T-79.5501 Cryptology

Lecture 10 (April 3, 2007):

- PKCS#1 and Bleichenbacher's attack (slides)
- OAEP (Slides and Stinson: Cryptosystem 5.3 Semantically Secure Public-key Cryptosystem)
- Rabin Oblivious Transfer
- 1-out-of-2 oblivious transfer
- The ElGamal cryptosystem (Stinson 6.1)
- Algorithms for the discrete logarithm problem (Shanks' algorithm Sec. 6.2.1, Pohlig-Hellman algorithm, Sec. 6.2.3

PKCS#1

PKCS#1 v 1.5 before it was corrected:

$$EB = 00 \,||\, BT \,||\, PS \,||\, 00 \,||\, B$$

BT	block type: 00, 01, tai 02. (In public key encryption $BT = 02$)
PS	padding string
B	data block

The leftmost byte 00 guarantees that the plaintext after conversion to an integer is less than the RSA modulus n.

PKCS#1 v 1.5

Bleichenbacher's attack :

- Bob sees ciphertext C obtained using RSA encryption with plaintext M formatted according to PKCS #1. Bob knows that then plaintext M is : $M = C^{a}$ mod n, where a is the private exponent and n is the modulus.
- Bob selects integers S and computes $C' = CS^{b}$ mod n and sends C' *to* Alice.
- Alice decrypts $(C')^{a}$ mod $n = MS$ mod n and checks if formatting of MS is as expected according to PKCS#1. If not, Alice tells it to Bob and rejects the message.
- If Alice accepts the message Bob knows that the first two bytes of MS mod n is equal to 00 || 02 .
- Then Bob knows that the following inequality holds:

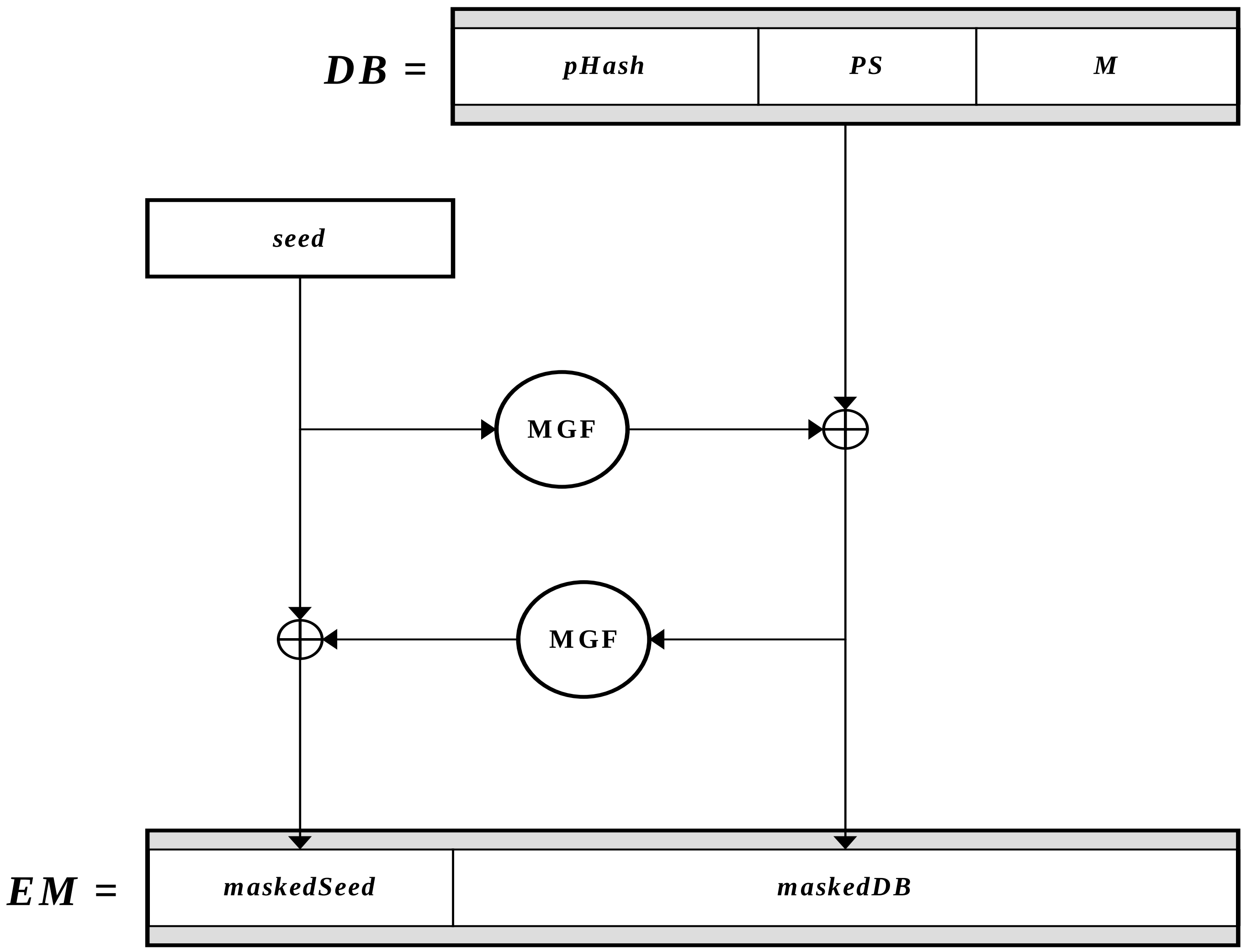
$$2L \leq MS \bmod n < 3L$$

where $L = 2^{8(k-2)}$ and k is the byte length of the RSA-modulus n .

- By collecting a number (~ 2^{20}) of such inequalities Bob can determine the value of M.

PKCS#1 v 2.1 EME-OAEP

Based on Bellare and Rogaway's Optimal Asymmetric Encryption scheme (1994)



Rabin's Oblivious Transfer

Two players: sender (Alice) and receiver (Bob)

Goal: Alice has one bit. Bob is allowed to try once to get the bit. His success probability is ½ . Alice does not know, if Bob gets the bit or not.

Protocol:

1. Alice sets up an RSA cryptosystem: $p, q, n, a, b,$ with $ab \equiv 1 \bmod \Phi(n)$.
2. Alice encrypts the bit s, to get $c = \{\text{encode}(s)\}^b \bmod n,$ and sends c, b and n to Bob.
3. Bob selects x, $0 < x < n$, at random, computes $y = x^2 \bmod n$, and sends y to Alice.
4. Alice finds the four square roots of y and picks one, say z, of them and sends it to Bob.
5. If $z \neq \pm x \bmod n$, Bob can factor n, compute $a = b^{-1} \bmod \Phi(n)$, and decrypt c, with probability ½. Alice does not know if $z \neq \pm x \bmod n$.

1-out-of-2 Oblivious Transfer

Two players: sender (Alice) and receiver (Bob)

Goal: Alice has two secret bits. Bob is allowed two see exactly one of them. Alice does not know, which of the two bits Bob gets.

Alice's inputs: two bits a_0 and a_1

Bob's input: one bit s

Protocol: OT(a_0, a_1; s)

Output to Alice: nothing

Output to Bob: $a_s = (s \oplus 1)\, a_0 \oplus s\, a_1$

We will see later how to implement OT(a_0, a_1; s) using the RSA.

1-out-of-2 Oblivious Transfer Application to Match Making

Two players: sender (Alice) and receiver (Bob)

Goal: Alice has a bit. Bob has a bit. A player sets its bit equal to one, if he/she is willing to make a match. Else, a player sets its bit equal to zero. A player should not get another player's bit unless he/she is willing to make a match.

Alice's inputs: one bit x

Bob's input: one bit y

Protocol: OT(0, x; y)

Output to Bob: $(y \oplus 1)0 \oplus xy = xy$.

Output to Alice: xy (Bob shows this to Alice).

Note. We shall assume that the players commit to their input bits, and that Bob is honest and shows the right value of xy to Alice ("semi-honest" model).

1-out-of-2 OT using RSA

Protocol:

1. Alice sets up RSA cryptosystem: p, q, n, a, b, with ab ≡ 1 mod ϕ(n), and sends n and b to Bob.

Assumption: Hard-core bit for the RSA function: For randomly chosen x, given y, n, b, where $y = x^b$ mod n finding the lsb of x is essentially as hard as finding all of x (see also Stinson, Section 5.9)

2. Bob selects a random m with lsb r_s and computes ciphertext $c_s = m^b$ mod n. Bob selects c_{1-s} at random, and sends c_s and c_{1-s} , that is, c_0 and c_1 to Alice.
3. Alice decrypts c_0 and c_1 and gets the lsb:s r_0 and r_1 of the decryptions. She then conceals the bits a_0 and a_1 by computing $a'_0 = r_0 + a_0$ (mod 2) and $a'_1 = r_1 + a_1$ mod 2, and sends a'_0 and a'_1 to Bob.
4. Bob then gets a_s from a'_s as he knows r_s. Alice does not know s.

Problem: How to make sure that Bob selects c_{1-s} at random and not similarly as c_s ?