Asymmetric Ciphers

Unit -3 [8Hours]

- Number Theory
  - Prime numbers
  - Fermat's and Euler's Theorems & ø(n)
  - Primality Testing
  - Primitive Roots
  - Discrete Logarithms

Prime Numbers

- prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
- eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- prime numbers are central to number theory
- list of prime number less than 200 is:

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83
89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167
173 179 181 191 193 197 199
```

An integer $p > 1$ is a prime number if and only if its only divisors are ± 1 and $\pm p$.

Prime Factorization

- Any integer $a > 1$ can be factored in a unique way as:

$$a = p_1^{a1} \times p_2^{a2} \times \ldots \times p_t^{at}$$

where p_1 , p_2 … p_t are prime numbers and where each *ai* is a positive integer. This is known as the fundamental theorem of arithmetic; a proof can be found in any text on number theory.

- If P is the set of all prime numbers, then any positive integer a can be written uniquely in the following form:

$$a = \prod_{p \in \mathrm{P}} p^{a_p} \quad \text{where each } a_p \geq 0$$

- to **factor** a number n is to write it as a product of other numbers:

  n=a x b x c
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number n is when its written as a product of primes

Example:

$91 = 7 \times 13$

$3600 = 24 \times 3^2 \times 5^2$

$11011 = 7 \times 11^2 \times 13$

Relatively Prime Numbers & GCD

- Two numbers a, b are **relatively prime** if have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- Conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - eg. $300=2^1 \times 3^1 \times 5^2$ $18=2^1 \times 3^2$ hence $GCD(18,300)=2^1 \times 3^1 \times 5^0=6$

Fermat's Theorem

- Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p *(i.e. relative prime number to p)*, then:

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

$a = 7, p = 19$
$7^2 = 49 \equiv 11 \pmod{19}$
$7^4 \equiv 121 \equiv 7 \pmod{19}$
$7^8 \equiv 49 \equiv 11 \pmod{19}$
$7^{16} \equiv 121 \equiv 7 \pmod{19}$
$a^{p-1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \pmod{19}$

- An alternative form of Fermat's theorem (also known as Fermat's Little Theorem) is also useful: If p is prime and a is a positive integer, then:

$$a^p = 1 \pmod{p}$$

$p = 5, a = 3 \quad a^p = 3^5 = 243 \equiv 3 (\text{mod } 5) = a (\text{mod } p)$
$p = 5, a = 10 \quad a^p = 10^5 = 100000 \equiv 10 (\text{mod } 5) \equiv 0 (\text{mod } 5) = a (\text{mod } p)$

Euler's Totient Function

- Euler's Totient Function , written as $\phi(n)$, is defined as the number of positive integers less than n and relatively prime to n.
- By convention, $\phi(1) = 1$.
- in general need prime factorization, but
  - for p (p prime) $\phi(p) = p-1$
  - for p.q (p,q prime, $p \neq q$) $\phi(pq) = (p-1)x(q-1)$

Determine $\phi(37)$ and $\phi(35)$.

Solution:

Because 35 is prime, $\phi(35) = \phi(7 \times 5) = (7-1)(5-1) = 24$

Therefore, positive integer less than 35 that are relatively prime to it:

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18
19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34

There are 24 numbers on the list, so $\phi(35) = 24$.

Euler's Totient Function

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	$\phi(n)$
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	$\phi(n)$
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

Figure: Some Values of Euler's Totient Function $\phi(n)$

$$\phi(pq) = \phi(p) \times \phi(q) = (p-1)(q-1)$$

Euler's Thorem

- Euler's theorem states that for every a and n that are relatively prime (such that gcd(a,n)=1):

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

- Example:
  $a=3; n=10; \phi(10)=4;$
  hence $3^4 = 81 = 1 \bmod 10$
  $a=2; n=11; \phi(11)=10;$
  hence $2^{10} = 1024 = 1 \bmod 11$

- As in the case for Fermat's theorem, an alternative form of the theorem is also useful:

$$a^{\phi(n)+1} \equiv a \pmod{n}$$

(Integers a and n need not be relatively prime)

Testing for Primality

- For many cryptographic algorithms, it is necessary to select one or more very large prime numbers at random.
- Thus, we are faced with the task of determining whether a given large number is prime.
- There is no simple yet efficient means of accomplishing this task.

- Traditionally **sieve** using **trial division**
  - ie. divide by all numbers (primes) in turn less than the square root of the number
  - only works for small numbers
- Alternatively can use statistical primality tests based on properties of primes
  - for which all primes numbers satisfy property
  - but some composite numbers, called pseudo-primes, also satisfy the property
- Can use a slower deterministic primality test

Miller Rabin Algorithm

- A test based on Fermat's Theorem to test a large number for primality.
- Background:
- For, any positive odd integer n≥3 can be expressed as

$$(n-1)=2^k q \text{ with } k>0, \ q \text{ odd}$$

- *To see this, note **n-1** is an even integer. Then divide **(n-1)** by **2** until the result is an odd number **q**, for a total of **k** divisions.*
- If ***n*** is expressed as a binary number, then the result is achieved by shifting the number to the right until the rightmost digit is a **1**, for a total of ***k*** shifts.

Two properties of prime numbers

First property is stated as follows:

- If p is prime and a is a positive integer less than p, then $a^2 \bmod p = 1$ if and only if either

$$a \bmod p = 1 \text{ or } a \bmod p = -1 \bmod p = p - 1.$$

By rule of modular arithmetic

$$\textbf{(a mod p) (a mod p) = a}^2 \textbf{ mod p.}$$

Thus, if either **a mod p = 1** or **a mod p = -1,** then $\mathbf{a^2 \bmod p = 1}$.

