

Cryptography CS 555

Lecture 15



The SRA Mental Poker Protocol and Semantic Security of RSA

Lecture Outline

- Commutative encryption
- The SRA Mental Poker Protocol
- Semantic Security of RSA



The Mental Poker Problem

- Alice and Bob want to play poker, we need a way to deal 5 cards to each of Alice and Bob so that
 - Alice's hand of 5 cards does not overlap with Bob's hand
 - Neither Alice nor Bob can control which cards they each get
 - Neither Alice nor Bob knows the other party's hand
 - Both hands should be random provided one party follows the protocol
- First solution due to Shamir, Rivest, and Adelman in 1980
 - uses commutative encryption schemes

Commutative Encryption

Definition: an encryption scheme is commutative if

$$E_{K_1}[E_{K_2}[M]] = E_{K_2}[E_{K_1}[M]]$$

- Given an encryption scheme that is commutative, then
 $D_{K_1}[D_{K_2}[E_{K_1}[E_{K_2}[M]]] = M$
- Most symmetric encryption scheme (such as DES and AES) are not commutative

Examples of Commutative Encryption Schemes

- Pohlig-Hellman Exponentiation Cipher with the same modulus p
 - encryption key is e , decryption key is d , where $ed \equiv 1 \pmod{p-1}$
 - $E_{e_1}[M] = M^{e_1} \pmod{p}$ and $D_{d_1}[C] = C^{d_1} \pmod{p}$
 - $E_{e_1}[E_{e_2}[M]] = M^{e_1 e_2} = E_{e_1}[E_{e_2}[M]] \pmod{p}$

Examples of Commutative Encryption Schemes

- The SRA encryption scheme
 - Alice and Bob share $n=pq$ and they both know p and q
 - Alice has encryption key e_1 and decryption key d_1 s.t. $e_1 \bullet d_1 = 1 \pmod{(p-1)(q-1)}$
 - $E_{e_1}[M] = M^{e_1} \pmod{n}$
 - Bob has e_2, d_2 s.t. $e_2 \bullet d_2 = 1 \pmod{(p-1)(q-1)}$
 - Also a commutative encryption scheme
 - Essentially RSA, except that e is kept private

The SRA Mental Poker Protocol

Setup: Alice and Bob share M_1, M_2, \dots, M_{52} denote the 52 cards, $n=pq$, p , and q . Alice has e_1, d_1 and Bob has e_2, d_2

Protocol:

- Alice encrypts M_1, M_2, \dots, M_{52} using her key, i.e., computes $C_j = M_j^{e_1} \pmod n$ for $1 \leq j \leq 52$, randomly permute them and send the ciphertexts to Bob
- Bob picks 5 cards as Alice's hand and sends them to Alice
- Alice decrypts them to get his hand
- Bob picks 5 other cards as his hand, encrypts them using his key, and sends them to Alice
- Alice decrypts the 5 ciphertexts and sends to Bob
- Bob decrypts what Alice sends and gets his hand
- Both Alice and Bob reveals their key pairs to the other party and verify that the other party was not cheating. (Why need this step?)

“Security Analysis” of the Protocol

- Bob sees 52 random ciphertexts, he doesn't know which ciphertext corresponds to which cards.
- Bob can only randomly pick Alice's hand, and Bob does not know what Alice's hand is.
- Bob can only randomly pick his hand, and Alice doesn't know Bob's hand, as it is encrypted under Bob's key.
- **This is not a security proof.**

An Attack on the SRA Mental Poker Protocol

- The encryption function $f(x)=x^e \bmod n$ leaks information about x !
 - $f(x)$ is QR modulo n iff. x is QR modulo n
 - $x^e \in \text{QR}_n \Leftrightarrow x^e \in \text{QR}_p$ and $x^e \in \text{QR}_q \Leftrightarrow x \in \text{QR}_p$ and $x \in \text{QR}_q$
 $\Leftrightarrow x \in \text{QR}_n$
 - Why this matters in the SRA mental poker protocol?
 - suppose that the cards that are QR are mostly large cards, and the cards that are not QR are mostly small cards, then Bob can choose large cards for him and small cards for Alice
 - Even when $f(x)$ is a trapdoor one-way function, some bits about x can be leaked.

