

# 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  $g$ ,  $\text{DLog}_g(h)$ , is defined as the smallest integer  $x > 0$  s.t.  $g^x = h$  in  $Z_p^*$  (or  $G_q$ ), and  $\text{DLog}_g(h) = \infty$  if no such  $x$  exists
- Conjectured hard to compute in many groups

## Recall: Comp. Diffie-Hellman Problem

---

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

## Recall: Decision Diffie-Hellman Problem

---

- **Decisional Diffie-Hellman Assumption (DDH):**  
Given a random  $n$ -bit prime  $q$ ,  $n_p(n)$ -bit prime  $p$  with  $q \mid p-1$ , and  $g \in \mathbb{Z}_p^*$  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  $\{1, \dots, q\}$ .

# Recall: Diffie-Hellman Key Exchange

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

A

$$x \leftarrow_{\mathbb{R}} \{1, \dots, p-1\}$$

$$h = g^x \in \mathbb{Z}_p^*$$



$$i = g^y \in \mathbb{Z}_p^*$$



$$K = i^x = (g^y)^x = g^{xy}$$

B

$$y \leftarrow_{\mathbb{R}} \{1, \dots, p-1\}$$

$$K = h^y = (g^x)^y = g^{xy}$$

# 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:  
$$\Pr [F(x) = F(x'); x \leftarrow_{\mathcal{R}} \{0,1\}^n, \\ y := F(x), x' \leftarrow A(n,y)]$$
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  $E = (E_n(K, \cdot))_{n \in \mathbb{N}}$  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  $F^{\text{DLog}}(x) = g^x \bmod p$
  - Inversion problem = DLog (believed hard)
  - Fastest DLog algorithm in  $Z_p^* = O(e^{(\log p)^{1/3}})$
  - Current world record: 90 digits (300 bits)

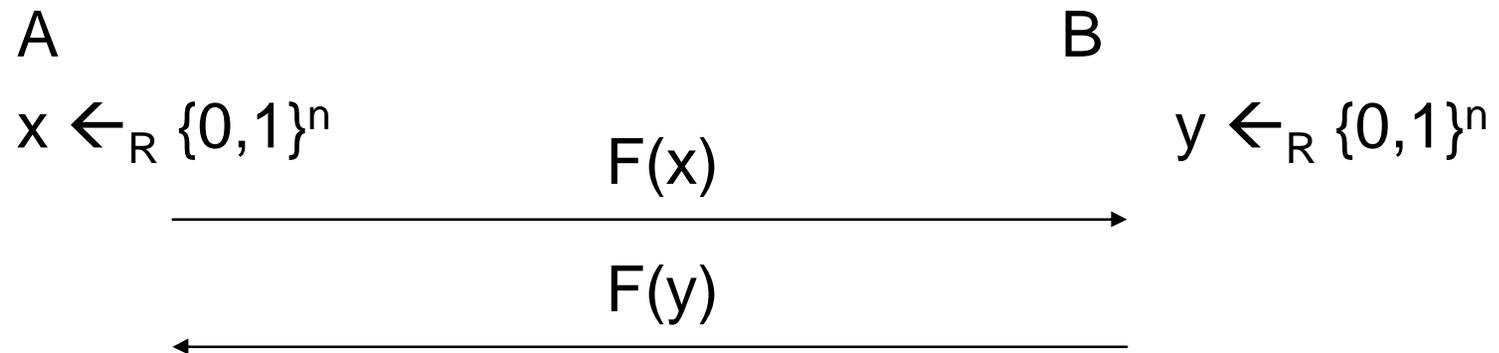
# One-way Functions so far (cont'd)

---

- Special properties of  $F^{\text{DLog}}$ ,
  - Given:  $F^{\text{DLog}}(x)$ ,  $F^{\text{DLog}}(y)$ , easy to compute  $F^{\text{DLog}}(x+y)$
  - And, given  $a$ ,  $F^{\text{DLog}}(x)$ , easy to compute  $F^{\text{DLog}}(ax)$
- → 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:
  - $sk: x \leftarrow_R \{0,1\}^n$
  - $pk: F(x)$
- $Enc(pk, m)$ :
  - $y \leftarrow_R \{0,1\}^n$
  - $c := (c_1, c_2) := ( F(y), m \oplus Hash(F(xy)) )$
- $Dec(sk, (c_1, c_2)) = c_2 \oplus Hash(F(xy))$   
 $\underbrace{\hspace{10em}}_{\text{From } x, c_1}$

# OWF and Diffie-Hellman

---

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

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

is negligible.

