2 b + z_2,$
$g_2 = g^{-ab\Delta} g^{\lambda}.$

Key Generation Phase 1 & 2: $\mathcal{A}$ issues private key queries for vectors. Consider a query for a vector $\vec{y} = (y_1, \ldots, y_n)$. $\mathcal{A}$ can request the private key query as long as $\sum y_i(1-2v_i) + (\sum v_i - k) = c_x \neq 0$.

$\mathcal{B}$ picks random exponents $\{r_{1,i}\}_{i=1}^n, \{r_{2,i}\}_{i=1}^n$, and $f_1', f_2', r_1, r_2,$. Then $\mathcal{B}$ computes:

$$\begin{aligned}
K_{1,i} &= g^{-\gamma_2 r_{1,i}} g^{(\frac{a}{2c_x} + f_1')(1-2y_i)u_{2,i}} \\
&= g^{\frac{a}{2c_x}(1-2y_i)u_{2,i}} g^{-\gamma_2 r_{1,i}} g^{f_1'(1-2y_i)u_{2,i}} \\
&= g^{\frac{a}{2c_x}(1-2y_i)u_{2,i}} \cdot K_{1,i}'. \\
K_{2,i} &= g^{\gamma_1 r_{1,i}} g^{-(\frac{a}{2c_x} + f_1')(1-2y_i)u_{1,i}} \\
&= g^{-\frac{a}{2c_x}(1-2y_i)u_{1,i}} g^{\gamma_1 r_{1,i}} g^{-f_1'(1-2y_i)u_{1,i}} \\
&= g^{-\frac{a}{2c_x}(1-2y_i)u_{1,i}} \cdot K_{2,i}'. \\
K_1 &= g^{-\gamma_2 r_1} g^{(\frac{a}{2c_x} + f_1')(\sum y_i - k)u_2} \\
&= g^{\frac{a}{2c_x}(\sum y_i - k)u_2} g^{-\gamma_2 r_1} g^{f_1'(\sum y_i - k)u_2} \\
&= g^{\frac{a}{2c_x}(\sum y_i - k)u_2} \cdot K_1'. \\
K_2 &= g^{\gamma_1 r_1} g^{-(\frac{a}{2c_x} + f_1')(\sum y_i - k)u_1} \\
&= g^{-\frac{a}{2c_x}(\sum y_i - k)u_1} g^{\gamma_1 r_1} g^{-f_1'(\sum y_i - k)u_1} \\
&= g^{-\frac{a}{2c_x}(\sum y_i - k)u_1} \cdot K_2',
\end{aligned}$$

which implicitly sets: $f_1 = \frac{a}{2c_x} + f_1'$. Next $\mathcal{B}$ computes:

$$\begin{aligned}
K_{3,i} &= g^{-\theta_2 r_{2,i}} g^{(\frac{a}{2c_x} + f_2')(1-2y_i)w_{2,i}} \\
&= g^{\frac{a}{2c_x}(1-2y_i)w_{2,i}} g^{-\theta_2 r_{2,i}} g^{f_2'(1-2y_i)w_{2,i}} \\
&= g^{\frac{a}{2c_x}(1-2y_i)w_{2,i}} \cdot K_{3,i}'. \\
K_{4,i} &= g^{\theta_1 r_{2,i}} g^{-(\frac{a}{2c_x} + f_2')(1-2y_i)w_{1,i}} \\
&= g^{-\frac{a}{2c_x}(1-2y_i)w_{1,i}} g^{\theta_1 r_{1,i}} g^{-f_2'(1-2y_i)w_{1,i}} \\
&= g^{-\frac{a}{2c_x}(1-2y_i)w_{1,i}} \cdot K_{4,i}'. \\
K_3 &= g^{-\theta_2 r_2} g^{(\frac{a}{2c_x} + f_2')(\sum y_i - k)w_2} \\
&= g^{\frac{a}{2c_x}(\sum y_i - k)w_2} g^{-\theta_2 r_2} g^{f_2'(\sum y_i - k)w_2} \\
&= g^{\frac{a}{2c_x}(\sum y_i - k)w_2} \cdot K_3'. \\
K_4 &= g^{\theta_1 r_2} g^{-(\frac{a}{2c_x} + f_2')(\sum y_i - k)w_1} \\
&= g^{-\frac{a}{2c_x}(\sum y_i - k)w_1} g^{\gamma_1 r_2} g^{-f_2'(\sum y_i - k)u_2} \\
&= g^{-\frac{a}{2c_x}(\sum y_i - k)u_1} \cdot K_4',
\end{aligned}$$

which implicitly sets: $f_2 = \frac{a}{2c_x} + f_2'$. Then K_B and K_A are computed as:

$$K_B = g^{-(r_1+r_2)} \prod_{i=1}^{n} g^{-(r_{1,i}+r_{2,i})}$$

$$K_A = g_2 \prod_{i=1}^{n} K_{1,i}^{-\overline{t_{1,i}}} K_1^{-\overline{t_1}} K_{2,i}^{-\overline{t_{2,i}}} K_2^{-\overline{t_2}} K_{3,i}^{-\overline{z_{1,i}}} K_3^{-\overline{z_1}} K_{4,i}^{-\overline{z_{2,i}}} K_4^{-\overline{z_2}}.$$

For K_A, we can compute

$$\begin{aligned}
K_{1,i}^{-\overline{t_{1,i}}} K_{2,i}^{-\overline{t_{2,i}}} &= (g^{\frac{a}{2c_x}(1-2y_i)u_{2,i}} \cdot K_{1,i}')^{-\overline{t_{1,i}}} \\
&\quad \cdot (g^{-\frac{a}{2c_x}(1-2y_i)u_{1,i}} \cdot K_{2,i}')^{-\overline{t_{1,i}}} \\
&= (g^{\frac{a}{2c_x}(1-2y_i)u_{2,i}})^{-(v_i\gamma_1 b + t_{1,i})} \cdot (K_{1,i}')^{-\overline{t_{1,i}}} \cdot (K_{2,i}')^{-\overline{t_{2,i}}} \\
&\quad \cdot (g^{-\frac{a}{2c_x}(1-2y_i)u_{1,i}})^{-(v_i\gamma_2 b + t_{1,i})} \\
&= g^{-\frac{ab}{2c_x} v_i(1-2y_i)(\gamma_1 u_{2,i} - \gamma_2 u_{1,i})} \cdot (K_{1,i}')^{-\overline{t_{1,i}}} \cdot (K_{2,i}')^{-\overline{t_{2,i}}} \\
&\quad \cdot g^{\frac{a}{2c_x}(1-2y_i)(u_{1,i}t_{2,i} - u_{2,i}t_{1,i})} \\
&= g^{-\frac{ab\Delta}{2c_x} v_i(1-2y_i)} g^{\frac{a}{2c_x}(1-2y_i)(u_{1,i}t_{2,i} - u_{2,i}t_{1,i})} \\
&\quad \cdot (K_{1,i}')^{-\overline{t_{1,i}}} \cdot (K_{2,i}')^{-\overline{t_{2,i}}} \\
&\quad \cdot (K_1')^{-\overline{t_1}} \cdot (K_2')^{-\overline{t_2}}.
\end{aligned}$$

$$\begin{aligned}
K_1^{-\overline{t_1}} K_2^{-\overline{t_2}} &= g^{\frac{a}{2c_x}(1-2y_i)u_2} \cdot K_1')^{-\overline{t_1}} \cdot (g^{-\frac{a}{2c_x}(1-2y_i)u_1} \cdot K_2')^{-\overline{t_1}} \\
&= (g^{\frac{a}{2c_x}(\sum y_i - k)u_{2,i}})^{-(\gamma_1 b + t_1)} \cdot (K_1')^{-\overline{t_1}} \cdot (K_2')^{-\overline{t_2}} \\
&\quad (g^{-\frac{a}{2c_x}(\sum y_i - k)u_1})^{-(\gamma_2 b + t_1)} \\
&= g^{-\frac{ab}{2c_x}(\sum y_i - k)(\gamma_1 u_2 - \gamma_2 u_1)} \cdot (K_1')^{-\overline{t_1}} \cdot (K_2')^{-\overline{t_2}} \\
&\quad g^{\frac{a}{2c_x}(\sum y_i - k)(u_1 t_2 - u_2 t_1)} \\
&= g^{-\frac{ab\Delta}{2c_x}(\sum y_i - k)} g^{\frac{a}{2c_x}(\sum y_i - k)(u_1 t_2 - u_2 t_1)} \\
&\quad \cdot (K_1')^{-\overline{t_1}} \cdot (K_2')^{-\overline{t_2}} K_{1,i}^{-\overline{t_{1,i}}} K_{2,i}^{-\overline{t_{2,i}}} K_1^{-\overline{t_1}} K_2^{-\overline{t_2}} \\
&= g^{-\frac{ab\Delta}{2c_x}(v_i(1-2y_i) + \sum y_i - k)} g^{\frac{a}{2c_x}(1-2y_i)(u_{1,i}t_{2,i} - u_{2,i}t_{1,i})} \\
&\quad g^{\frac{a}{2c_x}(\sum y_i - k)(u_1 t_2 - u_2 t_1)} \cdot (K_{1,i}')^{-\overline{t_{1,i}}} \cdot (K_{2,i}')^{-\overline{t_{2,i}}} \\
&\quad \cdot (K_1')^{-\overline{t_1}} \cdot (K_2')^{-\overline{t_2}}.
\end{aligned}$$

Similarly, we can compute

$$\begin{aligned}
&K_{3,i}^{-\overline{z_{1,i}}} K_{4,i}^{-\overline{z_{2,i}}} K_3^{-\overline{z_1}} K_4^{-\overline{z_2}} \\
&= (K_{3,i}')^{-\overline{z_{1,i}}} \cdot (K_{4,i}')^{-\overline{z_{2,i}}} \cdot (K_3')^{-\overline{z_1}} \cdot (K_4')^{-\overline{z_2}} \cdot \\
&\quad g^{\frac{a}{2c_x}(1-2y_i)(w_{1,i}z_{2,i} - w_{2,i}z_{1,i})} \\
&\quad g^{\frac{a}{2c_x}(\sum y_i - k)(w_1 z_2 - w_2 z_1)} g^{-\frac{ab\Delta}{2c_x}(v_i(1-2y_i) + \sum y_i - k)}.
\end{aligned}$$

