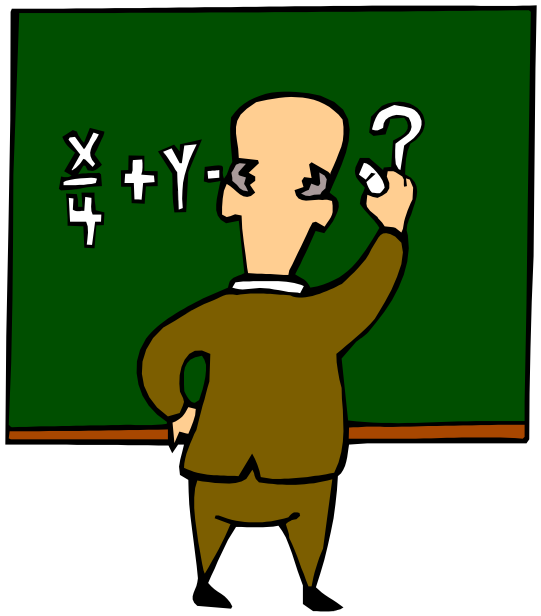
Cristina Nita-Rotaru

CS355: Cryptography

Lecture 13: Attacks against RSA.OAEP. Testing for primality

Math-Based Key Recovery Attacks

- Three possible approaches:
  1. Factor n = pq
  2. Determine $\Phi(n)$
  3. Find the private key d directly



- All the above are equivalent to factoring n

Knowing $\Phi(n)$ Implies Factorization

- Knowing both n and $\Phi(n)$, one knows

$$n = pq$$

$$\Phi(n) = (p-1)(q-1) = pq - p - q + 1$$

$$= n - p - n/p + 1$$

$$p\Phi(n) = np - p^2 - n + p$$

$$p^2 - np + \Phi(n)p - p + n = 0$$

$$p^2 - (n - \Phi(n) + 1)\, p + n = 0$$

- There are two solutions of p in the above equation.
- Both p and q are solutions.

Factoring Large Numbers

- **RSA-640 bits, Factored Nov. 2 2005**
- **RSA-200 (663 bits) factored in May 2005**
- **RSA-768 has 232 decimal digits and was factored on December 12, 2009, latest.**
- Three most effective algorithms are
  - quadratic sieve
  - elliptic curve factoring algorithm
  - number field sieve

Decryption attacks on RSA

RSA Problem: Given a positive integer n that is a product of two distinct large primes p and q, a positive integer e such that gcd(e, (p-1)(q-1))=1, and an integer c, find an integer m such that $m^e \equiv c \pmod{n}$

- widely believed that the RSA problem is computationally equivalent to integer factorization; however, no proof is known

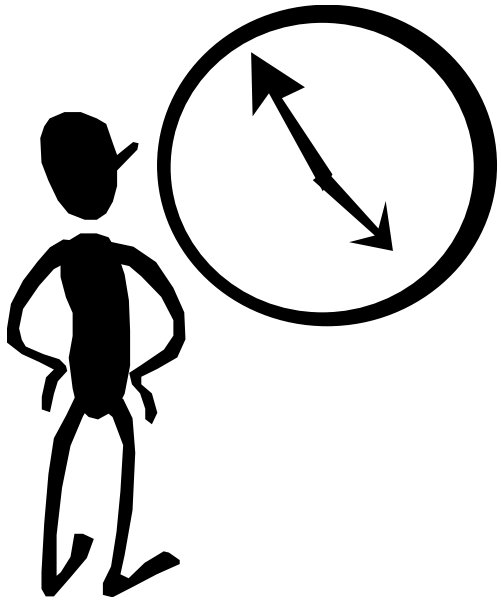
The security of RSA encryption's scheme depends on the hardness of the RSA problem.

Summary of Key Recovery Math-based Attacks on RSA

- Three possible approaches:
  1. Factor n = pq
  2. Determine $\Phi(n)$
  3. Find the private key d directly
- All are equivalent
  - finding out d implies factoring n
  - if factoring is hard, so is finding out d

Finding d: Timing Attacks

- *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems (1996), Paul C. Kocher*
- By measuring the time required to perform decryption (exponentiation with the private key as exponent), an attacker can figure out the private key
- Possible countermeasures:
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations



Timing Attacks (cont.)

- Is is possible in practice? YES.

OpenSSL Security Advisory [17 March 2003]
Timing-based attacks on RSA keys
==========
OpenSSL v0.9.7a and 0.9.6i vulnerability

Researchers have discovered a timing attack on RSA keys, to which OpenSSL is generally vulnerable, unless RSA blinding has been turned on.

