CS915/435 Advanced Computer Security - Elementary Cryptography

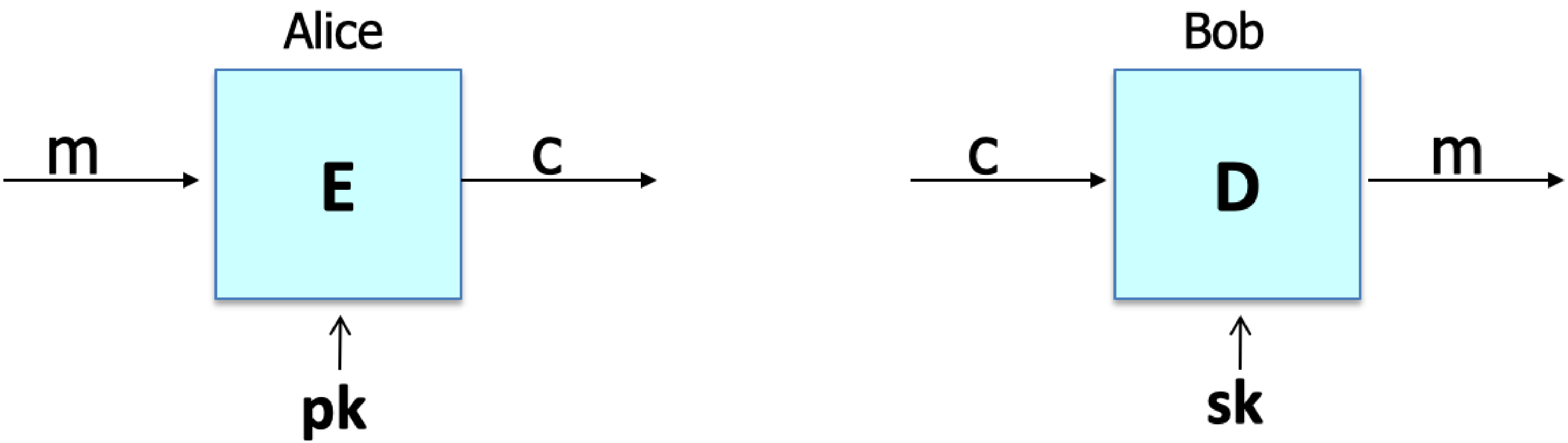
Public Key Encryption

Roadmap

- Symmetric cryptography
  - Classical cryptographic
  - Stream cipher
  - Block cipher I, II
  - Hash
  - MAC
- Asymmetric cryptography
  - Key agreement
  - **Public key encryption**
  - Digital signature

Public key encryption

Bob: generate a pair of keys (PK, SK), where PK is a public key and SK is a private key. He gives PK to Alice.



Public key encryption

<u>Def</u>: a public-key encryption system consists of 3 algs. (G, E, D)

- G(): randomised alg. outputs a key pair (pk, sk)
- E(pk, m): randomised alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): dec. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$(pk, sk) output by G :

$$\forall m \in M: \quad D(sk, E(pk, m)) = m$$

RSA

- Invented in 1977
- By Ron Rivest, Adi Shamir, Leonard Adleman
- The first widely used public key system
  - SSL/TLS   TLS 1.2   TLS 1.3
  - Secure email and file systems
  - many others

How great was this invention?

- Imagine someone designs a lock

1. One key to lock it and **another key** to unlock it.
2. Given the lock and one of the key, **you are unable to manufacture** the second key.

One-way function

One-way function

- We already saw one example of such functions The **DH protocol**:
  1. Given x, g, and p, we compute $g^x \bmod p = y$
  2. Given y, g, and p, it is hard to compute x
- **RSA** is based on the difficulty of factoring a prime number:
  1. Given p and q, it is easy to compute $n = p \times q$
  2. Factorizing n is hard (still an open problem)

Fermat's little theorem

Example: let $a = 4$, $p = 3$

- For any prime p not dividing a, we have

$$a^{p-1} = 1 \bmod p$$

$4^{3-1} = 4^2 = 1 \bmod 3$

Proof (sketch)

- Given the set $\{1, 2, \ldots, p-1\}$, we multiply it by a:

$$\{a, 2a, \ldots, (p-1)*a\}.$$

- The 2nd set has (p-1) distinct elements in [1, p-1], hence it's a permutation of the first set. Multiplying all elements in each set, we get: $(p-1)! = (p-1)!a^{p-1} \bmod p.$
- Therefore, $1 = a^{p-1} \bmod p.$