Since $g_2 = g^{ab\Delta} g^{\lambda}$ then K_A can be computed as:

$$\begin{aligned}
K_A = g^{\lambda} \prod_{i=1}^{n} & g^{\frac{a}{2c_x}(1-2y_i)(u_{1,i}t_{2,i} - u_{2,i}t_{1,i} + w_{1,i}z_{2,i} - w_{2,i}z_{1,i})} \\
& g^{\frac{a}{2c_x}(\sum y_i - k)(u_1 t_2 - u_2 t_1 + w_1 z_2 - w_2 z_1)} \cdot (K_{1,i}')^{-\overline{t_{1,i}}} \\
& \cdot (K_{2,i}')^{-\overline{t_{2,i}}} \cdot (K_1')^{-\overline{t_1}} \cdot (K_2')^{-\overline{t_2}} \cdot (K_3')^{-\overline{z_1}} \cdot (K_4')^{-\overline{z_2}} \\
& (K_{3,i}')^{-\overline{z_{1,i}}} \cdot (K_{4,i}')^{-\overline{z_{2,i}}}.
\end{aligned}$$

$\mathcal{B}$ gives $\mathcal{A}$ the private key: $SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^n, \{K_1, K_2, K_3, K_4\})$ for the queried vector $\vec{y}$.

Challenge Ciphertext: To generate a challenge ciphertext, $\mathcal{B}$ picks random $s_1', \alpha', \beta' \in \mathbb{Z}_p$. $\mathcal{B}$ implicitly sets:

$$s_1 = s_1', s_2 = c, \alpha = -bc + \alpha', \beta = -bc + \beta'.$$

Then $\mathcal{B}$ sets $A = g^c = g^{s_2}, B = g^{\Delta s_1} = g_1^{s_1}$. For i from 1 to n, $\mathcal{B}$ computes:

$$\begin{aligned}
C_{1,i} &= (g^{u_{1,i}})^{s_1} ((g^b)^{v_i\gamma_1} g^{t_{1,i}})^c g^{v_i\gamma_1(-bc+\alpha')} \\
&= U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i\alpha} \\
C_{2,i} &= (g^{u_{2,i}})^{s_1} ((g^b)^{v_i\gamma_2} g^{t_{2,i}})^c g^{v_i\gamma_2(-bc+\alpha')} \\
&= U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i\alpha} \\
C_1 &= (g^{u_1})^{s_1} ((g^b)^{\gamma_1} g^{t_1})^c g^{\gamma_1(-bc+\alpha')} = U_1^{s_1} T_1^{s_2} V_1^{\alpha} \\
C_2 &= (g^{u_2})^{s_1} ((g^b)^{\gamma_2} g^{t_2})^c g^{\gamma_2(-bc+\alpha')} = U_2^{s_1} T_2^{s_2} V_2^{\alpha} \\
C_{3,i} &= (g^{w_{1,i}})^{s_1} ((g^b)^{v_i\theta_1} g^{z_{1,i}})^c g^{v_i\theta_1(-bc+\beta')} \\
&= W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{v_i\beta} \\
C_{4,i} &= (g^{w_{2,i}})^{s_1} ((g^b)^{v_i\theta_2} g^{z_{2,i}})^c g^{v_i\theta_2(-bc+\beta')} \\
&= W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{v_i\beta} \\
C_3 &= (g^{w_1})^{s_1} ((g^b)^{\theta_1} g^{z_1})^c g^{\theta_1(-bc+\beta')} = W_1^{s_1} Z_1^{s_2} X_1^{\beta} \\
C_4 &= (g^{w_2})^{s_1} ((g^b)^{\theta_2} g^{z_2})^c g^{\theta_2(-bc+\beta')} = W_2^{s_1} Z_2^{s_2} X_2^{\beta}.
\end{aligned}$$

Next $\mathcal{B}$ computes $C_m = Z^{\Delta} \cdot e(g, g^c)^{\lambda} \cdot M_0$. If $Z = e(g, g)^{abc}$ the challenge ciphertext is distributed in $\mathbf{Game}_0$, otherwise if Z is randomly chosen in $\mathbb{G}_T$, then the challenge ciphertext is distributed in $\mathbf{Game}_1$. Hence, if $\mathcal{A}$ can distinguish these two games, $\mathcal{B}$ can solve the DBDH problem.

5.2. Indistinguishability between Game_1 and Game_2

Suppose that there exists an adversary $\mathcal{A}$ which can distinguish these two games with non-negligible advantage ϵ, we construct another algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve the Decision Linear problem with advantage ϵ. On input $(g, g^a, g^b, g^{ac}, g^d, Z) \in \mathbb{G}_6$, $\mathcal{B}$ simulates the game for $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ selects random elements $\gamma_1, \gamma_2, \theta_1, \theta_2, \lambda, u_1$, $\{u_{1,i}\}_{i=1}^n, t_1, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, t_2, w_1, \{w_{1,i}\}_{i=1}^n$, $z_1\{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n, z_2$ in $\mathbb{Z}_p$. Then it selects a random $\Delta \in \mathbb{Z}_p$ to obtain $\{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, u_2$ under the condition:

$$\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}, \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i},$$
$$\Delta = \gamma_1 u_2 - \gamma_2 u_1, \Delta = \theta_1 w_2 - \theta_2 w_1.$$

Then for $i = 1$ to n, $\mathcal{B}$ sets:

$$U_{1,i} = (g^a)^{u_{1,i}}, U_1 = (g^a)^{u_1}, U_{2,i} = (g^a)^{u_{2,i}},$$
$$T_{1,i} = g^{t_{1,i}}, T_1 = g^{t_1}, T_{2,i} = g^{t_{2,i}}, T_2 = g^{t_2}, U_2 = (g^a)^{u_2}$$
$$W_{1,i} = (g^a)^{w_{1,i}}(g^b)^{\theta_1 v_i}, W_1 = (g^a)^{w_1}(g^b)^{\theta_1},$$
$$W_{2,i} = (g^a)^{w_{2,i}}(g^b)^{\theta_2 v_i}, W_2 = (g^a)^{w_2}(g^b)^{\theta_2},$$
$$Z_{1,i} = g^{z_{1,i}}(g^b)^{\theta_1 v_i}, Z_1 = g^{z_1}(g^b)^{\theta_1}, Z_{2,i} = g^{z_{2,i}}(g^b)^{\theta_2 v_i},$$
$$Z_2 = g^{z_2}(g^b)^{\theta_2}.V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_1},$$
$$X_2 = g^{\theta_2}, g_1 = (g^a)^{\Delta}, g_2 = g^{\lambda}.$$

Each public key component is distributed properly following the random exponents:

$$\bar{u_{1,i}} = au_{1,i}, \bar{u_1} = au_1, \bar{u_{2,i}} = au_{2,i}, \bar{u_2} = au_2,$$
$$\bar{w_{1,i}} = aw_{1,i} + \theta_1 bv_i, \bar{w_1} = aw_1 + \theta_1 b,$$
$$\bar{w_{2,i}} = aw_{2,i} + \theta_2 bv_i, \bar{w_2} = aw_2 + \theta_2 b,$$
$$\bar{z_{1,i}} = v_i\theta_1 b + z_{1,i}, \bar{z_1} = \theta_1 b + z_1,$$
$$\bar{z_{2,i}} = v_i\theta_2 b + z_{2,i}, \bar{z_2} = \theta_2 b + z_2.$$

Key Generation Phase 1 & 2: $\mathcal{A}$ issues private key queries for vectors. Consider a query for a vector $\vec{y} = (y_1, \ldots, y_n)$. $\mathcal{B}$ picks random exponents $\{r'_{1,i}\}_{i=1}^n, \{r'_{2,i}\}_{i=1}^n$, and f_1, f_2, r'_1, r'_2,. Then $\mathcal{B}$ computes:

$$\begin{aligned}
K_{1,i} &= g^{-\gamma_2(-v_i(1-2y_i)b+r'_{1,i})}g^{f_1(1-2y_i)u_{2,i}} \\
&= g^{\gamma_2 v_i(1-2y_i)b}g^{-\gamma_2 r'_{1,i}}g^{f_1(1-2y_i)u_{2,i}} \\
&= g^{\gamma_2 v_i(1-2y_i)b} \cdot K'_{1,i}. \\
K_{2,i} &= g^{\gamma_1(-v_i(1-2y_i)b+r'_{1,i})}g^{-f_1(1-2y_i)u_{1,i}} \\
&= g^{-\gamma_1 v_i(1-2y_i)b}g^{\gamma_2 r'_{1,i}}g^{-f_1(1-2y_i)u_{1,i}} \\
&= g^{-\gamma_1 v_i(1-2y_i)b} \cdot K'_{2,i}. \\
K_1 &= g^{-\gamma_2(-(\sum y_i-k)b+r'_1)}g^{f_1(\sum y_i-k)u_2} \\
&= g^{\gamma_2(\sum y_i-k)b}g^{-\gamma_2 r'_1}g^{f_1(\sum y_i-k)u_2} \\
&= g^{\gamma_2(\sum y_i-k)b} \cdot K'_1. \\
K_2 &= g^{\gamma_1(-(\sum y_i-k))+r'_1)}g^{-f_1(\sum y_i-k)u_1} \\
&= g^{-\gamma_1(\sum y_i-k)b}g^{\gamma_2 r'_1}g^{-f_1(\sum y_i-k)u_1} \\
&= g^{-\gamma_1(\sum y_i-k)b} \cdot K'_2,
\end{aligned}$$

which implicitly sets:

$r_{1,i} = -(1-2y_i)v_i b + r'_{1,i}, r_1 = -(\sum y_i - k)b + r'_1$.

