We continue the discussion of public key encryption. Last time, we studied Hash Proof Systems as a mean of NIZK proofs for achieving a CCA-1 secure scheme. Now, we extend that to achieve CCA-2 security; this way, we present the Cramer-Shoup encryption scheme which was the first efficient scheme provably secure against adaptive chosen ciphertext attack. Then, we switch to pairing-based cryptography and look into identity based encryption.

1 CCA-Secure Public Key Encryption

1.1 CCA-1

Last time we studied a practical PKE scheme which is somewhat similar to ElGamal encryption:

\( G \) is a group of primer order \( q \)

PK: generators \( g_1, g_2 \in G \) and \( h = g_1^{x_1} g_2^{x_2} \)

SK: \( x_1, x_2 \)

Enc(\( m \)): \( u_1 = g_1^r, u_2 = g_2^r, v = h^r, w = v.m \)

\( c = (u_1, u_2, w, \pi) \), where \( \pi \) is a ”proof” that \( \log_{g_1} u_1 = \log_{g_2} u_2 \)

Dec(SK,\( c \)): check \( \pi \), \( m \leftarrow w/(u_1^{x_1} u_2^{x_2}) \)

At first, we considered the idea of implementing \( \pi \) using the Fiat-Shamir Heuristic. Then, we gave a construction without random oracles using a Hash Proof System for \( \pi \):

HPK: \( c = g_1^{y_1} g_2^{y_2} \)

HSK: \( y_1, y_2 \)

\( \pi: \text{c}^t \)

verify \( \pi \): \( \pi = \pi' \equiv u_1^{y_1} u_2^{y_2} \) (\( \pi' \) is the expected correct value of the proof)

If \( u_1 = g_1^{r_1}, u_2 = g_2^{r_2} \), and say \( g_2 = g_1^t \), then:

\[
\begin{pmatrix}
1 & t \\
\frac{r_1}{r_2t} & \frac{r_2}{r_2t}
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2
\end{pmatrix}
= 
\begin{pmatrix}
\log_{g_1} c \\
\log_{g_1} \pi'
\end{pmatrix}
\]

This gives CCA-1 security.
1.2 CCA2

For CCA-2 security we need 2-wise pseudorandom property for π.

- HPK: \( c = g_1^{y_1} g_2^{y_2}, d = g_1^{z_1} g_2^{z_2} \)
- HSK: \( y_1, y_2, z_1, z_2 \)
- \( \pi: c^d \alpha, \) where \( \alpha = H(u_1, u_2, w) \) and \( H \) is a CRHF
  (actually, TCR suffices for \( H \), but one should be careful because although the first input to \( H \) is "random", \( w \) depends on \( M \) chosen by \( A \); still, security could be shown)
- Verify: \( \pi = \pi' \equiv u_1^{y_1+\alpha z_1} u_2^{y_2+\alpha z_2}, \) where \( \alpha \) is computed in the same way
  (Note that if \( u_1 = g_1^{r_1}, u_2 = g_2^{r_2} \), and say \( g_2 = g_1^t \), then:
  \[
  (\ast) \quad u_1^{y_1+\alpha z_1} u_2^{y_2+\alpha z_2} = g_1^{r_1(y_1+\alpha z_1)} g_2^{r_2(y_2+\alpha z_2)} = g_1^{r_1 y_1 + r_1 \alpha z_1 + tr_2 y_2 + tr_2 \alpha z_2}
  \]

Consider two "invalid" ciphertexts:

\[
\begin{aligned}
  u_1 &= g_1^{r_1}, u_2 = g_2^{r_2}, w, \pi, r_1 \neq r_2 \\
  \bar{u}_1 &= g_1^{\bar{r}_1}, \bar{u}_2 = g_2^{\bar{r}_2}, \bar{w}, \bar{\pi}, \bar{r}_1 \neq \bar{r}_2
\end{aligned}
\]

for simplicity, we could assume \( t \neq 0, \alpha \neq \bar{\alpha}, r_1 \neq r_2, \bar{r}_1 \neq \bar{r}_2 \); then, define \( M \) to be

\[
M = \begin{pmatrix}
  1 & t & 0 & 0 \\
  0 & 0 & 1 & t \\
  r_1 & tr_2 & r_1 \alpha & tr_2 \alpha \\
  \bar{r}_1 & \bar{t}r_2 & \bar{r}_1 \bar{\alpha} & \bar{t}r_2 \bar{\alpha}
\end{pmatrix}
\]

