Semantic Security of RSA

Murat Kantarcioglu

Semantic Security

- As before our goal is to come up with a public key system that protects against more than "total break"
  - We want our system to be secure against
    - "total break" (i.e., can recover the private key)
    - "partial break" (i.e., can decrypt messages without knowing the key)
  - Also we want adversary to not to distinguish between any given ciphertexts!

Semantic Security (IND-CPA for Public Key Encryption)

- The IND-CPA game

Challenger | Adversary

picks a random key pair (K, K^{-1}), and picks random $b \in \{0,1\}$

$\xrightarrow{K}$

$\xleftarrow{M_0, M_1}$ picks M_0, M_1 of equal length

$\xrightarrow{C = E_K[M_b]}$

$\xleftarrow{b' \in \{0,1\}}$

Attacker wins game if b=b'

3

Semantic Insecurity of the RSA

- RSA encryption is not semantically secure because it is deterministic
- The encryption function $f(x)=x^e \bmod n$ leaks information about x !
  - it leaks the Jacobi symbol of x

$$\left(\frac{x^e}{N}\right)=\left(\frac{x^e}{p}\right)\left(\frac{x^e}{q}\right)=\left(\frac{x}{p}\right)\left(\frac{x}{q}\right)=\left(\frac{x}{N}\right)$$

  - it also leaks the whether x is a QR or not, but this is not a concern, why?

4

Partial Information Related to RSA function

- RSA does not leak certain type of partial information
- Given $y=x^e$ mod n, computing the parity(y) (i.e. parity(y)=0 if x is even parity(y)=1 if x is odd) is equivalent to inverting RSA.
- Given $y=x^e$ mod n, computing half(y) (i.e., half(y)=0 if $0 \leq x < N/2$ and half(y)=1 if $n/2 < x \leq n-1$) is equivalent to inverting RSA.

Reduction of half() to inverting RSA

- Note that for RSA $E_K(x_1)E_K(x_2) = x_1^e . x_2^e \bmod n = E_K(x_1 . x_2)$
- Also note that
  - half(y.$E_k(2^i)$ mod n)=half($E_k(x.2^i)$)
- Observe that half($E_k(2x)$)=0 iff $x \in [0,n/4) \cup [n/2, 3n/4)$ (why?)
- Using this idea, we can create an algorithm for inverting RSA

Oracle RSA Decryption Using Half()

$$
\begin{array}{l}
k \leftarrow \lfloor \log_2(n) \rfloor \\
\text{for } i \leftarrow 0 \text{ to } k\ \{ \\
\qquad h_i \leftarrow \text{half}(n,\ e,\ y);\ y \leftarrow y.2^e \bmod n \\
\ \ \} \\
lo \leftarrow 0;\ hi \leftarrow n \\
\text{for } i \leftarrow 0 \text{ to } k\ \{ \\
\qquad mid \leftarrow (hi + lo)/2; \\
\qquad \text{if } (h_i = 1) \text{ then } lo \leftarrow mid \text{ else } hi \leftarrow mid \\
\} \\
\text{return } \lfloor hi \rfloor
\end{array}
$$

Example

- Consider n=1457 e=779, ciphertext y=722
- Assume half() returns the following h_i values
- h_0=1, h_1 =0, h_2=1,h_3=0,h_4=1,h_5=1,h_6=1 h_7=1, h_8=1, h_9= 0, h_{10} =0
- Following the algorithm will find the plaintext as 999.

Parity()

- Similar ideas work for the parity() function as well. Note that

$$half\,(y) = parity\,(y.E_k(2) \bmod n)$$
$$parity\,(y) = half\,(y.E_k(2^{-1}) \bmod n)$$

The Goldwasser-Micali Probablistic Encryption Scheme

- First provably semantically secure public key encryption scheme, security based on the hardness of determining whether a number x is a QR modulo n, when the factoring of n is unknown and the Jacobi symbol $\left(\frac{x}{n}\right)$ is 1
- Encryption is bit by bit
- For each bit in the plaintext, the ciphertext is one number in Z_n*, expansion factor is 1024 when using 1024 moduli

10

The Goldwasser-Micali Probablistic Encryption Scheme

- Key generation
  - randomly choose two large equal-size prime number p and q, pick a random integer y such that

$$\left(\frac{y}{p}\right)=\left(\frac{y}{q}\right)=-1$$

  - public key is (n=pq, y)
  - private key is (p,q)
- Encryption
  - to encrypt one bit b, pick a random x in Z_n*, and let $C=x^2y^b$
  - that is, $C=x^2$ when b=0, and $C=x^2y$ when b=1

11

The Goldwasser-Micali Probablistic Encryption Scheme

- Consider the Jacobi symbol of the ciphertext C

$$\left(\frac{x^2}{n}\right)=\left(\frac{x^2}{p}\right)\left(\frac{x^2}{q}\right)=1\bullet 1=1 \qquad \left(\frac{yx^2}{n}\right)=\left(\frac{yx^2}{p}\right)\left(\frac{yx^2}{q}\right)=-1\bullet -1=1$$

- Consider whether the ciphertext C is QR modulo n
  - C is QR iff. the plaintext bit b is 0
- Decryption:
  - knowing p and q s.t. n=pq, one can determine whether x is QR modulo n and thus retrieves the plaintext (how?)

