## Public Key Cryptosystems Based on Discrete Logarithm Problem

Guevara Noubir
http://www.ccs.neu.edu/home/noubir/Courses/CSG252/F04

Textbook: "Cryptography: Theory and Applications",

Douglas Stinson, Chapman & Hall/CRC Press, 2002

Reading: 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, .):
  - $<\alpha> = \{\alpha^i : 0 \leq i \leq n-1\}$; $n$ is the order of $\alpha$
- Discrete Logarithm:
  - Given a multiplicative group ($G$, .), an element $\alpha \in G$ with order $n$, and an element $\beta \in G$ s.t. $\alpha^a = \beta$
  - Question: find the unique integer $0 \leq a \leq n-1$ s.t. $\alpha^a = \beta$
    - This is the same as finding $\log_\alpha(\beta)$

## El Gamal Cryptosystem

- Cryptosystem
  - $p$ prime s.t. ($Z_p^*$, .) is infeasible
  - Let $\alpha$ be a primitive element
  - $\mathcal{P} = Z_p^*$; $\mathcal{C} = Z_p^* \times Z_p^*$;
  - $\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \bmod p\}$
  - Public: $p, \alpha, \beta$; Private: $a$;
  - For K = ($p, \alpha, a, \beta$) and a secret number $k \in Z_p$;
  - $e_K(x, k) = (y_1, y_2)$ s.t.
    - $y_1 = \alpha^k \bmod p$ and $y_2 = x\,\beta^k \bmod p$;
  - $d_K(y_1, y_2)$ = ?



## Example:

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



## 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:
  - $p$ is carefully chosen against known attacks
  - $\alpha$ is a primitive element modulo $p$
  - Example: 300 digits, $p$-1 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, $\alpha$, $\beta$) [time-memory tradeoff]
  - $m \leftarrow \lceil \sqrt{n} \rceil$
  - **For** j=0 **to** m-1 **do** Compute $\alpha^{mj}$
  - **Sort** the $m$ pairs ($j$, $\alpha^{mj}$) with respect to second coordinate $\Rightarrow$ List $L_1$
  - **For** i=0 **to** m-1 **do** compute $\beta\alpha^{-i}$
  - **Sort** the $m$ pairs ($j$, $\beta\alpha^{-i}$) with respect to second coordinate $\Rightarrow$ List $L_2$
  - **Find** a pair ($j$, $y$) $\in L_1$ and a pair ($i$, $y$) $\in L_2$ [Note: same $y$]
  - $\text{Log}_\alpha\beta = (mj+i) \bmod n$
- Complexity of Shank's algorithm: Time? Space?



## Algorithms for Discrete Logarithm

- Pollard Rho Discrete log
  - Time: $O(\sqrt{n})$
- Pohlig-Hellman Algorithm
  - Time: $O(\max(c_i\sqrt{q_i}))$ s.t. $n = q_1^{c1} \ldots q_k^{ck}$
- Index Calculus Method:
  - Specialized algorithm for $Z_p^*$ and primitive element $\alpha$
  - Idea:
    - Use a factor base $B = \{p_1, p_2, \ldots, p_B\}$
    - Find the logarithms of the primes in the factor base
    - Use these logarithms to compute the logarithm of $\beta$
- 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: $\Omega(\sqrt{n})$



## Discrete Logarithm Algorithms in Practice

- Setups:
  - G = ($Z_p^*$, .), $p$ prime, $\alpha$ primitive element modulo $p$
  - G = ($Z_p^*$, .), $p$ and $q$ prime ($p \equiv 1 \bmod q$), $\alpha$ element having order $p$
  - G = ($F_{2^n}^*$, .), $\alpha$ primitive element modulo in $F_{2^n}^*$
  - Elliptic Curves modulo a prime or over a finite field
- Lenstra and Verheul report to be secure until year 2020:
  - $p \approx 2^{160}$ for elliptic curves
  - $p \approx 2^{1880}$ for ($Z_p^*$, .)
  - Elliptic Curve implementations are the most efficient
    - Mainly due to inexistence of an index calculus attack
    - Adequate for low power/resources devices such as PDAs and smartcards
- Latest challenge:
  - ECC2K-108 over $F_{2^{109}}^*$ (solved in April 2000) using 9500 computers about 50 times the computation effort required to factor the RSA challenge RSA-512

## Diffie-Hellman Problems

- Computational Diffie-Hellman
  - Given a multiplicative group ($G$, .), an element $\alpha \in G$ (order $n$), two elements $\alpha^b, \alpha^c \in <\alpha>$
  - Question: find $\alpha^{bc}$
- Decisional Diffie-Hellman
  - Given a multiplicative group ($G$, .), an element $\alpha \in G$ (order $n$), three elements $\alpha^b, \alpha^c, \alpha^d \in <\alpha>$
  - Question: Is $d = bc$ ?
- 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

| **Private: A** | **Public** | **Private: B** |
|---|---|---|
| **x** | **p: prime number,**<br>**$\alpha$: primitive element of $Z_p^*$** | **y** |
| **compute:**<br>**$\alpha^x$ mod p** | | **compute:**<br>**$\alpha^y$ mod p** |
| **receive:**<br>**$\alpha^y$ mod p** | | **receive:**<br>**$\alpha^x$ mod p** |
| **Compute shared key:**<br>**$(\alpha^y)^x$ mod p** | | **Compute shared key:**<br>**$(\alpha^x)^y$ mod p** |

- Based on the difficulty of computational Diffie-Hellman
- Works also in extension Galois fields: GF($p^q$), …