RSA blinding: the decryption time is no longer correlated to the value of the input ciphertext

Instead of computing c^d mod n, choose a secret random value r and compute $(r^e c)^d$ mod n.

A new value of r is chosen for each ciphertext.

Quadratic Residues

a is a quadratic residue modulo p if $\exists$ b $\in Z_p^*$ such that **$b^2 \equiv a \bmod p$**, otherwise a is a nonquadratic residue

Q_p is the set of all quadratic residues

$\overline{Q}_p$ is the set of all nonquadratic residues

$4^2 \equiv 6 \bmod 10$ so 6 is a quadratic residue (mod 10).

- If p is prime there are (p-1)/2 quadratic residues in Z_p^*, $|Q_p| = (p-1)/2$
- If $a^{(p-1)/2} \equiv 1 \bmod p$ then a is a quadratic residue (if -1 then r is a nonquadratic residue)

Legendre Symbol

- Let **p be an odd prime** and a an integer. The Legendre symbol is defined

$$\left(\frac{a}{p}\right) = \begin{cases} 0, \text{ if } p \mid a \\ 1, \text{ if } a \in Q_p \\ -1, \text{ if } a \in \overline{Q}_p \end{cases}$$

Jacobi Symbol

- let **n ≥ 3 be a composite** odd with prime factorization

$$n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$$

- the Jacobi symbol is defined to be

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \ldots \left(\frac{a}{p_k}\right)^{e_k}$$

