Vilde Haugsbakken Heggen

Functional Encryption for Inner Product Functionality

Explanation and Applications

Master's thesis in Industrial Mathematics
Supervisor: Jiaxin Pan
May 2022

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Vilde Haugsbakken Heggen

Functional Encryption for Inner Product Functionality

Explanation and Applications

Master's thesis in Industrial Mathematics
Supervisor: Jiaxin Pan
May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Abstract

The contents of this work is constructing functional encryption schemes for less general functionalities, which are still expressive enough for practical scenarios. This paper is restricted to only using inner product as functional. Encrypting a vector $\mathbf{x}$ using a vector $\mathbf{y}$ as a key, will only reveal $\langle \mathbf{x}, \mathbf{y} \rangle$. A functional encryption scheme for inner product functionality will be presented, and a proof for its security, which is based on the decisional Diffie-Hellman assumption. Further I will generalize this scheme, building its security on the matrix Diffie-Hellman assumptions.

Contents

Chapter 1

Introduction

1.1 Functional Encryption

In functional encryption a message is encrypted in the same way as for public key encryption, using a public key. But decryption for a functional encryption scheme differs from a public key encryption scheme, in the way that in public key encryption the whole message is retrieved by the receiver when decrypting. In functional encryption, only the functional value of the message is retrieved by the receiver.
Functional encryption is interesting since it allows users to control the amount of information retrieved by a given receiver. Decryption does no longer have to be an all or nothing affair, since we can decrypt only some of the information. Functional encryption could for example be of great use if one wanted to filtrate spam email, since one would want to leak as little information as possible from the email, just enough to decide if it is most likely spam or not.

1.2 Motivation

A great motivation of this paper is constructing functional encryption schemes for less general functionalities, which are still expressive enough for practical scenarios. This paper is therefore restricted to only using inner product as functional.
The work of this paper is based on *Simple Functional Encryption Schemes for Inner Products*, [1]. Further this work generalizes one of the schemes, such that the security is based on the matrix Diffie-Hellman (MDDH) assumptions. The matrix Diffie-Hellman assumptions are a generalization of the decisional Diffie-Hellman assumption, and will give various assumptions depending on the matrix distribution. In this paper I use the uniform distribution to define a generalized functional encryption scheme for inner product functionality, such that its security is based on the $\mathcal{U}_{k+1,k}-$MDDH assumption.

1.3 Further Directions

The functional encryption schemes introduced in this paper are proven to fulfill the security notion; selective indistinguishability against chosen plaintext attacks. Further directions from this paper could be also proving non-selective security, which is a stronger security notion.

Chapter 2

Preliminaries

2.1 Notation

2.1.1 Symbols

"$negl(\lambda)$" means neglibile given a security parameter λ.
"$\approx_c$" means computationally indistinguishable. The symbol states that no efficient algorithm can tell the difference between the right and the left side of the symbol, except with small probability.
"$\overset{\$}{\leftarrow}$" means the element on the left side of this symbol is randomly drawn from the set on the right side.

2.1.2 Overscore and Underscore Notation

Let $l > k$ for $l, k \in \mathbb{N}$. These are constants and usually small numbers. For a $l \times k$-matrix $\mathbf{A}$ with rank k, we define $\overline{\mathbf{A}}$ as the first k rows of $\mathbf{A}$ and $\underline{\mathbf{A}}$ as the last $l - k$ rows.
We can also define this notation for a vector in conjunction to a matrix with rank k. For a vector $\mathbf{a}$ of dimension l, let $\overline{\mathbf{a}}$ be the first k elements and $\underline{\mathbf{a}}$ be the last $l - k$.

2.1.3 Implicit Representation

In this work a cyclic group will often be considered. The elements of such a group can be represented implicit. This notation is introduced in [2]. This implicit representation is defined below.

Let $\mathbb{G}$ be a cyclic group of order q with generator g. For an element $a \in \mathbb{Z}_q$, we define $[a] = g^a$ as the implicit representation of a in the group $\mathbb{G}$.
A similar definition can be given for a vector $\mathbf{a} \in \mathbb{Z}_q^k$ of an arbitrary dimension k. The

implicit representation of such a vector is

$$[\mathbf{a}] = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} = \begin{pmatrix} g^{a_1} \\ g^{a_2} \\ \vdots \\ g^{a_k} \end{pmatrix}.$$

For a matrix $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}_q^{n \times m}$ we define

$$[\mathbf{A}] = \begin{pmatrix} g^{a_{1,1}} & g^{a_{1,2}} & ... & g^{a_{1,m}} \\ g^{a_{2,1}} & g^{a_{2,2}} & ... & g^{a_{2,m}} \\ \vdots & & & \\ g^{a_{n,1}} & g^{a_{n,2}} & ... & g^{a_{n,m}} \end{pmatrix}$$

as the implicit representation of $\mathbf{A}$ in $\mathbb{G}$.

2.2 Definitions

Definition 2.2.1. *PPT*

The notion PPT is short for probabilistic polynomial-time. This is a complexity class. If a decision problem is PPT, there exists an algorithm that makes random decisions which can solve the problem in polynomial time.

Definition 2.2.2. *Group Generator,* *GGen*

A group generator, GGen, is a PPT algorithm that outputs $\mathcal{G} = (\mathbb{G}, g, q) \overset{\$}{\leftarrow} GGen(1^\lambda)$, *where* $\mathbb{G}$ *is a cyclic group of prime-order* q $(2^{\lambda-1} \leq q \leq 2^\lambda)$ *and* g *is a generator of* $\mathbb{G}$. λ *is a security parameter.*

Definition 2.2.3. *Public-Key Encryption Scheme*

A public-key encryption scheme (PKE) consists of three PPT algorithms:

1. $Gen(1^\lambda)$ *which outputs public and secret keys,* (pk, sk), *for a security parameter* λ

2. $Enc(pk, m)$ *which inputs a public key and a message from the allowed message space, and outputs a ciphertext*

3. $Dec(sk, c)$ *inputs a chipertext and a secret key and outputs a corresponding message.*

Correctness of the PKE requires that for all $(pk, sk) \leftarrow Gen(1^\lambda)$ *and all messages* m, $Dec(sk, Enc(pk, m)) = m$ *except with negligible probability.*

Definition 2.2.4. ***Functionality***

A functionality F defined over (K, X) is a function $F : K \times X \to \Sigma \cup \{\perp\}$, where K is the key space, X the message space and Σ is the output space and $\perp$ is a special string not contained in Σ. The functionality is undefined when either the key is not in the key space or the message not in the message space.