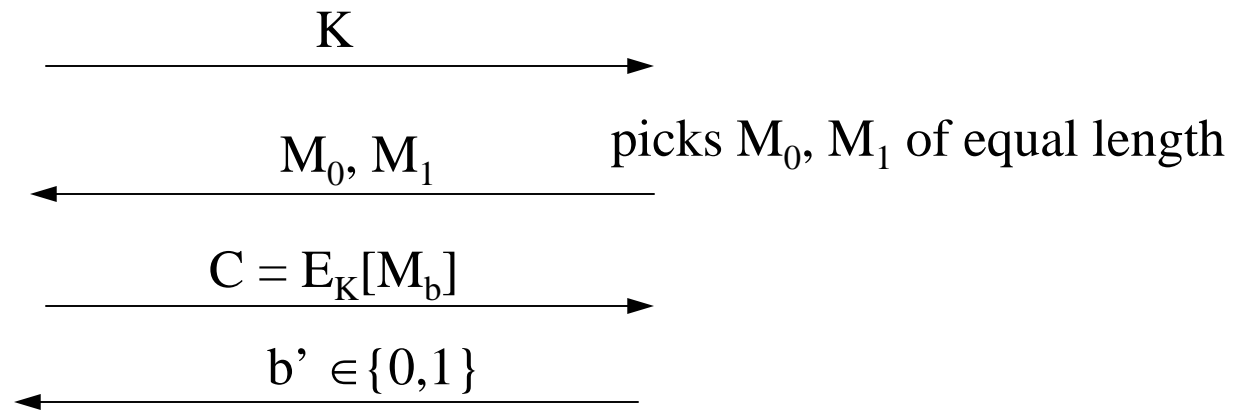
Semantic Security (IND-CPA for Public Key Encryption)

- The IND-CPA game

Challenger

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

Adversary



Attacker wins game if $b=b'$

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?**

The Goldwasser-Micali Probabilistic 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

The Goldwasser-Micali Probabilistic 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 $b=1$

The Goldwasser-Micali Probabilistic Encryption Scheme

- Consider the Jacobi symbol of the ciphertext C

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

- 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?)

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)?

A Padding Scheme for Semantically Secure Public-key Encryption

- Padding Scheme 1: given a public-key encryption scheme \mathbf{E} ,
 - to encrypt x , generates a random r , the ciphertext is $(\mathbf{E}_k[r], H(r) \oplus x)$, where H is a cryptographic hash function
 - to decrypt (y_1, y_2) , one compute $H(\mathbf{D}_k[y_1]) \oplus y_2$
 - 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) ,
 - The ciphertext for x is $(r^e \bmod n, x \oplus H(r))$
 - To decrypt a ciphertext (y_1, y_2) , compute $r = y_1^d \bmod n$, and $x = y_2 \oplus H(r)$
 - To encrypt a 128-bit message, the ciphertext has 1024+128 bits

Why is This Padding Scheme Secure?

- This padding scheme is provably IND-CPA secure, when H is modelled as a random oracle (i.e., H is a random function) and \mathbf{E}_K is a trapdoor one-way permutation
 - to learn any information about x from $(\mathbf{E}_K[r], x \oplus H(r))$, 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 \mathbf{E}_K

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 E_K 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 $E_K[x \oplus G(r) || r \oplus H(x \oplus G(r))]$
- How to decrypt given y ?

OAEP (cont.)

- OAEP: $E_K[x \oplus G(r) \parallel r \oplus H(x \oplus G(r))]$
 - $H: \{0,1\}^m \rightarrow \{0,1\}^t$ 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 E_K is a trapdoor one-way permutation.
- A ciphertext has size n (≈ 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

Summary

- The SRA Mental Poker Protocol
- The need for semantic security
- The Goldwasser-Micali probabilistic encryption scheme
- Padding schemes for asymmetric encryption



Next ...

- The Discrete Log Problem
- Diffie-Hellman Key Exchange
- El Gamal Encrytion

