Chapter 9. PUBLIC KEY CRYPTOGRAPHY AND RSA

Public key cryptography approach can be depicted by Figure 9.1:

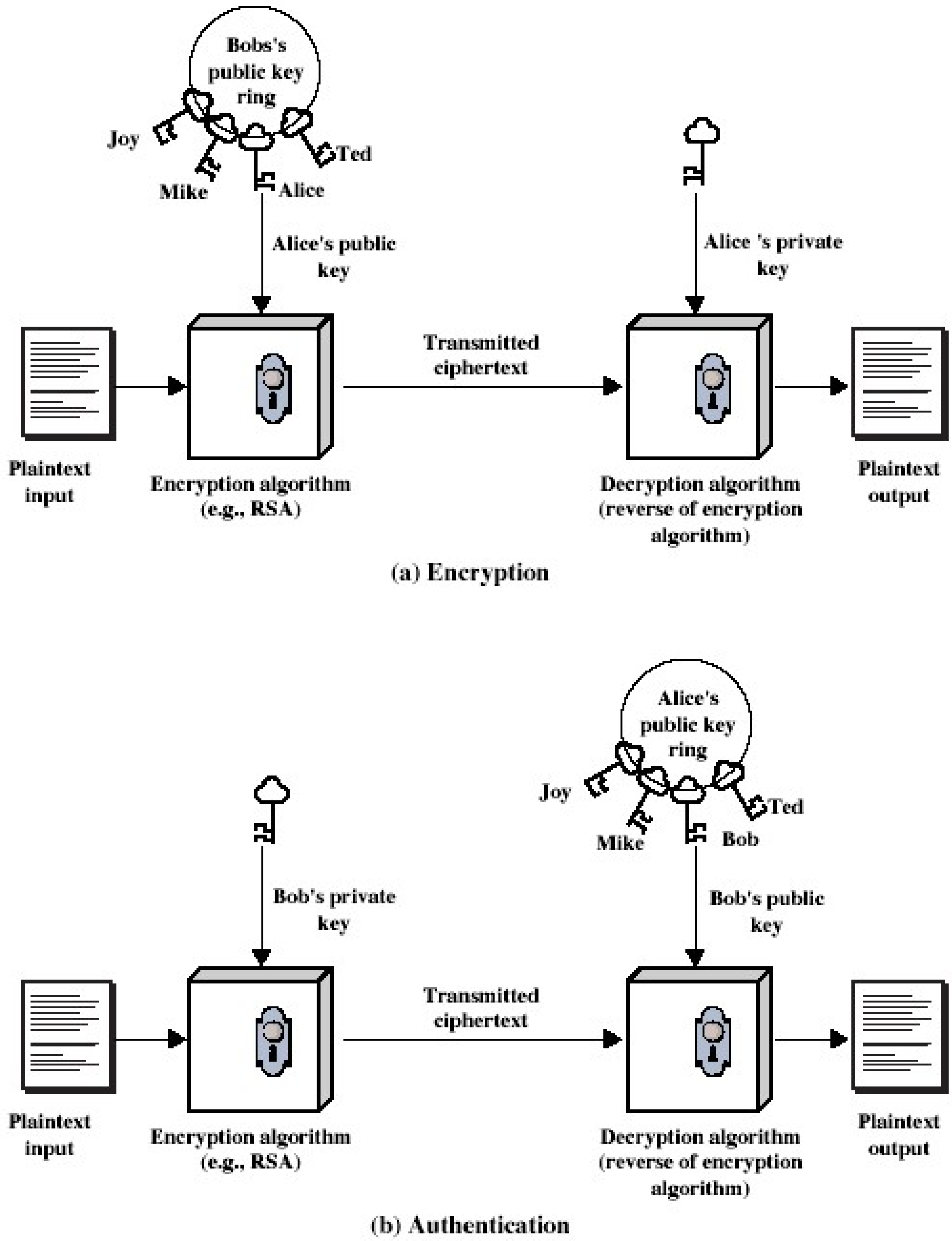

Figure 9.1 Public-Key Cryptography

The concept of public cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. The first problem concerns key distribution. The second problem is the problem of "digital signatures" (authentication). Whitfield Diffie and Martin

PUBLIC KEY CRYPTOGRAPHY AND RSA (CONT 1)

Hellman, US cryptologists, invented in 1976 a method that addressed both problems, and that was radically different from all previous approaches to cryptography, going back over four millennia.