Definition 2.2.5. ***Inner Product***

The inner product of two vectors $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_l \end{pmatrix}$ and $\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix}$ in some field, is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i \in |l|} x_i \cdot y_i.$$

In this paper the inner product is the same as the dot product.

Definition 2.2.6. ***Functional Encryption Scheme***

A functional encryption scheme $\mathcal{FE}$ for functionality F is a tuple
$\mathcal{FE} = (Setup, KeyDer, Encrypt, Decrypt)$ of 4 algorithms:
1. $Setup(1^\lambda)$ outputs public and master secret keys (mpk, msk) for security parameter λ;
2. $KeyDer(msk, k)$, on input a master secret key, msk, and a key $k \in K$ outputs secret key sk_k;
3. $Encrypt(mpk, x)$, on input public key mpk and message $x \in X$ outputs ciphertext Ct;
4. $Decrypt(mpk, Ct, sk_k)$ outputs $\sigma \in \Sigma \cup \{\perp\}$.

We make the following correctness requirement: for all $(mpk, msk) \leftarrow Setup(1^\lambda)$, all $k \in K$ and $x \in X$, for $sk_k \leftarrow KeyDer(msk, k)$ and $Ct \leftarrow Encrypt(mpk, x)$, we have that $Decrypt(mpk, Ct, sk_k) = F(k, x)$, whenever $F(k, x) \neq \perp$, except with negligible probability.

Definition 2.2.7. ***Code-Based Games***

In this paper I will use code-based game-playing to define security notions. Such games consist of a main procedure and the procedures of some oracles. These procedures varies according to the security notion. The game is played by executing the main procedure and responding to the oracle queries of the adversary. The advantage is defined accordingly.

Definition 2.2.8. ***The DDH Assumption***

Let $\mathbb{G}$ be a cyclic group of order q with generator g. Let $a, b \overset{\$}{\leftarrow} \mathbb{Z}_q$ be independently *chosen. The decisional Diffie-Hellman assumption states that*

$$(g^a, g^b, g^{ab}) \approx_c (g^a, g^b, g^c) \tag{2.1}$$

for $c \xleftarrow{\$} \mathbb{Z}_q$.

Definition 2.2.9. *The DDH assumption holds relative to GGen if for all PPT algorithms* A, $Adv^{ddh}_{GGen,A}(\lambda)$ *is negligible in* λ, *where* λ *is a security parameter. The games of figure 2.1 is used to define the advantage* $Adv^{ddh}_{GGen,A}(\lambda)$.

For an adversary A *playing the games of figure 2.1, the advantage* $Adv^{ddh}_{GGen,A}(\lambda)$ *will be defined as*

$$Adv^{ddh}_{GGen,A}(\lambda) = Adv(REAL^A, RAND^A) = |Pr[REAL^A \implies 1] - Pr[RAND^A \implies 1]|.$$

Game $\underline{REAL}$	**Game** $\underline{RAND}$
$\mathcal{G} \xleftarrow{\$} GGen(1^\lambda)$	$\mathcal{G} \xleftarrow{\$} GGen(1^\lambda)$
$x, y \xleftarrow{\$} \mathbb{Z}_q$	$x, y, z \xleftarrow{\$} \mathbb{Z}_q$
$\beta \xleftarrow{\$} A(\mathcal{G}, X = g^x, Y = g^y, Z = g^{xy})$	$\beta \xleftarrow{\$} A(\mathcal{G}, X = g^x, Y = g^y, Z = g^z)$
return β	return β

Figure 2.1: **Game** $REAL/RAND$

Definition 2.2.10. $\mathcal{U}_{l,k}$ ***Matrix Distribution***
A matrix from the matrix distribution $\mathcal{U}_{l,k}$, *is a matrix with* l *many rows and* k *many columns. It has rank* k *and all elements are drawn uniformly random from* $\mathbb{Z}_q$.

Definition 2.2.11. ***IND-FE-CPA/s-IND-FE-CPA Security***
For a functional encryption scheme $\mathcal{FE} = (Setup, KeyDer, Encrypt, Decrypt)$ *for functionality* F, *defined over* (K, X), *one can define security against chosen plaintext attacks (IND-FE-CPA security). Consider the games in figure 2.2. The advantage of an adversary* A *can be defined relative to the games as*

$$\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathcal{FE},\mathsf{A}}(\lambda) = \left| Pr[\textit{IND-FE-CPA}^A \implies 1] - \frac{1}{2} \right|.$$

We say that $\mathcal{FE}$ *is IND-FE-CPA secure if the advantage* $\mathsf{Adv}^{\mathsf{ind\text{-}fe\text{-}cpa}}_{\mathcal{FE},\mathsf{A}}(\lambda)$ *is negligible. The IND-FE-CPA security game is presented in figure 2.2.*

```
**Game** IND-FE-CPA($\lambda, k$)
$(mpk, msk) \xleftarrow{\$} Setup(1^\lambda)$
$x_0, x_1 \leftarrow A$
$V \leftarrow \emptyset$
$\beta \xleftarrow{\$} \{0,1\}$
$\hat{\beta} \xleftarrow{\$} A^{\textsc{KeyDer}(\cdot),\textsc{Encrypt}(\cdot,\cdot)}(mpk)$
If $\exists k \in V$ s.t. $F(k, x_0) \neq F(k, x_1)$:
    return $false$
return $\hat{\beta} = \beta$

**Oracle** KeyDer($k$)
$V \leftarrow V \cup \{k\}$
$sk_k \xleftarrow{\$} KeyDer(msk, k)$
return $sk_k$

**Oracle** Encrypt($x_0, x_1$)
$Ct \xleftarrow{\$} Encrypt(mpk, x_\beta)$
return $Ct$
```

Figure 2.2: **Game** IND-FE-CPA

We can also define selective security against chosen plaintext attacks for a functional encryption scheme (s-IND-FE-CPA security). This is the situation when the challenge messages x_0 and x_1 have to be chosen before hand.

```
**Game** s-IND-FE-CPA($\lambda, k, x_0, x_1$)
$(mpk, msk) \xleftarrow{\$} Setup(1^\lambda)$
$V \leftarrow \emptyset$
$\beta \xleftarrow{\$} \{0,1\}$
$\hat{\beta} \xleftarrow{\$} A^{\textsc{KeyDer}(\cdot),\textsc{Encrypt}(\cdot,\cdot)}(mpk)$
If $\exists k \in V$ s.t. $F(k, x_0) \neq F(k, x_1)$:
    return $false$
return $\hat{\beta} = \beta$

**Oracle** KeyDer($k$)
$V \leftarrow V \cup \{k\}$
$sk_k \xleftarrow{\$} KeyDer(msk, k)$
return $sk_k$

**Oracle** Encrypt($x_0, x_1$)
$Ct \xleftarrow{\$} Encrypt(mpk, x_\beta)$
return $Ct$
```

Figure 2.3: **Game** s-IND-FE-CPA

A functional encryption scheme is s-IND-FE-CPA secure if the advantage

$$\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}fe\text{-}cpa}}_{\mathcal{FE},\mathsf{A}}(\lambda) = \left| Pr[\textit{s-IND-FE-CPA}^A \implies 1] - \frac{1}{2} \right|$$

is negligible. The s-IND-FE-CPA security game is presented in figure 2.3.

