RSA Analysis

DR. ODELU VANGA

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRI CITY,
CHITTOOR, INDIA

Recap: RSA

$c = f(m) = m^e \bmod N$

$m = f^{-1}(m^e) = (m^e)^{1/e}$

$(m^e)^{1/e}$

$c^{1/e}$

$y = f(x)$

$x = f^{-1}(y)$

$= f^{-1}(f(x))$

$= x$

Parameters : $N=pq$. $N \approx 1024$ bits. $p,q \approx 512$ bits.
e – encryption exponent. $\gcd(e, \varphi(N)) = 1$.

Encryption : $RSA(M) = M^e \pmod N$ where $M \in Z_N^*$

Trapdoor : **d** – decryption exponent.

Where $e \cdot d = 1 \pmod{\varphi(N)}$

Decryption : $RSA(M)^d = M^{ed} = M^{k\varphi(N)+1} = M \pmod N$

(n,e,t,ε)-RSA Assumption: For any t-time alg. A:

$$\Pr\left[\, A(N,e,x) = x^{1/e} \ (N) : \begin{array}{l} p,q \xleftarrow{R} \text{n-bit primes}, \\ N \leftarrow pq, \quad x \xleftarrow{R} Z_N^* \end{array} \right] < \varepsilon$$

Recap: Φ(N) implies factorization

Knowing both ~~n~~ N and Φ(N), one knows

$$N = pq$$

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

$$= N - p - N/p + 1$$

$$p\Phi(N) = Np - p^2 - N + p$$

$$p^2 - Np + \Phi(N)p - p + N = 0$$

$$\boxed{p^2 - (N - \Phi(N) + 1)\, p + N = 0}$$

There are two solutions of p in the above equation.

Both p and q are solutions.

Decryption Attacks on RSA

e, d mod $\phi(N)$

$m^e \pmod N$

e–small

Small encryption exponent e

- When e=3, Alice sends the encryption of message m to three people (public keys (e, n_1), (e, n_2), (e,n_3))
  - $C_1 = M^3 \bmod n_1$, $C_2 = M^3 \bmod n_2$, $C_3 = M^3 \bmod n_3$,

A, B, C, D; e, n_1; e, n_2; e, n_3

$m < n_1$, $m < n_2$, $m < n_3$

$m^3 < n_1 n_2 n_3$

$$x \equiv c_1 \bmod n_1$$
$$x \equiv c_2 \bmod n_2$$
$$x \equiv c_3 \bmod n_3$$

- An attacker can compute a solution to linear system
- The solution x modulo $n_1 n_2 n_3$ must be M^3
- (No modulus!), one can compute integer cubit root
- **Countermeasure**: padding required

$(m \| r_1) * (m \| r_2) * (m \| r_3)$

Textbook RSA is insecure

Textbook RSA encryption:

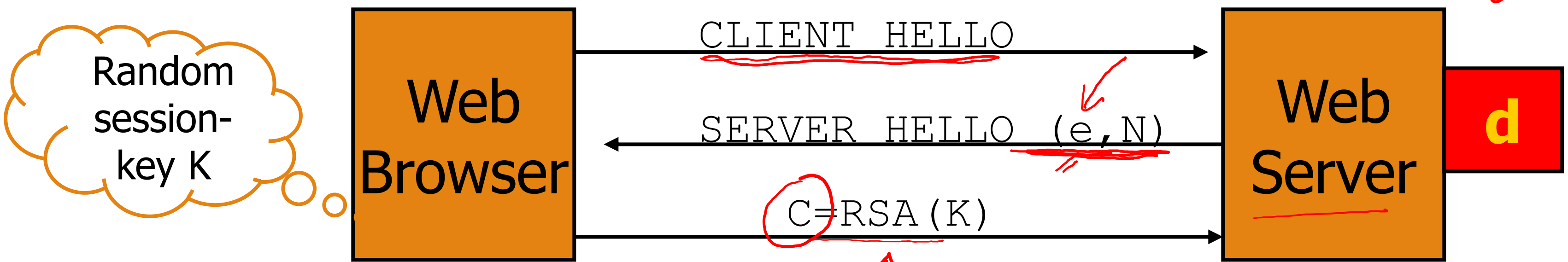
- public key: **(N, e)** Encrypt: $\mathbf{C = M^e}$ (mod N), where ($\mathbf{M} \in \mathbf{Z_N}^*$)
- private key: **d** Decrypt: $\mathbf{C^d = M}$ (mod N)