12

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 (why?)
  - the Goldwasser-Micali encryption scheme generate ciphertexts of size 1024m
  - suppose that all plaintexts have size m, what is the minimal size of ciphertexts to have an adequate level of security (e.g., takes 2^t to break the semantic security)?

13

A Padding Scheme for Semantically Secure Public-key Encryption

- Padding Scheme 1: given a public-key encryption scheme **E**,
  - to encrypt x, generates a random r, the ciphertext is (**f(r)**, H(r)$\oplus$x) , where H is a cryptographic hash function
  - to decrypt (y_1,y_2), one compute $H(\mathbf{f}^{-1}(y_1))\oplus y_2$
  - requires an extra random number generation and an XOR operation for each bit

14

Example of the Padding Scheme

- Example of the Padding Scheme for RSA
  - Public key: (n,e),
  - The ciphertext for x is (r^e mod n, x⊕H(r))
  - To decrypt a ciphertext (y_1, y_2), compute r= y_1^d mod n, and x= y_2⊕H(r)
  - To encrypt a 128-bit message, the ciphertext has 1024+128 bits

15

Why is This Padding Scheme Secure?

- This padding scheme is provably IND-CPA secure, when H is modeled as a random oracle (i.e., H is a random function) and **f** is a trapdoor one-way permutation
  - to learn any information about x from (**f(r)**, x⊕H(r)), one has to learn some information about H(r)
  - as H is a random function, the only way to learn any information about H(r) is to evaluate H at the point r
  - an adversary who can learn anything about x thus knows r
  - the adversary can thus invert **f**

16

Random Oracle Model

- Random Oracle Model
  - Use hash function H in your design
  - Give security proofs assuming that H is a random function. Replace H with some cryptographic hash function in practice.
- Random Oracle Assumption is
  - not valid in general
  - feasible and efficient in practice

Proof Sketch

- Assuming the existence of algorithm D() that can distinguish between the two ciphertexts with probability 0.5 + ϵ with at most q query queries to random oracle, we will show that we can define an algorithm that can invert given trapdoor function f with probability at least ϵ.
- In other words, if f is a secure trapdoor function then above scheme is secure in the random oracle model.

Proof Sketch

- Consider the following simulator simH() for random oracle H(). Given the y=f(x) that we want to invert, random y_2 , two plaintexts x_1, x_2
- SimH(r){

```
    if r is queried before in the i^th query then return Glist[i] ;
    else {
         if f(r)=y then {
              g ← y2⊕ xj for random j∈{1,2}; }
         else {
              g ← r for some random r; }
         l ← l+1; Glist[l] ←g; Rlist[l] ←r; }
   return g
 }
```

Proof Sketch

Invert(y)

{

y_1←y; y_2 ←r for random r;

Run D(x_1,x_2,(y_1,y_2)) for arbitrary $x_1 \neq x_2$

Answer D's queries to H using simH() until D stops.

if f(Rlist[i])=y for some i then return Rlist[i]

}

Proof Sketch

- Let us compute the success probability of invert(y) given that D() is successful with at least probabiliy 0.5+ ϵ

$$
\begin{aligned}
\Pr[D() \text{ succeeds}] = \\
& \Pr[D() \text{ succeeds} \mid f^{-1}(y) \in \text{Rlist}].\Pr[f^{-1}(y) \in \text{Rlist}] + \\
& \Pr[D() \text{ succeeds} \mid f^{-1}(y) \notin \text{Rlist}].\Pr[f^{-1}(y) \notin \text{Rlist}] \\
& \le \Pr[f^{-1}(y) \in \text{Rlist}] + 0.5 \Pr[f^{-1}(y) \notin \text{Rlist}] \\
& \le \Pr[f^{-1}(y) \in \text{Rlist}] + 0.5 \\
& \le \Pr[\text{inverse } (y) \text{succeed s}] + 0.5
\end{aligned}
$$

Proof Sketch

- If Distinguish algorithm D() runs with time t_1 using at most q random oracle queries, f() requires t_2 then Inverse() runs with time $t_1 + O(q^2 + qt_2)$
- Note Inverse()
  - calls f function O(q) times
  - calls Distinguish function once
  - each call to simh() may require search over list size O(q)

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.
- Uses one trapdoor permutation functions f and two hash functions: H: $\{0,1\}^m \rightarrow \{0,1\}^t$ and G: $\{0,1\}^t \rightarrow \{0,1\}^m$
- To encrypt $x \in \{0,1\}^m$, chooses random $r \in \{0,1\}^t$ and computes $f[x \oplus G(r) \,||\, r \oplus H(x \oplus G(r))]$
- How to decrypt given y?
- Security intuitions?

23

OAEP (cont.)

- OAEP: $f[x \oplus G(r) \,||\, r \oplus H(x \oplus G(r))]$
  - H: $\{0,1\}^m \rightarrow \{0,1\}^t$ and G: $\{0,1\}^t \rightarrow \{0,1\}^m$
- OAEP is provably IND-CPA secure when H and G are modeled as random oracles and f is a trapdoor one-way permutation.
- A ciphertext has size n ($\approx$1024 for RSA)
- The padding size t should be s.t. 2^t computing time is infeasible, why?
  - t $\approx$128
- The plaintext size m can be up to 1024-128=896
- Expansion is optimal

24