Next $\mathcal{B}$ computes:

$$\begin{aligned}
K_{3,i} &= g^{-\theta_2(v_i(1-2y_i)b+ar'_{2,i})}g^{f_2(1-2y_i)w_{2,i}} \\
&= g^{-\theta_2 v_i(1-2y_i)b}g^{-\gamma_2 r'_{2,i}a}g^{f_2(1-2y_i)u_{2,i}} \\
&= g^{-\theta_2 v_i(1-2y_i)b} \cdot K'_{3,i}. \\
K_{4,i} &= g^{\theta_1(v_i(1-2y_i)b+ar'_{2,i})}g^{-f_2(1-2y_i)w_{1,i}} \\
&= g^{\theta_1 v_i(1-2y_i)b}g^{\theta_2 ar'_{2,i}}g^{-f_2(1-2y_i)w_{1,i}} \\
&= g^{\theta_1 v_i(1-2y_i)b} \cdot K'_{4,i}.
\end{aligned}$$

$$\begin{aligned}
K_3 &= g^{-\theta_2((\sum y_i-k)b+ar'_2)}g^{f_2(\sum y_i-k)w_2} \\
&= g^{-\theta_2(\sum y_i-k)b}g^{-\theta_2 ar'_2}g^{f_2(\sum y_i-k)w_2} \\
&= g^{-\theta_2(\sum y_i-k)b} \cdot K'_3. \\
K_4 &= g^{\theta_1((\sum y_i-k))+ar'_2)}g^{-f_2(\sum y_i-k)w_1} \\
&= g^{\theta_1(\sum y_i-k)b}g^{\gamma_2 ar'_2}g^{-f_2(\sum y_i-k)w_1} \\
&= g^{\theta_1(\sum y_i-k)b} \cdot K'_4,
\end{aligned}$$

which implicitly sets:

$r_{2,i} = (1-2y_i)v_i b + ar'_{2,i}, r_2 = (\sum y_i - k)b + ar'_2$.

Then K_B and K_A are computed as :

$$\begin{aligned}
K_B &= g^{-(r_1+r_2)}\prod_{i=1}^n g^{-(r_{1,i}+r_{2,i})} \\
&= g^{-(-(\sum y_i-k)b+r'_1+(\sum y_i-k)b+ar'_2)} \\
&\quad \prod_{i=1}^n g^{-(-(1-2y_i)v_i b+r'_{1,i}+(1-2y_i)v_i b+ar'_{2,i})} \\
&= g^{-(r'_1+ar'_2)}\prod_{i=1}^n g^{-(r'_{1,i}+ar'_{2,i})}.
\end{aligned}$$

$$\begin{aligned}
K_A &= g_2\prod_{i=1}^n K_{1,i}^{-t_{1,i}}K_1^{-t_1}K_{2,i}^{t_{2,i}}K_2^{-t_2}K_{3,i}^{-\overline{z_{1,i}}}K_3^{-\overline{z_1}} \\
&\quad K_{4,i}^{-\overline{z_{2,i}}}K_4^{-\overline{z_2}}.
\end{aligned}$$

For K_A, we can compute

$$\begin{aligned}
K_{1,i}^{-t_{1,i}}K_{2,i}^{t_{2,i}}K_1^{-t_1}K_2^{t_2} &= g^{-\gamma_2 v_i(1-2y_i)bt_{1,i}}g^{\gamma_1 v_i(1-2y_i)bt_{2,i}} \\
&\quad \cdot g^{-\gamma_2(\sum y_i-k)bt_1}g^{\gamma_1(\sum y_i-k)bt_2} \\
&\quad \cdot (K'_{1,i})^{-t_{1,i}} \cdot (K'_{2,i})^{-t_{2,i}} \cdot (K'_1)^{-t_1} \cdot (K'_2)^{-t_2}. \\
&\quad K_{3,i}^{-\overline{z_{1,i}}}K_{4,i}^{-\overline{z_{2,i}}} \\
&= g^{-\theta_2(v_i(1-2y_i)b)(-z_{1,i}-\theta_1 bv_i)}g^{(-\theta_2 ar'_{2,i})(-z_{1,i}-\theta_1 bv_i)} \\
&\quad g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 bv_i)} \\
&\quad g^{(-f_2(1-2y_i)w_{1,i})(-z_{2,i}-\theta_2 bv_i)} \\
&\quad g^{\theta_1(v_i(1-2y_i)b)(-z_{2,i}-\theta_2 bv_i)}g^{(\theta_1 ar'_{2,i})(-z_{2,i}-\theta_2 bv_i)} \\
&= g^{-(v_i(1-2y_i)b+ar'_{2,i})\Delta}g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 bv_i)} \\
&\quad g^{(-f_2(1-2y_i)w_{1,i})(-z_{2,i}-\theta_2 bv_i)}. \\
K_3^{-\overline{z_1}}K_4^{-\overline{z_2}} &= g^{-\theta_2(\sum y_i-k)b(-z_1-\theta_1)}g^{(-\theta_2 ar'_2)(-z_1-\theta_1 b)} \\
&\quad g^{(f_2(\sum y_i-k)w_2)(-z_1-\theta_1 b)}g^{(-f_2(\sum y_i-k)w_1)(-z_2-\theta_2 b)} \\
&\quad g^{\theta_1((\sum y_i-k)b)(-z_2-\theta_2 b)}g^{(\theta_1 ar'_2)(-z_2-\theta_2 b)} \\
&= g^{-((\sum y_i-k)b+ar'_2)\Delta}g^{(f_2(\sum y_i-k)w_2)(-z_1-\theta_1 b)} \\
&\quad g^{(-f_2(\sum y_i-k)w_1)(-z_2-\theta_2 b)}.
\end{aligned}$$

Since $g_2 = g^{\lambda}$ then K_A is computed as :

$$\begin{aligned}
K_A &= g^{\lambda}\prod_{i=1}^n g^{-\gamma_2 v_i(1-2y_i)bt_{1,i}}g^{\gamma_1 v_i(1-2y_i)bt_{2,i}} \\
&\quad g^{-\gamma_2(\sum y_i-k)bt_1}g^{\gamma_1(\sum y_i-k)bt_2} \\
&\quad (K'_{1,i})^{-t1,i} \cdot (K'_{2,i})^{-t_{2,i}} \cdot (K'_1)^{-t_1} \cdot (K'_2)^{-t_2} \\
&\quad g^{-(v_i(1-2y_i)b+ar'_{2,i})\Delta}g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 bv_i)} \\
&\quad g^{(-f_2(1-2y_i)w_{1,i})(-z_{2,i}-\theta_2 bv_i)}g^{(-f_2(\sum y_i-k)w_1)(-z_2-\theta_2 b)} \\
&\quad g^{-((\sum y_i-k)b+ar'_2)\Delta}g^{(f_2(\sum y_i-k)w_2)(-z_1-\theta_1 b)}.
\end{aligned}$$

$\mathcal{B}$ gives $\mathcal{A}$ the private key $SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^n, \{K_1, K_2, K_3, K_4\})$ for the queried vector $\vec{y}$.

Challenge Ciphertext: To generate a challenge ciphertext, $\mathcal{B}$ picks random $s'_1, \alpha' \in \mathbb{Z}_p$. $\mathcal{B}$ implicitly sets:

$$s_1 = c, s_2 = d, \alpha = \alpha'$$

Then $\mathcal{B}$ sets: $A = g^d = g^{s_2}, B = (g^{ac})^{\Delta} = g_1^{s_1}$. For i from 1 to n, $\mathcal{B}$ computes:

$$C_{1,i} = (g^{au_{1,i}})^c (g^d)^{t_{1,i}} g^{v_i \gamma_1 (\alpha')} = U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i \alpha}$$
$$C_{2,i} = (g^{au_{2,i}})^c (g^d)^{t_{2,i}} g^{v_i \gamma_2 (\alpha')} = U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i \alpha}$$
$$C_1 = (g^{au_1})^c (g^d)^{t_1} g^{\gamma_1 (\alpha')} = U_1^{s_1} T_1^{s_2} V_1^{\alpha}$$
$$C_2 = (g^{au_2})^{s_1} (g^d)^{t_2} g^{\gamma_2 (\alpha')} = U_2^{s_1} T_2^{s_2} V_2^{\alpha}.$$

Next $\mathcal{B}$ computes for i from 1 to n:

$$C_{3,i} = (g^{aw_{1,i}})^c (g^d)^{z_{1,i}} Z^{\theta_1 v_i}, C_{4,i} = (g^{aw_{2,i}})^c (g^d)^{z_{2,i}} Z^{\theta_2 v_i}$$
$$C_3 = (g^{aw_1})^c (g^d)^{z_1} Z^{\theta_1}, C_{4i} = (g^{aw_2})^c (g^d)^{z_1} Z^{\theta_2}$$

If $Z = g^{b(c+d)g^r}$ for r chosen randomly in $\mathbb{Z}_p$, then $\mathcal{B}$ is simulating $\mathbf{Game}_1$ with $\beta = r$:

$$C_{3,i} = (g^{aw_{1,i}})^c (g^d)^{z_{1,i}} (g^{b(c+d)g^r})^{\theta_1 v_i} = W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{v_i \beta}$$
$$C_{4,i} = (g^{aw_{2,i}})^c (g^d)^{z_{2,i}} (g^{b(c+d)} g^r)^{\theta_2 v_i} = W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{v_i \beta}$$
$$C_3 = (g^{aw_1})^c (g^d)^{z_1} (g^{b(c+d)} g^r)^{\theta_1} = W_1^{s_1} Z_1^{s_2} X_1^{\beta}$$
$$C_4 = (g^{aw_2})^c (g^d)^{z_1} (g^{b(c+d)} g^r)^{\theta_2} = W_2^{s_1} Z_2^{s_2} X_1^{\beta}.$$

If $Z = g^{b(c+d)}$, then $\mathcal{B}$ is simulating $\mathbf{Game}_2$

$$C_{3,i} = (g^{aw_{1,i}})^c (g^d)^{z_{1,i}} (g^{b(c+d)})^{\theta_1 v_i} = W_{1,i}^{s_1} Z_{1,i}^{s_2}$$
$$C_{4,i} = (g^{aw_{2,i}})^c (g^d)^{z_{2,i}} (g^{b(c+d)})^{\theta_2 v_i} = W_{2,i}^{s_1} Z_{2,i}^{s_2}$$
$$C_3 = (g^{aw_1})^c (g^d)^{z_1} (g^{b(c+d)})^{\theta_1} = W_1^{s_1} Z_1^{s_2}$$
$$C_4 = (g^{aw_2})^c (g^d)^{z_1} (g^{b(c+d)})^{\theta_2} = W_2^{s_1} Z_2^{s_2}.$$

Therefore, if $\mathcal{A}$ can distinguish the two games, $\mathcal{B}$ can solve the DLIN problem.

