CS 578 – Cryptography

Prof. Michael Backes

Introduction to Number Theory – Part 1 Arithmetic modulo Primes

May 19, 2006

Administrative Announcements

- Mid-term evaluation:
  - Pick up your key in the break
- Handouts:
  - Today add-on to last lecture notes (on combining secrecy and integrity, still relevant for mid-term exam)
  - Handout for next Tuesday will be available on Monday on the course web page
- Link to more comprehensive number theory primer on the course web page
- Problems with notification emails for quizzes?

Summary of Symmetric Encryption

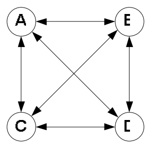
- Perfect secrecy and the OTP: Security against infinitely smart adversaries, but unpractical
- Stream ciphers: Fast, but keys only used once. Security treated via security of PRGs
- Block ciphers: Basic building blocks for larger encryption systems (modes of operation). Idealized into PRPs/PRFs in security proofs
- Modes of Operation: Semantic security: *The* definition of secure encryption. Captures security against all efficient adversaries. So far CT-only and CPA variants.

Summary of MACs

- MACs: Protect integrity, do not guarantee privacy
- CMA-Security: The definition of secure MACs
- Major problem: Building big-MACs from small-MACs
- CRHFs: Basic building blocks for enlarging domains of cryptographic primitives
- CHRFs + MACs: Give big-MACs. Also constructions immediately from hash functions (HMAC)
- Combining secrecy and integrity: Various modes, encrypt-then-MAC best, direct constructions exist (authenticated symmetric encryption)

Key Management

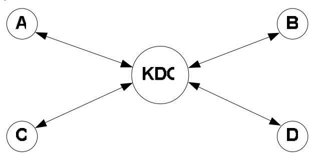
- Key management between n parties



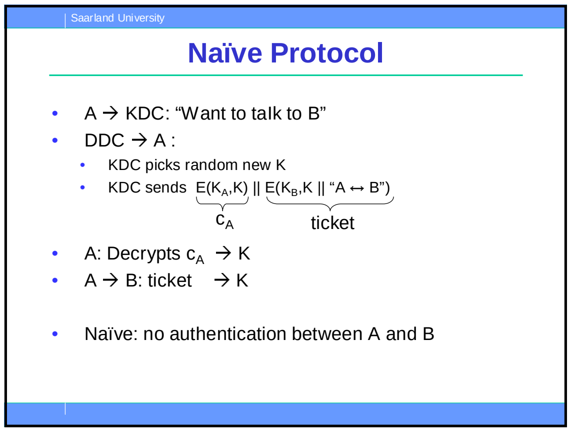
- Every party has to manage n keys, n^2 keys overall

Improvements via KDCs

- Improvement: Using a KDC (Key Distribution Center)