Conversely, if $\mathbf{a^2 \bmod p = 1}$, **then** $\mathbf{(a \bmod p)^2 = 1,}$ which is true only for **a mod p = 1** or **a mod p = -1.**

Second Property is states as follows:

- Let p be a prime number greater than 2. We can then write

$$p - 1 = 2^k q \text{ with } k > 0, \; q \text{ odd}.$$

- Let a be any integer in the range $1 < a < p - 1$. Then one of the two following conditions is true:

1. a^q is congruent to 1 modulo p. That is, $a^q \bmod p = 1$, or equivalently,

$$a^q \equiv 1 \pmod{p}.$$

2. One of the numbers $a^q, \; a^{2q}, \; a^{4q}, \; \ldots \; , \; a^{2^{k-1}q}$ is congruent to -1 modulo p.

That is, there is some number j in the range $(1 \leq j \leq k)$ such that

$$a^{2^{j-1}q} \bmod p = -1 \bmod p = p - 1$$

or equivalently,

$$a^{2^{j-1}q} \equiv -1 \pmod{p}$$

Algorithm: Miller Rabin Algorithm

- The procedure TEST takes a candidate integer ***n*** as input and returns the result `composite` if *n* is definitely not a prime, and the result `inconclusive` (`maybe prime`) if ***n*** may or may not be a prime.
- algorithm is:

TEST (*n*) is:

1. Find integers *k*, *q*, *k* > 0, *q* odd, (*n* is a +ve odd integer such that so that n≥3), $(n-1)=2^k q$
2. Select a random integer a, $1<a<n-1$
3. **if** $a^q \bmod n = 1$ **then** return ("maybe prime");
4. **for** *j* = 0 **to** *k* – 1 **do**
5. **if** ($a^{2^j q} \bmod n = n-1$)
   then return(" maybe prime ")
6. return ("composite")

Algorithm: Miller Rabin Algorithm

Let us apply the test to the prime number $n = 29$. We have $(n - 1) = 28 = 2^2(7) = 2^k q$. First, let us try $a = 10$. We compute $10^7 \bmod 29 = 17$, which is neither 1 nor 28, so we continue the test. The next calculation finds that $(10^7)^2 \bmod 29 = 28$, and the test returns `inconclusive` (i.e., 29 may be prime). Let's try again with $a = 2$. We have the following calculations: $2^7 \bmod 29 = 12$; $2^{14} \bmod 29 = 28$; and the test again returns `inconclusive`. If we perform the test for all integers a in the range 1 through 28, we get the same `inconclusive` result, which is compatible with n being a prime number.

Now let us apply the test to the composite number $n = 13 \times 17 = 221$. Then $(n - 1) = 220 = 2^2(55) = 2^k q$. Let us try $a = 5$. Then we have $5^{55} \bmod 221 = 112$, which is neither 1 nor $220 (5^{55})^2 \bmod 221 = 168$. Because we have used all values of j (i.e., $j = 0$ and $j = 1$) in line 4 of the TEST algorithm, the test returns `composite`, indicating that 221 is definitely a composite number. But suppose we had selected $a = 21$. Then we have $21^{55} \bmod 221 = 200$; $(21^{55})^2 \bmod 221 = 220$; and the test returns `inconclusive`, indicating that 221 may be prime. In fact, of the 218 integers from 2 through 219, four of these will return an inconclusive result, namely 21, 47, 174, and 200.

Probabilistic Considerations

- if Miller-Rabin returns “composite” the number is definitely not prime
- otherwise is a prime or a pseudo-prime
- chance it detects a pseudo-prime is $< {}^{1}/_{4}$
- hence if repeat test with different random a then chance n is prime after t tests is:
  - Pr(n prime after t tests) = $1-4^{-t}$
  - eg. for t=10 this probability is > 0.99999

Prime Distribution

- Prime number theorem states that primes near n are spaced on the average one every (`ln n`) integers.
- but can immediately ignore evens
- so in practice need only test `0.5 ln(n)` numbers of size `n` to locate a prime
  - note this is only the “average”
  - sometimes primes are close together
  - other times are quite far apart

- For example, if a prime on the order of magnitude of 2^{200} were sought, then about $0.5 \ln(2^{200}) = 69$ trials would be needed to find a prime.

Primitive Roots

- For any prime number p, if we have a number **a** such that **powers of a mod p** generate all the numbers between 1 to p−1 then a is called **Primitive Root** of **p.**
- from Euler's theorem have $a^{\phi(n)} \bmod n = 1$ i.e $a^{\phi(n)} \equiv 1 \bmod n$
- consider $a^m = 1 \pmod n$, $GCD(a,n) = 1$
  - must exist for $m = \phi(n)$ but may be smaller
  - once powers reach m, cycle will repeat
- if smallest is $m = \phi(n)$ then a is called a **primitive root**
- if p is prime, then successive powers of a "generate" the group mod p
- these are useful but relatively hard to find

Primitive Root

To see this last point, consider the powers of 7, modulo 19:

$$
\begin{aligned}
7^1 &\equiv & &\equiv 7 \pmod{19} \\
7^2 &= 49 = 2 \times 19 + 11 & &\equiv 11 \pmod{19} \\
7^3 &= 343 = 18 \times 19 + 1 & &\equiv 1 \pmod{19} \\
7^4 &= 2401 = 126 \times 19 + 7 & &\equiv 7 \pmod{19} \\
7^5 &= 16807 = 884 \times 19 + 11 & &\equiv 11 \pmod{19}
\end{aligned}
$$

There is no point in continuing because the sequence is repeating. This can be proven by noting that $7^3 \equiv 1 \pmod{19}$, and therefore, $7^{3+j} \equiv 7^3 7^j \equiv 7^j \pmod{19}$, and hence, any two powers of 7 whose exponents differ by 3 (or a multiple of 3) are congruent to each other (mod 19). In other words, the sequence is periodic, and the length of the period is the smallest positive exponent m such that $7^m \equiv 1 \pmod{19}$.

