Public Key Cryptosystems
Based on Discrete Logarithm Problem

Guevara Noubir
http://www.cs.nwu.edu/~noubi/Courses/CSC232V44
Readings: Chapter 6, Sections 6.1-6.4, and 6.7.3

Outline

- El Gamal Cryptosystem
- Algorithms for Discrete Logarithm
- Implementation Issues
- Diffie-Hellman Problems and Key Establishment

Element for El Gamal Scheme

- Motivation of design
  - RSA is based on the difficulty of factoring large numbers
  - El Gamal scheme is based on the difficulty of computing discrete logarithms
- Order of an element of a multiplicative group \((G, \cdot)\):
  - \(\langle \alpha \rangle = \{\alpha^i : i \in \mathbb{Z}_n\} \); \(n\) is the order of \(\alpha\)
- Discrete Logarithm:
  - Given a multiplicative group \((G, \cdot)\), an element \(\alpha \in G\) with order \(n\) and an element \(\beta \in G\) s.t. \(\alpha^n = \beta\)
  - Question: find the unique integer \(0 \leq a < n\) s.t. \(\alpha^a = \beta\)
    - This is the same as finding \(\log_\alpha(\beta)\)
El Gamal Cryptosystem

- Cryptosystem
  - \( p \) prime s.t. \( \langle Z_p^*, \cdot \rangle \) is infeasible
  - Let \( \alpha \) be a primitive element
  - \( \beta \) = \( Z_p^* \); \( \gamma \) = \( Z_p^* \times Z_p^* \)
  - \( x = \langle (\rho, \alpha, \beta) \rangle; \beta = \alpha^x \mod p \)
  - Public: \( \rho, \alpha, \beta \); Private: \( x \)
  - For \( K = (\rho, \alpha, \beta) \) and a secret number \( k \in Z_p^* \)
    - \( e_k(\gamma, \gamma) = (y_1, y_2) \) s.t.
      - \( y_1 = \alpha^x \mod p \) and \( y_2 = x \beta \mod p \)
      - \( d_k(y_1, y_2) = k \)

Example:

- \( p = 2579; \)
  - \( \alpha = 2 \) (primitive element modulo \( p \))
  - \( \beta = 765 \mod 2579 = 949 \)
  - Encrypt \( x = 1299; k = 853 \)
    - \( y_1 = 2^{853} \mod 2579 = 435; y_2 = 1299*949^{853} \mod 2579 = 2396 \)
  - Decrypt \( (y_1, y_2) = (435, 2396) \)
    - \( x = 2396/435^{853} \mod 2579 = 1299 \)

Algorithms for Discrete Logarithm

- El Gamal cryptosystem would be insecure if we can compute the discrete logarithm
  - Discrete logarithm is believed to be infeasible if:
    - \( \rho \) is carefully chosen against known attacks
    - \( \alpha \) is a primitive element modulo \( p \)
    - Example: 300 digits, \( p \) has at least one "large" prime factor
Algorithms for Discrete Logarithm

- Assumption:
  - Multiplication in $G$ can be done in $O(1)$
- Exhaustive search: Cost = $O(n)$
- Shank’s Algorithm ($G, n, u, v$) [time-memory tradeoff]
  - For $i=0$ to $m-1$ do Compute $u^i$
  - Sort the $m$ pairs $(i, u^i)$ with respect to second coordinate ⇒ List $l_u$
  - For $i=0$ to $m-1$ do compute $|u^i|
  - Sort the $m$ pairs $(|u^i|, u^i)$ with respect to second coordinate ⇒ List $l_{|u|}$
  - Find a pair $(i, u^i) = l_u$ and a pair $(|i|, u^i) = l_{|u|}$ (Note: same $i$)
  - Complexity of Shank’s algorithm: Time? Space?

Algorithms for Discrete Logarithm

- Pollard Rho Discrete log
  - Time: $O(1)$
- Pohlig-Hellman Algorithm
  - Time: $O(n)$
- Index Calculus Method:
  - Specialized algorithm for $Z^*_p$ and primitive element $a$
  - Idea:
    - Use a factor base $B = \{r_1, r_2, \ldots, r_k\}$
    - Find the logarithms of the primes in the factor base
    - Use these logarithms to compute the logarithm of $b$
- Lower bound on generic algorithms:
  - Definition: a generic algorithm applies to any group and does not use any properties of the element of the group s.t. factorization, ...
  - Any generic algorithm for discrete logarithm has a lower bound of time complexity: $\Omega(n)$

Discrete Logarithm Algorithms in Practice

- Setup:
  - $G = \{Z^*_p, \cdot\}$, $p$ prime, $a$ primitive element modulo $p$
  - $G = \{Z^*_q, \cdot\}$, $p$ and $q$ prime ($p = 1 \mod q$), $a$ element having order $p$
  - $G = \{Z^*_q, \cdot\}$, $a$ primitive element modulo in $\mathbb{F}_p$
  - Elliptic Curves modulo a prime or over a finite field
- Lenstra and Verheul report to be secure until year 2020:
  - $p = 2^{256}$ for elliptic curves
  - Elliptic Curve implementations are the most efficient
  - Adequate for low power/resource devices such as PDAs and smartcards
- Latest challenge:
  - ECCM-108 over $F_{2^{108}}$ (solved in April 2000) using 9500 computers about 59 times the computation effort required to factor the RSA challenge RSA-512
**Diffie-Hellman Problems**

- Computational Diffie-Hellman
  - Given a multiplicative group \((G, \cdot)\), an element \(a \in G\) (order \(r\)), two elements \(a^r, a^{r'} \in <a>\)
  - Question: find \(a^{r'}\)
- Decisional Diffie-Hellman
  - Given a multiplicative group \((G, \cdot)\), an element \(a \in G\) (order \(r\)), three elements \(a^r, a^{r'}, a^{r''} \in <a>\)
  - Question: Is \(d = ac\)?
- Turing Reductions:
  - Decision Diffie-Hellman can be reduced to Computational Diffie-Hellman
  - Computational Diffie-Hellman can be reduced to Discrete Logarithm
- Computational Diffie-Hellman can be used to decrypt El Gamal ciphertext and vice versa

**Diffie-Hellman Key Exchange**

<table>
<thead>
<tr>
<th>Private: A</th>
<th>Public</th>
<th>Private: B</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>(p): prime number, (\alpha): primitive element of (\mathbb{Z}_p^*)</td>
<td>(y)</td>
</tr>
<tr>
<td>compute: (\alpha^r \mod p)</td>
<td>compute: (\alpha^{r'} \mod p)</td>
<td>receive: (\alpha^r \mod p)</td>
</tr>
<tr>
<td>receive: (\alpha^{r'} \mod p)</td>
<td>receive: (\alpha^{r''} \mod p)</td>
<td></td>
</tr>
</tbody>
</table>

Compute shared key: \((\alpha^r)^y \mod p\)

- Based on the difficulty of computational Diffie-Hellman
- Works also in extension Galois fields: \(\text{GF}(p^t), \ldots\)