- Decision Diffie-Hellman analogously generalized
- Thus  $F^{\text{DLog}}(x) = g^x \bmod p$  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  $F: \{0,1\}^* \rightarrow \{0,1\}^*$  is an efficiently computable Boolean predicate  $\pi: \{0,1\}^* \rightarrow \text{Bool}$  such that for all efficient  $A$ , we have that  $|\Pr[z = \pi(x); x \leftarrow_{\mathcal{R}} \{0,1\}^n, y := F(x), z \leftarrow A(n,y)] - \frac{1}{2}|$  is negligible.
- Example: Given prime  $p$  and generator  $g$  of  $Z_p^*$ . Then  $\pi(x) := 0$  if  $x < p/2$  and  $\pi(x) := 1$  if  $x > p/2$  constitutes a hardcore predicate/bit of the  $F^{\text{DLog}}$  function.

# OWF and Diffie-Hellman

---

- Facts about DLog
  - $F^{\text{DLog}}$  leaks  $\text{lsb}(x)$  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  $F^{\text{DLog}}$
  - 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:  $Z_N = \{0, 1, 2, \dots, N-1\}$  with addition and multiplication modulo  $N$
- Which elements in  $Z_N$  have an inverse?

# Extended Euclidian Algorithm

---

- Lemma: For all  $a, b \in \mathbb{Z}$ , there exist  $x, y \in \mathbb{Z}$  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       $\xleftarrow{-1}$       30      (1,-1)                       $\xleftarrow{-1}$       (0,1)
- 23       $\xrightarrow{-1}$       7      (1,-1)                       $\xrightarrow{-1}$       (-1,2)
- 2       $\xleftarrow{-3}$       7      (4,-7)                       $\xleftarrow{-3}$       (-1,2)
- 2       $\xrightarrow{-3}$       1      (4,-7)                       $\xrightarrow{-3}$       (-13,23)
- 0       $\xleftarrow{-2}$       1      (30,-53)                       $\xleftarrow{-2}$       (-13,23)
- Indeed  $53 \cdot (-13) + 30 \cdot 23 = -689 + 690 = 1$

# Arithmetic Modulo Composites

---

- Lemma:  $x \in \mathbb{Z}_N$  is invertible in  $\mathbb{Z}_N$  if and only if  $\gcd(x, N) = 1$ .

[proof on the board]

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

# Arithmetic Modulo Composites

- Note: If  $0 \neq x \notin Z^*_N$ , then we can factor  $N$   
 $\rightarrow \gcd(x, N)$  is a non-trivial factor of  $N$
- Size of  $Z^*_N$ ? Denoted by  $\varphi(N) = |Z^*_N|$
- $\varphi$  called Euler's totient function; easily computable:
  1.  $\varphi(p) = p - 1$  if  $p$  prime
  2.  $\varphi(N) = N \cdot \prod_i \left(1 - \frac{1}{p_i}\right)$  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):  
$$\forall a \in \mathbb{Z}_N^*: a^{\varphi(N)} = 1 \pmod{N}$$
- Special case:  $p$  prime, then  $\varphi(p) = p-1$  and  
$$\forall a \in \mathbb{Z}_p^*: a^{\varphi(p)} = a^{p-1} = 1 \pmod{p}$$

