CS 578 – Cryptography

Prof. Michael Backes

One-way Functions, Basics of Number Theory Trapdoor Permutations, Introduction to RSA

June 13, 2006

Administrative Announcements

- Handouts:
  - New exercise sheet
  - New lecture notes
- Investigating your exam:
  - In case you missed the investigation yesterday, please come to this Friday's office hour

Recall: Discrete Logarithm Problem

- Fix a prime p (approx. 1024 bits), and element $g \in Z^*_p$ (or of a subgroup G_q of prime order q of Z^*_p),
- Definition (Discrete Logarithm). The discrete logarithm of h with respect to g, $DLog_g(h)$, is defined as the smallest integer $x>0$ s.t. $g^x = h$ in Z^*_p (or G_q), and $DLog_g(h) = \infty$ if no such x exists
- Conjectured hard to compute in many groups

Recall: Comp. Diffie-Hellman Problem

- Let $DH_g(g^x, g^y) := g^{xy}$ denote the Diffie-Hellman function
- Computational Diffie-Hellman Assumption (CDH):
  Given a random n-bit prime p, a random generator $g \in Z^*_p$ (or a subgroup G_q of large prime order q of Z^*_p), and random elements $x,y \in \{1,\ldots,p-1\}$ no efficient adversary (in n) can compute $DH_g(g^x, g^y)$.

Recall: Decision Diffie-Hellman Problem

- Decisional Diffie-Hellman Assumption (DDH):
  Given a random n-bit prime q, $n_p(n)$-bit prime p with q | p-1, and $g \in Z^*_p$ of order q, no efficient adversary (in n) can distinguish
  (g^x, g^y, g^{xy}) and (g^x, g^y, g^z)
  for x, y, z random in $\{1,\ldots,q\}$.

Recall: Diffie-Hellman Key Exchange

- Basic Diffie-Hellman protocol in subgroups
- Choose:
  - Primes q and p such that q | p-1
  - An element $g \in Z^*_p$ of order q

A		B
$x \leftarrow_R \{1,\ldots,p-1\}$		$y \leftarrow_R \{1,\ldots,p-1\}$
	$h = g^x \in Z^*_p$ →	
	← $i = g^y \in Z^*_p$	
$K = i^x = (g^y)^x = g^{xy}$		$K = h^y = (g^x)^y = g^{xy}$

One-way Functions (OWF)

- Unifying all this into one-way functions
- Definition (One-way Function): An efficiently computable function F: $\{0,1\}^* \rightarrow \{0,1\}^*$ is one-way iff it is hard to invert: given F(x), hard to get x. More precisely:
  $\Pr[F(x) = F(x'); x \leftarrow_R \{0,1\}^n, y := F(x), x' \leftarrow A(n,y)]$
  is negligible.
- Remark: Sometimes families of functions with different domains/ranges and explicit generation algorithm for x

One-way Functions so far

1. Assume that $E = (E_n(K,\cdot))_{n \in N}$ is a family of PRPs, where E_n has domain and range $\{0,1\}^n$.
- Then e.g. $F^E(K) := E_n(K,0)$ if $K \in \{0,1\}^n$ is a one-way function.

- This function is bad for key-exchange (leaves only Merkle puzzles)

One-way Functions so far (cont'd)

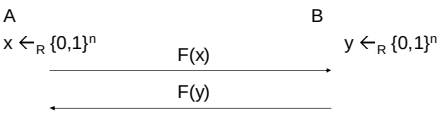
2. p prime, g generator of Z^*_p
- Then define $F^{DLog}(x) = g^x \bmod p$
- Inversion problem = DLog (believed hard)
- Fastest DLog algorithm in $Z^*_p = O(e^{(\log p)^{1/3}})$
- Current world record: 90 digits (300 bits)

One-way Functions so far (cont'd)

- Special properties of F^{DLog},
  - Given: $F^{DLog}(x)$, $F^{DLog}(y)$, easy to compute $F^{DLog}(x+y)$
  - And, given a, $F^{DLog}(x)$, easy to compute $F^{DLog}(ax)$
- → DH Key exchange + ElGamal encryption

Generalized DH Key Exchange

- DH Key Exchange



Key = F(xy)
A: given x, F(y), compute F(xy)
B: given y, F(x), compute F(xy)

Generalized ElGamal Encryption

- Setup:
  - sk: $x \leftarrow_R \{0,1\}^n$
  - pk: F(x)
- Enc(pk,m):
  - $y \leftarrow_R \{0,1\}^n$
  - $c := (c_1,c_2) := (F(y), m \oplus \text{Hash}(F(xy)))$
- Dec(sk, (c_1,c_2)) = $c_2 \oplus \text{Hash}(\underbrace{F(xy)}_{\text{From } x,\, c_1})$

OWF and Diffie-Hellman

- Recasting (Computational) Diffie-Hellman Assumption: For all efficient adversaries A:

  $\Pr [z = F(xy); x,y \leftarrow_R \{0,1\}^n, z \leftarrow A(n,F(x),F(y))]$

  is negligible.
- Decision Diffie-Hellman analogously generalized
- Thus $F^{DLog}(x) = g^x \bmod p$ is believed to be one-way and to satisfy the DH assumption

OWF as Naïve Encryption Functions

- OWFs on their own are typically bad encryption functions:
- Let F be a OWF. Then H(x,y) := F(x) || y is a OWF as well, but leaks half of the plaintext!

