CS408
Cryptography & Internet Security

Lectures 16, 17:

Security of RSA

El Gamal Cryptosystem

Announcement

- Final exam will be on May 11, 2015 between 11:30am – 2:00pm in FMH 319
  - http://www.njit.edu/registrar/exams/finalexams.php

FROM LAST WEEK: RSA

Z^*_n (multiplicative group of integers mod n)

- Let n>1 be an integer and let $Z^*_n = \{a \in Z_n \mid \gcd(a,n)=1\}$. $(Z^*_n, \bullet)$ is a group, where • is multiplication modulo n. $(Z^*_n, \bullet)$ is called the multiplicative group of Z_n.

- Let p and q be two large primes
- Denote their product n=pq
- $Z^*_n = Z^*_{pq}$ contains all integers in the range [1, n-1] that are relatively prime to both p and q
- The size of Z^*_n is $\phi(pq) = (p-1)(q-1)$
- For every $x \in Z_n^*$, $x^{(p-1)(q-1)} \equiv 1 \bmod n$

RSA Public Key Cryptosystem

Key generation:

- Select 2 large prime numbers of about the same size, p and q
- Compute n = pq, and $\phi(n) = (q-1)(p-1)$
- Select a random integer e, $1 < e < \phi(n)$, s.t. $\gcd(e, \phi(n)) = 1$
- Compute d, $1 < d < \phi(n)$ s.t. $ed \equiv 1 \bmod \phi(n)$

Public key: (n, e)
Private key: d

Note: p and q must remain secret

RSA Public Key Cryptosystem

Encryption

- Obtain the recipient's public key (n,e)
- Represent the message as an integer M , $0 \le M < n$
- Compute $C = M^e \bmod n$
- Send ciphertext C to recipient

Decryption

- Given a ciphertext C, use private key d to recover M: $M = C^d \bmod n$

Why is RSA Encryption Secure?

- Because there is no known efficient algorithm to solve the RSA problem
- The RSA Problem (RSAP):
  Given an integer c, find an integer m such that $m^e \equiv c \pmod{n}$
  (where e, n are defined like in the RSA setting:
  - n is a product of two distinct large primes: n = pq
  - e is a positive integer s.t. gcd(e, ϕ(n))=1
  - n and e are public
  - c $\in$ {0, 1, …, n-1})

  In other words, the RSA problem is that of finding e^{th} roots modulo n
  (this problem is difficult when n, p, q are very large; there are no known efficient algorithms to solve it)

The Integer Factorization Problem (FACTORING)

- Given a positive integer n, find its prime factorization. That is, write $n = p_1^{e1} p_2^{e2} \dots p_k^{ek}$, where p_i are pairwise distinct primes.
- When n is a very large number (e.g., n is represented using thousands of bits), there is no known efficient algorithm to solve FACTORING.
- In particular, let n be a product of two distinct large primes n = pq. Then FACTORING becomes: given n, find p and q.

The Relationship between RSAP and FACTORING

- If an adversary can solve FACTORING, then the adversary can also solve RSAP.

  (we say that the RSA problem reduces to the integer factorization problem)
- Why?

  If adversary knows primes p and q such that n = pq, then the adversary can compute $\phi(n) = (p-1)(q-1)$.

  This allows the adversary to compute d such that $de \equiv 1 \bmod \phi(n)$.
  If adversary knows d, then it can decrypt any RSA ciphertext.

- It is widely believed that if an adversary can solve RSAP, then it can also solve FACTORING (although no one has been able to prove this).
  - If this were true, then RSAP and FACTORING would be equivalent.

Factoring Large Numbers

- RSA Factoring Challenge: a challenge put forward by RSA Labs to encourage research into computational number theory and the practical difficulty of factoring large integers.
- RSA-768 (modulus n has 768 bits) is the largest RSA Challenge number factored to date (December 12, 2009)
  - The effort took almost 2000 2.2GHz-Opteron-CPU years according to the submitters, just short of 3 years of calendar time, using a combination of sophisticated techniques
- Factoring a 1024-bit modulus is still estimated to be about one thousand times harder to factor than a 768-bit one
  - It may be possible to factor a 1024-bit RSA modulus within the next decade
- NIST standards recommend phasing out 1024-bit moduli by 2010, after 2010 the recommended size is 2048 bits.

Attacks on RSA Encryption

Small message space (e.g., m ∈ {yes, no})

- Attacker can try to encrypt all possible plaintext messages until the ciphertext c is obtained.
- Solution: use *salting* (append a random bitstring to the plaintext message before encryption):
  - For a message m = 1001…10, generate a random bitstring r = 11001…11 (at least 64 bits) and encrypt r || m (|| denotes bitstring concatenation)

Attacks on RSA Encryption (continued)

Common modulus attack

Assume two entities A and B use the same modulus n.
A's public key is (n, e_1), B's public key is (n, e_2).