The Jacobi symbol can be computed without factoring n

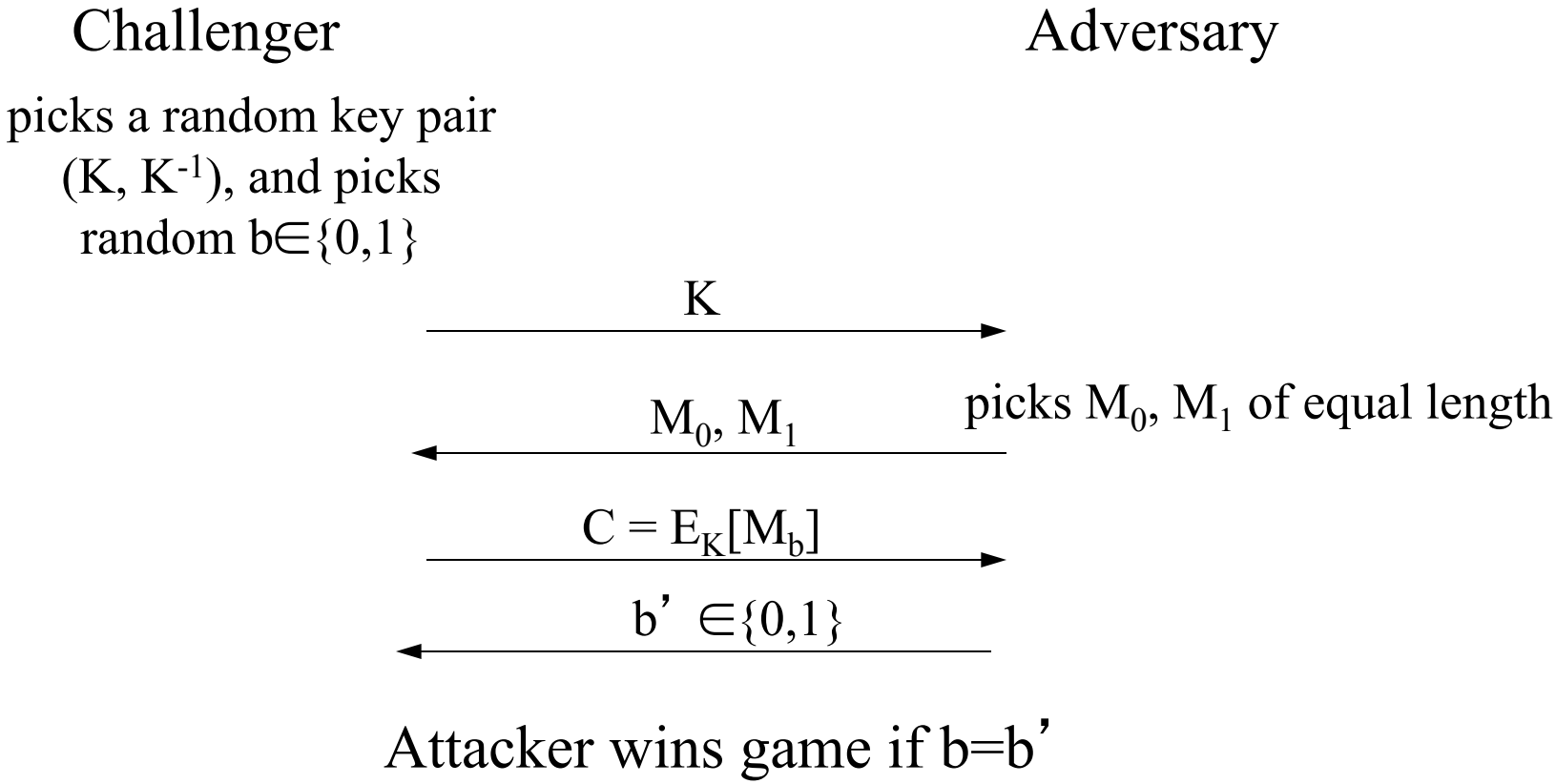
Semantic Insecurity of the RSA

- 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, so it allows an attacker to distinguish between ciphertexts

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

Reminder: Semantic Security (IND-CPA)

- The IND-CPA game



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 **E**,
  - to encrypt x, generates a random r, the ciphertext is (y1 = EK[r], y2 = H(r)⊕x) , where H is a cryptographic hash function
  - to decrypt (y_1,y_2), one compute H(DK[y1]) ⊕y2
  - 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 mod n, $x \oplus H(r)$)

  To decrypt a ciphertext (y_1, y_2), compute r= y_1^d mod n, and x= $y_2 \oplus H(r)$

  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.
- 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 $\mathbf{E_K[x \oplus G(r) \,||\, r \oplus H(x \oplus G(r))]}$
- OAEP is provably IND-CPA secure when H and G are modeled as random oracles and E_K is a trapdoor one-way permutation.

Cristina Nita-Rotaru

Testing for primality.

Pseudoprime Numbers

- **Fermat pseudoprime base a**: a composite number p which satisfies the Fermat Theorem

$$a^{p-1} \equiv 1 \pmod{p}$$

- If a = 2: p is called simply a pseudoprime.
- If p is a pseudoprime, so is 2p-1.
- **Carmichael number**: a pseudoprime for every base *a* relatively prime to it.
- The use of a Carmichael number instead of a prime factor in the modulus of an RSA cryptosystem is likely to make the system fatally vulnerable, so such numbers may be detected.

Strong Pseudoprimes

- **Strong pseudoprime** to a base a is an odd composite number p with $p-1 = d*2^r$, d is odd, such that

$$a^d \equiv 1 \pmod{p}$$

Testing for Primality

- There are applicationsthat need large prime numbers.
- The usual algorithm to generate prime numbers is to generate random odd numbers and test them for primality.
- Primality testing is easier than prime factorization, and is P-class.

How can we tell if a number is prime or not without factoring the number?

Complexity

- **Complexity theory**: mathematical discipline that classifies problems based on the difficulty to solve them.
- **P-class** (polynomial-time): class of questions for which some algorithm can provide an answer in polynomial time. i.e. number of steps needed to solve a problem is bounded by some power of the problem's size.
- **NP-class** (nondeterministic polynomial-time): Class of questions for which there is no known way to find an answer in polynomial time, but if the answer is provided, it can be verified in polynomial time, i.e. it permits a nondeterministic solution and the number of steps to verify the solution is bounded by some power of the problem's size.

P vs NP problem

- **P ? NP**: Asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer

- Quickly means polynomial

NP-Complete

- **NP-complete**: problems to each of which any other **NP**-problem can be reduced in polynomial time, and whose solution may still be verified in polynomial time, i.e. , any NP problem can be transformed into any of the NP-complete problems.
- An **NP**-complete problem is at least as "tough" as any other problem in **NP**.

Integer Factoring

- **Computation**: determine the prime factorization of a given integer
- **Decision**: determine if a given integer has a factor less than k
- No efficient integer factorization algorithm is known
- Integer factorization is in NP

Testing for Primality

Theorem

Composite numbers have a divisor below their square root.

Proof idea:

n composite, so n = ab, $0 < a \leq b < n$, then $a \leq \sqrt{n}$, otherwise we obtain ab > n (contradiction).

Algorithm 1

```
for (i=2, i < sqrt(n) + 1); i++) {
      If i a divisor of n {
            n is composite
      }
}
n is prime
```

Running time is O(sqrt(n))

Rabin-Miller Test

- Primality test that provides an efficient probabilistic algorithm for determining if a given number is prime.
  - Given an odd integer $p = 2^d s + 1$, with s odd.
  - Choose a random integer a, $1 \leq a \leq p-1$.
  - If $a^d \equiv 1 \bmod p$ then p passes the test.
- A prime will pass the test for all a, while a composite number passes the test for at most 1/4 of the possible bases a.
- If N multiple independent tests are performed on a composite number, then the probability that it passes each test is or less than $1/4^N$.
- The test is fast, very used in practice.