From \((\ast)\) and the definitions of \( c \) and \( d \), we get:

\[
M \begin{pmatrix}
  y_1 \\
  y_2 \\
  z_1 \\
  z_2
\end{pmatrix} = \begin{pmatrix}
  1 & t & 0 & 0 \\
  0 & 0 & 1 & t \\
  r_1 & tr_2 & r_1 \alpha & tr_2 \alpha \\
  \bar{r}_1 & \bar{t}r_2 & \bar{r}_1 \bar{\alpha} & \bar{t}r_2 \bar{\alpha}
\end{pmatrix} \begin{pmatrix}
  y_1 \\
  y_2 \\
  z_1 \\
  z_2
\end{pmatrix} = \begin{pmatrix}
  \log_{g_1} c \\
  \log_{g_1} d \\
  \log_{g_1} \pi' \\
  \log_{g_1} \bar{\pi}'
\end{pmatrix}
\]

Claim: matrix \( M \) is non-singular (one could verify \( \det M = t^2*(r_2-r_1)(\bar{r}_2-\bar{r}_1)(\alpha-\bar{\alpha}) \neq 0 \))

So, if \( y_1, y_2, z_1, z_2 \) are chosen at random, the log values will be uniform and random;
\( \log_{g_1} \bar{\pi}' \) will be independent and random when conditioned on the other three log values.

2 Pairing Based Cryptography

The Discrete Logarithm (DL) is a nice problem that is widely used in cryptography. People have looked into number of groups in which the problem appears to be hard but computations could be done efficiently. Although there is no algorithm for computing DL in polynomial time, for \( \mathbb{Z}_p^* \) there are so-called "index calculus methods" which compute discrete logs in time \( exp(c(\log p)^{1/3}(\log \log p)^{2/3}) \); whereas the brute-force attack uses \( \approx \sqrt{q} \) steps for a subgroup of order \( q \). So, one would want to use \( q \geq 2^{160} \) to avoid brute-force and \( p \geq 2^{1024} \) to avoid sub-exponential attacks. An advantage of using elliptic curves is that there is no known sub-exponential attacks for the DL problem.
2.1 What is an elliptic curve?

An elliptic curve is defined by an equation \( y^2 = f(x) \) where \( f(x) \) is a cubic polynomial with no repeated roots. The “curve” consists of the points \( \{(x, y) : y = f(x)\} \cup \{O\} \), where \( O \) is the point at infinity. When working over \( \mathbb{R} \), for two points \( P \) and \( Q \) on the curve, \( P + Q \) is equal to \( R \) such that \( P, Q, \) and \( R \) lie on the same line; we make relations between the coordinates and transfer those when working in a field. It is a non-trivial fact that this is an abelian group. Hasse’s theorem states that \(|\#\text{points on the curve} - p + 1| \leq 2\sqrt{p}\) when working over field \( \mathbb{F}_p \).

The original motivation for Elliptic Curve Cryptography was to work with group that has more compact representation. An initial product of Certicom Corp (one of co-founders of which is S. Vanstone) was based on a “special” elliptic curve, called a supersingular curve, due to the nice implementation features. But, unfortunately, it was insecure.

2.2 Pairings

Let \( E \) be an elliptic curve over \( \mathbb{Z}_p \).

**Definition** A pairing is a function \( e : E \times E \to \mathbb{F}_p^* \) (for some \( k \)) and has the following properties:

- bilinear
  \[
  e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q)
  
  e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2)
  \]
- non-degenerate: not everything maps to 1

- if \( P \) is a point on \( E \) of prime order \( q \), due to bilinearity \( e(aP, bP) = e(P, P)^{ab} \) and \( e(P, P) = \gamma \), where \( \gamma \) has order \( q \)
- for supersingular curve a pairing exists with \( k = 1 \) or \( 2 \) and \( e \) is easy to compute

**MOV reduction**

\[ \text{given: } P \text{ and } Q = xP, \text{ we can find } x \text{ as follows: } e(P, Q) = e(P, xP) = e(P, P)^x = \gamma^x \]
\[ \Rightarrow \log_p \gamma = \log_p e(P, Q), \text{ so } x \text{ could be computed using subexponential methods if } k \text{ is small} \]

pairings can be useful too
- we can choose elliptic curbes w/ pairings into \( \mathbb{F}_p^* \) for ”small k”
- Joux observed that the Diffie-Hellman key exchange protocol could be extended to 3-way key exchange protocol (both protocols are single round):
  
  - In the original Diffie-Hellman key exchange idea, \( A \) chooses \( x \) and sends \( g^x \), and \( B \) chooses \( y \) and sends \( g^y \). Then, each of them could compute \( g^{xy} = (g^y)^x = (g^x)^y \).
    However, any eavesdropper has to compute \( \text{CDH}(g, g^x, g^y) \) in order to obtain the key used by \( A \) and \( B \)
  