If an attacker eavesdrops an encryption for A: $c_1 = m^{e1} \bmod n$, and also eavesdrops an encryption for B for the same message: $c_2 = m^{e2} \bmod n$, then the attacker can find m!

- if $\gcd(e_1, e_2) = 1$, then the attacker can find integers u and v s.t. $ue_1 + ve_2 = 1$
- The attacker then computes
  $c_1^u c_2^v = (m^{e1})^u (m^{e2})^v = m^{ue1 + ve2} = m$

Thus, each entity should choose its own RSA modulus n.

Attacks on RSA Encryption (continued)

Malleability (multiplicative property)

Let m_1 and m_2 be two plaintext messages and let c_1 and c_2 be their respective RSA encryptions. Observe that:
$(m_1 m_2)^e = m_1^e m_2^e = c_1 c_2 \pmod n$

Assume that an adversary observes a ciphertext $c = m^e \bmod n$. Even if the adversary doesn't know m, it can transform c so that it is still valid upon decryption: e.g., multiply c by $(10)^e$. As a result, recipient gets $c' = (10)^e c = (10)^e m^e = (10m)^e$ and will incorrectly decrypt 10m instead of m.

Definitions of Security

- Semantic Security:
  An adversary should be unable to learn *any partial information* about the plaintext from the ciphertext (besides the length of the plaintext)
- Ciphertext Indistinguishability:
  An adversary should be unable to distinguish pairs of ciphertexts based on the plaintext they encrypt

- These two notions are equivalent, but the latter one is usually used in proofs of security
  - They were proven equivalent under chosen-plaintext (CPA) attacks
- Under chosen-plaintext attacks, these are basic requirements for any modern cryptosystem (IND-CPA)
  - Some cryptosystems achieve stronger security (IND-CCA)

Ciphertext Indistinguishability

- If the adversary knows that a ciphertext results from one of two possible plaintexts, the adversary should not be able to tell which one plaintext is more likely to be the one that was encrypted

Ciphertext Indistinguishability (for public key encryption)

Let (G,E,D) be a public-key encryption scheme.

IND-CPA security game: between Challenger (Chal) and Adversary (Adv)

1. Chal chooses a key pair (Pub,Priv). Pub is made public, but Priv is kept secret and not revealed to Adv.
2. Adv is allowed to perform any number of encryptions (using Pub) or other operations (we say Adv uses Chal as an "encryption oracle")
3. Eventually, Adv chooses two distinct plaintexts of equal length m_0 and m_1 and sends them to Chal
4. Chal chooses a bit $b \in \{0,1\}$ uniformly at random, computes the *challenge* ciphertext $c = E_{Pub}(m_b)$, and sends c back to Adv
5. Adv is allowed to perform any number of encryptions (using Pub) or other operations (i.e., Adv continues to have oracle access to E)
6. Adv outputs a bit b'

The Adversary wins the game if b' = b

Ciphertext Indistinguishability (continued)

- A public-key encryption scheme is IND-CPA secure if every probabilistic polynomial-time (PPT) adversary wins the IND-CPA security game with probability 0.5 + $\varepsilon(k)$, where $\varepsilon(k)$ is a *negligible function* in the security parameter k
  - i.e., the adversary has a negligible "advantage" over random guessing

Deterministic vs. Probabilistic Encryption

- Probabilistic encryption implies the use of *randomness* in encryption: when encrypting the same plaintext several times, it will result in different ciphertexts
  - Each plaintext will map into a large number of possible ciphertexts
- To achieve semantic security, an encryption algorithm ***must*** be probabilistic
- Why can't deterministic public encryption achieve semantic security?
- Textbook RSA encryption is not semantically secure because it is deterministic

(BEGIN) CRYPTOGRAPHIC HASH FUNCTION

Functions

Definition

Given two sets, X and Y, a function f : X → Y (from set X to set Y), is a relation which uniquely associates members of set X with members of set Y.

Cryptographic Hash Functions

- Takes as input a string of any size and outputs a fixed-size string (usually output is much smaller than input)
  - E.g., output can be 160 bits regardless of input size
- A hash is a many-to-one function, so **collisions can happen (but should be unlikely to happen)**.
- Two fundamental properties: **compression and easy to compute**.