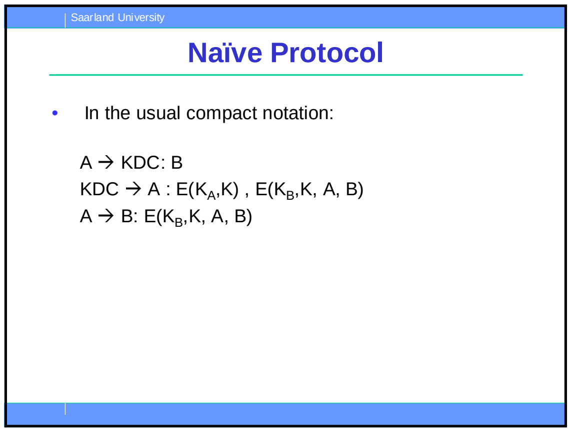


- KDC has linear number of keys
- KDC has to be online all the time



Naïve Protocol

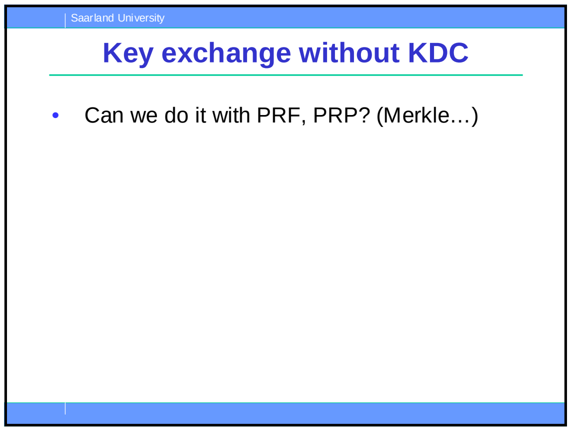
- A $\rightarrow$ KDC: “Want to talk to B”
- DDC $\rightarrow$ A :
  - KDC picks random new K
  - KDC sends $\underbrace{E(K_A,K)}_{c_A} \,||\, \underbrace{E(K_B,K \,||\, \text{“}A \leftrightarrow B\text{”})}_{\text{ticket}}$
- A: Decrypts c_A $\rightarrow$ K
- A $\rightarrow$ B: ticket $\rightarrow$ K
- Naïve: no authentication between A and B



Naïve Protocol

- In the usual compact notation:

A $\rightarrow$ KDC: B
KDC $\rightarrow$ A : $E(K_A,K)$, $E(K_B,K, A, B)$
A $\rightarrow$ B: $E(K_B,K, A, B)$



Key exchange without KDC

- Can we do it with PRF, PRP? (Merkle…)

Merkle Puzzles

- Key exchange between Alice and Bob (1974)
- Step 1 (done by Alice):
  - Create puzzles i := $E(K_i$, "Puzzle X_i" || $K^*_i)$, X_i, P_i, K_i random, $|K_i|$ = 20 bits, $|K^*_i|$ = 128 bits
  - Send (Puzzle 1, …, Puzzle 2^{20})
- Step 2 (done by Bob):
  - Pick random puzzle j
  - Find K_j by exhaustive search, retrieve X_j, K^*_j
  - Sends X_j to Alice
- Step 3: Alice extracts K^*_j from puzzle j

Merkle Puzzles (cont'd)

- K^*_j is the shared (secret) key
- Time:
  - Alice: 2^{20} for building the puzzles
  - Bob: 2^{20} for solving one puzzle
- Attacker: Tries to get K^*_j
- Must solve 2^{20} puzzles
  $\rightarrow$ requires time $2^{20} \cdot 2^{20} = 2^{40}$
- Time for n puzzles: O(n) for Alice/Bob, $O(n^2)$ for attacker $\rightarrow$ quadratic gap
- Not enough! Goal: Key exchange with exponential gap between A/B and attacker

Key Exchange without KDC

- Goal: Key exchange without KDC but exponential gap (go beyond Merkle puzzles)
- $\rightarrow$ PGP, SSL, …

Modular Arithmetic, modulo primes

- Let p be a prime, p huge (approx 1024 bits)
- Notation: Z_p= {0,1,2,…, p-1} with addition and multiplication modulo p
- (Not speaking about residue classes here, well-definedness of addition/multiplication, etc.)
- Convention: All values in Z are mapped to their corresponding number in Z_p
  e.g., -3 mod 7 = 4

Fermat's Little Theorem, Inverses

- Classic theorem (Fermat's little theorem):
  $\forall\, a \in Z_p \setminus \{0\}$ (mod p): $a^{p-1} = 1$ (mod p)
- Example: 3^4 mod 5 = 81 mod 5 = 1 mod 5
- Definition (Inverse): The inverse of $x \in Z_p$ is an element $y \in Z_p$ such that $x \cdot y = 1$ ($\in Z_p$).
- Denote the inverse of x by x^{-1}
- Example: Inverse of $2 \in Z_p$ (p odd prime) is $2^{-1} = (p+1)/2$