For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15

Ordinary Logarithms

- With ordinary positive real numbers, the logarithm function is the inverse of exponentiation.
- The logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal the number. That is, for base *x* and for a value y

$$y = x^{\log_x(y)}$$

- The properties of logarithms include

$$\log_x(1) = 0$$
$$\log_x(x) = 1$$
$$\log_x(yz) = \log_x(y) + \log_x(z)$$
$$\log_x(y^r) = r \times \log_x(y)$$

Discrete Logarithm

- Consider a primitive root **a** for some prime number **p** (the argument can be developed for nonprimes as well).
  – Then we know that the powers of **a** from 1 through $(p - 1)$ produce each integer from 1 through $(p - 1)$ exactly once.
- By the definition of modular arithmetic, any integer **b** satisfies

$$b \equiv r \pmod{p} \quad \textit{for some } r\textit{, where } 0 \leq r \leq (p-1)$$

- It follows that for any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$b \equiv a^i \pmod{p} \quad \textit{where } 0 \leq r \leq (p-1)$$

- This exponent i is referred to as the **discrete logarithm** of the number b for the base $a \pmod{p}$. We denote this value as $\text{dlog}_{a,p}(b)$
- Note the following:

$\text{dlog}_{a,p}(1) = 0$ because $a^0 \bmod p = 1 \bmod p = 1$
$\text{dlog}_{a,p}(a) = 1$ because $a^1 \bmod p = a$

Private Key Cryptosystems

- traditional **private/secret/single key** cryptography uses **one** key