Chapter 3

Functional Encryption Scheme based on DDH

In this chapter a functional encryption scheme for the inner product functionality is presented. The security of the scheme is based on the decisional Diffie-Hellman (DDH) assumption. A proof that the scheme is s-IND-FE-CPA secure will be given.

3.1 Construction

Let $IP = (Setup, KeyDer, Encrypt, Deckrypt)$ be the functional encryption scheme for inner product functionality. The 4 algorithms are defined as in the figure below.

```
$\underline{Setup(1^\lambda, 1^l)}$
$(\mathbb{G}, q, g) \xleftarrow{\$} GGen(1^\lambda)$
$\mathbf{s} = (s_1, ..., s_l)^T \xleftarrow{\$} \mathbb{Z}_q^l$
$mpk = (h_i = [s_i])_{i\in|l|}$
$msk = \mathbf{s}$
return $(mpk, msk)$

$\underline{KeyDer(msk, \mathbf{y} = (y_1, ..., y_l)^T \in \mathbb{Z}_q^l)}$
return $sk_y = \langle \mathbf{y}, \mathbf{s} \rangle$

$\underline{Encrypt(mpk, \mathbf{x} = (x_1, ..., x_l)^T \in \mathbb{Z}_q^l)}$
$r \xleftarrow{\$} \mathbb{Z}_q$
$[ct_0] = [r]$ and
$[ct_i] = [s_i \cdot r + x_i]$ for $i \in |l|$
return $[\mathbf{Ct}] = [ct_0, (ct_i)_{i\in|l|}]$

$\underline{Decrypt(mpk, [\mathbf{Ct}], sk_y)}$
return $\prod_{i\in|l|} \frac{[y_i \cdot ct_i]}{[sk_y \cdot ct_0]}$
```

Figure 3.1: **FE Scheme for Inner Product Functionality**

The encryption of this functional encryption scheme is just the same as in the public key encryption scheme ElGamal [5]. The scheme differs from a public key encryption scheme, since a user secret key, sk_y, is computed and used in the decryption.

3.1.1 Correctness

For all $(mpk, msk) \leftarrow Setup(1, 1^l)$, for all $\mathbf{y}, \mathbf{x} \in \mathbb{Z}_q^l$, $sk_y \leftarrow KeyDer(msk, \mathbf{y})$ and $[\mathbf{Ct}] \leftarrow Encrypt(mpk, \mathbf{x})$ we have that

$$\begin{aligned}
Decrypt(mpk, [\mathbf{Ct}], sk_y) = \frac{\prod_{i\in|l|}[y_i \cdot ct_i]}{[sk_y \cdot ct_0]} &= \left[\left(\sum_{i\in|l|} y_i \cdot ct_i\right) - sk_y \cdot ct_0\right] \\
&= \left[\left(\sum_{i\in|l|} y_i(s_i \cdot r + x_i)\right) - \left(\sum_{i\in|l|} y_i \cdot s_i\right) r\right] \\
&= \left[\sum_{i\in|l|} y_i \cdot s_i \cdot r + \sum_{i\in|l|} y_i \cdot x_i - \sum_{i\in|l|} y_i \cdot s_i \cdot r\right] \\
&= \left[\sum_{i\in|l|} y_i \cdot x_i\right] = [\langle \mathbf{y}, \mathbf{x} \rangle].
\end{aligned}$$

Notice that decrypting an encrypted message $\mathbf{x}$ will only reveal the inner product of $\mathbf{x}$ and the key $\mathbf{y}$ in the group $\mathbb{G}$.

3.2 Security

Theorem 3.2.1. *Under the DDH assumption, the above IP scheme is s-IND-FE-CPA secure. In particular, for an adversary A there exists an adversary B with roughly the same running time such that*

$$\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}fe\text{-}cpa}}_{\mathcal{FE},\mathsf{A}}(\lambda) = Adv^{ddh}_{GGen,B}(\lambda).$$

Proof:
For the proof consider the three games of figure 3.2.

```
Game $G_0/G_1/G_2$
$(\mathbb{G}, q, g) \xleftarrow{\$} GGen(1^\lambda)$
$a \xleftarrow{\$} \mathbb{Z}_q$
$\mathbf{s}' = \mathbf{x}_1 - \mathbf{x}_0$
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^l$
$h_i = [a \cdot s'_i + r_i]$
$mpk = (h_i)_{i \in |l|}$
$V \leftarrow \emptyset$
$\beta \xleftarrow{\$} \{0,1\}$
$\hat{\beta} \xleftarrow{\$} A^{\text{KeyDer}(\cdot),\text{Encrypt}(\cdot,\cdot)}(mpk)$
If $\exists \mathbf{y} \in V$ s.t. $\langle \mathbf{y}, \mathbf{x}_0 \rangle \neq \langle \mathbf{y}, \mathbf{x}_1 \rangle$:
   return $false$
return $\hat{\beta} = \beta$

Oracle KeyDer($\mathbf{y}$)
$V \leftarrow V \cup \{\mathbf{y}\}$
return $sk_y = \langle \mathbf{r}, \mathbf{y} \rangle$

Oracle Encrypt($\mathbf{x}_0, \mathbf{x}_1$)
$b \xleftarrow{\$} \mathbb{Z}_q$
$\mathbf{z} \xleftarrow{\$} \mathbb{Z}_q^l$
$[ct_0] = [b]$
$[ct_i] = [(a \cdot s'_i + r_i) \cdot b + x_{\beta,i}]$   //$G_0$
$[ct_i] = [z_i + x_{\beta,i}]$   //$G_1$
$[ct_i] = [z_i]$   //$G_2$
return $[\mathbf{Ct}] = [ct_0, (ct_i)_{i \in |l|}]$
```

Figure 3.2: **Game** $G_0/G_1/G_2$

Notice that we have the restriction $\langle \mathbf{y}, \mathbf{x}_0 \rangle = \langle \mathbf{y}, \mathbf{x}_1 \rangle$ for the key $\mathbf{y}$ when given the messages $\mathbf{x}_0$ and $\mathbf{x}_1$. This restriction causes $\mathbf{y} \in \{\mathbf{x}_1 - \mathbf{x}_0\}^\perp$, where $\{\mathbf{x}_1 - \mathbf{x}_0\}$ is the space spanned by $\mathbf{x}_1 - \mathbf{x}_0$. The master secret key will in this case be $a \cdot \mathbf{s}' + \mathbf{r}$, where $\mathbf{s}' = \mathbf{x}_1 - \mathbf{x}_0$. Hence simulation of $sk_y = \langle \mathbf{r}, \mathbf{y} \rangle$ is of the same form as $\langle msk, \mathbf{y} \rangle$.