# Chinese Remainder Theorem

- **Chinese Remainder Theorem (CRT)** for  $N=pq$ :  
Let  $p \neq q$  be primes and  $N = pq$ . Then for all  $a, b \in \mathbb{Z}$ :  
 $a = b \pmod{N} \Leftrightarrow a = b \pmod{p} \wedge a = b \pmod{q}$
- Further, given  $x_p \in \mathbb{Z}_p$  and  $x_q \in \mathbb{Z}_q$  there exists a unique element  $s \in \mathbb{Z}_N$  such that  
$$s = x_p \pmod{p} \text{ and } s = x_q \pmod{q}.$$
- $s$  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  $x := up x_q + vq x_p \pmod{N}$ .
- **Example: Compute  $1027 \pmod{55}$ ,  $p = 5$ ,  $q = 11$ .**
  - $u = -2$ ,  $v = 1$  and  $1027 = 2 \pmod{5}$ , and  $1027 = 4 \pmod{11}$ .
  - Then:  $1027 = -2 \cdot 5 \cdot 4 + 1 \cdot 11 \cdot 2 = -40 + 22 = -18 = 37 \pmod{55}$ .

# 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  $F = (F(pk, \cdot))_{(pk, *) \in [\text{Gen}(n)]}$ ,  $F(pk, \cdot): \mathcal{M}_{pk} \rightarrow \mathcal{T}_{pk}$  is a family of trapdoor permutations iff
  1.  $\Pr [F(pk, x) = F(pk, x'); (pk, sk) \leftarrow \text{Gen}(n), x \leftarrow_R \mathcal{M}_{pk}, y := F(pk, x), x' \leftarrow A(n, pk, y)]$  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:  $F: \mathbb{Z}_N \rightarrow \mathbb{Z}_N$ 
  - $F(x) = x^e \bmod N$
- 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  $x, y \in \mathbb{Z}$ . Then  $g^x = g^y$  iff  $x = y \bmod \text{ord}(g)$ .

[proof on the board]

- Lemma 2: Let  $(N, e)$  be an RSA public-key,  $d$  the corresponding trapdoor, and  $m \in \mathbb{Z}_N$ . 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  $c \in \mathbb{Z}_N$ , 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  $\rightarrow$  ability to factor?
- Reduction to show that no shortcut exist would look as follows:
  - Efficient alg. for  $e$ 'th roots mod  $N \rightarrow$  efficient alg. for factoring  $N$ .
  - One of the oldest problems in public-key crypto
- Evidence no reduction exists (Boneh-Venkatesan 1998)
  - Algebraic reduction  $\rightarrow$  factoring easy
  - Unlike Diffie-Hellman (conjectured equivalent to DLog (Maurer 1994)).

# Naïve Use of RSA as an Encryption Scheme

---

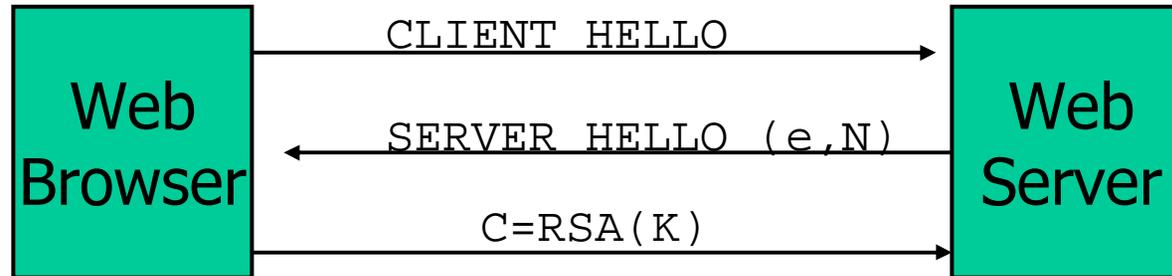
- Naively (wrong):
- Setup:
  - Generate public parameters  $pk = (N, e)$
  - Keep trapdoor  $sk = d$  secret
- $Enc(pk, m) := m^e \bmod N$
- $Decrypt(sk, c) := c^d \bmod N$

# Naïve RSA is not even CPA-secure

---

- Naïve RSA is deterministic  $\rightarrow$  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.,  $K \in \{0, \dots, 2^{64}\}$   
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  $\approx 20\% \rightarrow C / K_1^e = K_2^e \pmod{N}$
- Build table:  $C / 1^e, C / 2^e, C / 3^e, \dots, C / 2^{34e}$ .
- For  $K_2 = 0, \dots, 2^{34}$  test if  $K_2^e$  is in table.
- Attack time for usual values  $e$ :  $\approx 2^{39} \ll 2^{64}$