5.3. *Indistinguishability of* Game_2 *and* Game_3

Suppose that there exists an adversary $\mathcal{A}$ which can distinguish these two games with a non-negligible advantage ϵ, we construct another algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve the Decision Linear problem with advantage ϵ. On input $(g, g^a, g^b, g^{ac}, g^d, Z) \in \mathbb{G}_6$, $\mathcal{B}$ simulates the game for $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ selects random elements γ_1, γ_2, θ_1, θ_2, λ, u_1, $\{u_{1,i}\}_{i=1}^n, t_1, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, t_2, w_1, \{w_{1,i}\}_{i=1}^n$, $z_1\{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n, z_2$ in $\mathbb{Z}_p$. Then it selects a random $\Delta \in \mathbb{Z}_p$ to obtain $\{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, u_2$ under the condition:

$$\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}, \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}.$$
$$\Delta = \gamma_1 u_2 - \gamma_2 u_1, \Delta = \theta_1 w_2 - \theta_2 w_1.$$

Then for $i = 1$ to n, $\mathcal{B}$ sets:

$$U_{1,i} = (g^a)^{u_{1,i}}, U_1 = (g^a)^{u_1}, U_{2,i} = (g^a)^{u_{2,i}}, U_2 = (g^a)^{u_2},$$
$$T_{1,i} = g^{t_{1,i}}, T_1 = g^{t_1}, T_{2,i} = g^{t_{2,i}}, T_2 = g^{t_2}$$
$$W_{1,i} = (g^a)^{w_{1,i}} (g^b)^{\theta_1 v_i}, W_1 = (g^a)^{w_1} (g^b)^{\theta_1},$$
$$W_{2,i} = (g^a)^{w_{2,i}} (g^b)^{\theta_2 v_i}, W_2 = (g^a)^{w_2} (g^b)^{\theta_2},$$
$$Z_{1,i} = g^{z_{1,i}} (g^b)^{\theta_1 v_i}, Z_1 = g^{z_1} (g^b)^{\theta_1},$$
$$Z_{2,i} = g^{z_{2,i}} (g^b)^{\theta_2 x_i}, Z_2 = g^{z_2} (g^b)^{\theta_2}.$$
$$V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_1}, X_2 = g^{\theta_2},$$
$$g_1 = (g^a)^{\Delta}, g_2 = g^{\lambda}.$$

Each public key component is distributed properly following the random exponents:

$$\bar{u_{1,i}} = au_{1,i}, \bar{u_1} = au_1, \bar{u_{2,i}} = au_{2,i} \bar{u_2} = au_2,$$
$$\bar{w_{1,i}} = aw_{1,i} + \theta_1 b v_i, \bar{w_1} = aw_1 + \theta_1 b,$$
$$\bar{w_{2,i}} = aw_{2,i} + \theta_2 b x_i, \bar{w_2} = aw_2 + \theta_2 b,$$
$$\bar{z_{1,i}} = v_i \theta_1 b + z_{1,i}, \bar{z_1} = \theta_1 b + z_1,$$
$$\bar{z_{2,i}} = v_i \theta_2 b + z_{2,i}, \bar{z_2} = \theta_2 b + z_2.$$

Key Generation Phase 1 & 2: $\mathcal{A}$ issues private key queries for vectors. Consider a query for a vector $\vec{y} = (y_1, \ldots, y_n)$. Notice that $\mathcal{A}$ obeys the restrictions defined in the model. That is $\sum v_i(1 - 2y_i) + \sum y_i - k = 0 \bmod p$ if and only if $\sum x_i(1 - 2y_i) + \sum y_i - k = 0 \bmod p$. There are two cases we need to consider.

- **Case 1:** $\sum v_i(1 - 2y_i) + \sum y_i - k = 0 = \sum x_i(1 - 2y_i) + \sum y_i - k = 0 \bmod p$. In this case, $\mathcal{B}$ picks random exponents $\{r'_{1,i}\}_{i=1}^n, \{r'_{2,i}\}_{i=1}^n$, and f_1, f_2, r'_1, r'_2,. Then $\mathcal{B}$ computes:

$$K_{1,i} = g^{-\gamma_2(-v_i(1-2y_i)b + r'_{1,i})} g^{f_1(1-2y_i)u_{2,i}}$$
$$= g^{\gamma_2 v_i (1-2y_i) b} g^{-\gamma_2 r'_{1,i}} g^{f_1(1-2y_i)u_{2,i}}$$
$$= g^{\gamma_2 v_i (1-2y_i) b} \cdot K'_{1,i}.$$
$$K_{2,i} = g^{\gamma_1(-v_i(1-2y_i)b + r'_{1,i})} g^{-f_1(1-2y_i)u_{1,i}}$$
$$= g^{-\gamma_1 v_i (1-2y_i) b} g^{\gamma_2 r'_{1,i}} g^{-f_1(1-2y_i)u_{1,i}}$$
$$= g^{-\gamma_1 v_i (1-2y_i) b} \cdot K'_{2,i}.$$
$$K_1 = g^{-\gamma_2(-(\sum y_i - k)b + r'_1)} g^{f_1(\sum y_i - k)u_2}$$
$$= g^{\gamma_2(\sum y_i - k)b} g^{-\gamma_2 r'_1} g^{f_1(\sum y_i - k)u_2}$$
$$= g^{\gamma_2(\sum y_i - k)b} \cdot K'_1.$$
$$K_2 = g^{\gamma_1(-(\sum y_i - k)) + r'_1)} g^{f_1(\sum y_i - k)u_1}$$
$$= g^{-\gamma_1(\sum y_i - k)b} g^{\gamma_2 r'_1} g^{f_1(\sum y_i - k)u_1}$$
$$= g^{-\gamma_1(\sum y_i - k)b} \cdot K'_2,$$

which implicitly sets:
$r_{1,i} = -(1 - 2y_i)v_i b + r'_{1,i}, r_1 = -(\sum y_i - k)b + r'_1$.
Next $\mathcal{B}$ computes:

$$K_{3,i} = g^{-\theta_2(x_i(1-2y_i)b + ar'_{2,i})} g^{f_2(1-2y_i)w_{2,i}}$$
$$= g^{-\theta_2 x_i (1-2y_i) b} g^{-\gamma_2 r'_{2,i} a} g^{f_2(1-2y_i)u_{2,i}}$$
$$= g^{-\theta_2 x_i (1-2y_i) b} \cdot K'_{3,i}.$$
$$K_{4,i} = g^{\theta_1(x_i(1-2y_i)b + ar'_{2,i})} g^{-f_2(1-2y_i)w_{1,i}}$$
$$= g^{\theta_1 x_i (1-2y_i) b} g^{\theta_2 a r'_{2,i}} g^{-f_2(1-2y_i)w_{1,i}}$$
$$= g^{\theta_1 x_i (1-2y_i) b} \cdot K'_{4,i}.$$
$$K_3 = g^{-\theta_2((\sum y_i - k)b + ar'_2)} g^{f_2(\sum y_i - k)w_2}$$
$$= g^{-\theta_2(\sum y_i - k)b} g^{-\theta_2 a r'_2} g^{f_2(\sum y_i - k)w_2}$$
$$= g^{-\theta_2(\sum y_i - k)b} \cdot K'_3.$$
$$K_4 = g^{\theta_1((\sum y_i - k)) + ar'_2)} g^{-f_2(\sum y_i - k)w_1}$$
$$= g^{\theta_1(\sum y_i - k)b} g^{\gamma_2 a r'_2} g^{-f_2(\sum y_i - k)w_1}$$
$$= g^{\theta_1(\sum y_i - k)b} \cdot K'_4,$$

which implicitly sets: $r_{2,i} = (1 - 2y_i)x_i b + ar'_{2,i}, r_2 = (\sum y_i - k)b + ar'_2$.
$\mathcal{B}$ also compute K_A and K_B as follows.

$$K_B = g^{-(r_1 + r_2)} \prod_{i=1}^{n} g^{-(r_{1,i} + r_{2,i})}$$
$$= g^{-(-(\sum y_i - k)b + r'_1 + (\sum y_i - k)b + ar'_2)}$$
$$\prod_{i=1}^{n} g^{-(-(1-2y_i)v_i b + r'_{1,i} + (1-2y_i)x_i b + ar'_{2,i})}$$
$$= g^{(\sum v_i(1-2y_i) + \sum y_i - k) - (\sum x_i(1-2y_i) + \sum y_i - k)}$$
$$\cdot g^{-(r'_1 + ar_2)} \cdot \prod_{i=1}^{n} g^{-(r'_{1,i} + ar'_{2,i})}$$
$$= g^{-(r'_1 + ar'_2)} \prod_{i=1}^{n} g^{-(r'_{1,i} + ar'_{2,i})}.$$

$$K_A = g_2 \prod_{i=1}^{n} K_{1,i}^{-t_{1,i}} K_1^{-t_1} K_{2,i}^{t_{2,i}} K_2^{-t_2} K_{3,i}^{-\overline{z_{1,i}}} K_3^{-\overline{z_1}} K_{4,i}^{-\overline{z_{2,i}}} K_4^{-\overline{z_2}}.$$