By definition 2.2.11, G_0 is the s-IND-FE-CPA-game for the scheme IP. Hence

$$Pr[\text{s-IND-FE-CPA}^A \implies 1] = Pr[G_0^A \implies 1].$$

G_1 differs from G_0 only in how $[ct_i]$ is computed. In G_1, $\mathbf{z} \xleftarrow{\$} \mathbb{Z}_q^l$ is used in the encryption. We state the following lemma.

Lemma 3.2.2. *There exists an adversary B with roughly the same running time as A such that*

$$Adv(G_0^A, G_1^A) = Adv_{GGen,B}^{ddh}(\lambda).$$

G_2 is exactly the same as G_1 except that in G_2 we do not encrypt the message, only the random vector $\mathbf{z}$. This is just a conceptual change. Hence the distribution of $[\mathbf{Ct}]$ does not change and therefore

$$Pr[G_1^A \implies 1] = Pr[G_2^A \implies 1].$$

In G_2, $[\mathbf{Ct}] = [b, (z_i)_{i \in |l|}]$ which is independent of the challenge bit β. Hence

$$Pr[G_2^A \implies 1] = \frac{1}{2}.$$

Combining the equations derived above we obtain

$$\begin{aligned}Pr[s-IND-FE-CPA^A \implies 1] &= Pr[G_0^A \implies 1]\\ &= Pr[G_1^A \implies 1] + Adv^{ddh}_{GGen,B}(\lambda)\\ &= Pr[G_2^A \implies 1] + Adv^{ddh}_{GGen,B}(\lambda)\\ &= \frac{1}{2} + Adv^{ddh}_{GGen,B}(\lambda),\end{aligned}$$

which proves the theorem since $\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}fe\text{-}cpa}}_{\mathcal{FE},\mathsf{A}}(\lambda) = \left|Pr[\text{s-IND-FE-CPA}^A \implies 1] - \frac{1}{2}\right| = Adv^{ddh}_{GGen,B}(\lambda)$. □

Proof of Lemma 3.2.2:
Let B be an adversary against the DDH assumption. Adversary B inputs $(\mathcal{G}, [a], [b], [c])$, for $a, b, c \xleftarrow{\$} \mathbb{Z}_q$, and needs to distinguish $c = a \cdot b$ (game $REAL$) from c uniformly random (game $RAND$).

```
Procedure $B(\mathcal{G}, [a], [b], [c])$
$\mathbf{s}' = \mathbf{x}_1 - \mathbf{x}_0$
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^l$
$h_i = [a \cdot s'_i + r_i]$
$mpk = (h_i)_{i \in |l|}$
$V \leftarrow \emptyset$
$\beta \xleftarrow{\$} \{0, 1\}$
$\hat{\beta} \xleftarrow{\$} A^{\textsc{KeyDer}(\cdot),\textsc{Encrypt}(\cdot,\cdot)}(mpk)$
If $\exists \mathbf{y} \in V$ s.t. $\langle \mathbf{y}, \mathbf{x}_0 \rangle \neq \langle \mathbf{y}, \mathbf{x}_1 \rangle$:
    return $false$
return $\hat{\beta} = \beta$

Oracle KeyDer($\mathbf{y}$)
$V \leftarrow V \cup \{\mathbf{y}\}$
return $sk_y = \langle \mathbf{r}, \mathbf{y} \rangle$

Oracle Encrypt($\mathbf{x}_0, \mathbf{x}_1$)
$[ct_0] = [b]$
$[ct_i] = [c \cdot s'_i + b \cdot r_i + x_{\beta,i}]$
return $[\mathbf{Ct}] = [ct_0, (ct_i)_{i \in |l|}]$
```

Figure 3.3: **Adversary** B

If B runs the $REAL$ game, the Encrypt simulation is the same as in G_0 from figure 3.2, and hence $Pr[REAL^B \implies 1] = Pr[G_0^A \implies 1]$. If B runs the $RAND$ game, the Encrypt simulation is the same as in G_1. Also, in both games the adversary will be provided with answers to the oracle queries of the KeyDer oracle. The answers of the KeyDer oracle are of the form $\langle msk, \mathbf{y} \rangle = \langle \mathbf{r}, \mathbf{y} \rangle$, since $\mathbf{y}$ has to be element in $\{\mathbf{x}_1 - \mathbf{x}_0\}^\perp$ if $\langle \mathbf{y}, \mathbf{x}_0 \rangle = \langle \mathbf{y}, \mathbf{x}_1 \rangle$. Hence sk_y will leak some information about $\mathbf{r}$. But since $[a]$ is random, $[a \cdot s'_i + r_i]$ will still be random. Hence $Pr[RAND^B \implies 1] = Pr[G_1^A \implies 1]$. We can therefore conclude that

$$\begin{aligned}Adv^{ddh}_{GGen,B}(\lambda) &= |Pr[REAL^B \implies 1] - Pr[RAND^B \implies 1]|\\ &= |Pr[G_0^A \implies 1] - Pr[G_1^A \implies 1]|\\ &= Adv(G_0^A, G_1^A). \quad \square\end{aligned}$$

Chapter 4

Matrix Diffie-Hellman Assumptions

Further in this paper I will try to generalize the scheme IP from last chapter. The security of this more general scheme is based on the matrix Diffie-Hellman (MDDH) assumptions. This chapter will include the definition of the MDDH assumptions and some examples. The chapter is taken from my project thesis, *Matrix Diffie-Hellman Assumptions* [3].

4.1 Definition

The decisional Diffie-Hellman assumption can be generalized to the $\mathcal{D}_{l,k}$-matrix Diffie-Hellman ($\mathcal{D}_{l,k}-$MDDH) assumption.