Euler's theorem

Example: let n = 9 $\varphi(n) = 6$ Set of coprimes {1,2,4,5,7,8}

- Euler's phi (or totient) function: $\phi(n)$ is the number of positive integers less than n with which it has no divisor in common.
  - E.g., $\phi(n) = (p-1)(q-1)$ if $n=pq$
- Euler's theorem (more general than Fermat's): for any modulus n and any integer a coprime to n, we have

$$a^{\phi(n)} = 1 \bmod n$$

The Euclidean Algorithm

Algorithm 5.1: EUCLIDEAN ALGORITHM(a, b)

$r_0 \leftarrow a$
$r_1 \leftarrow b$
$m \leftarrow 1$
while $r_m \neq 0$

$$\textbf{do} \begin{cases} q_m \leftarrow \left\lfloor \frac{r_{m-1}}{r_m} \right\rfloor \\ r_{m+1} \leftarrow r_{m-1} - q_m r_m \\ m \leftarrow m + 1 \end{cases}$$

$m \leftarrow m - 1$
return $(q_1, \ldots, q_m; r_m)$
comment: $r_m = \gcd(a, b)$

GCD(a,b) = GCD(b, a mod b)

a = 40, b = 15

40 = 2 × 15 + 10

Gcd(40, 15) = gcd(15, 10)

Gcd(15, 10) = gcd(10, 5) = 5

Chapter 5 “The RSA Cryptosystem and Factoring Integers”, Cryptography - Theory and Practice, 3rd edition, 2006 by Doug Stinson.

Extended Euclidean Algorithm

Algorithm 5.2: EXTENDED EUCLIDEAN ALGORITHM(a, b)

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$s_0 \leftarrow 1$
$s \leftarrow 0$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$

$$\textbf{do} \begin{cases} temp \leftarrow t_0 - qt \\ t_0 \leftarrow t \\ t \leftarrow temp \\ temp \leftarrow s_0 - qs \\ s_0 \leftarrow s \\ s \leftarrow temp \\ a_0 \leftarrow b_0 \\ b_0 \leftarrow r \\ q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor \\ r \leftarrow a_0 - qb_0 \end{cases}$$

$r \leftarrow b_0$
return (r, s, t)
comment: $r = \gcd(a, b)$ and $\boxed{sa + tb = r}$

40 = 2 × 15 + 10
15 = 1 × 10 + 5
10 = 2 × 5 + 0

15 = 1 × 10 + 5
15 - 1 × 10 = 5

40 = 2 × 15 + 10
Or 10 = 40 - 2 × 15

So 15 - 1 × 10 = 5
is 15 - 1 × (40 - 2 × 15) = 5

Finally - 1 × 40 + 3 × 15 = 5

Extended Euclidean Algorithm

- We are interested in the special case where r = 1
- So, `sa + bt = 1` in this case
- In other words, `sa = 1 - bt`
- And **`sa = 1`** `mod b`

Computing the inverse of a

Given an element **a** in Z_N where **a** is relatively prime to N, we can compute its inverse $\mathbf{a^{-1}}$

$$a \times a^{-1} = 1 \bmod N$$

Hint: use Extended Euclidean Algorithm with **a** and N as inputs: $s \times a + t \times N = GCD(a,N) = 1$.

We have $s \times a = 1 \bmod N$. Obviously, $a^{-1} = s$.

Summary: arithmetic mod composites

Let $N = p \times q$ where p, q are primes

$Z_N = \{0, 1, 2, \ldots, N-1\}$;

$Z_N^* = \{\text{invertible elements in } Z_N\}$

$\boxed{\text{Facts:}}$ (1) $x \in Z_N$ is invertible $\Leftrightarrow$ $\gcd(x, N) = 1$

(2) Number of elements in Z_N^* is $\phi(N) = (p-1)(q-1)$

Euler's theorem: $\forall a \in Z_N^*$: $\boxed{a^{\phi(N)} = 1 \bmod N}$

Chinese Remainder Theorem

A method of solving systems of congruences.

$$\begin{aligned}
x &\equiv \overset{2}{a_1} \pmod{\overset{3}{m_1}} \\
x &\equiv \overset{3}{a_2} \pmod{\overset{5}{m_2}} \\
&\vdots \\
x &\equiv \overset{2}{a_r} \pmod{\overset{7}{m_r}}.
\end{aligned}$$