Inverses

- Which elements in Z_p have an inverse?
- Given $x \in Z_p$, set $y = x^{p-2} \in Z_p$.
- Then we have
  $x \cdot y = x \cdot x^{p-2} = x^{p-1} = 1 \in Z_p$ if $x \neq 0$
  $\rightarrow \forall\, x \neq 0$: x is invertible
- Efficient way of computing the inverse: $x^{-1} := x^{p-2}$
- Set of invertible elements in Z_p is {1,2,3,…,p-1} := Z^*_p

Solving Equations

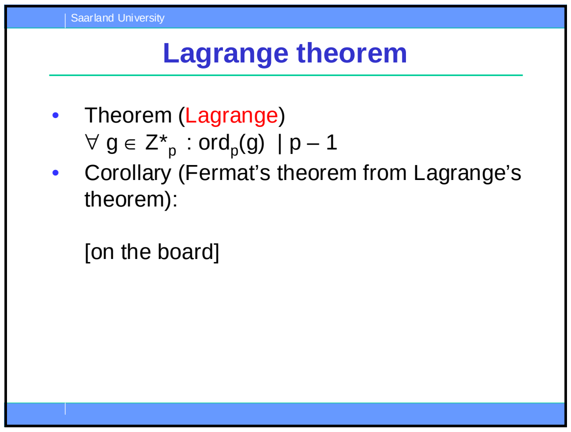
- Now solving equations ax+b = 0 (mod p)
- Simple algorithm: Compute: $x = -b \cdot a^{p-2}$
- What about quadratic equations in Z_p?
  → First investigate the structure of Z^*_p
- Theorem (Euler): Z^*_p is a cyclic group, i.e.,
  $\exists g \in Z^*_p$: $\langle g \rangle := \{1,g,g^2,g^3,\ldots,g^{p-2}\} = Z^*_p$
- Such an element g is called a generator of Z^*_p

Generators

- Examples: 3 is a generator of Z^*_7 :
- $\langle 3 \rangle = \{1,3,2,6,4,5\}$ (mod 7) = Z^*_7
- Not every element of Z^*_p is a generator, e.g.,
- 2 is not a generator of Z^*_7:
- $\langle 2 \rangle = \{1,2,4\}$ (mod 7)
- Note that $\langle 2 \rangle = 3 \mid 7 - 1$ (size of the generated group divided the size of Z^*_p)

Order of Elements

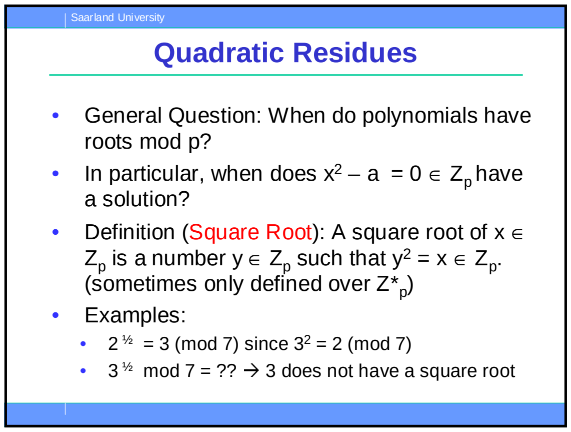
- Definition (Order): The order of $g \in Z^*_p$ is the smallest positive integer a such that
  $g^a = 1 \in Z^*_p$
- Order of g denoted $ord_p(g)$ (which is $|\langle g \rangle|$).
- Examples:
  - $ord_7(3) = 6$,
  - $ord_7(2) = 3$.