For K_A, its components are computed as follows:

$$\begin{aligned}
&K_{1,i}^{-t_{1,i}} K_{2,i}^{-t_{2,i}} K_1^{-t_1} K_2^{-t_2} \\
&= g^{-\gamma_2 v_i(1-2y_i)bt_{1,i}} g^{-\gamma_1 v_i(1-2y_i)bt_{2,i}} \\
&\quad \cdot g^{-\gamma_2(\sum y_i - k)bt_1} g^{-\gamma_1(\sum y_i - k)bt_2} \\
&\quad \cdot (K'_{1,i})^{-t_{1,i}} \cdot (K'_{2,i})^{-t_{2,i}} \cdot (K'_1)^{-t_1} \cdot (K'_2)^{-t_2}. \\
&K_{3,i}^{-\overline{z_{1,i}}} K_{4,i}^{-\overline{z_{2,i}}} \\
&= g^{-\theta_2(x_i(1-2y_i)b)(-z_{1,i}-\theta_1 b v_i)} \\
&\quad g^{(-\theta_2 ar'_{2,i})(-z_{1,i}-\theta_1 b v_i)} \\
&\quad g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 b v_i)} \\
&\quad g^{(-f_2(1-y_i)w_{1,i})(-z_{2,i}-\theta_2 b v_i)} \\
&\quad g^{\theta_1(x_i(1-2y_i)b)(-z_{2,i}-\theta_2 b v_i)} \\
&\quad g^{(\theta_1 ar'_{2,i})(-z_{2,i}-\theta_2 b v_i)} \\
&= g^{-(v_i(1-2y_i)b+ar'_{2,i})\Delta} g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 b v_i)} \\
&\quad g^{(-f_2(1-2y_i)w_{1,i})(-z_{2,i}-\theta_2 b v_i)} \\
&K_3^{-\overline{z_1}} K_4^{-\overline{z_2}} \\
&= g^{-\theta_2(\sum y_i - k)b(-z_1-\theta_1)} g^{(-\theta_2 ar'_2)(-z_1-\theta_1 b)} \\
&\quad g^{(f_2(\sum y_i - k)w_2)(-z_1-\theta_1 b)} g^{(-f_2(\sum y_i - k)w_1)(-z_2-\theta_2 b)} \\
&\quad g^{\theta_1((\sum y_i - k)b)(-z_2-\theta_2 b)} g^{(\theta_1 ar'_2)(-z_2-\theta_2 b)} \\
&= g^{-((\sum y_i - k)b+ar'_2)\Delta} g^{(f_2(\sum y_i - k)w_2)(-z_1-\theta_1 b)} \\
&\quad g^{(-f_2(\sum y_i - k)w_1)(-z_2-\theta_2 b)}.
\end{aligned}$$

Since $g_2 = g^{\lambda}$ then K_A can be computed as:

$$\begin{aligned}
K_A = g^{\lambda} &\prod_{i=1}^{n} g^{-\gamma_2 v_i(1-2y_i)bt_{1,i}} g^{\gamma_1 x_i(1-2y_i)bt_{2,i}} \\
&\cdot g^{-\gamma_2(\sum y_i - k)bt_1} g^{\gamma_1(\sum y_i - k)bt_2} \\
&\cdot (K'_{1,i})^{-t1,i} \cdot (K'_{2,i})^{-t_{2,i}} \cdot (K'_1)^{-t_1} \cdot (K'_2)^{-t_2} \\
&\cdot g^{-(x_i(1-2y_i)b+ar'_{2,i})\Delta} g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 b v_i)} \\
&\cdot g^{(-f_2(1-2y_i)w_{1,i})(-z_{2,i}-\theta_2 b v_i)} \\
&\cdot g^{-((\sum y_i - k)b+ar'_2)\Delta} g^{(f_2(\sum y_i - k)w_2)(-z_1-\theta_1 b)} \\
&\cdot g^{(-f_2(\sum y_i - k)w_1)(-z_2-\theta_2 b)}.
\end{aligned}$$

$\mathcal{B}$ gives $\mathcal{A}$ the private key $SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^n, \{K_1, K_2, K_3, K_4\})$ for the queried vector $\vec{y}$.

- **Case 2:** $\sum v_i(1-2y_i) + \sum y_i - k = c_v \neq 0$ and $\sum x_i(1-2y_i) + \sum y_i - k = c_x \neq 0$. In this case, $\mathcal{B}$ picks random exponents $\{r'_{1,i}\}_{i=1}^n$, $\{r'_{2,i}\}_{i=1}^n$, and f_1, f_2, r'_1, r'_2,. Then $\mathcal{B}$ computes:

$$\begin{aligned}
K_{1,i} &= g^{-\gamma_2(-c_x v_i(1-2y_i)b+r'_{1,i})} g^{f_1(1-2y_i)u_{2,i}} \\
&= g^{\gamma_2 c_x v_i(1-2y_i)b} g^{-\gamma_2 r'_{1,i}} g^{f_1(1-2y_i)u_{2,i}} \\
&= g^{\gamma_2 c_x v_i(1-2y_i)b} \cdot K'_{1,i}. \\
K_{2,i} &= g^{\gamma_1(c_x - v_i(1-2y_i)b+r'_{1,i})} g^{-f_1(1-2y_i)u_{1,i}} \\
&= g^{-\gamma_1 c_x v_i(1-2y_i)b} g^{\gamma_2 r'_{1,i}} g^{-f_1(1-2y_i)u_{1,i}} \\
&= g^{-\gamma_1 c_x v_i(1-2y_i)b} \cdot K'_{2,i}. \\
K_1 &= g^{-\gamma_2 c_x(-(\sum y_i - k)b+r'_1)} g^{f_1(\sum y_i - k)u_2} \\
&= g^{\gamma_2 c_x(\sum y_i - k)b} g^{-\gamma_2 r'_1} g^{f_1(\sum y_i - k)u_2} \\
&= g^{\gamma_2 c_x(\sum y_i - k)b} \cdot K'_1. \\
K_2 &= g^{\gamma_1 c_x(-(\sum y_i - k))+r'_1)} g^{-f_1(\sum y_i - k)u_1} \\
&= g^{-\gamma_1 c_x(\sum y_i - k)b} g^{\gamma_2 r'_1} g^{-f_1(\sum y_i - k)u_1} \\
&= g^{-\gamma_1 c_x(\sum y_i - k)b} \cdot K'_2,
\end{aligned}$$

which implicitly sets:

$$r_{1,i} = -c_x(1-2y_i)v_i b + r'_{1,i}, r_1 = -c_x(\sum y_i - k)b + r'_1.$$

Next $\mathcal{B}$ computes:

$$\begin{aligned}
K_{3,i} &= g^{-\theta_2 c_v(x_i(1-2y_i)b+ar'_{2,i})} g^{f_2(1-2y_i)w_{2,i}} \\
&= g^{-\theta_2 c_v x_i(1-2y_i)b} g^{-\gamma_2 r'_{2,i} a} g^{f_2(1-2y_i)u_{2,i}} \\
&= g^{-\theta_2 c_v x_i(1-2y_i)b} \cdot K'_{3,i}. \\
K_{4,i} &= g^{\theta_1 c_v(x_i(1-2y_i)b+ar'_{2,i})} g^{-f_2(1-2y_i)w_{1,i}} \\
&= g^{\theta_1 c_v x_i(1-2y_i)b} g^{\theta_2 ar'_{2,i}} g^{f_2(1-2y_i)w_{1,i}} \\
&= g^{\theta_1 c_v x_i(1-2y_i)b} \cdot K'_{4,i}. \\
K_3 &= g^{-\theta_2(c_v(\sum y_i - k)b+ar'_2)} g^{f_2(\sum y_i - k)w_2} \\
&= g^{-\theta_2 c_v(\sum y_i - k)b} g^{-\theta_2 ar'_2} g^{f_2(\sum y_i - k)w_2} \\
&= g^{-\theta_2 c_v(\sum y_i - k)b} \cdot K'_3. \\
K_4 &= g^{\theta_1 c_v((\sum y_i - k))+ar'_2)} g^{-f_2(\sum y_i - k)w_1} \\
&= g^{\theta_1 c_v(\sum y_i - k)b} g^{\gamma_2 ar'_2} g^{-f_2(\sum y_i - k)w_1} \\
&= g^{\theta_1 c_v(\sum y_i - k)b} \cdot K'_4,
\end{aligned}$$

which implicitly sets: $r_{2,i} = c_v(1-2y_i)x_i b + ar'_{2,i}$, $r_2 = c_v(\sum y_i - k)b + ar'_2$. Then K_B and K_A are computed as follows:

$$\begin{aligned}
K_B &= g^{-(r_1+r_2)} \prod_{i=1}^{n} g^{-(r_{1,i}+r_{2,i})} \\
&= g^{-(-c_x(\sum y_i - k)b+r'_1+c_v(\sum y_i - k)b+ar'_2)} \\
&\quad \prod_{i=1}^{n} g^{-(-c_x(1-2y_i)v_i b+r'_{1,i}+c_v(1-2y_i)x_i b+ar'_{2,i})} \\
&= g^{(c_x(\sum v_i(1-2y_i)+\sum y_i - k)-c_v(\sum x_i(1-2y_i)+\sum y_i - k)} \\
&\quad g^{-(r'_1+ar'_2)} \prod_{i=1}^{n} g^{-(r'_{1,i}+ar'_{2,i})} \\
&= g^{-(r'_1+ar'_2)} \prod_{i=1}^{n} g^{-(r'_{1,i}+ar'_{2,i})}.
\end{aligned}$$

$$\begin{aligned}
K_A = g_2 &\prod_{i=1}^{n} K_{1,i}^{-t_{1,i}} K_1^{-t_1} K_{2,i}^{-t_{2,i}} K_2^{-t_2} \\
&K_{3,i}^{-\overline{z_{1,i}}} K_3^{-\overline{z_1}} K_{4,i}^{-\overline{z_{2,i}}} K_4^{-\overline{z_2}}.
\end{aligned}$$