Cryptographic Hash Functions (continued)

- Informal requirements
  - One-way (non-invertible)
  - Produces different outputs for different inputs (with high likelihood)

(END) CRYPTOGRAPHIC HASH FUNCTION

Cost of Semantic Security in Public Key Encryption

- In order to have semantic security, some expansion is necessary
  - i.e., the ciphertext must be larger than its corresponding plaintext

A Padding Scheme for Semantically Secure Public-key Encryption

- Given a public-key encryption scheme (G,E,D):
  - to encrypt m, generate a random r, and the ciphertext is $c = (c_1, c_2) = (E_{Pub}(r), H(r) \oplus m)$, where H is a cryptographic hash function
    - The ciphertext consists of a pair of values
  - to decrypt $c = (c_1, c_2)$, compute $m = H(D_{Priv}(c_1)) \oplus c_2$
  - requires an extra random number generation and an XOR operation for each bit

Example of the Padding Scheme

- Example of the Padding Scheme for RSA

  Public key: (n,e)

  Private key: d

  To encrypt m, generate random r and compute ciphertext $(c_1, c_2) = (r^e \bmod n, m \oplus H(r))$

  To decrypt a ciphertext (c_1, c_2), compute $r = c_1^d \bmod n$, and $m = c_2 \oplus H(r)$

  For a 1024-bit modulus n, to encrypt a 128-bit message, the ciphertext has 1024+128 bits

OAEP

- *M. Bellare and P. Rogaway, Optimal asymmetric encryption, Advances in Cryptology - Eurocrypt '94, Springer-Verlag (1994), 92-111.*
- [Optimal Asymmetric Encryption Padding (OAEP)]: method for encoding messages.
- To encode a message $m \in \{0,1\}^u$:
  - choose random $r \in \{0,1\}^t$
  - use two cryptographic hash functions: $H: \{0,1\}^u \rightarrow \{0,1\}^t$ and $G: \{0,1\}^t \rightarrow \{0,1\}^u$
  - compute $m \oplus G(r) \,||\, r \oplus H(m \oplus G(r))$
- To encrypt m, compute $\mathbf{E_K[m \oplus G(r) \,||\, r \oplus H(m \oplus G(r))]}$, where E_K is a trapdoor one-way permutation function (such as RSA encryption).
  - So, to encrypt we do $[m \oplus G(r) \,||\, r \oplus H(m \oplus G(r))]^e \bmod n$
- OAEP is provably IND-CPA secure when H and G are modeled as random oracles and E_K is a trapdoor one-way permutation.

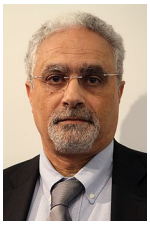
El Gamal Cryptosystem

Reminder: Z^*_p

- Let p be a prime integer and let Z^*_p be the set {1, 2, …, p-1}.
  $(Z^*_p, \bullet)$ is a group, where • is multiplication modulo p.
- If p is a prime, then $(Z^*_p, \bullet)$ is always cyclic (it has primitive roots)

ElGamal Public Key Cryptosystem

- Proposed by Taher ElGamal in 1985
- Message expansion: the ciphertext is twice as big as the original message
- Uses randomization, each message has p-1 possible different encryptions



ElGamal Public Key Encryption

Key Generation

Each entity A should do the following:

1. Generate a large random prime p. Find a generator g of the multiplicative group Z^*_p.
2. Select a random integer x, $1 \le x \le p-2$, and compute $y = g^x \bmod p$
3. A's public key is (p, g, y)

   A's private key is x

ElGamal Public Key Encryption

Encryption (B encrypts a message m for A):

1. Obtain A's authentic public key (p, g, y)
2. Represent the message as an integer m in the range $[0 .. p-1]$
3. Select a random integer r, $1 \le r \le p-2$
4. Compute $c_1 = g^r \bmod p$, $c_2 = m \cdot y^r \bmod p$
5. Send ciphertext $c = (c_1, c_2)$ to A

Decryption (A decrypts a ciphertext $c = (c_1, c_2)$ from B):

1. Compute $m = c_2 \cdot c_1^{-x} \bmod p$

Why is decryption correct?

$c_2 \cdot c_1^{-x} = m \cdot y^r \cdot (g^r)^{-x} = m \cdot g^{xr} \cdot g^{-xr} = m$

Security of ElGamal Encryption

- Security of ElGamal encryption relies on the difficulty of two problems:
  - Computational Diffie-Hellman Problem (CDH)
  - Decisional Diffie-Hellman Problem (DDH)

  (will see these later in the course)

- Also, one cannot find x from g^x mod p
  - this is called the Discrete Log Problem (DLP)

- Note that ElGamal produces randomized ciphertexts
  - Two identical plaintexts will be encrypted into different ciphertexts

Recommended Reading

- Chapter 6.1, 6.2, 6.7 for RSA
- Chapter 7.5 for ElGamal