  [proof on the board]

Hardcore Predicates of OWF

- Definition (Hardcore Predicate): A hardcore predicate of a function F: {0,1}* → {0,1}* is an efficiently computable Boolean predicate π: {0,1}* → Bool such that for all efficient A, we have that
  $|\Pr[z = \pi(x); x \leftarrow_R \{0,1\}^n, y := F(x), z \leftarrow A(n,y)] - ½|$
  is negligible.
- Example: Given prime p and generator g of Z^*_p. Then π(x) := 0 if x < p/2 and π(x) := 1 if x > p/2 constitutes a hardcore predicate/bit of the F^{DLog} function.

OWF and Diffie-Hellman

- Facts about DLog
  - F^{DLog} leakes lsb(x) in Z^*_p
    → DDH easy in Z^*_p
  - For performance, schemes in practice use subgroup of Z^*_p (exponents are smaller)
  - $\pi(x)$ constitutes a hardcore predicate/bit of F^{DLog}
  - DLog has random self-reduction
- Today: A new type of one-way functions (trapdoor one-way functions)

Arithmetic Modulo Composites

- RSA: requires arithmetic modulo composites
- $N = p \cdot q$, p and q large primes of same size
  - $N \approx 1024$ bits
  - $p,q \approx 512$ bits
- Notation: $Z_N = \{0,1,2,\ldots, N-1\}$ with addition and multiplication modulo N
- Which elements in Z_N have an inverse?

Extended Euclidian Algorithm

- Lemma: For all $a,b \in Z$, there exist $x,y \in Z$ such that $a \cdot x + b \cdot y = \gcd(a,b)$.
- These values x,y can be efficiently computed by means of the extended Euclidian algorithm.

Extended Euclidian Algorithm

- Given a,b, find x,y such that
  $a \cdot x + b \cdot y = \gcd(a,b)$
- Similar to Euclidean algorithm:
  1. Start with: a, b, (1,0), (0,1)
  2. Divide the larger of the two numbers by the smaller using division with remainder. Call this quotient q.
  3. Subtract q times the smaller from the larger.
  4. Subtract q times the vector corresponding to the smaller from the vector corresponding to the larger.
  5. Repeat steps 2 through 4 until the result is zero.

Extended Euclidian Algorithm

- An example for a=53, b=30 :

53		30	(1,0)		(0,1)
23	←-1	30	(1,-1)	←-1	(0,1)
23	-1→	7	(1,-1)	-1→	(-1,2)
2	←-3	7	(4,-7)	←-3	(-1,2)
2	-3→	1	(4,-7)	-3→	(-13,23)
0	←-2	1	(30,-53)	←-2	(-13,23)

- Indeed $53 \cdot (-13) + 30 \cdot 23 = -689 + 690 = 1$

Arithmetic Modulo Composites

- Lemma: $x \in Z_N$ is invertible in Z_N if and only if $\gcd(x,N) = 1$.

  [proof on the board]

- Note: Given $x \in Z_N$, $\gcd(x,N) = 1$, we can find $x^{-1} \in Z_N$ in time $O(\log^2 N)$
- Denote the of invertible elements in Z_N by Z^*_N

Arithmetic Modulo Composites

- Note: If $0 \neq x \notin Z^*_N$, then we can factor N
  → gcd(x,N) is a non-trivial factor of N
- Size of Z^*_N? Denoted by $\varphi(N) = |Z^*_N|$
- φ called Euler's totient function; easily computable:
  1. $\varphi(p) = p - 1 \quad if\ p\ prime$
  2. $\varphi(N) = N \cdot \prod_i (1 - \frac{1}{p_i}) \qquad if\ N = \prod_i p_i^{e_i}$

Arithmetic Modulo Composites

- For RSA moduli: $N = p \cdot q$, then
  $\varphi(N) = (p-1)(q-1) = N - q - p + 1$
- Theorem (Euler, generalization of Fermat's theorem):
  $\forall\ a \in Z^*_N: a^{\varphi(N)} = 1 \pmod N$
- Special case: p prime, then $\varphi(p) = p-1$ and
  $\forall\ a \in Z^*_p: a^{\varphi(p)} = a^{p-1} = 1 \pmod p$

Chinese Remainder Theorem

- Chinese Remainder Theorem (CRT) for N=pq:
  Let $p \neq q$ be primes and $N = pq$. Then for all $a,b \in Z$:
  $a = b \bmod N \Leftrightarrow a = b \bmod p \wedge a = b \bmod q$
- Further, given $x_p \in Z_p$ and $x_q \in Z_q$ there exists a unique element $s \in Z_N$ such that
  $s = x_p \bmod p$ and $s = x_q \bmod q$.
- s can be computed from x_p and x_q as follows:
  - Use the extended Euclidean algorithm to compute integers u, v with
    $up + vq = 1$,
  - Output $x := up\ x_q + vq\ x_p \bmod N$.
- Example: Compute 1027 mod 55, p = 5, q = 11.
  - u = -2, v = 1 and 1027 = 2 mod 5, and 1027 = 4 mod 11.
  - Then: $1027 = -2 \cdot 5 \cdot 4 + 1 \cdot 11 \cdot 2 = -40 + 22 = -18 = 37 \bmod 55$.