- shared by both sender and receiver
- if this key is disclosed communications are compromised
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender

Public-Key Cryptography

- Public-key encryption, first publicly proposed by *Diffie and Hellman* in 1976.
- public-key cryptography is **asymmetric**
  - uses two keys - a **public** & a **private** key.
  - parties are **not equal**
  - The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication.
- uses clever application of **number theoretic concepts** to function

Why Public-Key Cryptography?

- developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **digital signatures** – how to verify a message comes intact from the claimed sender

Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public Key Cryptosystem

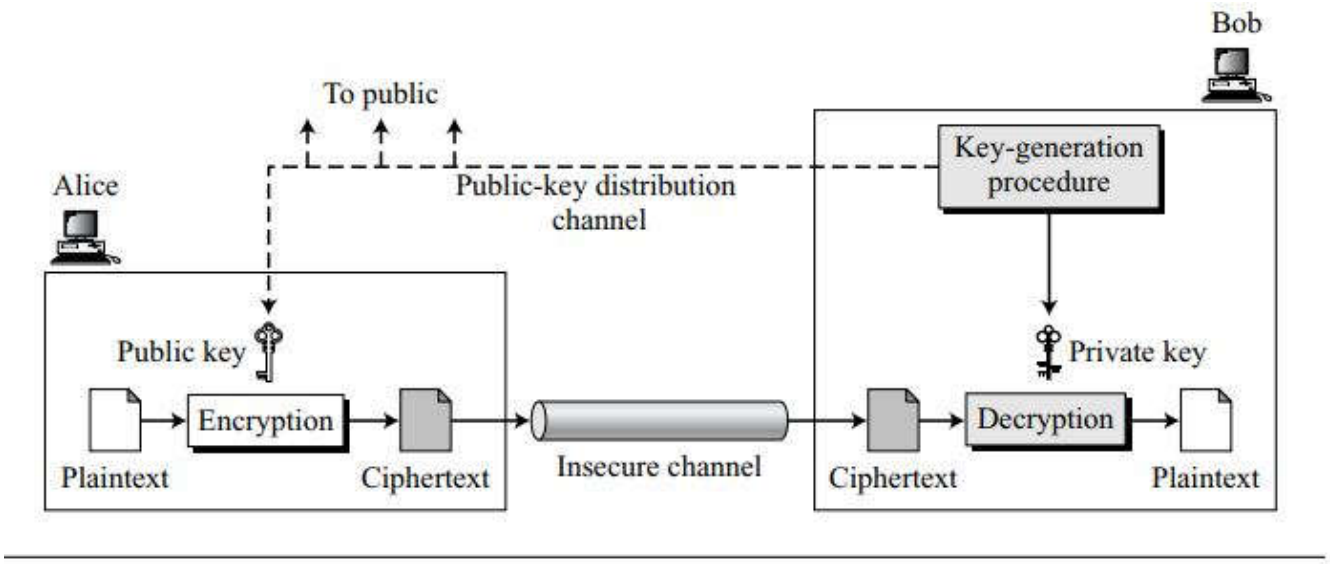


Figure: Encryption with public key

Six ingredients:

1. Plaintext
2. Encryption algorithm
3. Public and Private keys
4. Ciphertext
5. Decryption algorithm

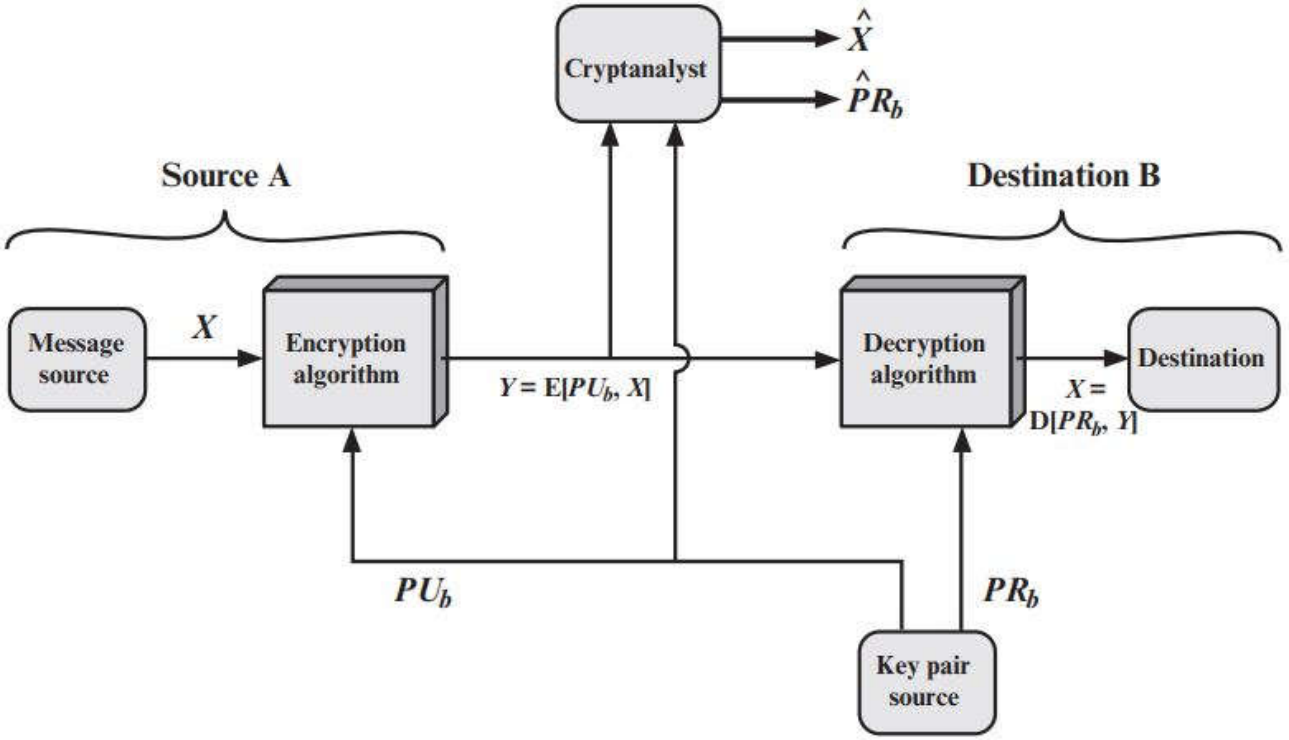
Public Key Cryptosystem

1. **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
2. **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext
3. **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
4. **Ciphertext:** This is the encrypted message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
5. **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext

