Attacks on RSA

RSA Security

- possible approaches to attacking RSA are:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing ø(n), by factoring modulus n)
  - Side channel attacks (on running of decryption)
  - chosen ciphertext attacks (given properties of RSA)

Factoring Problem

- mathematical approach takes 3 forms:
  - factor $n=p.q$, hence compute $ø(n)$ and then d
  - determine $ø(n)$ directly and compute d
  - find d directly [$e.d \equiv 1 \pmod{ø(n)}$]

- currently believe all equivalent to factoring
  - have seen slow improvements over the years
    - as of May-05 best is 200 decimal digits (663) bit with LS
  - 1024 bit RSA is no more secure
  - currently assume 2048-4096 bit RSA is secure
    - ensure p, q of similar size and matching other constraints

Chosen Ciphertext Attacks

- RSA is vulnerable to a Chosen Ciphertext Attack (CCA).

- - Adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key.
- - The adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, yield information needed for cryptanalysis.
- - Can counter simple attacks with random pad of plaintext. More sophisticated variants need to modify the plaintext using a procedure known as optimal asymmetric encryption padding (OAEP).

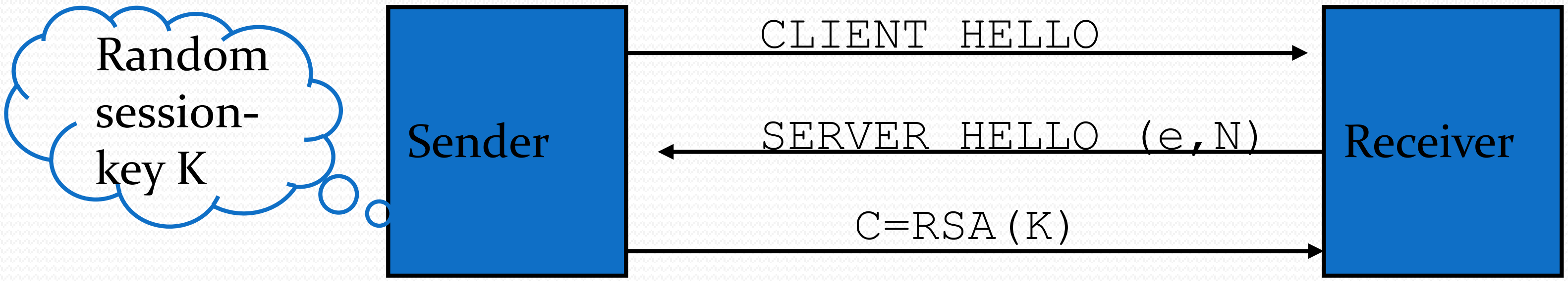
Textbook RSA is insecure

- Textbook RSA encryption:
  - public key: **(N,e)** Encrypt: $\mathbf{C = M^e} \pmod{N}$
  - private key: **d** Decrypt: $\mathbf{C^d = M} \pmod{N}$

$(\mathbf{M} \in \mathbf{Z}_N^*)$

- Completely insecure cryptosystem:
  - Does not satisfy basic definitions of security.
  - Many attacks exist.

A simple attack on textbook RSA



- Session-key K is 64 bits. View $K \in \{0,\ldots,2^{64}\}$
- Eavesdropper sees: $C = K^e \pmod{N}$.
- Suppose $\mathbf{K = K_1 \cdot K_2}$ where $K_1, K_2 < 2^{34}$. (prob. $\approx$20%)