Definition 4.1.1. $\mathcal{D}_{l,k}$***-Matrix Diffie-Hellman Assumption*** *Let $l, k \in \mathbb{N}$ and $l > k$. These are constants and usually small numbers. Let $\mathcal{D}_{l,k}$ be a matrix distribution over $\mathbb{Z}_q^{l\times k}$, such that a matrix of this distribution has rank k. Let $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{l\times k}$ form the distribution $\mathcal{D}_{l,k}$, $\boldsymbol{r} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$ and $\boldsymbol{u} \overset{\$}{\leftarrow} \mathbb{G}^l$. Then the $\mathcal{D}_{l,k}$-matrix Diffie-Hellman assumption is defined as*

$$[\boldsymbol{A}||\boldsymbol{A}\boldsymbol{r}] \approx_c [\boldsymbol{A}||\boldsymbol{u}] \in \mathbb{G}^{l\times(k+1)}. \tag{4.1}$$

The $\mathcal{D}_{l,k}$-matrix Diffie-Hellman assumption holds if for all PPT adversaries A,

$$Adv^{mddh}_{GGen,A}(\lambda) = |Pr[REAL^A \implies 1] - Pr[RAND^A \implies 1]| = negl(\lambda),$$

where $negl(\lambda)$ means negligible for a given security parameter λ. $Pr[REAL^A \implies 1]$ means the adversary outputs 1 given a real $\mathcal{D}_{l,k}$-matrix distribution and $Pr[RAND^A \implies 1]$ means the adversary outputs 1 given a random distribution.

In the following it will be reveled that the matrix Diffie-Hellman (MDDH) assumptions with $l = k + 1$ are, among others, the Linear, Cascade- and Symmetric Cascade assumptions. Since these assumptions are the ones of interest, I will restrict this paper to using $l = k + 1$. The only interesting case for $l > k + 1$ is when $\mathcal{D}_{l,k}$ is a random distribution. If so, one can generalize the results of this paper by adding $l - (k + 1)$ random rows to the matrix.

4.2 Examples

4.2.1 The DDH Assumption

Let us look at an example of the MDDH assumptions when $k = 1$ and $l = k + 1$. Let

$$\begin{aligned} \mathbf{A} &= \begin{pmatrix} a \\ 1 \end{pmatrix} \in \mathcal{L}_1 \quad \text{and} \\ r, z \xleftarrow{\$} \mathbb{Z}_q, \quad [\mathbf{u}] &= \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \in \mathbb{G}^2, \quad \text{then} \\ [\mathbf{A}||\mathbf{Ar}] &= \begin{pmatrix} g^a & g^{ar} \\ g & g^r \end{pmatrix} \\ &\quad \text{and} \\ [\mathbf{A}||\mathbf{u}] &= \begin{pmatrix} g^a & g^{u_1} \\ g & g^{u_2} \end{pmatrix}. \end{aligned}$$

The MDDH assumption, by equation 4.1, then becomes

$$\begin{pmatrix} g^a & g^{ar} \\ g & g^r \end{pmatrix} \approx_c \begin{pmatrix} g^a & g^{u_1} \\ g & g^{u_2} \end{pmatrix}$$

We can rewrite this assumption, in a shorter notation. Then it becomes

$$([a], [ar], [r]) \approx_c ([a], [u_1], [u_2]).$$

Since $a \xleftarrow{\$} \mathbb{Z}_q^*$ and $r, u_1, u_2 \xleftarrow{\$} \mathbb{Z}_q$, the distributions of the triplets in the assumption above matches the distributions of the triplets in the DDH assumption (2.1). Hence this is just the DDH assumption.

4.2.2 $\mathcal{D}_{3,2}$-MDDH Assumptions

Let $k = 2$ and $l = k + 1$ then we can consider the following examples of distributions for the matrix $\mathbf{A}$

$$\mathcal{L}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \quad \mathcal{C}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 1 & a_2 \\ 0 & 1 \end{pmatrix} \quad \mathcal{SC}_2 : \mathbf{A} = \begin{pmatrix} a & 0 \\ 1 & a \\ 0 & 1 \end{pmatrix},$$

for uniform $a, a_1, a_2 \xleftarrow{\$} \mathbb{Z}_q^*$.

If we now consider $\mathbf{A} \leftarrow \mathcal{L}_2$ and $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$. Let

$$\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}, \quad [\mathbf{u}] = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \in \mathbb{G}^3.$$

The corresponding MDDH assumption can be written

$$([a_1], [a_2], [a_1 r_1], [a_2 r_2], [r_1 + r_2]) \approx_c ([a_1], [a_2], [u_1], [u_2], [u_3]).$$

This is the $2 - Lin$ assumption.

Let us now consider $\mathbf{A} \leftarrow \mathcal{C}_2$ and $\mathbf{r}$ and $\mathbf{u}$ as before. The $2 - Casc$ assumption (Cascade Assumption) can be defined as $2 - Casc := \mathcal{C}_2$-MDDH assumption. The assumption will become

$$([a_1], [a_2], [a_1 r_1], [r_1 + a_2 r_2], [r_2]) \approx_c ([a_1], [a_2], [u_1], [u_2], [u_3]).$$

Considering $\mathbf{A} \leftarrow \mathcal{SC}_2$ and $\mathbf{r}$ and $\mathbf{u}$ as before. The $2 - SCasc$ assumption (Symmetric Cascade Assumption) can be defined as $2 - SCasc := \mathcal{SC}_2$-MDDH assumption. The assumption will become

$$([a], [a \cdot r_1], [r_1 + a \cdot r_2], [r_2]) \approx_c ([a], [u_1], [u_2], [u_3]).$$

4.2.3 $k - Lin$ Assumption

Let $\mathcal{L}_k$ be the linear matrix distribution such that a $(k+1) \times k$-matrix $\mathbf{A}$ of this distribution has the form

$$\mathbf{A} = \begin{pmatrix} a_1 & 0 & ... & 0 & 0 \\ 0 & a_2 & ... & 0 & 0 \\ 0 & 0 & & \ddots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & ... & 0 & a_k \\ 1 & 1 & ... & 1 & 1 \end{pmatrix} \in \mathbb{Z}_q^{(k+1) \times k},$$

where $a_i \leftarrow \mathbb{Z}_q^*$.
Let a matrix $\mathbf{A} \leftarrow \mathcal{L}_k$, then the $k - Lin$ assumption (where $k \in \mathbb{N}$) is equivalent to the $\mathcal{L}_k$-MDDH assumption. The assumption can be written as $([a_1], ..., [a_k], [a_1 r_1], ..., [a_k r_k], [r_1 + ... + r_k]) \approx_c ([a_1], ..., [a_k], [u_1], ..., [u_k], [u_{k+1}])$, where $a_i \leftarrow \mathbb{Z}_q^*$, and $r_i, u_j \leftarrow \mathbb{Z}_q$ for $i = 1, ..., k$ and $j = 1, ..., k + 1$.
Examples of this when $k = 1$ and $k = 2$ have been shown.