Public-Key Characteristics

- Public-Key algorithms rely on two keys where:
  - it is computationally infeasible to find decryption key knowing only algorithm & encryption key
  - it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)

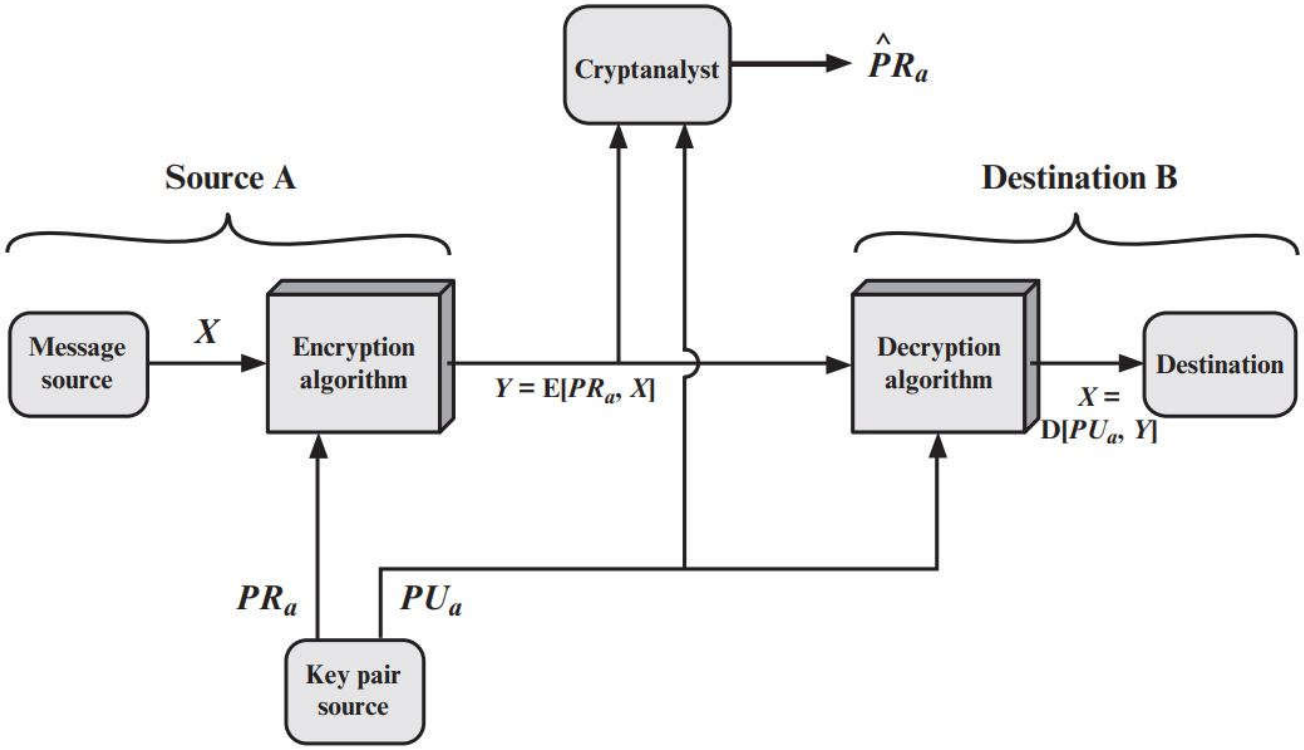
Public-Key Cryptosystem: Confidentiality



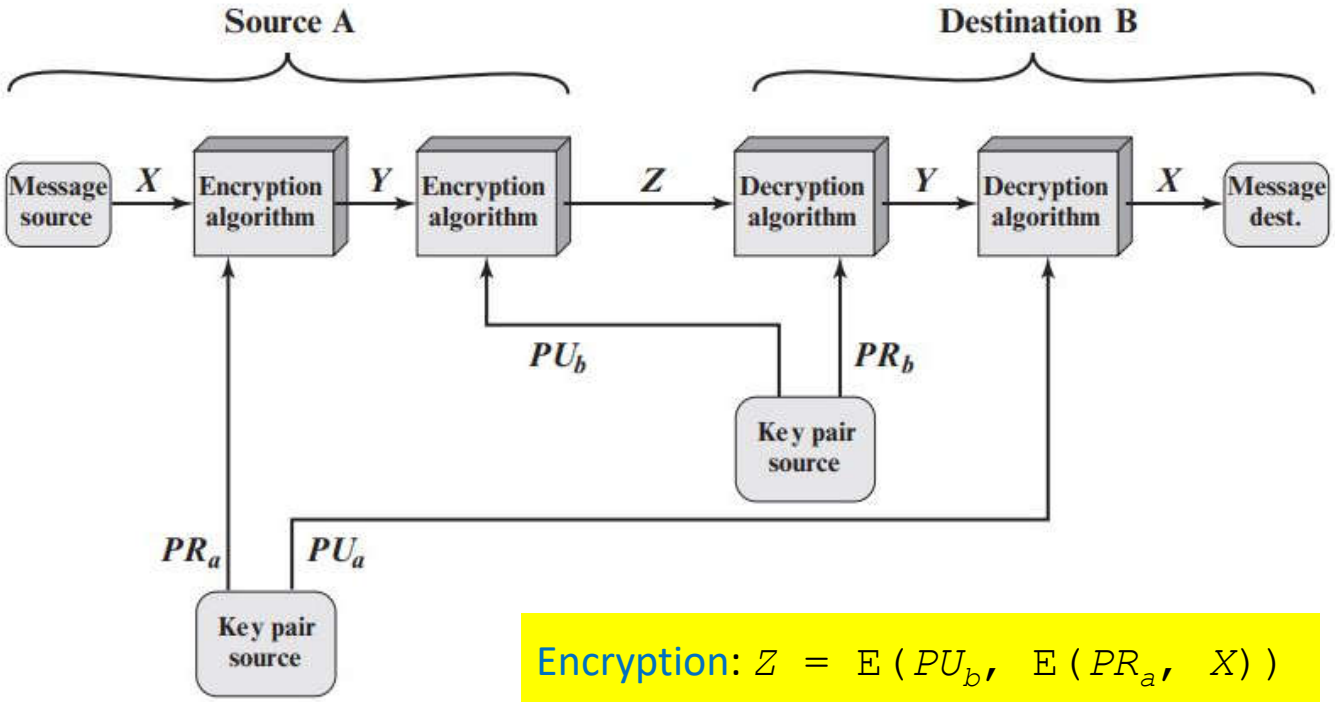
Encryption: $Y = E(PU_b, X)$
Decryption: $X = D(PR_b, Y)$

Figure: Public-Key Cryptosystem: Confidentiality

Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Authentication & Confidentiality



Encryption: $Z = E(PU_b, E(PR_a, X))$

Decryption: $X = D(PU_a, D(PR_b, Z))$

Begin by encrypting a message, using the sender's private key, which provides the digital signature

Next, encrypt with the receiver's public key, thus providing the confidentiality.

Figure: Public-Key Cryptosystem: Authentication & Secrecy

Applications of Public Key Cryptosystems

- can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
- some algorithms are suitable for all uses, others are specific to one.

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

Public-Key Cryptoanalysis

- vulnerable to brute-force attack → increase key size
- compute the private key given the public key → not proven yet
- probable message attack
  - an adversary could encrypt all possible keys using the public key and could discover the encrypted key by matching the transmitted ciphertext.