For K_A, the components are computed as follows:

$$\begin{aligned}
&K_{1,i}^{-t_{1,i}} K_{2,i}^{-t_{2,i}} K_1^{-t_1} K_2^{-t_2} \\
&= g^{-\gamma_2 c_x v_i(1-2y_i)bt_{1,i}} g^{-\gamma_1 c_x v_i(1-2y_i)bt_{2,i}} \\
&\quad \cdot g^{-\gamma_2 c_x(\sum y_i - k)bt_1} g^{-\gamma_1 c_x(\sum y_i - k)bt_2} \\
&\quad \cdot (K'_{1,i})^{-t_{1,i}} \cdot (K'_{2,i})^{-t_{2,i}} \cdot (K'_1)^{-t_1} \cdot (K'_2)^{-t_2}. \\
&K_{3,i}^{-\overline{z_{1,i}}} K_{4,i}^{-\overline{z_{2,i}}} \\
&= g^{-\theta_2 c_v(x_i(1-2y_i)b)(-z_{1,i}-\theta_1 b x_i)} \\
&\quad g^{(-\theta_2 ar'_{2,i})(-z_{1,i}-\theta_1 b v_i)} \\
&\quad g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 b x_i)} \\
&\quad g^{\theta_1 c_v(x_i(1-2y_i)b)(-z_{2,i}-\theta_2 b x_i)} \\
&\quad g^{(\theta_1 ar'_{2,i})(-z_{2,i}-\theta_2 b v_i)} \\
&\quad g^{(-f_2(1-2y_i)w_{1,i})(-z_{2,i}-\theta_2 b x_i)} \\
&= g^{-c_v(x_i(1-2y_i)b+ar'_{2,i})\Delta} \\
&\quad g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 b x_i)} \\
&\quad g^{(-f_2(1-2y_i)w_{1,i})(-z_{2,i}-\theta_2 b x_i)} \\
&\quad K_3^{-\overline{z_1}} K_4^{-\overline{z_2}}
\end{aligned}$$

$$= g^{-\theta_2 c_v(\sum y_i - k)b(-z_1-\theta_1)} g^{(-\theta_2 a r'_2)(-z_1-\theta_1 b)}$$
$$g^{(f_2(\sum y_i-k)w_2)(-z_1-\theta_1 b)}$$
$$g^{\theta_1 c_v((\sum y_i-k)b)(-z_2-\theta_2 b)} g^{(\theta_1 a r'_2)(-z_2-\theta_2 b)}$$
$$g^{(-f_2(\sum y_i-k)w_1)(-z_2-\theta_2 b)}$$
$$= g^{-(c_v(\sum y_i-k)b+ar'_2)\Delta} g^{(f_2(\sum y_i-k)w_2)(-z_1-\theta_1 b)}$$
$$g^{(-f_2(\sum y_i-k)w_1)(-z_2-\theta_2 b)}.$$

Since $g_2 = g^{\lambda}$ then K_A is computed as:

$$K_A = g^{\lambda} \prod_{i=1}^{n} g^{-\gamma_2 c_x v_i(1-2y_i)bt_{1,i}} g^{\gamma_1 c_x v_i(1-2y_i)bt_{2,i}}$$
$$\cdot g^{-\gamma_2 c_x(\sum y_i-k)bt_1} g^{\gamma_1 c_x(\sum y_i-k)bt_2}$$
$$\cdot (K'_{1,i})^{-t1,i} \cdot (K'_{2,i})^{-t_{2,i}} \cdot (K'_1)^{-t_1} \cdot (K'_2)^{-t_2}$$
$$\cdot g^{-(c_v x_i(1-2y_i)b+ar'_{2,i})\Delta}$$
$$g^{(f_2(1-2y_i)w_{2,i})(-z_{1,i}-\theta_1 b x_i)}$$
$$\cdot g^{(-f_2(1-2y_i)w_{1,i})(-z_{2,i}-\theta_2 b x_i)}$$
$$\cdot g^{-(c_v(\sum y_i-k)b+ar'_2)\Delta}$$
$$g^{(f_2(\sum y_i-k)w_2)(-z_1-\theta_1 b)}$$
$$\cdot g^{(-f_2(\sum y_i-k)w_1)(-z_2-\theta_2 b)}.$$

$\mathcal{B}$ gives $\mathcal{A}$ the private key $SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^n, \{K_1, K_2, K_3, K_4\})$ for the queried vector $\vec{y}$.

Challenge Ciphertext: To generate a challenge ciphertext, $\mathcal{B}$ picks random $s'_1, \alpha' \in \mathbb{Z}_p$. $\mathcal{B}$ implicitly sets:

$$s_1 = c, s_2 = d, \alpha = \alpha'$$

Then $\mathcal{B}$ sets: $A = g^d = g^{s_2}, B = (g^{ac})^{\Delta} = g_1^{s_1}$. For i from 1 to n, $\mathcal{B}$ computes:

$$C_{1,i} = (g^{au_{1,i}})^c (g^d)^{t_{1,i}} g^{v_i \gamma_1(\alpha')} = U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_i \alpha}$$
$$C_{2,i} = (g^{au_{2,i}})^c (g^d)^{t_{2,i}} g^{v_i \gamma_2(\alpha')} = U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i \alpha}$$
$$C_1 = (g^{au_1})^c (g^d)^{t_1} g^{\gamma_1(\alpha')} = U_1^{s_1} T_1^{s_2} V_1^{\alpha}$$
$$C_2 = (g^{au_2})^{s_1} (g^d)^{t_2} g^{\gamma_2(\alpha')} = U_2^{s_1} T_2^{s_2} V_2^{\alpha}.$$

Next $\mathcal{B}$ computes for i from 1 to n:

$$C_{3,i} = (g^{aw_{1,i}})^c (g^d)^{z_{1,i}} Z^{\theta_1 x_i}$$
$$C_{4,i} = (g^{aw_{2,i}})^c (g^d)^{z_{2,i}} Z^{\theta_2 x_i}$$
$$C_3 = (g^{aw_1})^c (g^d)^{z_1} Z^{\theta_1}$$
$$C_4 = (g^{aw_2})^c (g^d)^{z_1} Z^{\theta_2}.$$

If $Z = g^{b(c+d)}$ then, $\mathcal{B}$ is playing **Game**$_2$ with $\mathcal{A}$

$$C_{3,i} = (g^{aw_{1,i}})^c (g^d)^{z_{1,i}} (g^{b(c+d)g^r})^{\theta_1 v_i}$$
$$= W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{x_i \beta}$$
$$C_{4,i} = (g^{aw_{2,i}})^c (g^d)^{z_{2,i}} (g^{b(c+d)} g^r)^{\theta_2 v_i}$$
$$= W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{x_i \beta}$$
$$C_3 = (g^{aw_1})^c (g^d)^{z_1} (g^{b(c+d)} g^r)^{\theta_1}$$
$$= W_1^{s_1} Z_1^{s_2} X_1^{\beta}$$
$$C_4 = (g^{aw_2})^c (g^d)^{z_1} (g^{b(c+d)} g^r)^{\theta_2}$$
$$= W_2^{s_1} Z_2^{s_2} X_1^{\beta}.$$

Otherwise, if $Z = g^{b(c+d)g^r}$ for r chosen randomly in $\mathbb{Z}_p$, then $\mathcal{B}$ is playing **Game**$_3$ with $\mathcal{A}$ by setting $\beta = r$

$$C_{3,i} = (g^{aw_{1,i}})^c (g^d)^{z_{1,i}} (g^{b(c+d)})^{\theta_1 v_i} = W_{1,i}^{s_1} Z_{1,i}^{s_2}$$
$$C_{4,i} = (g^{aw_{2,i}})^c (g^d)^{z_{2,i}} (g^{b(c+d)})^{\theta_2 v_i} = W_{2,i}^{s_1} Z_{2,i}^{s_2}$$
$$C_3 = (g^{aw_1})^c (g^d)^{z_1} (g^{b(c+d)})^{\theta_1} = W_1^{s_1} Z_1^{s_2}$$
$$C_4 = (g^{aw_2})^c (g^d)^{z_1} (g^{b(c+d)})^{\theta_2} = W_2^{s_1} Z_2^{s_2}$$

Therefore, if $\mathcal{A}$ can distinguish *Game*$_2$ from *Game*$_3$, then $\mathcal{B}$ can solve the DLIN problem.

The rest of the proof is similar to the above proofs:

- the indistinguishability between *Game*$_3$ and *Game*$_4$ can be proved in the same way as for *Game*$_2$ and *Game*$_3$;
- the indistinguishability between *Game*$_4$ and *Game*$_5$ can be proved in the same way as for *Game*$_1$ and *Game*$_2$;
- the indistinguishability of *Game*$_5$ and *Game*$_6$ can be proved in the same way as for *Game*$_0$ and *Game*$_1$.

Theorem 2. *Assume the Decision Bilinear Diffie-Hellman assumption and Decisional Linear Assumption hold in group* $\mathbb{G}$*, then our SAFE-2 scheme is secure.*

We can obtain the security proof by modifying the proof of Theorem 1. When the adversary $\mathcal{A}$ announces two character strings in Σ^n as the challenge, they will be converted to two vectors as in (*i*) and (*ii*): $\vec{v} = (v_{1_0}, \ldots, v_{1_{15}}, v_{2_0}, \ldots, v_{2_{15}}, \ldots, v_{n_0}, \ldots, v_{n_{15}})$;, similarly (*ii*), $\vec{x} = (x_{1_0}, \ldots, x_{1_{15}}, x_{2_0}, \ldots, x_{2_{15}}, \ldots, x_{n_0}, \ldots, x_{n_{15}})$. Such a convertion will also be done in answering the key generation queries. The rest of the proof is similar to that of Theorem 1 and hence is omitted.