Applications of Public-Key Cryptosystems

Encryption/decryption
Digital signature
Key exchange

Table 9.2 Applications for Public-Key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a pair (public key KUb, private key KRb)
2. It is computationally easy for a sender A, knowing the public key and the message M to be encrypted, to generate corresponding ciphertext:
$$C=E_{KUb}(M)$$
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key:
$$M=D_{KRb}(C)= D_{KRb}(E_{KUb}(M))$$
4. It is computationally infeasible for an opponent, knowing the public key KUb to determine the private key KRb
5. It is computationally infeasible for an opponent, knowing the public key KUb and a ciphertext C to recover original message M.

Requirements for public-key cryptography can be met if to discover trap-door one-way functions, which are defined as follows:

A trap-door one-way function is a family of invertible functions fk, such that:

Y=fk(X) easy, if k and X are known

Requirements for Public-Key Cryptography (Cont 1)

$X=f_k^{-1}(Y)$ easy, if k and Y are known
$X=f_k^{-1}(Y)$ infeasible, if Y is known but k is not known

Public-Key Cryptanalysis

It is vulnerable to brute force attack -> use large keys.

Another form of attack is to find some way to compute the private key given the public key. To date, it has not been mathematically proven that this form of attack is infeasible for a particular public-key algorithm. Thus, any given algorithm, including the widely used RSA algorithm, is suspect.

Finally, there is a form of attack that is peculiar to public-key systems. This is, in essence, a probable-message attack. Suppose, for example, that a message were to be sent that consisted solely of a 56-bit DES key. An opponent could encrypt all possible keys using the public key and could decipher any message by matching the transmitted ciphertext. Thus, no matter how large the key size of the public-key scheme, the attack is reduced to a brute-force attack on a 56-bit key. This attack can be thwarted by appending some random bits to such simple messages.

The RSA Algorithm

It was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. The Rivest-Shamir-Adleman (RSA) has since that time reigned supreme as most widely accepted and implemented general-purpose approach to public-key encryption.

Description of the Algorithm

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some integer n. That is, the block size must be less or equal to $\log_2 n$; in practice, the block size is k bits, where $2^k < n \le 2^{k+1}$. Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both sender and receiver must know the value of n. The sender knows the value of e, and only receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of KU={e,n}, and a private key of KR={d,n}. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

Description of the Algorithm (Cont 1)

1. It is possible to find values of e,d,n such that $M^{ed} = M \bmod n$ for all M<n
2. It is relatively easy to calculate M^e and C^d for all values of M<n
3. It is infeasible to determine d given e and d.

For now, we focus on the 1st requirement and consider the other questions later. We need to find a relationship of the form

$$M^{ed} = M \bmod n$$

A corollary to Euler's theorem
(For every a and n that are relatively prime

$$a^{\varphi(n)} \equiv 1 \bmod n$$

where $\varphi(n)$ is the Euler's totient function – number of positive integers less than n and relatively prime to n),
fits the bill:
Given two prime numbers, p and q, and two integers, n and m, such that n=pq and 0<m<n, and arbitrary integer k, the following relationship holds:

$$m^{k\varphi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m \bmod n \qquad (*)$$

(as far as for p,q prime, $\varphi(n) = (p-1)(q-1)$)
Thus, we can achieve the desired relationship if $ed = k\varphi(n) + 1$