A special case:

x = a mod p

x = a mod q

We must have

x = a mod pq

One possible solution: x = 23 (general solution is x = 23 + 105k)

There is a unique solution mod (m_1 x m_2 x … x m_r)

RSA Key Generation

GenRSA(1^n)

Input: key length n

Miller-Rabin primality test

Generate two large n-bit **distinct primes** p and q

Compute $N = p \cdot q$ and $\varphi(N) = (p\text{-}1) \cdot (q\text{-}1)$

Choose a random integer e, $\mathbf{gcd}(e, \varphi(N)) = 1$

Compute e's inverse d: $d \cdot e = 1 \ (\mathbf{mod}\ \varphi(N))$

Output: $\boxed{pk = (N, e),}$ $\boxed{sk = (N, d)}$

Textbook RSA encryption

KeyGen: $pk=(N, e)$, $sk=(N, d)$

Enc: Given $pk=(N, e)$ and message $m \in Z_N$: **[0, N-1]**

$$c = \boxed{m^e} \quad (\textbf{mod } N)$$

Dec: Given $sk=(d, N)$ and ciphertext c:

$$m = \boxed{c^d} \quad (\textbf{mod } N)$$

Correctness

Need to show:

$$\boxed{\mathbf{Dec}_{sk}(\mathbf{Enc}_{pk}(m)) = m}$$

Key: $\mathbf{gcd}(e, \varphi(N)) = 1$ and $ed = 1$ ($\mathbf{mod}\ \varphi(N)$)

Case 1: If m is relatively prime to N, i.e., $m \in Z_N^*$

$$c^d = (m^e)^d = m^{de} = \underline{m^{de \bmod \varphi(N)}} = m \bmod N$$

Case 2: Else (i.e., $m \in Z_N \backslash Z_N^*$)

$$c^d = (m^e)^d = m^{de} = m^{de \bmod (p-1)} = \underline{m \bmod \mathbf{p}}$$

$$c^d = (m^e)^d = m^{de} = m^{de \bmod (q-1)} = \underline{m \bmod \mathbf{q}}$$

Hence $\boxed{c^d = m \bmod p \times q}$ (Chinese Remainder Theorem)

RSA Example - Key Setup

1. Select primes: p=17 and q=11
2. Compute $n = pq$ =17 × 11=187
3. Compute $\phi(n)=(p-1)(q-1)$=16 × 10=160
4. Select e: gcd(e,160)=1; choose **e = 7**
5. Determine d: d.e ≡ 1 mod 160 and d < 160
    1. Use Euclid's Inverse algorithm
    2. Value is **d = 23** since 23 × 7=161= 10 × 160+1
6. Publish public key PU={**7,**187}
7. Keep secret private key PR={**23,**187}

RSA Example - En/Decryption

- Given a message $M = 88$ (with $88<187$)
- Its encryption is:

  $C = 88^7 \bmod 187 = 11$
- Its decryption is:

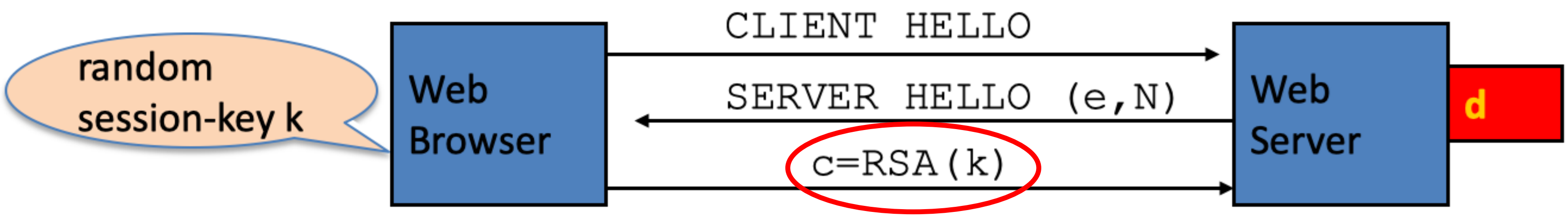
  $M = 11^{23} \bmod 187 = 88$

Square and multiply algorithm

How Secure is Textbook RSA?