Completely insecure cryptosystem:

- Does not satisfy basic definitions of security.
- Many attacks exist.

Simple Attack on RSA

RSA: $N = pq$
e, d



Session-key K is 64 bits. View $K \in \{0,\dots,2^{64}\}$

Eavesdropper sees: $C = K^e \pmod N$

Suppose $\mathbf{K = K_1 \cdot K_2}$ where $K_1, K_2 < 2^{34}$ (prob. ≈20%).

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

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

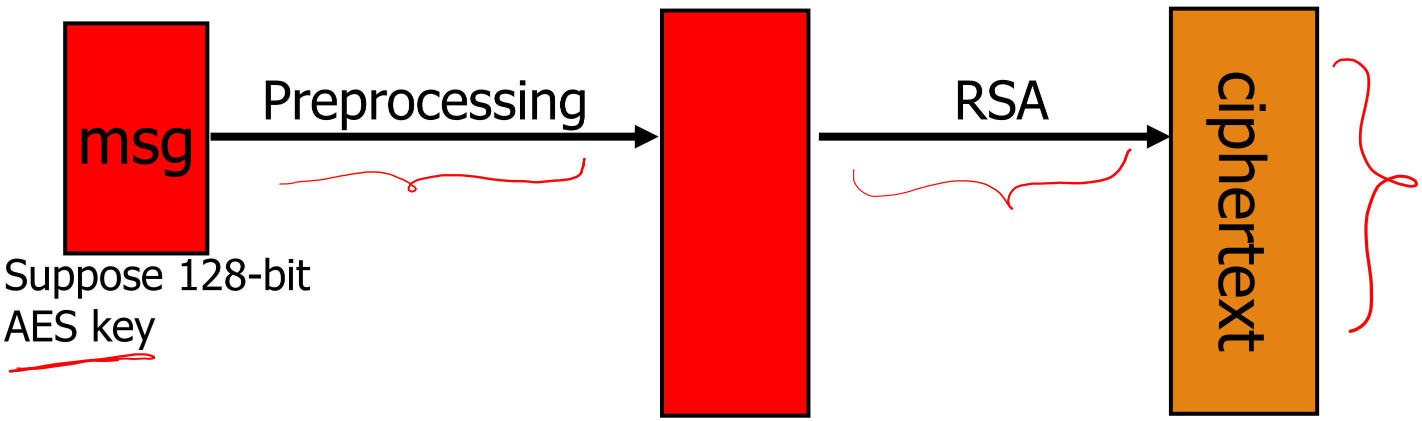
For $K_2 = 0,\dots, 2^{34}$ test if K_2^e is in table. time: $2^{34} \cdot 34$

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

Common RSA encryption

Never use textbook RSA.

RSA in practice:



Main question:

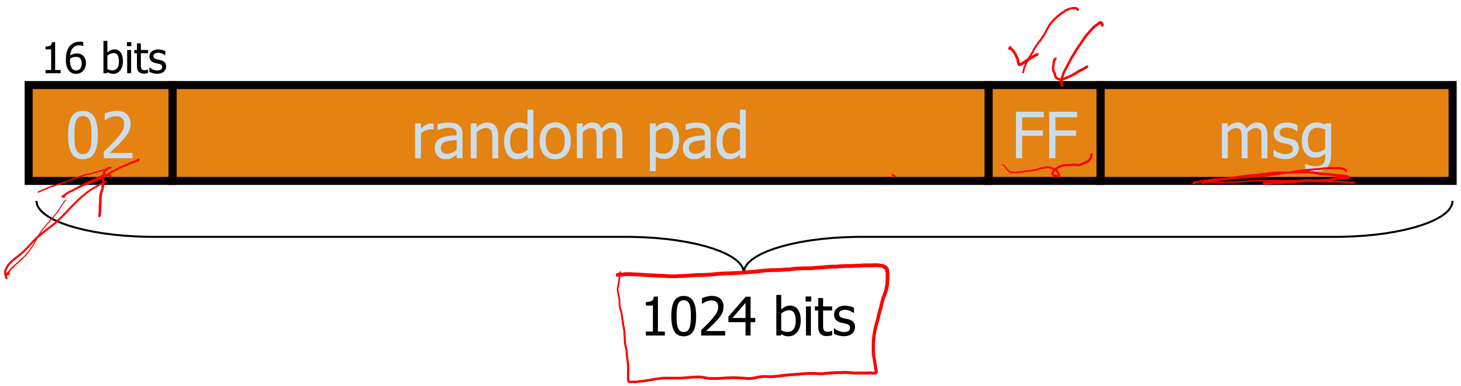
- How should the preprocessing be done?
- Can we argue about security of resulting system?

https://www.coursera.org/lecture/crypto/pkcs-1-JwjDq

PKCS1 V1.5

(Public-Key Cryptography Standards)

PKCS1 mode 2:   (encryption)



Resulting value is RSA encrypted.

Widely deployed in web servers and browsers.

No security analysis !!

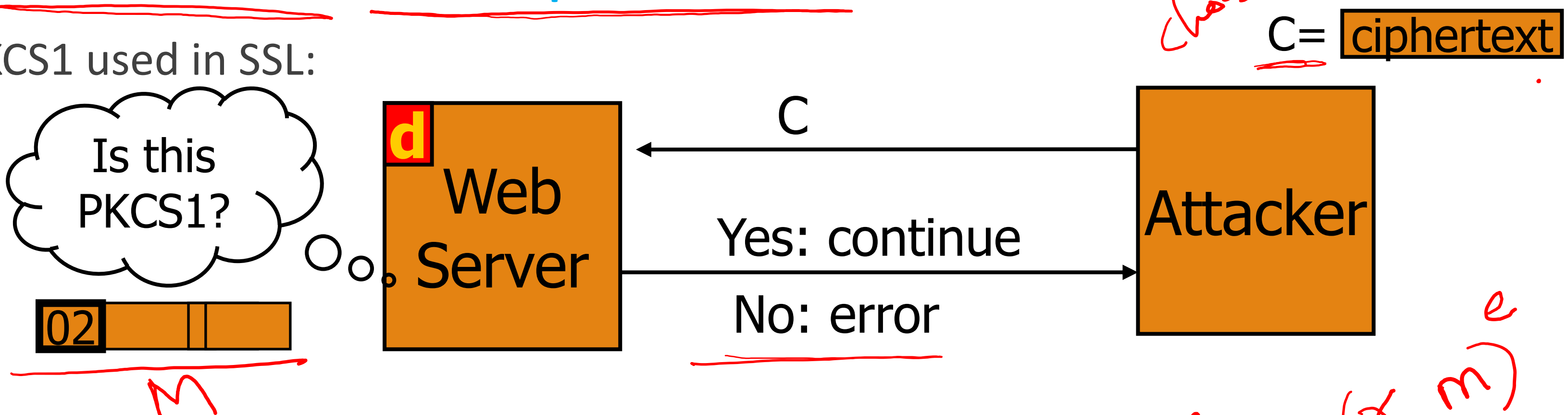
Attack on PKCS1

$\frac{r \cdot m}{r} = m$

Bleichenbacher 98. **Chosen-ciphertext attack**.

PKCS1 used in SSL:

choose r



$\Rightarrow$ attacker can test if 16 MSBs of plaintext = '02'.