This is equivalent to saying: $$ed \equiv 1 \bmod \varphi(n)$$ $$d \equiv e^{-1} \bmod \varphi(n)$$

That is, e and d are multiplicative inverses $\bmod \varphi(n)$. Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\varphi(n)$. Equivalently, $\gcd(\varphi(n), d) = 1$.
We are now ready to state the RSA scheme. The ingredients are the following:

p,q, two prime numbers (private, chosen)
n=pq (public, calculated)
e, with $\gcd(\varphi(n), e) = 1; 1 < e < \varphi(n)$ (public, chosen)
$d \equiv e^{-1} \bmod \varphi(n)$ (private, calculated)

The private key consists of {d,n}, and the public key consists of {e,n}. Suppose that user A has published its public key and that user B wishes to send message M to A. Then B calculates $C = M^e \bmod n$ and transmits C. On receipt of this ciphertext, user A decrypts by calculating $M = C^d \bmod n$. It is worthwhile to summarize the justification for this algorithm. We have chosen e and d such that $d \equiv e^{-1} \bmod \varphi(n)$.

Description of the Algorithm (Cont 2)

Therefore $ed \equiv 1 \bmod \varphi(n)$. Therefore, ed is of the form $k\varphi(n)+1$. But by the corollary to Euler's theorem (*), given two prime numbers, p and q, and integer n=pq and M, with 0<M<n: $M^{k\varphi(n)+1} = M^{k(p-1)(q-1)+1} \equiv M \bmod n$

So, $M^{ed} \equiv M \bmod n$. Now

$C = M^e \bmod n$

$M = C^d \bmod n \equiv (M^e)^d \bmod n \equiv M^{ed} \bmod n \equiv M \bmod n$

Figure 9.5 summarizes RSA algorithm:

Key Generation

Select p, q	p and q both prime, p q
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \bmod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod n$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod n$

Figure 9.5 The RSA Algorithm

Description of the Algorithm (Cont 3)

An example is shown in Figure 9.6

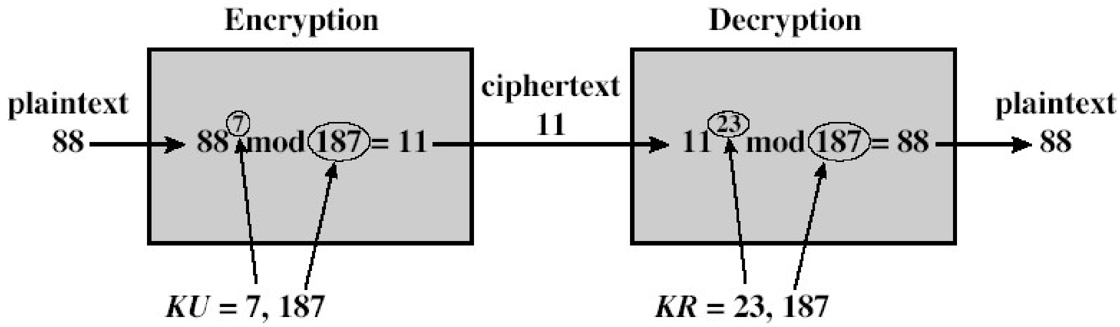

Figure 9.6 Example of RSA Algorithm

For this example, the keys were generated as follows:

1. Select two prime numbers, p=17, q=11
2. Calculate n=pq=17x11=187
3. Calculate $\varphi(n)$=(p-1)(q-1)=16x10=160
4. Select e such that e is relatively prime to $\varphi(n)$=160 and less than $\varphi(n)$; we choose e=7.
5. Determine d such that $de \equiv 1 \bmod 160$ and d<160. The correct value is d=23, because 23x7=161=1x160+1; d can be calculated using the extended Euclid's algorithm:

M=160, b=7
A=(1,0,m), B=(0,1,b)
Q=A3/B3=160/7=22
T=A-QB=(1,-22,6)
A=(0,1,b), B=(1,-22,6)
Q=A3/B3=7/6=1
T=(-1,23,1)
A=(1,-22,6), B=(-1,23,1)
B3=1 =>b^{-1}=B2=23.