Conventional and Public Key Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

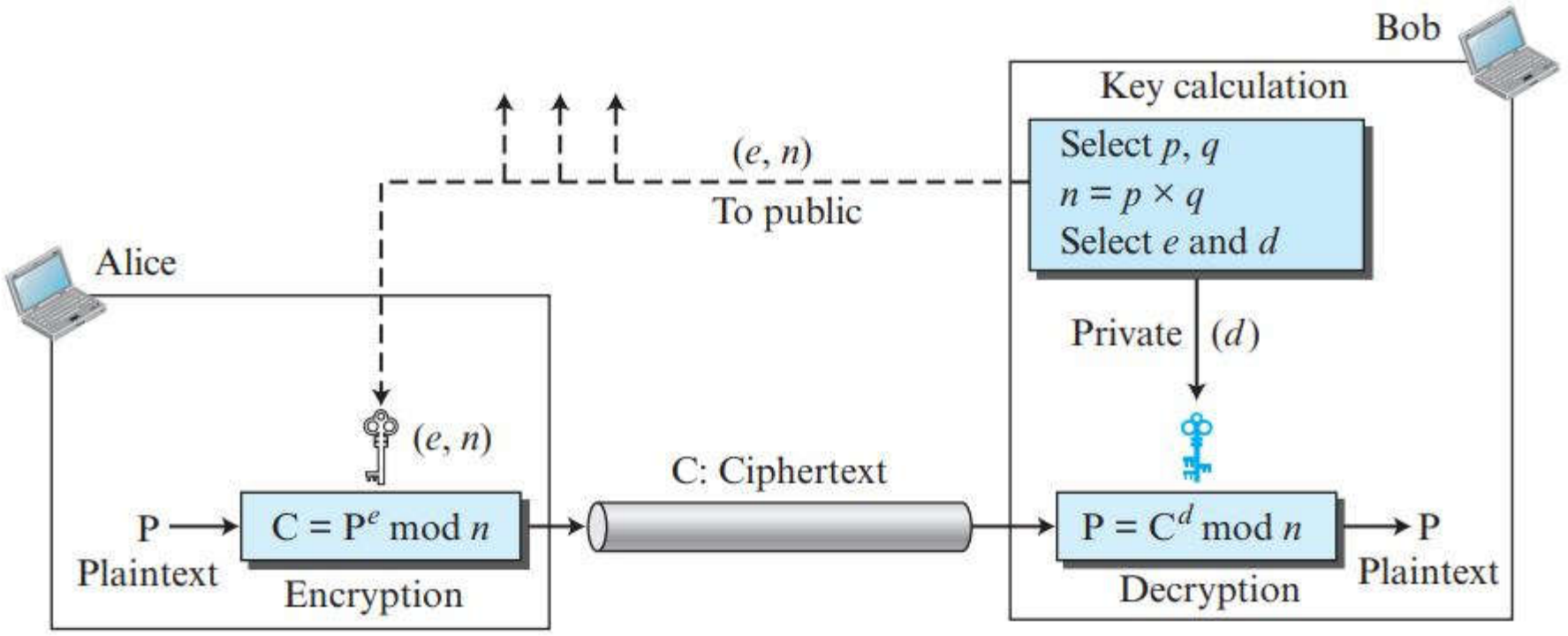
RSA Algorithm

- RSA is one of the first public key cryptosystem and is widely used for secure data transmission.
- Developed in 1977 by Ron **R**ivest, Adi **S**hamir, and Len **A**dleman at MIT and first published in 1978.
- Most widely used asymmetric key encryption.
- Used in security protocol such as IPSEC, SSH, TLS etc.
- The **RSA** scheme is a cipher in which the plaintext and ciphertext are integers between 0 and *n* - 1 for some *n*. A typical size for *n* is 1024 bits, or 309 decimal digits. That is, *n* is less than 2^{1024} .
- RSA uses two exponents, *e* and *d*, where *e* is public and *d* is private.
- Suppose *P* is the plaintext and *C* is the ciphertext. Alice uses $C = P^e \bmod n$ to create ciphertext C from plaintext P; Bob uses $P = C^d \bmod n$ to retrieve the plaintext sent by Alice. The modulus *n*, a very large number, is created during the key generation process.

RSA Algorithm

- A user of RSA creates and then published a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret.
- The RSA algorithm involves four steps:
  - Key generation
  - Key distribution
  - Encryption
  - Decryption

RAS Alogirthm



RSA: Key Generation

- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus $n=p.q$
  - note $\varnothing(n)=(p-1)(q-1)$
- selecting at random the encryption key e
    - where $1<e<\varnothing(n)$, $\gcd(e,\varnothing(n))=1$
- solve following equation to find decryption key d
  - $e.d \equiv 1 \bmod \varnothing(n)$ and $0 \le d \le n$
  - [i.e. $j.\varnothing(n)-1 = e.d \rightarrow d = (j \times \varnothing(n)-1)/e$], $j = 1, 2, \ldots$
  - [for j, d should be integer.]
- publish their public encryption key: PU={e,n}
- keep secret private decryption key: PR={d,n}

RSA: Encryption and Decryption

- Given public key (e , n) and private key (d, n) are computed
- To encrypt bit pattern, *P*, compute cipher text C as:

$$C = P^e \ mod \ n$$

- To decrypt bit pattern, *C*, compute

$$P = C^d \ mod \ n$$

RSA: Example

- Pick two prime numbers: $p=3,\ q = 5$.
- $n=p \times q = 3 \times 5=15$
- $\phi(\mathbf{n})=(p-1)(q-1) = (3-1)(5-1) = 2 \times 4 = 8$.
- Choose e satisfying $1<e<\Phi(\mathbf{n})$.
- Let us choose $e=3$,which do not share any common factors with 8 rather than 1.
- Compute d satisfying $de\{mod\Phi(\mathbf{n})\}=1$
- $So\ d \times 3\{mod\ 8\} = 1$
- Let us choose $d = 11$ which satisfy the relation

RSA: Example

- So public key (e,n) is (3,15) which is released publicly and the persons that want to send the message use this key to encrypt the message and send it to the receiver.