4.2.4 $k - Casc$ Assumption

Let $\mathcal{C}_k$ be a matrix distribution such that a $(k+1) \times k$-matrix $\mathbf{A}$ of this distribution has the form

$$\mathbf{A} = \begin{pmatrix} a_1 & 0 & \dots & 0 & 0 \\ 1 & a_2 & \dots & 0 & 0 \\ 0 & 1 & & \ddots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & a_k \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \in \mathbb{Z}_q^{(k+1)\times k},$$

where $a_i \leftarrow \mathbb{Z}_q^*$.
Let a matrix $\mathbf{A} \leftarrow \mathcal{C}_k$, then the $k - Casc$ assumption is defined as the $\mathcal{C}_k$-MDDH assumption. The assumption can be written as $([a_1], ..., [a_k], [a_1 \cdot r_1], [r_1 + a_2 \cdot r_2], ..., [r_{k-1} + a_k \cdot r_k], [r_k]) \approx_c ([a_1], ..., [a_k], [u_1], [u_2], ..., [u_k], [u_{k+1}])$, where $a_i \leftarrow \mathbb{Z}_q^*$, and $r_i, u_j \leftarrow \mathbb{Z}_q$ for $i = 1, ..., k$ and $j = 1, ..., k+1$.

Chapter 5

Functional Encryption Scheme based on MDDH

In chapter 3, a functional encryption scheme for inner product functionality is presented. Its security is based on the decisional Diffie-Hellman (DDH) assumption. Chapter 4 describes how the DDH assumption can be generalized to the matrix Diffie-Hellman (MDDH) assumptions, among these is the $\mathcal{U}_{k+1,k}$-MDDH assumption.
This chapter includes the generalization of the scheme IP from chapter 3, and the proof of its security based on the $\mathcal{U}_{k+1,k}$-MDDH assumption. The scheme is proven s-IND-FE-CPA secure under the $\mathcal{U}_{k+1,k}$-MDDH assumption.

5.1 Construction

Let $IP_{MDDH} = (Setup, KeyDer, Encrypt, Deckrypt)$ be the functional encryption scheme for inner product functionality based on the $\mathcal{U}_{k+1,k}$-MDDH assumption. The matrix distribution $\mathcal{U}_{k+1,k}$ is described in definition 2.2.10 in the preliminaries. The 4 algorithms for the generalized scheme are defined as in the figure below.

```
$\underline{Setup(1^\lambda, 1^l)}$
$(\mathbb{G}, q, g) \xleftarrow{\$} GGen(1^\lambda)$
$\mathbf{A} \leftarrow \mathcal{U}_{k+1,k}$
$\mathbf{T}_A = \underline{\mathbf{A}}\overline{\mathbf{A}}^{-1}$
for $i = 1, ..., l$ :
    $s_i \xleftarrow{\$} \mathbb{Z}_q^*$
    $\underline{\mathbf{A}}_i = s_i \underline{\mathbf{A}}$
    $\mathbf{T}_i = s_i \mathbf{T}_A$
$mpk = \begin{bmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}}_1 \\ \vdots \\ \underline{\mathbf{A}}_l \end{bmatrix} \in \mathbb{G}^{(k+l)\times k}$
$msk = \mathbf{T} = \begin{pmatrix} \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_l \end{pmatrix} \in \mathbb{Z}_q^{l\times k}$
return $(mpk, msk)$
```

```
$\underline{KeyDer(msk, \mathbf{y} = (y_1, ..., y_l)^T \in \mathbb{Z}_q^l)}$
$sk_y = \mathbf{y}^T\mathbf{T} = \sum_{i\in|l|} y_i \mathbf{T}_i$
return $sk_y$

$\underline{Encrypt(mpk, \mathbf{x} = (x_1, ..., x_l)^T \in \mathbb{Z}_q^l)}$
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^k$
$[\mathbf{ct}_0] = [\overline{\mathbf{A}} \cdot \mathbf{r}]$
$[ct_i] = [\underline{\mathbf{A}}_i \cdot \mathbf{r} + x_i]$ for $i \in |l|$
return $[\mathbf{Ct}] = [ct_0, (ct_i)_{i\in|l|}]$

$\underline{Decrypt(mpk, [\mathbf{Ct}], sk_y)}$
return $\prod_{i\in|l|} \frac{[y_i \cdot ct_i]}{[sk_y \cdot ct_0]}$
```

Figure 5.1: **FE Scheme for Inner Product Functionality based on MDDH**

Notice that the user secret key, sk_y, is the matrix-vector product of the key $\mathbf{y}$ with the master secret key $\mathbf{T}$. The matrix-vector product of $\mathbf{y}$ with $\mathbf{T}$ is just the dot product of $\mathbf{y}$ with each row of $\mathbf{T}$. Notice that only the user secret key is used in the decryption.

5.1.1 Correctness

For all $(mpk, msk) \leftarrow Setup(1^\lambda, 1^l)$, for all $\mathbf{y}, \mathbf{x} \in \mathbb{Z}_q^l$, $sk_y \leftarrow KeyDer(msk, \mathbf{y})$ and $[\mathbf{Ct}] \leftarrow Encrypt(mpk, \mathbf{x})$ we have that

$$\begin{aligned}
Decrypt(mpk, [\mathbf{Ct}], sk_y) = \frac{\prod_{i\in|l|}[y_i \cdot ct_i]}{[sk_y \cdot \mathbf{ct}_0]} &= \left[\left(\sum_{i\in|l|} y_i \cdot ct_i\right) - sk_y \cdot \mathbf{ct}_0\right] \\
&= \left[\sum_{i\in|l|} y_i(\underline{\mathbf{A}}_i \cdot \mathbf{r} + x_i) - \left(\sum_{i\in|l|} y_i \mathbf{T}_i\right)\overline{\mathbf{A}} \cdot \mathbf{r}\right] \\
&= \left[\sum_{i\in|l|} y_i \underline{\mathbf{A}}_i \cdot \mathbf{r} + \sum_{i\in|l|} y_i \cdot x_i - \left(\sum_{i\in|l|} y_i \cdot s_i \underline{\mathbf{A}}\overline{\mathbf{A}}^{-1}\right)\overline{\mathbf{A}} \cdot \mathbf{r}\right] \\
&= \left[\sum_{i\in|l|} y_i \cdot s_i \underline{\mathbf{A}} \cdot \mathbf{r} + \sum_{i\in|l|} y_i \cdot x_i - \sum_{i\in|l|} y_i \cdot s_i \underline{\mathbf{A}} \cdot \mathbf{r}\right] \\
&= \left[\sum_{i\in|l|} y_i \cdot x_i\right] = [\langle \mathbf{y}, \mathbf{x} \rangle].
\end{aligned}$$

Notice that when decrypting an encrypted message $\mathbf{x}$, only the inner product of $\mathbf{x}$ and the key $\mathbf{y}$ in the group $\mathbb{G}$ is revealed.