  Then: $\mathbf{C/K_1^e = K_2^e} \pmod{N}$

- Build table: $C/1^e, C/2^e, C/3^e, \ldots, C/2^{34e}$. time: 2^{34}

  For $K_2 = 0,\ldots, 2^{34}$ test if K_2^e is in table. time: $2^{34} \cdot 34$

- Attack time: $\approx 2^{40} << 2^{64}$

Attack on RSA by Re-encryption

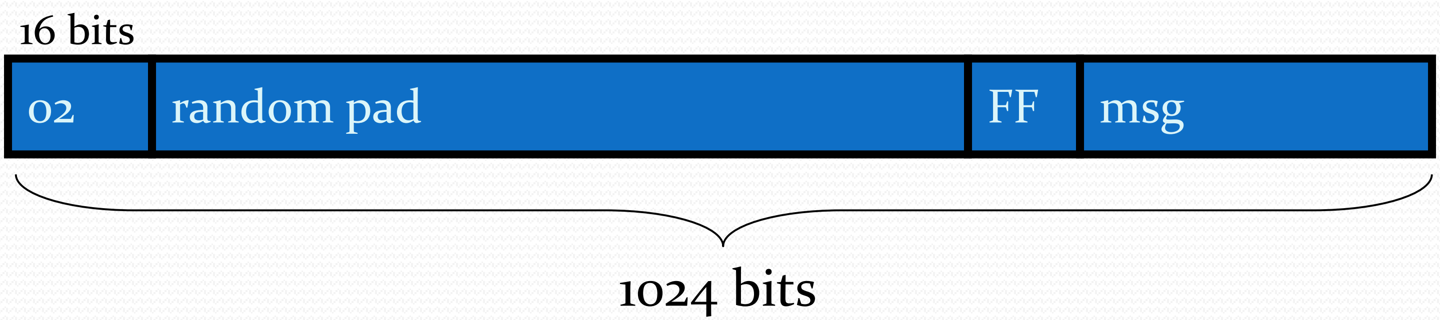
- Property of RSA encryption:
  - For each M there exists a unique number k called the iteration exponent or period of M such that

    $$C_{k+1} = C_0, \text{ where } C_{k+1} = C_k^{\,e} \pmod{N} \text{ and } C_0 = M$$

  - Efficiently applied only for relatively small p, q and e

- Attack Principle:
  - Attacker has to re-encrypt (as encryption exponent and the modulus are public)
  - Iterate the encryption step on each new cipher text, until the message is recovered

Practical RSA



- Resulting value is RSA encrypted
- Widely deployed in communications for portable wireless systems
- Coppersmith "Short Pad Attack" that exploit random padding to determine M

Attacks on RSA using Low-exponent

- Commonly chosen exponent e for practical implementation of RSA:
  - $e = 2^1 + 1 = 3$, $e = 2^4 + 1 = 17$, or $e = 2^{16} + 1 = 65537$
  - Eve sees k cipher texts $\mathbf{M^e}$ (mod Ni)
  - To uniquely decipher the message the following condition must hold

  $M < N_i$ for $i = 1, 2, \ldots, k$ and so $\mathbf{M^e} < N_1 . N_2 . \ldots N_k$.

  If the Ni are relatively prime, Eve can compute $\mathbf{M^e}$ (mod $N_1 . N_2 . \ldots N_k$), where $e < k$ using Chinese Remainder Theorem. Then she has a perfect integer power over the integers, namely $\mathbf{M^e}$

  She can calculate eth root and recover M. Alternately, Eve can factor the Ni's and compute M

  To avoid this attack, a large encryption e must be selected

Improving RSA's performance

- To speed up RSA decryption use small private key d. $C^d = M \pmod{N}$

  - Wiener87: if $d < N^{0.25}$ then RSA is insecure
  - Wiener 90: method to find decryption key when a small d is used

  - Decryption key d can be found from (N,e).

  - Small d should never be used

Wiener's attack

- Theorem:

  Let $N = pq$ with $q < p < 2q$. Let $d < (1/3)(N)^{¼}$.
Given (n, e) such that $e \cdot d \equiv 1 \pmod{\varphi(N)}$ then an attacker can efficiently recover d.

Wiener's attack

- Sketch: $e \cdot d = 1 \pmod{\varphi(N)}$

$$\Rightarrow \quad \exists\, k \in Z: \quad e \cdot d = k \cdot \varphi(N) + 1$$

$$\Rightarrow \quad \left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| \leq \frac{1}{d\varphi(N)}$$

$$\varphi(N) = N-p-q+1 \quad \Rightarrow \quad |N- \varphi(N)| \leq p+q \leq 3\sqrt{N}$$

$$d \leq N^{0.25}/3 \quad \Rightarrow \quad \left| \frac{e}{N} - \frac{k}{d} \right| \leq \frac{1}{2d^2}$$

Continued fraction expansion of e/N gives k/d.

$$e \cdot d = 1 \pmod{k} \quad \Rightarrow \quad \gcd(d,k)=1$$

Prime Recognition and Factorization

- The key problems for the development of RSA cryptosystem are that of prime recognition and integer factorization.

- August 2002 first polynomial time algorithm has been discovered that allows to determine whether a given m bit integer is a prime. Algorithm works in time $O(m^{12})$.