Lagrange theorem

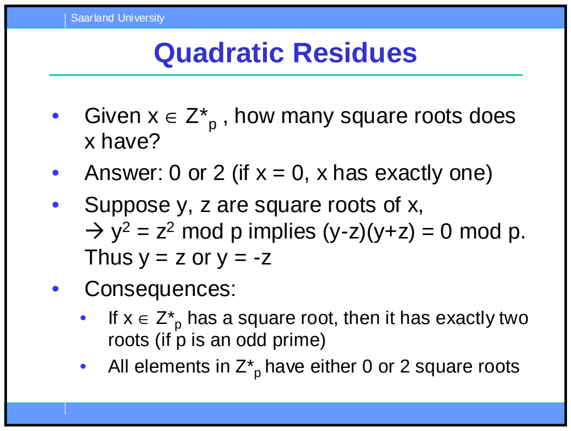
- Theorem (Lagrange)
  $\forall$ g $\in$ Z^*_p : $ord_p(g)$ | p – 1
- Corollary (Fermat’s theorem from Lagrange’s theorem):

  [on the board]



Quadratic Residues

- General Question: When do polynomials have roots mod p?
- In particular, when does $x^2 – a = 0 \in Z_p$ have a solution?
- Definition (Square Root): A square root of x $\in$ Z_p is a number y $\in$ Z_p such that $y^2 = x \in Z_p$. (sometimes only defined over Z^*_p)
- Examples:
  - $2^{½} = 3$ (mod 7) since $3^2 = 2$ (mod 7)
  - $3^{½}$ mod 7 = ?? → 3 does not have a square root



Quadratic Residues

- Given x $\in$ Z^*_p , how many square roots does x have?
- Answer: 0 or 2 (if x = 0, x has exactly one)
- Suppose y, z are square roots of x,
  → $y^2 = z^2$ mod p implies (y-z)(y+z) = 0 mod p.
  Thus y = z or y = -z
- Consequences:
  - If x $\in$ Z^*_p has a square root, then it has exactly two roots (if p is an odd prime)
  - All elements in Z^*_p have either 0 or 2 square roots

Quadratic Residues

- Definition (Quadratic Residues): An element $g \in Z^*_p$ is called a quadratic residue if g has a square root.
- Fact: For an odd prime, the number of quadratic residues is (p-1)/2.

  [proof on the board]

Euler's Theorem

- Theorem (Euler): $g \in Z^*_p$ is a quadratic residue if and only if $g^{(p-1)/2} = 1 \pmod p$.

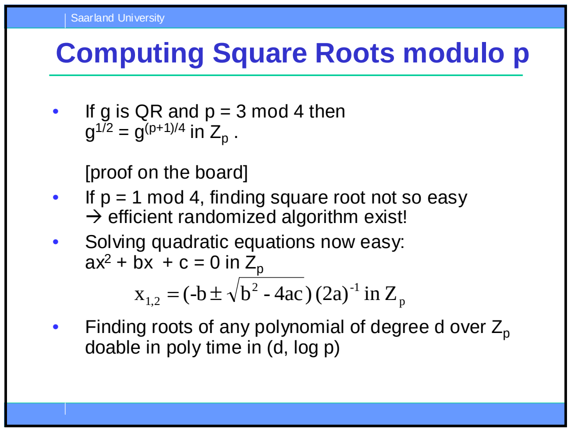
  [proof on the board]

Legendre Symbol

- Note: For any $g \in Z^*_p$:
  - $g^{(p-1)/2}$ is a square root of 1
  - 1 only has two square roots: 1 and -1
    $\rightarrow g^{(p-1)/2} \in \{-1,1\}$.
- Definition (Legendre Symbol). The Legendre symbol of g over p is defined as

$$\left(\frac{g}{p}\right) = \begin{cases} 1 & \text{if g is QR in } Z^*_p \\ -1 & \text{if g is not a QR in } Z^*_p \\ 0 & \text{if g=0} \end{cases}$$