5.2 Security

Theorem 5.2.1. *Under the $\mathcal{U}_{k+1,k}$-MDDH assumption, the scheme IP_{MDDH} is s-IND-FE-CPA secure. In particular, for an adversary A there exists an adversary B with roughly the same running time such that*

$$\mathsf{Adv}^{\mathsf{s\text{-}ind\text{-}fe\text{-}cpa}}_{\mathcal{FE},\mathsf{A}}(\lambda) = Adv^{mddh}_{GGen,B}(\lambda).$$

Proof:
For the proof consider the three games of figure 5.2.

```
Game $G_0/G_1/G_2$
$(\mathbb{G}, q, g) \xleftarrow{\$} GGen(1^\lambda)$
$\mathbf{A} \leftarrow \mathcal{U}_{k+1,k}$
$\mathbf{s}' = \mathbf{x}_1 - \mathbf{x}_0$
for $i = 1, ..., l$ :
    $\mathbf{R}_i \xleftarrow{\$} \mathbb{Z}_q^{1\times k}$
    $[\underline{\mathbf{A}}_i] = [s'_i\underline{\mathbf{A}} + \mathbf{R}_i\overline{\mathbf{A}}]$
$mpk = \begin{bmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}}_1 \\ \vdots \\ \underline{\mathbf{A}}_l \end{bmatrix}, \quad \mathbf{R} = \begin{pmatrix} \mathbf{R}_1 \\ \vdots \\ \mathbf{R}_l \end{pmatrix} \in \mathbb{Z}_q^{l\times k}$
$V \leftarrow \emptyset$
$\beta \xleftarrow{\$} \{0,1\}$
$\hat{\beta} \xleftarrow{\$} A^{\textsc{KeyDer}(\cdot),\textsc{Encrypt}(\cdot,\cdot)}(mpk)$
If $\exists \mathbf{y} \in V$ s.t. $\langle \mathbf{y}, \mathbf{x}_0\rangle \neq \langle \mathbf{y}, \mathbf{x}_1\rangle$:
    return $false$
return $\hat{\beta} = \beta$
```

```
Oracle KeyDer($\mathbf{y}$)
$V \leftarrow V \cup \{\mathbf{y}\}$
$sk_y = \mathbf{y}^T\mathbf{R} = \sum_{i\in|l|} y_i\mathbf{R}_i$
return $sk_y$

Oracle Encrypt($\mathbf{x}_0, \mathbf{x}_1$)
$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^k$
$\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^{k+1}$
$[\mathbf{ct}_0] = [\overline{\mathbf{A}} \cdot \mathbf{r}]$   $//G_0$
$[\mathbf{ct}_0] = [\overline{\mathbf{b}}]$   $//G_1, G_2$
for $i = 1, ..., l$ :
    $z_i \xleftarrow{\$} \mathbb{Z}_q$
    $[ct_i] = [s'_i(\underline{\mathbf{A}}_i \cdot \mathbf{r}) + \mathbf{R}_i(\overline{\mathbf{A}} \cdot \mathbf{r}) + x_{\beta,i}]$   $//G_0$
    $[ct_i] = [s'_i\underline{\mathbf{b}} + \mathbf{R}_i\overline{\mathbf{b}} + x_{\beta,i}]$   $//G_1$
    $[ct_i] = [z_i]$   $//G_2$
return $[\mathbf{Ct}] = [\mathbf{ct}_0, (ct_i)_{i\in|l|}]$
```

Figure 5.2: **Game** $G_0/G_1/G_2$

Notice that the master secret key will be $\mathbf{T} = \mathbf{s}'\mathbf{T}_A + \mathbf{R}$, where $\mathbf{T}_A = \underline{\mathbf{A}}\overline{\mathbf{A}}^{-1}$. Because of the restriction $\langle \mathbf{y}, \mathbf{x}_0\rangle = \langle \mathbf{y}, \mathbf{x}_1\rangle$ we have that $\mathbf{y} \in \{\mathbf{x}_1 - \mathbf{x}_0\}^\perp$. $\{\mathbf{x}_1 - \mathbf{x}_0\}$ is the space spanned by $\mathbf{x}_1 - \mathbf{x}_0$, where $\mathbf{x}_0$ and $\mathbf{x}_1$ are two predefined messages. Notice also that we choose $\mathbf{s}' = \mathbf{x}_1 - \mathbf{x}_0$, hence $sk_y = \mathbf{y}^T\mathbf{R}$ is simulated in the same way as $\mathbf{y}^T\mathbf{T}$.

By definition 2.2.11, G_0 is the s-IND-FE-CPA-game for the scheme IP_{MDDH}. Hence

$$Pr[\text{s-IND-FE-CPA}^A \implies 1] = Pr[G_0^A \implies 1].$$

G_1 only differs from G_0 in how $[\mathbf{ct}_0]$ and $[ct_i]$ are computed. In G_1 a random vector $\mathbf{b} \overset{\$}{\leftarrow} \mathbb{Z}_q^{k+1}$ is used in the encryption. We state the following lemma.