- Fast randomized algorithms for prime recognition has been known since 1977. One of the simplest one is due to Rabin.

Integer Factorization

- No polynomial time classical algorithm is known.
- Simple, but not efficient factorization algorithms are known.
- Several sophisticated distributed factorization algorithms are known that allowed to factorize, using enormous computation power, surprisingly large integers.
- Progress in integer factorization, due to progress in algorithms and technology, has been recently enormous.
- Polynomial time quantum algorithms for integer factorization are known since 1994 (P. Shor).

Pollard's *p-1* Factoring Algorithm

- Principle:

n is the product of two large primes p and q.

The number (p-1) is uniquely expressible as the product of prime powers.

$p-1 = p_1^{a1} . p_2^{a2} \ldots p_s^{as}$, where $p_1, p_2 \ldots . p_s$ are the distinct primes dividing (p-1).

Thus no two of $p_1^{a1} . p_2^{a2} \ldots p_s^{as}$ are equal

We assume that $p_1^{a1} < p_2^{a2} < \ldots < p_s^{as}$ and also assume that p_s^{as} is less than or equal to some small number B.

Then, p_i^{ai} is less than or equal to B, for $1 \leq i \leq s$

Pollard's *p-1* Factoring Algorithm

- Pick some integer t that is a multiple of all integers less than or equal to B, t = *factorial*(B). Or choose t to be the LCM of {1, 2, 3 …. B}
- Choose the integer x randomly, with n-2 > x > 2
- Calculate y = x^t by repeated squaring
- Let d be the gcd of (x^t -1) and n
- We have that d divides n. we can guarantee that d > 1. This means that unless (x^t -1) is a multiple of n, d is a proper factor of n.

Pollard's *p-1* Factoring Algorithm

- Any two of the numbers $p_1^{a1} . p_2^{a2} \ldots p_s^{as}$ are relatively prime and each is less than B.
- By choice, their product (p-1) divides t. so, t = v(p-1).
- Therefore, $x^t = x^{v(p-1)}$
- By Fermat's Little Theorem, $(x^{(p-1)})^v \equiv 1 \pmod p$.

Thus, $x^t \equiv 1 \pmod p$. Therefore, p divides $(x^t - 1)$, and p divides n.

Now, d = gcd of $((x^t - 1), n)$, it follows that p divides d.

This means that we have factored n.

Ref: Pollard 74, Lenstra 87 with Elliptic Curves.

Pollard's *p-1* Factoring Algorithm

- Example:

Let n = 2117, B = 7

Then choose t as LCM of { 1, 2, 3, 4, 5, 6, 7} = 420

Choose x = 2 (randomly)

Then 2^{420} (mod 2117) = 1451, thus, y = 1451

So, d= gcd(y-1, n) = gcd(1450, 2117) = 29.

It follows that n = 29.73

Implementation attacks

- Attack the implementation of RSA.
- Timing attack: (Kocher 97)
  The time it takes to compute $C^d \pmod N$
  can expose d.
- Power attack: (Kocher 99)
  The power consumption of a smartcard while
  it is computing $C^d \pmod N$ can expose d.
- Faults attack: (BDL 97)
  A computer error during $C^d \pmod N$
  can expose d.

Timing Attacks

- developed by Paul Kocher in mid-1990's
- exploit timing variations in operations
  - eg. multiplying by small vs large number
  - or IF's varying which instructions executed
- infer operand size based on time taken
- RSA exploits time taken in exponentiation
- countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations

Key lengths

- Security of public key system should be comparable to security of block cipher.

NIST:

Cipher key-size	Modulus size
$\leq$ 64 bits	512 bits.
80 bits	1024 bits
128 bits	3072 bits.
256 bits (AES)	**15360** bits

- High security $\Rightarrow$ very large moduli.

Private-key versus public-key cryptography

- The prime advantage of public-key cryptography is increased security.

- Public key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure.
- Example: RSA and DES are usually combined as follows

  1. The message is encrypted with a random DES key
  2. DES-key is encrypted with RSA
  3. DES-encrypted message and RSA-encrypted DES-key are sent.

- In software (hardware) DES is generally about 100 (1000) times faster than RSA.
- If n users communicate with secrete-key cryptography, they need $n\ (n - 1) / 2$ keys. In the case they use public key cryptography $2n$ keys are sufficient.