- Euler $\rightarrow$ $\left(\frac{g}{p}\right) = g^{(p-1)/2}$ in Z_p

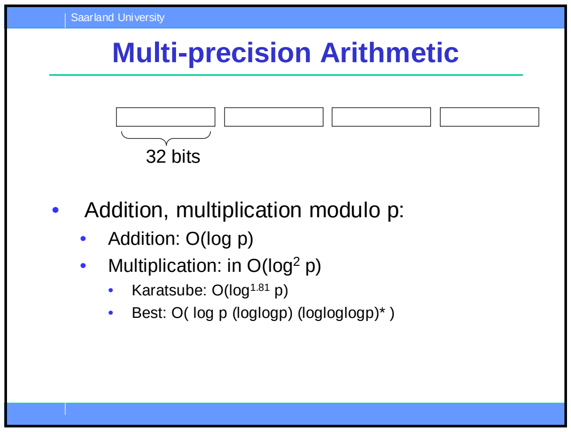


Computing Square Roots modulo p

- If g is QR and p = 3 mod 4 then
  $g^{1/2} = g^{(p+1)/4}$ in Z_p .

  [proof on the board]
- If p = 1 mod 4, finding square root not so easy
  → efficient randomized algorithm exist!
- Solving quadratic equations now easy:
  $ax^2 + bx + c = 0$ in Z_p

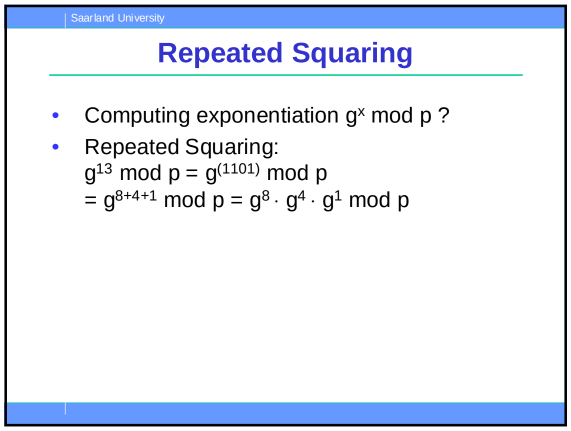
  $$x_{1,2} = (-b \pm \sqrt{b^2 - 4ac})\,(2a)^{-1} \text{ in } Z_p$$
- Finding roots of any polynomial of degree d over Z_p
  doable in poly time in (d, log p)



Multi-precision Arithmetic

$\underbrace{\boxed{}}_{\text{32 bits}}\ \boxed{}\ \boxed{}\ \boxed{}$

- Addition, multiplication modulo p:
  - Addition: O(log p)
  - Multiplication: in $O(\log^2 p)$
    - Karatsube: $O(\log^{1.81} p)$
    - Best: O(log p (loglogp) (logloglogp)*)



Repeated Squaring

- Computing exponentiation g^x mod p ?
- Repeated Squaring:
  g^{13} mod p = $g^{(1101)}$ mod p
  = g^{8+4+1} mod p = $g^8 \cdot g^4 \cdot g^1$ mod p

Repeated Squaring

- General algorithm
  - Let $x = x_n x_{n-1} \ldots x_1 x_0$ be a binary representation of x
  - Set z := 1, y := g
  - For i = 0,1,…,n:
    - If $x_i = 1$ set $z := z \cdot y \pmod p$
    - $y := y^2 \pmod p$
  - Output z
- Running time:
  - n squarings + (in average) n/2 multiplications (if x random)
  - → On average O(log x) multiplications:
  - → Exponentiation takes time $O(\log x \cdot \log^2 p)$

Finding Large Primes

- How to find large primes?
- (Strong) recent result: PRIMES is in P!
  → deterministic algorithm for deciding if an n-bit number is prime or not
- But complexity $O(n^{12})$, can be reduced to $O(n^6)$
- (Efficient) remedy: Randomized primality tests

Fermat’s Primality Test

- Input: a potential prime p
  - Pick lots of a_i with $1 < a_i < p-1$ random
  - Check if $a_i^{p-1} = 1 \bmod p$ for all i
  - If no, p is not a prime
  - If yes, p “presumably prime”
- Idea: exploit Fermat’s little theorem for primes (does not hold for composites)
- Problem: Carmichael numbers