The resulting keys are public key KU={7,187} and private key KR={23,187}. The example shows the use of these keys for a plaintext input of M=88. For encryption, we need to calculate C=88^7 mod 187. Exploiting the properties of modular arithmetic, we can do this as follows:

88^7 mod 187= [(88^4 mod 187)x(88^2 mod 187)x(88 mod 187)] mod 187
88 mod 187 = 88

Description of the Algorithm (Cont 4)

88^2 mod 187 = 7744 mod 187 = 77
88^4 mod 187 = 77^2 mod 187 =132
88^7 mod 187 = (88x77x132) mod 187 = [((88x77) mod 187) x (132 mod 187)] mod 187 = (44x132) mod 187 = 5808 mod 187 = 11
For decryption, we calculate M=11^{23} mod 187:
11^{23} mod 187 = [(11 mod 187)x(11^2 mod 187)x(11^4 mod 187)x(11^8 mod 187)x
(11^8 mod 187)] mod 187 = [11x121x 14641 mod 187 x (11^8 mod 187)x
(11^8 mod 187)] mod 187 = [11x121x55x (11^8 mod 187)x
(11^8 mod 187)] mod 187 = [11x121x55x (3025 mod 187)x
(3025 mod 187)] mod 187 = [11x121x55x 33x33
] mod 187 = [((11x121) mod 187)x((55x 33) mod 187) x 33
] mod 187 = [(1331 mod 187)x(1815 mod 187)x33] mod 187 =
[22x132x33]mod 187 = [2904mod187x33]mod187= [99x33] mod 187=
3267 mod 187 = 88

The Security of RSA

Three possible approaches to attacking the RSA algorithm are as follows:
Brute force – use large keys
Mathematical attacks
Timing attacks

Mathematical attacks

We can identify three approaches to attacking RSA mathematically:
- Factor n into two prime factors, this enables calculation of $\varphi(n)$=(p-1)(q-1), which, in turn, enables determination of d=e^{-1} mod $\varphi(n)$.
- Determine $\varphi(n)$ directly, without first determining p and q.
- Determine d directly, without first determining $\varphi(n)$

Most discussions of cryptanalysis of RSA have focused on the task of factoring n into its two prime numbers. Determining $\varphi(n)$ given n is equivalent to factoring n. With presently known algorithms, determining d given e and n appears to at least as time consuming as the factoring problem. Hence, we can use factoring performance as a benchmark against which to evaluate the security of RSA.
Table 9.3 shows the progress in factoring performance:

Mathematical attacks (Cont 1)

Table 9.3 Progress in Factorization

Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
100	332	April 1991	7	quadratic sieve
110	365	April 1992	75	quadratic sieve
120	398	June 1993	830	quadratic sieve
129	428	April 1994	5000	quadratic sieve
130	431	April 1996	1000	generalized number field sieve
140	465	February 1999	2000	generalized number field sieve
155	512	August 1999	8000	generalized number field sieve

The level of effort is measured in MIPS-years: a million-instructions-per-second-processor running for 1 year, which is about 3e13 instructions executed. A 1-GHz Pentium is about a 250-MIPS machine.
We see that progress in factoring is impressive, and for the near future, a key size in the range of 1024 to 2048 bits seems reasonable.

Timing Attacks

A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number. In this case, time for exponentiation may be used for attacking.
There are simple counter-measures against timing attacks:

- constant exponentiation time – ensure that all exponentiations take the same time, but this will degrade performance
- Random delay – better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack
- Blinding – multiply the ciphertext by a random number before performing exponentiation. This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack. RSA Data Security reports a 2 to 10% performance penalty for blinding.