Other Public-Key Cryptosystems

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

Audio/Video recordings of this lecture are available at:
http://www.cse.wustl.edu/~jain/cse571-11/
1. Diffie-Hellman Key Exchange
2. ElGamal Cryptosystem
3. Elliptic Curve Arithmetic
4. Elliptic Curve Cryptography
5. Pseudorandom Number Generation using Asymmetric Cipher

Diffie-Hellman Key Agreement

- Allows two party to agree on a secret key using a public channel
- A selects q=large prime, and $\alpha$=a primitive root of q
- A selects a random # $X_A$, B selects another random # $S_B$

\[
X_A, \alpha, q \\
Y_A = \alpha^{X_A} \mod q \\
Y_{AB} = Y_B^{X_A} \mod q
\]

\[
\alpha, q, Y_A \\
Y_B = \alpha^{X_B} \mod q \\
Y_{AB} = Y_A^{X_B} \mod q
\]

\[
Y_{AB} = g^{X_A \times X_B} \mod q
\]

- Eavesdropper can see $Y_A$, $\alpha$, q but cannot compute $X_A$
- Computing $X_A$ requires discrete logarithm - a difficult problem
Diffie-Hellman (Cont)

- Example: $\alpha=5$, $q=19$
  - A selects 6 and sends $5^6 \mod 19 = 7$
  - B selects 7 and sends $5^7 \mod 19 = 16$
  - A computes $K = 16^6 \mod 19 = 7$
  - B computes $K = 7^7 \mod 19 = 7$

- Preferably $(q-1)/2$ should also be a prime.
- Such primes are called safe prime.
Man-in-Middle Attack on Diffie-Hellman

- Diffie-Hellman does not provide authentication

- $X$ can then intercept, decrypt, re-encrypt, forward all messages between Alice & Bob

- You can use RSA authentication and other alternatives
ElGamal Cryptography

- Public-key cryptosystem related to D-H
- Uses exponentiation in a finite (Galois)
- Security based difficulty of computing discrete logarithms
- $X_A$ is the private key, $\{\alpha, q, Y_A\}$ is the public key

\[
\begin{align*}
X_A, \alpha, q \\
Y_A &= \alpha^{X_A} \mod q \\
K &= (C_1)^{X_A} \\
M &= C_2 K^{-1} \mod q
\end{align*}
\]

\[
\begin{align*}
\alpha, q, Y_A \\
C_1, C_2 \\
K &= \alpha^{kx_a} \\
K &= (Y_A)^k \mod q \\
C_1 &= \alpha^k \mod q \\
C_2 &= KM \mod q
\end{align*}
\]

- $k$ must be unique each time. Otherwise insecure.
ElGamal Cryptography Example

- Use field GF(19) q=19 and α=10
- Alice chooses $x_A=5$,
- Bob wants to send message $M=17$, selects a random key $k=6$

- $X_A=5$, $\alpha=10$, $q=19$
  - $Y_A = \alpha^{X_A} \mod q = 10^5 \mod 19 = 3$
  - $K = (C_1)^{X_A} = 11^5 \mod 19 = 7$
  - $K^{-1} = 7^{-1} = 11$
  - $M = C_2 K^{-1} \mod q = 5 \times 11 \mod 19 = 17$

- Select random key $k = 6$
  - $K = (Y_A)^k \mod q = 3^6 \mod 19 = 7$
  - $C_1 = \alpha^k \mod q = 10^6 \mod 19 = 11$
  - $C_2 = K M \mod q = 7 \times 17 \mod 19 = 5$
Elliptic Curve Cryptography

- Majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- Imposes a significant load in storing and processing keys and messages
- An alternative is to use elliptic curves
- Offers same security with smaller bit sizes
- Newer, but not as well analyzed
Elliptic Curves over Real Numbers

- An elliptic curve is defined by an equation in two variables $x$ & $y$,
  - $y^2 = x^3 + ax + b$
  - Where $x$, $y$, $a$, $b$ are all real numbers
  - $4a^3 + 27b^2 \neq 0$

- The set of points $E(a, b)$ forms an abelian group with respect to “addition” operation defined as follows:
  - $P+Q$ is reflection of the intersection $R$
  - $O$ (Infinity) acts as additive identity
  - To double a point $P$, find intersection of tangent and curve
  - Closure: $P+Q \in E$
  - Associativity: $P+(Q+R) = (P+Q)+R$
  - Identity: $P+O = P$
  - Inverse: $-P \in E$
  - Commutative: $P+Q = Q+P$
Elliptic Curve over Real Numbers (Cont)

- Slope of line PQ is:
  - \( D = \frac{(y_Q-y_P)}{(x_Q-x_P)} \)
- The sum \( R = P + Q \) is:
  - \( x_R = D^2 - x_P - x_Q \)
  - \( y_R = -y_P + D(x_P - x_R) \)
- \( P + P = 2P = R \)

\[
\begin{align*}
    x_R &= \left( \frac{3x_P^2 + a}{2y_P} \right)^2 - 2x_P \\
y_R &= \left( \frac{3x_P^2 + a}{2y_P} \right) (x_P - x_R) - y_P
\end{align*}
\]
Finite Elliptic Curves

Elliptic curve cryptography uses curves whose variables & coefficients are defined over GF

- **Prime curves**: $E_p(a, b)$ defined over $\mathbb{Z}_p$
  - Use integers modulo a prime
  - Easily implemented in software

