We introduce Bilinear Maps and two of its applications: NIKE, Non-Interactive Key Exchange; and IBE, Identity Based Encryption.

1 Diffie-Hellman Key Exchange

![Diffie-Hellman Key Exchange Diagram]

Fig 1 illustrates Diffie-Hellman key exchange. Alice and Bob each has a private key (a and b respectively), and they want to build a shared key for symmetric encryption communication. They can only communicate over an insecure link, which is eavesdropped by Eve. So Alice generates a public key $A$ and Bob generates a public key $B$, and they send their public key to each other at the same time. Then Alice generates the shared key $K$ from $a$ and $B$, and likewise, Bob generates the shared key $K$ from $b$ and $A$. And we have $\forall$ PPT Eve, $Pr[k = Eve(A, B)] = neg(k)$, where $k$ is the length of $a$.

1.1 Discussion 1

Assume that $\forall(g, p)$, and $a_1, b_1 \xleftarrow{} Z_p^*$, and $a_2, b_2, r \xleftarrow{} Z_p^*$, we have $(g^{a_1}, g^{b_1}, g^{a_1b_1}) \xleftarrow{} (g^{a_2}, g^{b_2}, g^r)$. How to apply this to Diffie-Hellman Key Exchange?

Make $A = g^a$, $B = g^b$, $K = A^b = g^{ab}$, and $K = B^a = g^{ab}$.

1.2 Discussion 2

How does Diffie-Hellman Key Exchange imply Public Key Encryption?

Alice $pk = A$, $sk = a$, $Enc(pk, m \in \{0, 1\})$.

Bob $b, r \xleftarrow{} Z^*_p (g^b, mA^b + (1 - m)g^r)$
Alice $Dec(sk, (c_1, c_2))$
\[ c_1^a = c_2 \]

\section{Bilinear Maps}

\textbf{Definition 1} \textit{Bilinear Maps}

\textit{Bilinear Maps} is $(G, P, G_T, g, e)$, where $e$ is an efficient function $G \times G \rightarrow G_T$ such that

- if $g$ is generator of $G$, then $e(g, g)$ is the generator of $G_T$.
- $\forall a, b \in \mathbb{Z}_p$, we have $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

\subsection{Discussion 1}

How does Bilinear Maps apply to Diffie-Hellman?

Make $A = g^a$, $B = g^b$, and $T = g^{ab}$, then Diffie-Hellman has $e(A, B) = e(g, T)$.

\section{Tripartite Diffie-Hellman}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{tripartite_diffie_hellman.png}
\caption{Tripartite Diffie-Hellman Key Exchange}
\end{figure}
Fig 3 illustrates Tripartite Diffie-Hellman key exchange. \( a, b, \) and \( c \) are private key of Alice, Bob, and Carol, respectively. They use \( g^a, g^b, g^c \) as public key, and the shared key \( K = e(g, g)^{abc} \).

Formally, we have
\[
\begin{align*}
a, b, c & \xleftarrow{\$} Z_p^*, r \xleftarrow{\$} Z_p^* \\
A &= g^a, B = g^b, C = g^c \\
K &= e(g, g)^{abc}
\end{align*}
\]

4 IBE: Identity-Based Encryption

IBE contains four steps: \textit{Setup}, \textit{KeyGen}, \textit{Enc}, and \textit{Dec}. We illustrate it in Figure 4. In first step, Key authority get a Master Public Key (MPK) and Master Signing Key (MSK) from \textit{Setup}(1^k). Then a user with an ID (in this example, “Mike”), sends his ID to the key authority. The key authority generates the Signing Key of Mike with \textit{KeyGen}(MSK, ID) ans sends it back. Another use, Alice, wants to send an encrypted message to Mike. She only has MPK and Mike’s ID. So she encrypts the message with \( c = Enc(MPK, ID = Mike, m) \), and sends the encrypted message \( c \) to Mike. Mike decodes \( c \) with \( m = Dec(c, SK_{Mike}) \). Notice that Alice never need to know Mike’s public key. She only needs to remember MPK and other people’s IDs.

![Figure 3: Identity-Based Encryption](image)

Formally, we have

\[
Pr \left[ \begin{array}{l}
(MPK, MSK) \leftarrow Setup(1^k), \\
SK_{ID} \leftarrow KeyGen(MSK, ID), \\
c \leftarrow Enc(MPK, ID, m), \\
m \leftarrow Dec(SK_{ID}, c)
\end{array} \right] = 1
\]
4.1 Security Descriptions

We have different security descriptions for IBE, as discussed in this section.

4.1.1 CCA1

<table>
<thead>
<tr>
<th>Challenger</th>
<th>Adversary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(MPK, MSK) \leftarrow Setup(1^k)$</td>
<td>$\xrightarrow{MPK}$</td>
</tr>
<tr>
<td>$SK_{ID_1} \leftarrow KeyGen(MSK, ID_1)$</td>
<td>$\xrightarrow{SK_{ID_1}}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$SK_{ID_i} \leftarrow KeyGen(MSK, ID_i)$</td>
<td>$\xrightarrow{SK_{ID_i}}$</td>
</tr>
<tr>
<td>$b \overset{$}{\leftarrow} {0,1}$, $c^* = Enc(MPK, ID^*, m_b)$</td>
<td>$c^*$</td>
</tr>
<tr>
<td>Output 1 if $b' = b$, otherwise 0</td>
<td>$b'$</td>
</tr>
</tbody>
</table>

4.1.2 CCA2

In CCA2, we allow adversary to send further queries after getting $c^*$.