RSA Trapdoor Permutation

- RSA (Rivest, Shamir, Adleman 1977): Trapdoor Permutation (Permutation that is hard to compute unless one knows a secret trapdoor)
- Definition ((Keyed) trapdoor permutation): A keyed family of efficiently computable functions $F = (F(pk, \cdot))_{(pk,*) \in [Gen(n)]}$, $F(pk, \cdot): \mathcal{M}_{pk} \rightarrow \mathcal{T}_{pk}$ is a family of trapdoor permutations iff
  1. $Pr[F(pk,x) = F(pk,x'); (pk,sk) \leftarrow Gen(n), x \leftarrow_R \mathcal{M}_{pk}, y := F(pk,x), x' \leftarrow A(n,pk,y)]$ is negligible for all efficient A.
  2. There exists an efficient algorithm B such that $B(n, sk, pk, F(pk,x)) = x$ for all $x \in \mathcal{M}_{pk}$.

RSA Trapdoor Permutation

- Setup:
  - Pick random n-bit primes p, q (e.g., n = 512 bits)
  - Set $N = p \cdot q$.
  - Choose arbitrary value e such that $\gcd(e, \varphi(N)) = 1$
- Function: $F: Z_N \rightarrow Z_N$
  - $F(x) = x^e \bmod N$
- Trapdoor for given pk = (e,N):
  $d := e^{-1} \bmod \varphi(N)$

RSA Trapdoor Permutation

- Claim: $D(y) := y^d \bmod N$ is inverse of F.
- Lemma 1: Let G be a group, $g \in G$, and $x,y \in Z$. Then $g^x = g^y$ iff $x = y \bmod ord(g)$.

  [proof on the board]

- Lemma 2: Let (N,e) be an RSA public-key, d the corresponding trapdoor, and $m \in Z_N$. Then $m^{ed} = m \bmod n$.

  [proof on the board]

Is RSA a One-way Permutation?

- Eve sees $c = m^e \bmod N$
- Goal: Compute $c^{1/e} \bmod N$
- Currently best algorithm for computing e'th roots:
  - Step 1: factor N. (conjectured hard)
  - Step 2: Find e'th roots modulo p and q. (easy)

Is RSA a One-way Permutation?

- RSA Assumption: Given random n-bit primes p,q, a (random or fixed) value e such that $\gcd(e, \varphi(N)) = 1$, and a random number $c \in Z_N$, there does not exist any efficient algorithm that computes the e'th root of c modulo N up to negligible probability.
- Major open problem: Given algorithm for computing e'th roots modulo N, does it imply a factoring algorithm?

Is RSA a One-way Permutation?

- Breaking RSA → ability to factor?
- Reduction to show that no shortcut exist would look as follows:
  - Efficient alg. for e'th roots mod N → efficient alg. for factoring N.
  - One of the oldest problems in public-key crypto
- Evidence no reduction exists (Boneh-Venkatesan 1998)
  - Algebraic reduction → factoring easy
  - Unlike Diffie-Hellman (conjectured equivalent to DLog (Maurer 1994)).

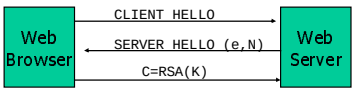
Naïve Use of RSA as an Encryption Scheme

- Naively (wrong):
- Setup:
  - Generate public parameters pk = (N,e)
  - Keep trapdoor sk = d secret
- Enc(pk,m) := m^e mod N
- Decrypt(sk,c) := c^d mod N

Naïve RSA is not even CPA-secure

- Naïve RSA is deterministic → not even CPA-secure.
- Concrete leakage known: 1 bit: ciphertext c leaks the so-called Jacobi symbol of the plaintext
- Various active attacks against naïve RSA:
  - Simple attack in CCA game using two ciphertexts
  - Slightly more complicated attack based on one ciphertext ("blind decryption")

A Practical Attack against Naïve RSA



- Session-key K is 64 bits, i.e., $K \in \{0,\ldots,2^{64}\}$
  Adversary sees: $C = K^e \pmod N$.
- Suppose that $K = K_1 \cdot K_2$, where $K_1, K_2 < 2^{34}$.
  Holds with probability ≈ 20% → $C/K_1^e = K_2^e \pmod N$
- Build table: $C/1^e$, $C/2^e$, $C/3^e$, …, $C/2^{34e}$.
- For $K_2 = 0,\ldots, 2^{34}$ test if K_2^e is in table.
- Attack time for usual values e: $\approx 2^{39} << 2^{64}$