Lemma 5.2.2. *There exists an adversary B with roughly the same running time as A such that*

$$Adv(G_0^A, G_1^A) = Adv_{GGen,B}^{mddh}(\lambda).$$

G_2 only differs from G_1 in how $[ct_i]$ is computed. In G_2, $[ct_i]$ is just an element drawn uniformly random from the group. In G_1, $[ct_i] = [s_i'\underline{\mathbf{b}} + \mathbf{R}_i\overline{\mathbf{b}} + x_{\beta,i}]$. One can argue that the distribution of this element is also a uniformly random distribution. The vector $\mathbf{b}$ is a random vector, and multiplying this vector by a non-zero element makes the product also random. Hence $[ct_i]$ has the same distribution in G_2 as in G_1 and therefore

$$Pr[G_1^A \implies 1] = Pr[G_2^A \implies 1].$$

In G_2, $[\mathbf{Ct}] = [\overline{\mathbf{b}}, (z_i)_{i \in |l|}]$, which just consist of all random elements. Also $[\mathbf{Ct}]$ is independent of the challenge bit β. Hence

$$Pr[G_2^A \implies 1] = \frac{1}{2}.$$

Combining the equations derived above we obtain

$$\begin{aligned} Pr[s-IND-FE-CPA^A \implies 1] &= Pr[G_0^A \implies 1] \\ &= Pr[G_1^A \implies 1] + Adv_{GGen,B}^{mddh}(\lambda) \\ &= Pr[G_2^A \implies 1] + Adv_{GGen,B}^{mddh}(\lambda) \\ &= \frac{1}{2} + Adv_{GGen,B}^{mddh}(\lambda), \end{aligned}$$

which proves the theorem since $\mathsf{Adv}_{\mathcal{FE},\mathsf{A}}^{\text{s-ind-fe-cpa}}(\lambda) = \left| Pr[\text{s-IND-FE-CPA}^A \implies 1] - \frac{1}{2} \right| = Adv_{GGen,B}^{mddh}(\lambda)$. $\square$

Proof of Lemma 5.2.2:
Let B be an adversary against the $\mathcal{U}_{k+1,k}$-MDDH assumption. Adversary B inputs a matrix $[\mathbf{A}]$, where $\mathbf{A} \leftarrow \mathcal{U}_{k+1,k}$, and a vector $[\mathbf{b}] \in \mathbb{G}^{k+1}$. $[\mathbf{b}]$ is either from a random distribution or $[\mathbf{b}] = [\mathbf{A} \cdot \mathbf{r}]$ for a vector $\mathbf{r} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$. The adversary B needs to distinguish a real $\mathcal{U}_{k+1,k}$-MDDH distribution (game *REAL*) from a random distribution (game *RAND*).

```
Procedure $B(\mathcal{G}, [\mathbf{A}], [\mathbf{b}])$
$\mathbf{s}' = \mathbf{x}_1 - \mathbf{x}_0$
for $i = 1, ..., l$ :
    $\mathbf{R}_i \xleftarrow{\$} \mathbb{Z}_q^{1\times k}$
    $[\underline{\mathbf{A}}_i] = [s'_i\underline{\mathbf{A}} + \mathbf{R}_i\overline{\mathbf{A}}]$
$mpk = \begin{bmatrix} \overline{\mathbf{A}} \\ \underline{\mathbf{A}}_1 \\ \vdots \\ \underline{\mathbf{A}}_l \end{bmatrix}, \quad \mathbf{R} = \begin{pmatrix} \mathbf{R}_1 \\ \vdots \\ \mathbf{R}_l \end{pmatrix} \in \mathbb{Z}_q^{l\times k}$
$V \leftarrow \emptyset$
$\beta \xleftarrow{\$} \{0,1\}$
$\hat{\beta} \xleftarrow{\$} A^{\text{KeyDer}(\cdot),\text{Encrypt}(\cdot,\cdot)}(mpk)$
If $\exists \mathbf{y} \in V$ s.t. $\langle \mathbf{y}, \mathbf{x}_0\rangle \neq \langle \mathbf{y}, \mathbf{x}_1\rangle$:
    return $false$
return $\hat{\beta} = \beta$

Oracle $\text{KeyDer}(\mathbf{y})$
$V \leftarrow V \cup \{\mathbf{y}\}$
$sk_y = \mathbf{y}^T\mathbf{R} = \sum_{i\in|l|} y_i\mathbf{R}_i$
return $sk_y$

Oracle $\text{Encrypt}(\mathbf{x}_0, \mathbf{x}_1)$
$[\mathbf{ct}_0] = [\overline{\mathbf{b}}]$
$[ct_i] = [s'_i\underline{\mathbf{b}} + \mathbf{R}_i\overline{\mathbf{b}} + x_{\beta,i}]$
return $[\mathbf{Ct}] = [\mathbf{ct}_0, (ct_i)_{i\in|l|}]$
```

Figure 5.3: **Adversary** B

Notice that if B runs the $REAL$ game, the ENCRYPT simulation is the same as in G_0, from figure 5.2. Hence $Pr[REAL^B \implies 1] = Pr[G_0^A \implies 1]$. In the simulation above, the master secret key will be $msk = \mathbf{s}'\mathbf{T}_A + \mathbf{R}$. In both games the adversary will be provided with answers to the oracle queries of the KEYDER oracle. The answers of the KEYDER oracle are of the form $\mathbf{y}^T\mathbf{R}$. This will leak some information about $\mathbf{R}$. Since $\mathbf{A} \leftarrow \mathcal{U}_{k+1,k}$ the matrix $\mathbf{T}_A = \underline{\mathbf{A}}\overline{\mathbf{A}}^{-1}$ will be a $(1 \times k)$-matrix where each entry is a random element from $\mathbb{Z}_q$. Hence the master secret key is still random even if some information about $\mathbf{R}$ is leaked. If B runs the $RAND$ game, the ENCRYPT simulation is the same as in G_1, hence $Pr[RAND^B \implies 1] = Pr[G_1^A \implies 1]$. This leads to

$$\begin{aligned} Adv_{GGen,B}^{mddh}(\lambda) &= |Pr[REAL^B \implies 1] - Pr[RAND^B \implies 1]| \\ &= |Pr[G_0^A \implies 1] - Pr[G_1^A \implies 1]| \\ &= Adv(G_0^A, G_1^A). \quad \square \end{aligned}$$

Bibliography

[1] Abdalla, M., Bourse, F., De Caro, A. & Pointcheval, D. (2015) *Simple Functional Encryption Schemes for Inner Products.* In: Katz J. (eds) Public-Key Cryptography – PKC 2015. PKC 2015. Lecture Notes in Computer Science, vol 9020. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-46447-2_33

[2] Escala, A., Herold, G., Kiltz, E., Ràfols, C. & Villar, J. (2013) *An Algebraic Framework for Diffie-Hellman Assumptions.* In: Canetti R., Garay J.A. (eds) Advances in Cryptology – CRYPTO 2013. CRYPTO 2013. Lecture Notes in Computer Science, vol 8043. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40084-1_8

[3] Heggen, V. (2021) *Matrix Diffie-Hellman Assumptions.* [Project thesis, NTNU]

[4] Kiltz, E. (2021) *Cryptographic Protocols.* Ruhr-Universitat Bochum.

[5] Elgamal, T. (1985). *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.* IEEE Transactions on Information Theory, VOL. IT-31, NO. 4. https://doi.org/10.1109/TIT.1985.1057074.