6. Sequence aware keyword search scheme

In this section, we propose a Sequence Aware Keyword Search (SAKS) Scheme which is an application of our SAFE-2 scheme. Using our SAKS scheme, an encrypter can create a ciphertext *CT* corresponding to a user public key and a DNA string such as AGTAC; the owner of the public key can use the corresponding secret key to create a trapdoor *T* for another string such as AGTAT and a distance *d*. A test function on *CT* and *T* will return true if and only if the hamming distance of the two strings is equal to *d*.

A SAKS scheme consists of the following four probabilistic polynomial-time algorithms:

- **Key Gen**$(1^n, \Sigma)$: on input a security parameter 1^n, an alphabet Σ, the algorithm outputs a public key *PK* and the corresponding secret key *MSK*.
- **PEKS**$(PK, W = (w_1, w_2, \ldots, w_n) \in \Sigma^n)$: on input a public key *PK*, a keyword $W = (w_1, w_2, \ldots, w_n)$, the algorithm outputs a searchable encryption *CT*.
- **Trapdoor**$(MSK, \bar{w} = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_n) \in \Sigma^n, k)$: on input a secret key *MSK* and a keyword w of length n, and a distance k the algorithm outputs a trapdoor $T_{\bar{w}}$.
- **Test**$(CT, T_{\bar{w}})$: on input a ciphertext *CT* with keywords $W = (w_1, w_2, \ldots, w_n)$ and a trapdoor $T_{\bar{w}}$ with keyword $w = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w})$, the algorithm outputs 1 if $HammingDistance(W, \bar{w}) = k$, or 0 otherwise.

SAKS Security Model The security model for a SAKS scheme is defined via the following game between an adversary *A* and a challenger *B*.

- **Init**: *A* submits two strings W_0^*, W_1^* of equal length and a distance value τ.
- **Setup**: The challenger *B* runs **Setup**$(1^n, \Sigma)$ to generate *PK* and *MSK*. *PK* is then passed to *A*.
- **Query Phase 1**: The challenger answers trapdoor queries for any keyword $\bar{w}$ such that $HammingDistance(W_0^*, \bar{w}) \neq \tau$ and $HammingDistance(W_1^*, \bar{w}) \neq \tau$ by returning $T_{\bar{w}} \leftarrow Trapdoor(MSK, \bar{w}, \tau)$.
- **Challenge**: The challenger flips a coin $\mu \leftarrow \{0, 1\}$. Then the challenger computes the challenge ciphertext $C^* \leftarrow Encrypt(PK, W_{\beta}^*)$ which is given to *A*.

- **Query Phase 2**: same as Query Phase 1.
- **Output**: A outputs a bit μ' as her guess for μ.
Define the advantage of A as $\mathbf{Adv}_A^{\mathsf{SAKS}}(k) = |\Pr_A^{\mathsf{SAKS}}(k)[\mu' = \mu] - 1/2|$.

6.1. SAKS Construction

- **Setup**$(1^k, n)$: The setup algorithm first randomly generates $(g, \mathbb{G}, \mathbb{G}_T, p, e)$. It then chooses randomly $\gamma_1, \gamma_2, \theta_1, \theta_2, u_1, \{u_{1,i}\}_{i=1}^n, t_1, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, t_2, w_1, \{w_{1,i}\}_{i=1}^n$, $z_1\{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n, z_2$ in $\mathbb{Z}_p$. Then it selects a random $\Delta \in \mathbb{Z}_p$ and obtains $\{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, u_2$ under the condition:

$$\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}, \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i},$$
$$\Delta = \gamma_1 u_2 - \gamma_2 u_1, \Delta = \theta_1 w_2 - \theta_2 w_1.$$

For i from 1 to n, it creates:

$$U_{1,i} = g^{u_{1,i}}, U_{2,i} = g^{u_{2,i}}, U_1 = g^{u_1}, U_2 = g^{u_2},$$
$$W_{1,i} = g^{w_{1,i}}, W_{2,i} = g^{w_{2,i}}, W_1 = g^{w_1}, W_2 = g^{w_2},$$
$$T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}}, T_1 = g^{t_1}, T_2 = g^{t_2},$$
$$Z_{1,i} = g^{z_{1,i}}, Z_{2,i} = g^{z_{2,i}}, Z_1 = g^{z_1}, Z_2 = g^{z_2},$$
$$V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}.$$

Next it sets $g_1 = g^{\Delta}$, and the public key PK and master key MSK as

$$PK = (g, \mathbb{G}, \mathbb{G}_T, p, e, g_1, \{U_{1,i}, U_{2,i}, T_{1,i}, T_{2,i}, W_{1,i}, W_{2,i}, Z_{1,i}, Z_{2,i}\}^n{}_{i=1}, \{U_i, T_i, V_i, X_i\}_{i=1}{}^2)$$
$$MSK = (\{u_{1,i}, u_{2,i}, t_{1,i}, t_{2,i}, w_{1,i}, w_{2,i}, z_{1,i}, z_{2,i}\}^n{}_{i=1}, \{u_i, t_i, w_i, z_i, \gamma_i, \theta_i\}^2{}_{i=1}).$$

- **PEKS**$(PK, W = (w_1, w_2, \ldots, w_n) \in \Sigma^n)$: The encryption algorithm chooses random $s_1, s_2, \alpha, \beta \in \mathbb{Z}_p$. Each characters w_i in W is converted to (a_1, a_2). Then algorithm creates vector $\vec{v}$ as in (i):

$$\vec{v} = (v_{1_0}, \ldots, v_{1_{15}}, v_{2_0}, \ldots, v_{2_{15}}, \ldots, v_{n_0}, \ldots, v_{n_{15}})$$

Next, the ciphertext is created as follows

$$C_A = g^{s_2}, \quad C_B = g_1^{s_1},$$
$$\{C_{1,i,j}, C_{2,i,j}\} = \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{v_{i_j}\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{v_i\alpha}\},$$
$$\{C_1, C_2\} = \{U_1^{s_1} T_1^{s_2} V_1^{\alpha}, U_2^{s_1} T_2^{s_2} V_2^{\alpha}\},$$
$$\{C_{3,i,j}, C_{4,i,j}\} = \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{v_{i_j}\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{v_{i_j}\beta}\},$$
$$\{C_3, C_4\} = \{W_1^{s_1} Z_1^{s_2} X_1^{\beta}, W_2^{s_1} Z_2^{s_2} X_2^{\beta}\}.$$

Then ciphertext CT is set as

$$CT_W = (C_A, C_B, C_1, C_2, C_3, C_4, \{\{C_{1,i,j}, C_{2,i,j}, C_{3,i,j}, C_{4,i,j}\}_{j=0}^{15}\}_{i=1}^n).$$

- **Trapdoor**$(MSK, \bar{w} = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_n) \in \Sigma^n, k)$: Choose random $r_{i,1}, r_{i,2}$ for $i = 1$ to n, and $f_1, f_2, r_1, r_2 \in \mathbb{Z}_p$. Each characters $\bar{w}_i$ in $\bar{w}$ is converted to (b_1, b_2). Then create vector $\vec{x}$ as in (ii):

$$\vec{x} = (x_{1_0}, \ldots, x_{1_{15}}, x_{2_0}, \ldots, x_{2_{15}}, \ldots, x_{n_0}, \ldots, x_{n_{15}}).$$

Then compute

$$\{K_{1,i,j}, K_{2,i,j}\} = \{g^{-\gamma_2 r_{1,i}} g^{f_1 x_{i_j}} u_{2,i}, g^{\gamma_1 r_{1,i}} g^{f_1 x_{i,j} u_{1,i}}\},$$
$$\{K_1, K_2\} = \{g^{-\gamma_2 r_1} g^{f_1(-k)u_2}, g^{\gamma_1 r_1} g^{f_1(-k)u_1}\},$$
$$\{K_{3,i}, K_{4,i}\} = \{g^{-\theta_2 r_{2,i}} g^{f_2 x_{i_j} w_{2,i}}, g^{\theta_1 r_{2,i}} g^{f_2 x_{i_j} w_{1,i}}\},$$
$$\{K_3, K_4\} = \{g^{-\theta_2 r_2} g^{f_2(-k)w_2}, g^{\theta_1 r_2} g^{-f_2(-k)w_1}\},$$
$$K_A = g_2 \cdot K_1^{-t_1} K_2^{-t_2} K_3^{-z_1} K_4^{-z_2} \prod_{i=1}^n K_{1,i,j}^{-t_{1,i}} K_{2,i,j}^{-t_{2,i}} K_{3,i,j}^{-z_{1,i}} K_{4,i,j}^{-z_{2,i}},$$
$$K_B = g^{-(r_1+r_2)} \prod_{i=1}^n g^{-(r_{1,i}+r_{2,i})}.$$

The trapdoor is set as

$$T_{\bar{w}} = (K_A, K_B, \{\{K_{1,i,j}, K_{2,i,j}, K_{3,i,j}, K_{4,i,j}\}_{j=0}{}^{15}\}_{i=1}{}^n, \{K_1, K_2, K_3, K_4\}).$$

- **Test**(Tw, CT): Given a ciphertext CT associated with a alphabet string $W = (w_1, w_2, \ldots, w_n)$ and trapdoor $T_{\bar{w}}$ associated with another alphabet string $\bar{w} = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_n)$, the algorithm tests

$$e(C_A, K_A)^{16} \cdot e(C_B, K_B)^{16} \cdot \prod_{t=1}^4 \prod_{i=1}^n \prod_{j=0}^{15} e(C_t, K_t)^{16} \cdot e(C_{t,i,j}, K_{t,i,j}) \stackrel{?}{=} 1.$$

Return 1 if and only if the equation holds.

The correctness of the SAKS scheme directly follows that of the SAFE-2 scheme.

Theorem 3. *Assume the Decisional Linear Assumption holds in $\mathbb{G}$, then our SAKS scheme is selectively secure.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ which can win the game with a non-negligible advantage ϵ, we construct another algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve the Decision Linear problem with advantage ϵ. On input $(g, g^a, g^b, g^{ac}, g^d, Z) \in \mathbb{G}_6$, $\mathcal{B}$ simulates the game for $\mathcal{A}$.