  - Using pairings: \( A : xP, \quad B : yP, \quad C : zP \)
    and each party could compute \( \gamma^{xy} = e(yP, zP)^x = e(xP, zP)^y = e(xP, yP)^z \)
    The security of this protocol is based on the BDH assumption, a natural extension of the CDH assumption, which is defined in the next chapter.
2.3 Bilinear Diffie Hellman Assumption

Given prime order groups and a pairing \( e: G_1 \times G_1 \to G_2 \), where \( |G_1| = |G_2| = q \) and \( e(P, P) = \gamma \), the bilinear versions of the CDH and DDH assumptions are as follows:

**Bilinear Diffie Hellman Assumption (BDH):**

Given \( P, xP, yP, zP \), it is hard to compute \( \gamma^{xyz} \)

**Decisional Bilinear Diffie Hellman Assumption (DBDH):**

The decisional version says that \( (P, xP, yP, zP, \gamma^{xyz}) \approx_c (P, xP, yP, zP, \gamma^c) \)

**Fact:** DDH in \( G_1 \) is easy

Given \( P, xP, yP, zP \), the question whether \( z \approx xy \) could easily be answered by checking \( e(xP, yP) \approx e(P, zP) \)

**Fact:** BDH \( \Rightarrow \) CDH in \( G_1 \) & CDH in \( G_2 \)

**Proof:**

- assume we can break CDH in \( G_1 \)
  - then given \( P, xP, yP, zP \), one could compute \( \gamma^{xy} = e(xyP, zP) \), where \( xyP = CDH(P, xP, yP) \)

- assume we can break CDH in \( G_2 \)
  - then given \( P, xP, yP, zP \), here is how to break BDH: \( \gamma^{xy} = e(xP, yP), \gamma^z = e(P, zP) \)
  - and, in the end, \( \gamma^{xyz} = CDH(\gamma, \gamma^{xy}, \gamma^z) \)

Two interesting results: Maurer showed an example of groups where CDH=DL; Joux extended that line of work and showed examples where CDH=DL, but DDH is easy

2.4 Identity Based Encryption Scheme

Identity Based Encryption (IBE) was proposed by Shamir in 1984. Generally speaking, it provides an encryption mechanism where one’s identity, e.g. bob@company.com, is used as his public key. The private keys are generated by a so called Secret Key Authority (SKA). Bob obtains his private key by contacting the SKA and authenticating himself. Although the model was proposed a long time ago, and it was not until the 2001 paper by Boneh and Franklin that a formal security model and a practical implementation were proposed.

An IBE is specified by the following four PPT algorithms:

- \( KeyGen() = (MPK, MSK) \), that is a master PK and a master SK
- \( SKExtract(MSK, ID) = SK_{ID} \) - that’s ID’s personal secret key
- \( E(ID, M) = C \)
- \( D(SK_{ID}, C) = M \)

The identity \( ID \) could be any binary string, \( ID \in \{0,1\}^* \), and the message and the cyphertext belong to the corresponding spaces \( (M \in \mathcal{M}, C \in \mathcal{C}) \).
Semantic Security (IND-ID-CPA):

An IBE scheme is semantically secure if any PPT adversary $A$ has a negligible advantage in the following game:

- **Setup**: The challenger takes a security parameter and runs the $KeyGen$ algorithm. It send the adversary the $MPK$ and keeps the $MSK$ to itself.

- **Phase 1**: The challenger answers extraction queries for $IDs$ chosen by $A$:
  - for each "extract: $\tilde{I}D$" the adversary sends
  - the challenger replies with $SKExtract(MSK, \tilde{I}D)$

- **Challenge**: $A$ sends "encrypt: $I$D, $M_0$, $M_1$. Then, the challenger chooses a random bit ($b \leftarrow R 0, 1$), encrypts $M_b$ ($c \leftarrow E(ID, M_b)$), and sends back $c$.

- **Phase 2**: The adversary asks more extract queries which are answered as before.

- **Guess**: The adversary outputs a guess $\hat{b}$

Note that the adversary is not allowed to extract $SK_{ID}$ and the advantage in the above game in defined as usual to be $Adv = |Pr[b = \hat{b}] - \frac{1}{2}|$.

CCA-Security (IND-ID-CCA):

The attack game above is enhanced to allow decryption queries in phase 1 and 2:
- $A$ could send "decrypt: $\tilde{I}D, c"$
- the challenger responds with $D(SK_{\tilde{I}D}, c)$, where $SK_{\tilde{I}D} = SKExtract(MSK, \tilde{I}D)$

The additional constraint is that $A$ can’t ask "decrypt: $I$D, c" in phase 2.

A silly observation: no decryption queries are needed in phase 2 for $\tilde{I}D \neq I$D.