- **Binary curves**: $E_{2^m}(a, b)$ defined over $\text{GF}(2^n)$
  - Use polynomials with binary coefficients
  - Easily implemented in hardware

Cryptography: Addition in elliptic = multiplication in Integer

- Repeated addition = Exponentiation
- Easy to compute $Q=P+P+\ldots+P=kP$, where $Q, P \in E$
- Hard to find $k$ given $Q, P$ (Similar to discrete log)
Finite Elliptic Curve Example

- \( E_p(a,b): y^2 = x^3 + ax + b \mod p \)
- \( E_{23}(1,1): y^2 = x^3 + x + 1 \mod 23 \)
- Consider only +ve \( x \) and \( y \)
- \( R = P + Q \)
  - \( x_R = (\lambda^2 - x_P - x_Q) \mod p \)
  - \( y_R = (\lambda(x_P - x_R) - y_P) \mod p \)
  - Where

\[
\lambda = \begin{cases} 
\frac{y_Q - y_P}{x_Q - x_P} \mod p & \text{if } P \neq Q \\
\frac{3x_P^2 + a}{2y_P} \mod p & \text{if } P = Q 
\end{cases}
\]

- Example: \( (3, 10) + (9, 7) \)
  - \( \lambda = \left( \frac{3(3^2) + 1}{2 \times 10} \right) \mod 23 = \frac{1}{4} \mod 23 = 6 \)
  - \( x_R = (6^2 - 3 - 3) \mod 23 = 7 \)
  - \( y_R = (6(3 - 7) - 10) \mod 23 = 12 \)
ECC Diffie-Hellman

- Select a suitable curve $E_q(a, b)$
- Select base point $G = (x_1, y_1)$ with large order $n$ such that $nG = O$
- A & B select private keys $n_A < n$, $n_B < n$
- Compute public keys: $Y_A = n_A G$, $Y_B = n_B G$
- Compute shared key: $K = n_A Y_B$, $K = n_B Y_A$
  - Same since $K = n_A n_B G$
- Attacker would need to find $K$, hard

```
+----------------+-----------------+----------------+  
| n_A, G, {q,a,b} | G, {q, a, b}, Y_A | n_B           | 
| Y_A = n_A G    |     Y_A         | Y_B = n_B G   |  
| Y_AB = n_A Y_B |     Y_B         | Y_AB = n_B Y_A|  
| Y_AB = n_A n_B G|               |               |  
+----------------+-----------------+----------------+
```
ECC Encryption/Decryption

- Several alternatives, will consider simplest
- Select suitable curve & point G
- Encode any message M as a point on the elliptic curve \( P_m \)
- Each user chooses private key \( n_A < n \)
- Computes public key \( P_A = n_A G \)
- Encrypt \( P_m : C_m = \{ kG, \ P_m + kP_b \} \), \( k \) random
- Decrypt \( C_m \) compute:
  \[
  P_m + kP_b - n_B (kG) = P_m + k (n_B G) - n_B (kG) = P_m
  \]
ECC Security

- Relies on elliptic curve logarithm problem
- Can use much smaller key sizes than with RSA etc
- For equivalent key lengths computations are roughly equivalent
- Hence for similar security ECC offers significant computational advantages

<table>
<thead>
<tr>
<th>Symmetric scheme (key size in bits)</th>
<th>ECC-based scheme (size of n in bits)</th>
<th>RSA/DSA (modulus size in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>112</td>
<td>512</td>
</tr>
<tr>
<td>80</td>
<td>160</td>
<td>1024</td>
</tr>
<tr>
<td>112</td>
<td>224</td>
<td>2048</td>
</tr>
<tr>
<td>128</td>
<td>256</td>
<td>3072</td>
</tr>
<tr>
<td>192</td>
<td>384</td>
<td>7680</td>
</tr>
<tr>
<td>256</td>
<td>512</td>
<td>15360</td>
</tr>
</tbody>
</table>
PRNG based on Asymmetric Ciphers

- Asymmetric encryption algorithms produce apparently random output
- Hence can be used to build a pseudorandom number generator (PRNG)
- Much slower than symmetric algorithms
- Hence only use to generate a short pseudorandom bit sequence (e.g., key)
PRNG based on RSA

- Micali-Schnorr PRNG using RSA
  - in ANSI X9.82 and ISO 18031
PRNG based on ECC

- Dual elliptic curve PRNG
  - NIST SP 800-9, ANSI X9.82 and ISO 18031
- Some controversy on security /inefficiency
- Notation: \( x(P) = x \) coordinate of \( P \). \( \text{lsb}_i(x) = i \) least sig bits of \( x \)

Algorithm

\[
\text{for } i = 1 \text{ to } k \text{ do}
\]
\[
\text{set } s_i = x(s_{i-1} P )
\]
\[
\text{set } r_i = \text{lsb}_{240} (x(s_i Q))
\]
\[
\text{end for}
\]
\[
\text{return } r_1, \ldots, r_k
\]

- Only use if just have ECC
1. Diffie-Hellman key exchange allows creating a secret in public based on exponentiation
2. ElGamal cryptography uses D-H
3. Elliptic Curve cryptography is based on defining addition of points on an elliptic curve in GF(p) or GF(2^n)
4. Public key cryptography (both RSA and ECC) can also be used to generate cryptographically secure pseudorandom numbers.
Homework 10

- Submit answers to problems 10.6 and 10.15