- **Init**: $\mathcal{A}$ declares two challenge strings $\overrightarrow{W_0^*} = (w_{00}^*, \ldots, w_{0n}^*)$, $\overrightarrow{W_1^*} = (w_{10}^*, \ldots, w_{1n}^*)$ and a distance τ. B flips a coin $\mu \in \{0, 1\}$. Let $W_\mu^* = (w_{\mu 0}^*, \ldots, w_{\mu n_1}^*)$. Then each character $w_{\mu_i}^*$ in W_μ^* is converted to a 2-bit string (a_1, a_2) and the algorithm creates vector $\vec{v_\mu^*}$ as in (i), then vector $\vec{v_\mu^*}$ is $\vec{v_\mu^*} = v_{\mu_{1_0}}^*, \ldots, v_{\mu_{1_{15}}}^*, v_{\mu_{2_0}}^*, \ldots, v_{\mu_{2_{15}}}^*, \ldots, v_{\mu_{n_0}}^*, \ldots, v_{\mu_{n_{15}}}^*)$
The rest is similar to the Public key and Master Secret Key generation in Theorem 1 - indistinguishably of *Game 2* and *Game 3*.
- **Query Phase 1**: A queries trapdoor for word $\bar{w} = (\bar{w}_1, \ldots, \bar{w}_n)$ under the restriction *Hamming Distance* $(W_\mu^*, \bar{w}) \neq \tau$. Each character $\bar{w}_i$ in $\bar{w}$ is converted to a 2-bit string (b_1, b_2). Then the algorithm creates vector $\vec{x}$ as in (ii), then vector $\vec{x}$ is : $\vec{x} = (x_{1_0}, \ldots, x_{1_{15}}, v_{2_0}, \ldots, x_{2_{15}}, \ldots, x_{n,0}, \ldots, x_{n_{15}})$. Then B simulates the trapdoor by following the simulation of Key Generation query in Theorem 1 - indistinguishably of *Game 2* and *Game 3*, Case 2.
- **Challenge**: This is similar to the simulation of the Challenge Ciphertext in Theorem 1 - indistinguishably of *Game 2* and *Game 3*
- **Query Phase 2**: Repeat Phase 1.
- **Outputs**: A outputs a bit μ'. Then B outputs 1 if $\mu' = \mu$ and 0 if $\mu' \neq \mu$.

Therefore, by following the same analysis as in Theorem 1 - indistinguishably of *Game 2* and *Game 3*, if $\mathcal{A}$ can distinguish the ciphertext, then $\mathcal{B}$ can solve the DLIN problem. □

7. Comparison and discussion

We give a detailed comparison among our SAFE1 scheme and SAFE2 scheme in Table 1. The schemes are compared in terms of the order of the number of characters, number expression of bit ,ciphertext size, key size, decryption cost. In addition, we show

Table 1
Performance Comparison.

Scheme	Number of character	Number expression of bit	Ciphertext Size	Key Size	Decryption Cost
SAFE 1	No	1	$(n+6)\|\mathbb{G}\|+\mathbb{G}_T$	$(n+6)\|\mathbb{G}\|$	$(4n+6)$p
SAFE 2	4	2	$(16n+6)\|\mathbb{G}\|+\mathbb{G}_T$	$(16n+6)\|\mathbb{G}\|$	$(64n+6)$p
SAFE 2-General	26	8	$(8^{26}\times n+6)\|\mathbb{G}\|+\mathbb{G}_T$	$(8^{26}\times n+6)\|\mathbb{G}\|$	$(8^{26}\times 4\times n+6)$p

p: pairing operation, n : length of input string

that our SAFE 2 scheme can be applied for general alphabets, in this case we choose the typical English alphabet. Instead of using 2-bit to express for (*A, T, G, C*) in the SAFE 2 scheme, we use 8-bit to express for each English letter in the extension of general case. For example, from the letter A is converted to 0100001, to the letter Z is converted to 01011010. Then we can check the similar two letters between two strings by applying (5). Since this is our first consideration for the general characters, it is inevitable to increase the computation cost.

8. Conclusion

In this paper, we proposed two Sequence Aware Functional Encryption (SAFE) schemes based on Hamming Distance. The first construction is for matching bit strings and the second construction is for character strings. Our SAFE schemes not only achieves confidentiality but also attribute hiding under some standard assumptions. We also showed that our SAFE scheme can be extended to a searchable encryption scheme. Our future work is to construct SAFE schemes based on other similarity metrics such as edit distance and longest common subsequence.

References

[1] Abdalla M, Catalano D, Dent A, Malone-Lee J, Neven G, Smart N. Identity-based encryption gone wild. In: Bugliesi M, Preneel B, Sassone V, Wegener I, editors. Automata, Languages and Programming. Lecture Notes in Computer Science, 4052. Springer Berlin Heidelberg; 2006. p. 300–11. doi:10.1007/11787006_26.

[2] Abdalla M, De Caro A, Phan DH. Generalized key delegation for wildcarded identity-based and inner-product encryption. Inf Forensics Secur, IEEE Trans 2012;7(6):1695–706. doi:10.1109/TIFS.2012.2213594.

[3] Agrawal S, Freeman D, Vaikuntanathan V. Functional encryption for inner product predicates from learning with errors. In: Lee D, Wang X, editors. Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science, 7073. Springer Berlin Heidelberg; 2011. p. 21–40. doi:10.1007/978-3-642-25385-0_2.

[4] Attrapadung N, Libert B, Panafieu E. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano D, Fazio N, Gennaro R, Nicolosi A, editors. Public Key Cryptography – PKC 2011. Lecture Notes in Computer Science, 6571. Springer Berlin Heidelberg; 2011. p. 90–108. doi:10.1007/978-3-642-19379-8_6.

[5] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: Security and Privacy, 2007. SP '07. IEEE Symposium on; 2007. p. 321–34. doi:10.1109/SP.2007.11.

[6] Boneh D, Boyen X. Efficient selective-id secure identity-based encryption without random oracles. In: Cachin C, Camenisch J, editors. Advances in Cryptology - EUROCRYPT 2004. Lecture Notes in Computer Science, 3027. Springer Berlin Heidelberg; 2004. p. 223–38. doi:10.1007/978-3-540-24676-3_14.

[7] Boneh D, Franklin M. Identity-based encryption from the weil pairing. In: Kilian J, editor. Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, 2139. Springer Berlin Heidelberg; 2001. p. 213–29. doi:10.1007/3-540-44647-8_13.

[8] Boneh D, Waters B. Conjunctive, subset, and range queries on encrypted data. In: Proceedings of the 4th conference on Theory of cryptography. In: TCC'07. Berlin, Heidelberg: Springer-Verlag; 2007. p. 535–54.

[9] Cheung DW, Mamoulis N, Wong WK, Yiu S, Zhang Y. Anonymous fuzzy identity-based encryption for similarity search. In: Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15–17, 2010, Proceedings, Part I; 2010. p. 61–72.

[10] Freeman DM. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques. In: EUROCRYPT'10; 2010. p. 44–61.

[11] Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security. In: CCS '06. New York, NY, USA: ACM; 2006. p. 89–98. doi:10.1145/1180405.1180418.

[12] Guillevic A. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In: Applied Cryptography and Network Security, 7954; 2013. p. 357–72.

[13] Iovino V, Persiano G. Hidden-vector encryption with groups of prime order. In: Proceedings of the 2nd international conference on Pairing-Based Cryptography. In: Pairing '08. Berlin, Heidelberg: Springer-Verlag; 2008. p. 75–88. doi:10.1007/978-3-540-85538-5_5.

[14] Katz J, Sahai A, Waters B. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology. In: EUROCRYPT'08. Berlin, Heidelberg: Springer-Verlag; 2008. p. 146–62.

[15] Lee K, Lee DH. Improved hidden vector encryption with short ciphertexts and tokens. Des Codes Cryptography 2011;58(3):297–319. doi:10.1007/s10623-010-9412-x.

[16] Okamoto T, Takashima K. Adaptively attribute-hiding (hierarchical) inner product encryption. In: Pointcheval D, Johansson T, editors. Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, 7237. Springer Berlin Heidelberg; 2012. p. 591–608. doi:10.1007/978-3-642-29011-4_35.

[17] Park J. Inner-product encryption under standard assumptions. Des, Codes Cryptography 2011;58(3):235–57. doi:10.1007/s10623-010-9405-9.

[18] Park JH, Lee K, Susilo W, Lee DH. Fully secure hidden vector encryption under standard assumptions. Inf Sci 2013;232:188–207.

[19] Sahai A, Waters B. Fuzzy identity-based encryption. In: Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Techniques. In: EUROCRYPT'05. Berlin, Heidelberg: Springer-Verlag; 2005. p. 457–73. doi:10.1007/11426639_27.

[20] Sedghi S, Liesdonk P, Nikova S, Hartel P, Jonker W. Searching keywords with wildcards on encrypted data. In: Garay J, Prisco R, editors. Security and Cryptography for Networks. Lecture Notes in Computer Science, 6280. Springer Berlin Heidelberg; 2010. p. 138–53. doi:10.1007/978-3-642-15317-4_10.

[21] Shamir A. Identity-based cryptosystems and signature schemes. In: Proceedings of CRYPTO 84 on Advances in cryptology. New York, NY, USA: Springer-Verlag New York, Inc.; 1985. p. 47–53.

[22] Shi E, Waters B. Delegating capabilities in predicate encryption systems. In: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II. In: ICALP '08. Berlin, Heidelberg: Springer-Verlag; 2008. p. 560–78. doi:10.1007/978-3-540-70583-3_46.

[23] Waters B. Functional encryption for regular languages. In: Safavi-Naini R, Canetti R, editors. Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, 7417. Springer Berlin Heidelberg; 2012. p. 218–35. doi:10.1007/978-3-642-32009-5_14.