Attack: to decrypt a given ciphertext C do:

- Pick $r \in Z_N$.
- Compute $C' = r^e \cdot C = \left(r \cdot \mathrm{PKCS1}(M)\right)^e$
- Send C' to web server and use response.

$c' = r^e c = (r\, m)^e$

$(c')^d = ((r m)^e)^d = r \cdot m$

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.

Key lengths

public-key
× encryption

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

NIST:

symmetric

A ⟷ B $\boxed{SK}$

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

C ⟷ S
SK

High security $\Rightarrow$ very large moduli.

Not necessary with Elliptic Curve Cryptography.

Thank You

Factoring when knowing e and d

Knowing ed such that $ed \equiv 1 \pmod{\Phi(N)}$

write $ed - 1 = 2^s r$ (r odd)

choose w at random such that 1<w<n-1

if w not relative prime to N then return gcd(w,N)

(if gcd(w,N)=1, what value is ($w^{2^s r}$ mod N)?)

compute $w^r, w^{2r}, w^{4r}, \ldots$, by successive
squaring until find $w^{2^t r} \equiv 1 \pmod{N}$

Fails when $w^r \equiv 1 \pmod{N}$ or $w^{2^t r} \equiv -1 \pmod{N}$

Failure probability is less than ½ (Proof is complicated)

Example: Factoring n given (e,d)

Input: N=2773, e=17, d=157

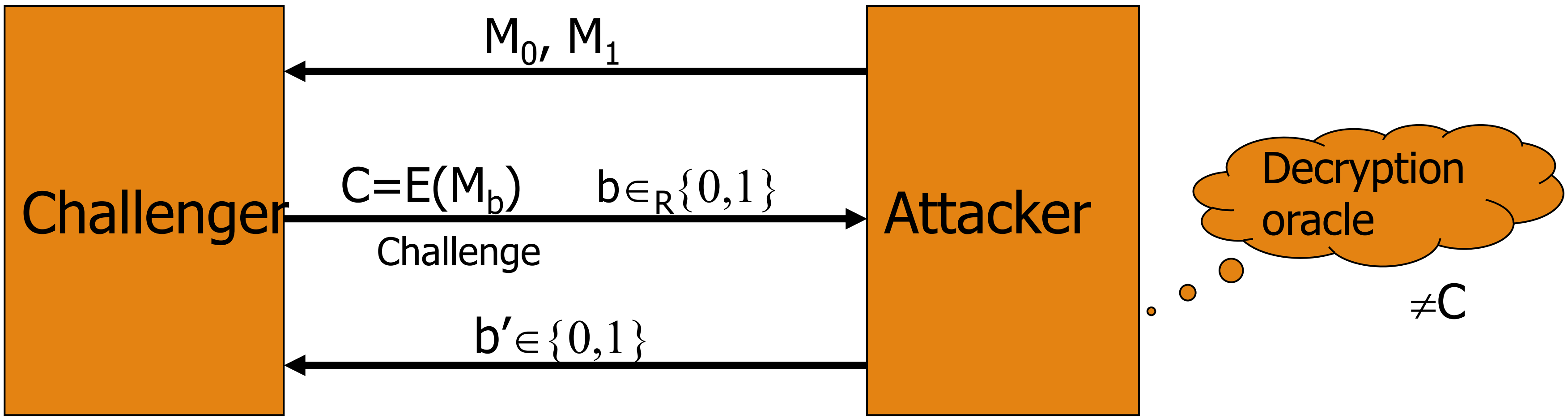
ed-1=2668=2^2•667 **(r=667)**

Pick random **w**, compute w^r mod n

- **w=7, 7^{667}=1 not good**
- w=8, 8^{667}=471, and **471^2=1**, so 471 is a nontrivial square root of 1 mod 2773
- compute gcd(471+1, 2773)=59 and gcd(471-1, 2773)=47.
- 2773=59•47

Chosen ciphertext security (CCS)

No efficient attacker can win the following game:
(with non-negligible advantage)



Attacker wins if $b=b'$