<table>
<thead>
<tr>
<th>Challenger</th>
<th>Adversary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(MPK, MSK) \leftarrow Setup(1^k)$</td>
<td>$\xrightarrow{MPK}$</td>
</tr>
<tr>
<td>$SK_{ID_1} \leftarrow KeyGen(MSK, ID_1)$</td>
<td>$\xrightarrow{SK_{ID_1}}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$SK_{ID_q} \leftarrow KeyGen(MSK, ID_q)$</td>
<td>$\xrightarrow{SK_{ID_q}}$</td>
</tr>
<tr>
<td>$b \overset{$}{\leftarrow} {0,1}$, $c^* = Enc(MPK, ID^*, m_b)$</td>
<td>$c^*$</td>
</tr>
<tr>
<td>$\xleftarrow{ID_{q+1}}$</td>
<td>$ID_{q+1}$</td>
</tr>
<tr>
<td>$SK_{ID_{q+1}} \leftarrow KeyGen(MSK, ID_{q+1})$</td>
<td>$\xrightarrow{SK_{ID_{q+1}}}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$SK_{ID_{q'}} \leftarrow KeyGen(MSK, ID_{q'})$</td>
<td>$\xrightarrow{SK_{ID_{q'}}}$</td>
</tr>
<tr>
<td>Output 1 if $b' = b$, otherwise 0</td>
<td>$b'$</td>
</tr>
</tbody>
</table>

4.1.3 Selective Security

In selective security, the adversary sends $ID^*$ before everything.
4.2 Discussion 1

How does Bilinear Maps apply to IBE?
Given Bilinear Maps: \( (G, P, G_T, g, e) \), we have

1. \((G, P, G_T, g, e) \leftarrow \text{Setup}(1^k)\)

2. \(s \leftarrow Z_p^*\), and \(H_1 : \{0,1\}^* \rightarrow G, \ H_2 : G_T \rightarrow \{0,1\}^n\)

3. \(MPK = (G, g^s, H_1, H_2)\), and \(MSK = (s)\)

Let’s look at how we construct each function in IBE.

\(\text{KeyGen}(s, ID)\):

1. Output \(SK_{ID} = (H_1(ID))^s\)

\(\text{Enc}(MPK, ID, m)\):

1. \(r \leftarrow Z_p^*\)

2. \(c_1 = g^r\)

3. \(c_2 = m \oplus H_2(e(A, H_1(ID)^r)), \text{ where } A = g^s\)

4. Output \((c_1, c_2)\)
$Dec(SK_{ID}, (c_1, c_2))$:

1. Get $e(A, H_1(ID)^r) = e(H_1(ID)^s, c_1) = e(SK_{ID}, c_1)$

2. Get $m = c_2 \oplus H_2(e(A, H_1(ID))^r)$

**Proof.** To prove this, we use a hybrid argument. Assume we have two oracles with exact random functions, denoted as $O_{H_1}$ and $O_{H_2}$. One can request a random string from them with a query ID. The random strings are denoted as $H_1(ID)$ and $H_2(ID)$, respectively. These two oracles keep track of query IDs and corresponding responses. If a query ID was seen before, they return the exact same response corresponding to it. If not, they generate a random string, correspond the string to the ID, and return the string.

We first define $\mathcal{H}_0$, in which $H_1(ID)$ and $H_2(ID)$ are generated by the oracles. We use the construction described above.

\[
\begin{align*}
\text{Challenger} & \quad \text{Adversary} \\
& \quad \begin{array}{c}
G, ID^* \\
\xleftarrow{s} \\
\xleftarrow{g} \\
\xleftarrow{O_{H_1}} \\
\xleftarrow{O_{H_2}} \\
\xleftarrow{ID_1} \\
SK_{ID_1} & \leftarrow KeyGen(s, ID_1) \\
& \quad \xrightarrow{SK_{ID_1}} \\
& \quad \vdots \\
& \quad \vdots \\
& \quad \xleftarrow{ID_q} \\
SK_{ID_q} & \leftarrow KeyGen(s, ID_q) \\
& \quad \xrightarrow{SK_{ID_q}} \\
& \quad \xrightarrow{m_0, m_1} \\
b & \xleftarrow{s} \{0, 1\}, \ c^* = Enc(MPK, ID^*, m_0) \\
& \quad \xrightarrow{c^*} \\
& \quad \xrightarrow{b'} \\
\end{array}
\end{align*}
\]

Then we discard oracle’s $H_1$, and use $H_1(ID) = g^{\alpha_{ID}}$, where $\alpha_{ID} \xleftarrow{\$} Z_p$. We denote this as $\mathcal{H}_1$.

Then we change $SK_{ID}$ to $SK_{ID} = (H_1(ID))^s = (g^{\alpha_{ID}})^s$. We denote this as $\mathcal{H}_2$.

We have Bilinear Decision Diffie-Hellman (DDH). If $\mathcal{H}_2$ breaks DDH, then $\mathcal{H}_0$ can as well.

In DDH, we have $(g^a, g^b, g^c, e(g, g)^{abc}) \xleftarrow{\$} (g^a, g^b, g^c, e(g, g)^r)$. We denote $A = g^a$, $B = g^b$, $C = g^c$. And in $\mathcal{H}_2$, we have $A = g^s, B = H_1(ID^s), C = c_1 = g^r$. And in $c_2 = m \oplus H_2(e(g^s, H_1(ID^s))^r)$, we have $T = H_2(e(g^s, H_1(ID^s))^r) = e(g, g)^{abc}$. 

\[\blacksquare\]