- Security definition
  - **Semantic security**: ciphertext indistinguishable from random data
  - But textbook RSA is not semantically secure; many attacks exist

A meet-in-the-middle attack on textbook RSA



Suppose k is 64 bits: $k \in \{0,..,2^{64}\}$. Eve sees $c = k^e$ in Z_N

If $k=k_1 * k_2$ where $k_1, k_2 < 2^{34}$ (prob=20%) then

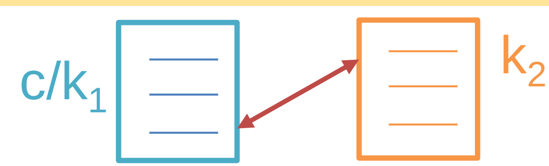
$c/(k_1)^e = (k_2)^e$ in Z_N

$c = (k_1 k_2)^e$

Step 1: build table: $c/1^e, c/2^e, \ldots, c/2^{34e}$. Time 2^{34}

Step 2: for $k_2 = 0,\ldots,2^{34}$ test if k_2^e is in table. Time 2^{34}

Output matching $(c/k_1, k_2)$



Mangling Ciphertexts

Example: Alice sends bid m=**1000** in an auction.

$$c = m^e \ (\mathbf{mod}\ N)$$

$$c^* = 2^e \cdot c \ (\mathbf{mod}\ N)$$

$$(c^*)^d = (2^e \cdot m^e)^d = (2 \cdot m)^{de} = 2 \cdot m = 2000$$

Common modulus attack

Assume organisation uses **common modulus** N for all employees.

Each employee receives key pair ($pk=e$, $sk=d$)

What can go wrong?

Knowledge of d $\Longleftrightarrow$ factorization of N

[Fact 1 from Boneh “Twenty Years of Attacks on the RSA Cryptosystem”]

RSA with padding

Padding is to randomize the encryption

- PKCS #1 v1.5
- RSA OAEP

RSA with PKCS #1 v1.5 Padding

Encryption:

Choose random byte-string r (k-D-3>8 bytes).

16 bit		8 bit	
2	**Random padding r**	**0**	**m**

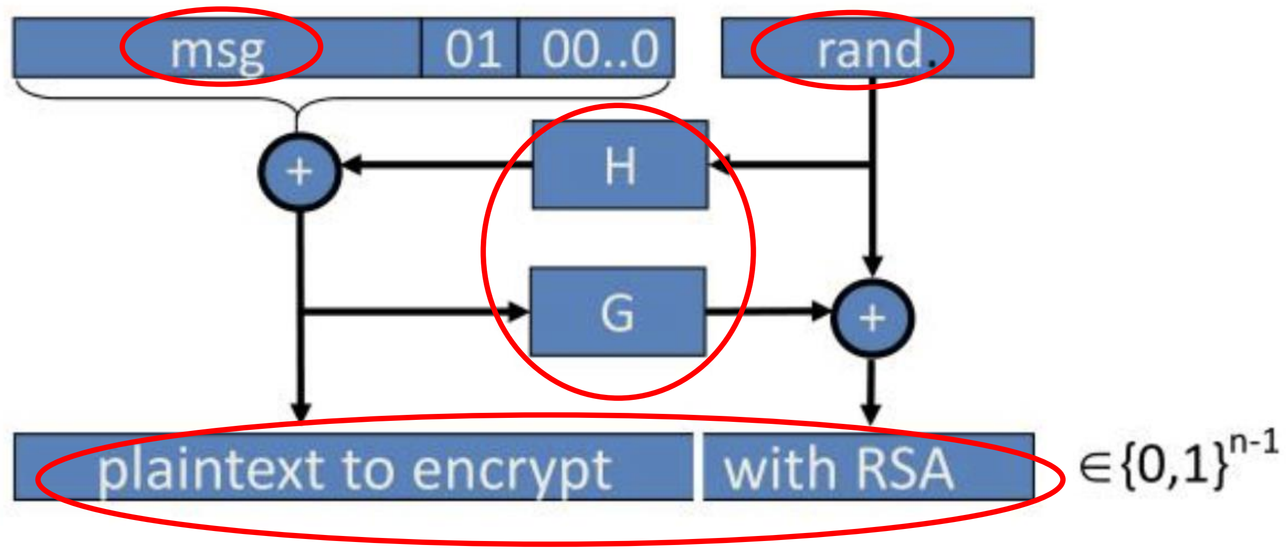
$(00000000||00000010||r||00000000||m)^e$ (**mod** N)

Decryption:

As usual, check that the padding is ok!