Using Arithmetic mod primes

- Secret Key Exchange (against passive eavesdroppers – listen only)

$$\begin{tikzcd}
A \arrow[rr, shift left=4] \arrow[d] & & B \arrow[ll, shift left=2] \arrow[d] \\
K & & K
\end{tikzcd}$$

$$\begin{tikzcd}
 \arrow[rr, shift left=2] & & \arrow[ll]
\end{tikzcd}$$

- Example: Merkle puzzle (inefficient)

Diffie-Hellman Key Exchange

- Basic Diffie-Hellman protocol, 1976
- Fix:
  - A prime p (approx. 1024 bits)
  - An element $g \in Z^*_p$ where g is a generator of Z^*_p

A | B

$a \leftarrow_R \{1,\ldots,p-1\}$ | $b \leftarrow_R \{1,\ldots,p-1\}$

$$\xrightarrow{X = g^a \in Z^*_p}$$

$$\xleftarrow{Y = g^b \in Z^*_p}$$

$K = Y^a = (g^b)^a = g^{ab}$ | $K = X^b = (g^a)^b = g^{ab}$

Diffie-Hellman Key Exchange

- Both parties obtain the same key
  $K = g^{ab} \in Z^*_p$
- Then K can be used to derive other keys, e.g.,
  - an AES key
  - a MAC key, …
- Called key derivation function (KDF)

Diffie-Hellman Key Exchange

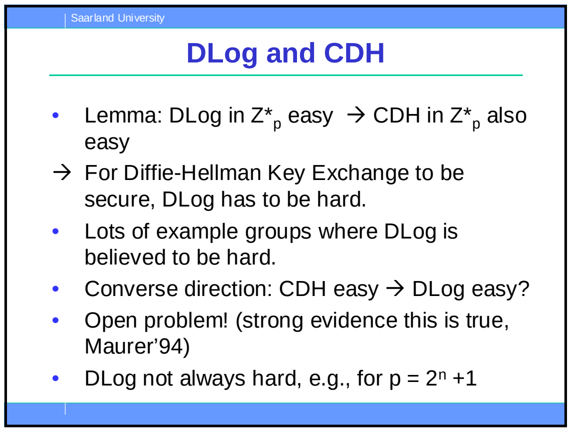
- Why is basic DH secure against eavesdroppers?
- Question: Can Eve compute $K = g^{ab}$? More precisely, can Eve distinguish $K = g^{ab}$ from g^c for a random c?
- Can Eve compute K:
  - Sees: $X = g^a$, $Y = g^b \in Z^*_p$
  - Goal: Compute $g^{ab} \in Z^*_p$

Comp. Diffie-Hellman Problem

- Let $DH(g, g^a, g^b) := g^{ab}$ denote the Diffie-Hellman function
- Computational Diffie-Hellman Assumption (CDH):
  Given a random n-bit prime p and a random generator $g \in Z^*_p$, no efficient adversary (in n) can compute the function $DH(g, g^a, g^b)$

Discrete Logarithm Problem

- Related function: Discrete Logarithm
- Fix:
  - A prime p (approx. 1024 bits)
  - An element $g \in Z^*_p$
- Definition (Discrete Logarithm). The discrete logarithm of x with respect to g, $DLog_g(x)$, is defined as the smallest integer $i \geq 0$ s.t. $g^i = x$ in Z^*_p
  (and $DLog_g(x) = \infty$ if no such i exists)
- Examples: p = 7, g = 3
  - $Dlog_3(2) = 2$, $Dlog_3(4) = 5$



DLog and CDH

- Lemma: DLog in Z^*_p easy $\rightarrow$ CDH in Z^*_p also easy

$\rightarrow$ For Diffie-Hellman Key Exchange to be secure, DLog has to be hard.

- Lots of example groups where DLog is believed to be hard.
- Converse direction: CDH easy $\rightarrow$ DLog easy?
- Open problem! (strong evidence this is true, Maurer'94)
- DLog not always hard, e.g., for $p = 2^n + 1$