Idea: Prefix D-byte message m with random padding

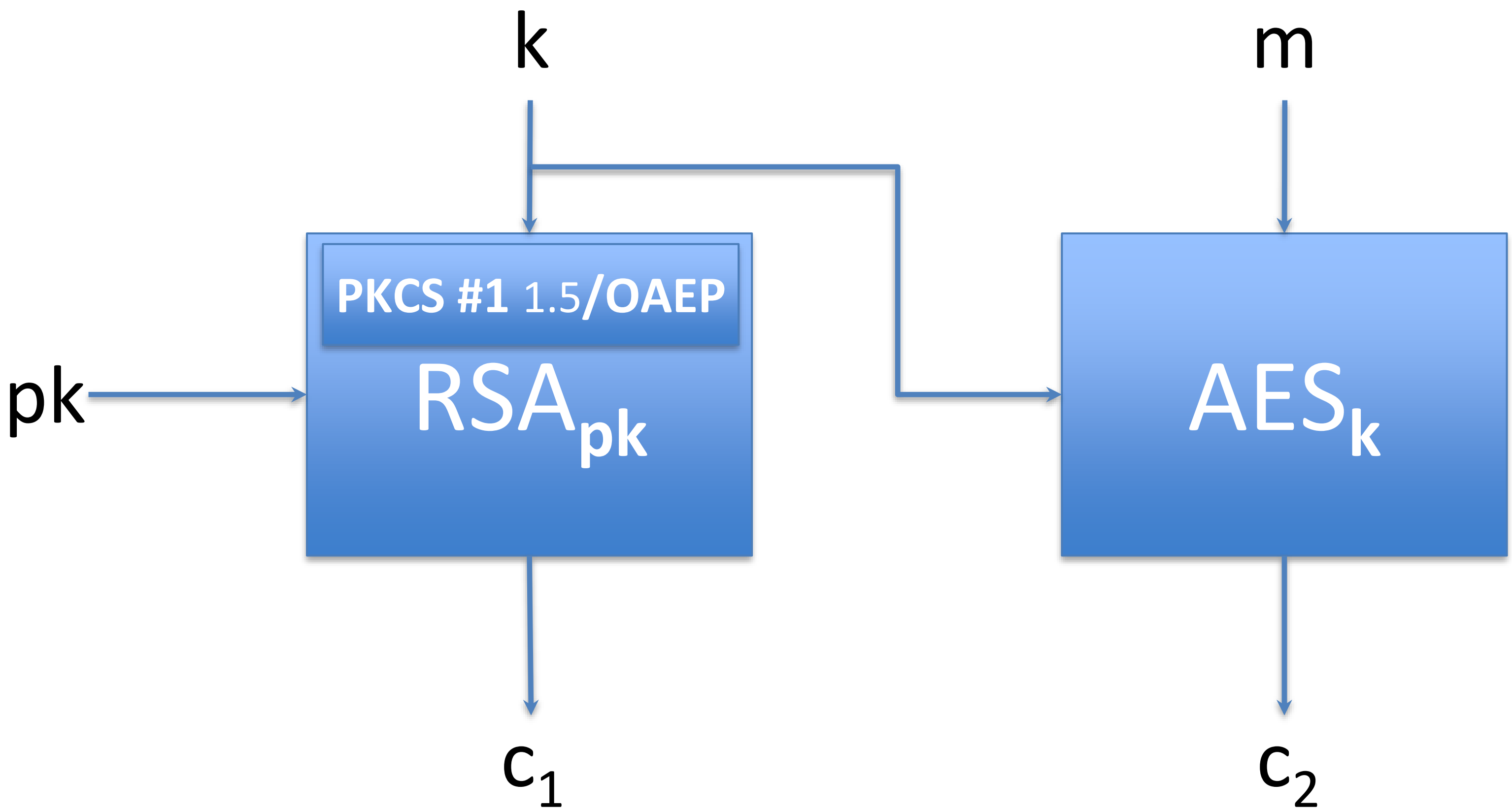
RSA OAEP



PKCS1 v2.0

- H and G are hash function

RSA in Practice: Hybrid with padding



Choose unique N for each user and fresh random k