- Private key(d,n) is (11,15) which is kept secret by the receiver.

- Let us consider the message be 2.
- So, at encryption process, the sender uses the public key to encrypt the message. Resulting cipher text will be:
- $C = P^e \pmod{n} = 2^3 \pmod{15} = 8$.
- At decryption process, the private key is used to decrypt the cipher text. Plain text is obtained as:
- $P = C^d \pmod{n} = 8^{11} \pmod{15} = 2$.
- Hence the original message 2 is obtained at receiver end after decryption.

RSA Example: Key Setup

1. Select primes: $p=17$ & $q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e: $\gcd(e,160) = 1$; choose $e=7$
5. Determine d: $de \equiv 1 \bmod 160$ and $d < 160$ Value is $d=23$ since $23 \times 7 = 161 = 10 \times 160 + 1$
6. Publish public key $PU = \{7, 187\}$
7. Keep secret private key $PR = \{23, 187\}$

RSA Example - En/Decryption

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88<187$)
- encryption:

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

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

Modular Arithmetic

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$
$$88^1 \bmod 187 = 88$$
$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$
$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$
$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$
$$11^1 \bmod 187 = 11$$
$$11^2 \bmod 187 = 121$$
$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$
$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$
$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79,720,245 \bmod 187 = 88$$

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)
  - timing attacks (on running of decryption)
  - chosen ciphertext attacks (given properties of RSA)

Distribution of Public Keys

- Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:
  - public announcement
  - publicly available directory
  - public-key authority
  - public-key certificates

Public Announcement of Public Keys

- users distribute public keys to recipients or broadcast to community at large
  - eg. append PGP keys to email messages or post to news groups or email list
- major weakness is forgery
  - anyone can create a key claiming to be someone else and broadcast it
  - until forgery is discovered can masquerade as claimed user

Figure: Uncontrolled Public-Key Distribution

Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
  - contains {name,public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically
- still vulnerable to tampering or forgery

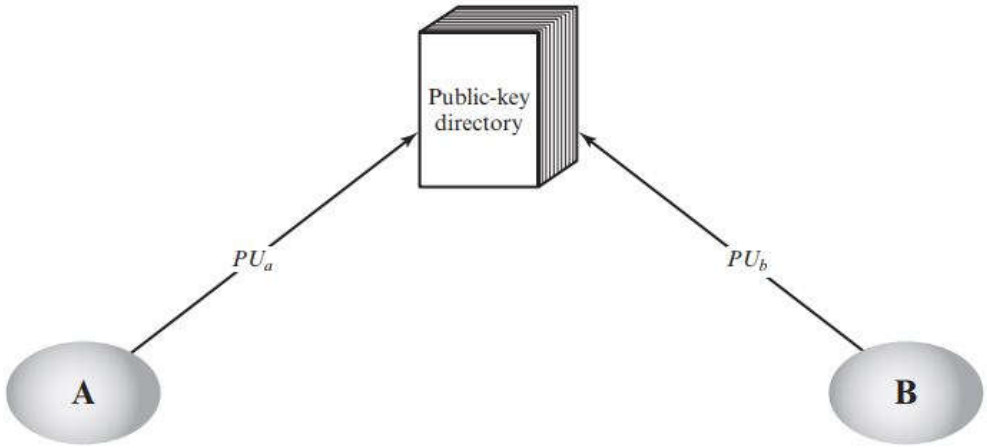
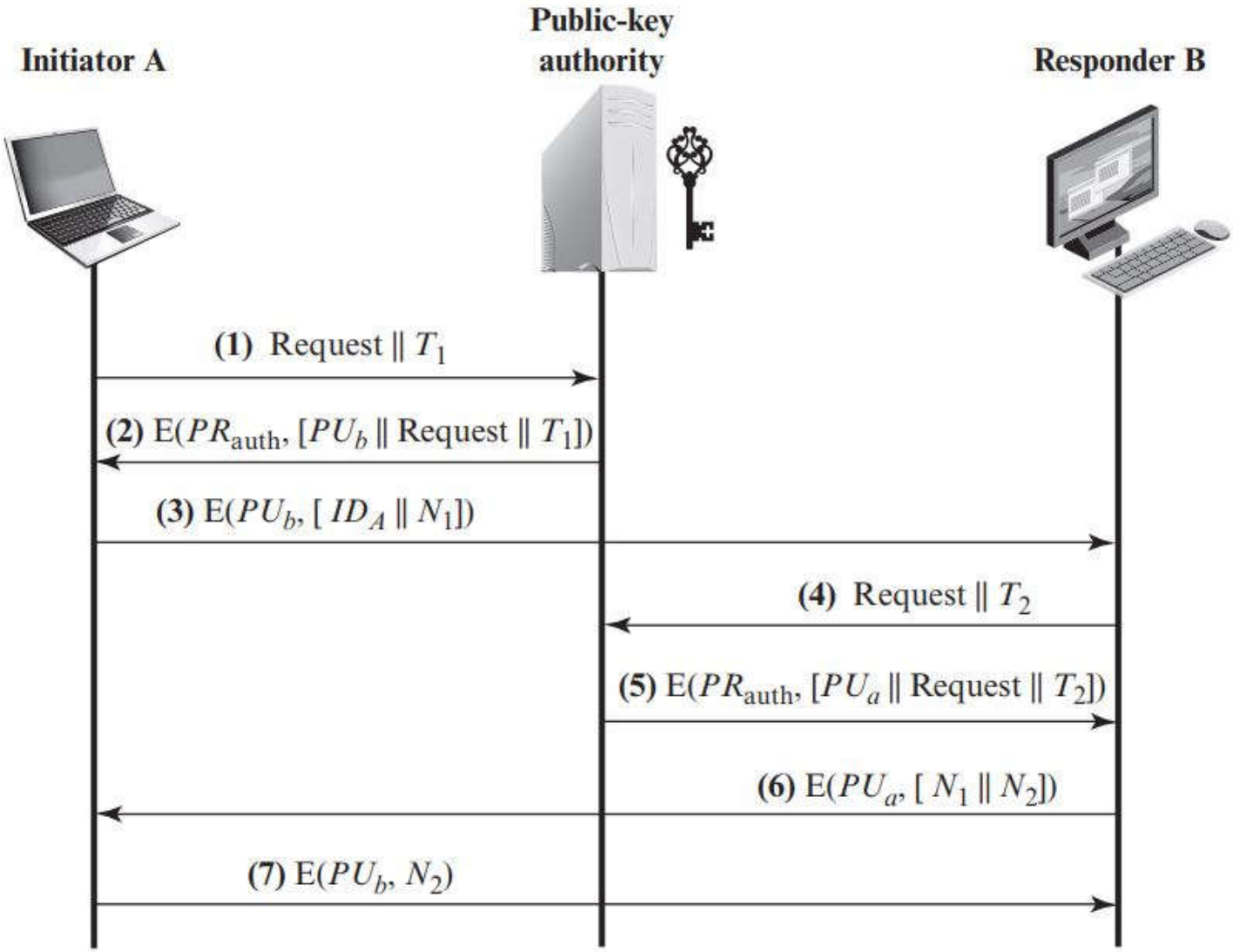


Figure: Public-Key Publication

Public-Key Authority

- improve security by tightening control over distribution of keys from directory
- has properties of directory and requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed

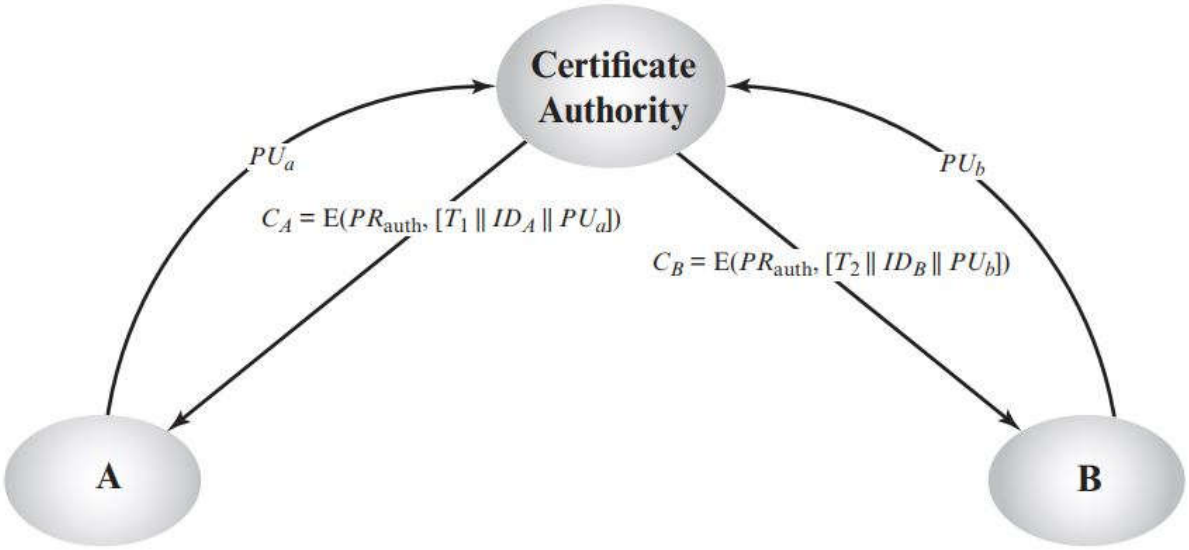


T = Timestamp
PR = Private key
PU = Public key
ID = identifier
N = Nonce, used to identify transaction uniquely

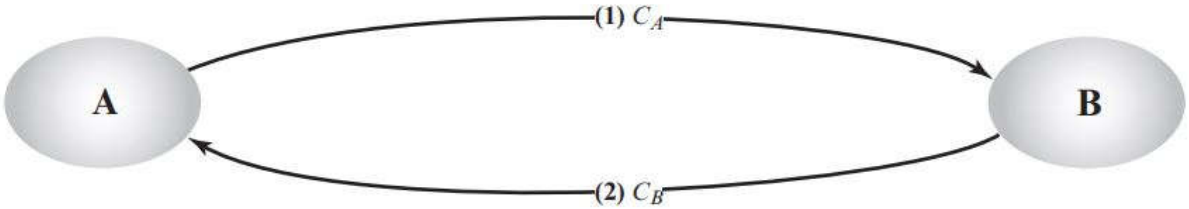
Figure: Public-Key Distribution Scenario

Public-Key Certificates

- certificates allow key exchange without real-time access to public-key authority
- a certificate binds **identity** to **public key**
  - usually with other info such as period of validity, rights of use etc
- with all contents **signed** by a trusted Public-Key or **Certificate Authority** (CA)
- can be verified by anyone who knows the public-key authorities public-key



(a) Obtaining certificates from CA



(b) Exchanging certificates

Figure: Exchange of Public-Key Certificates

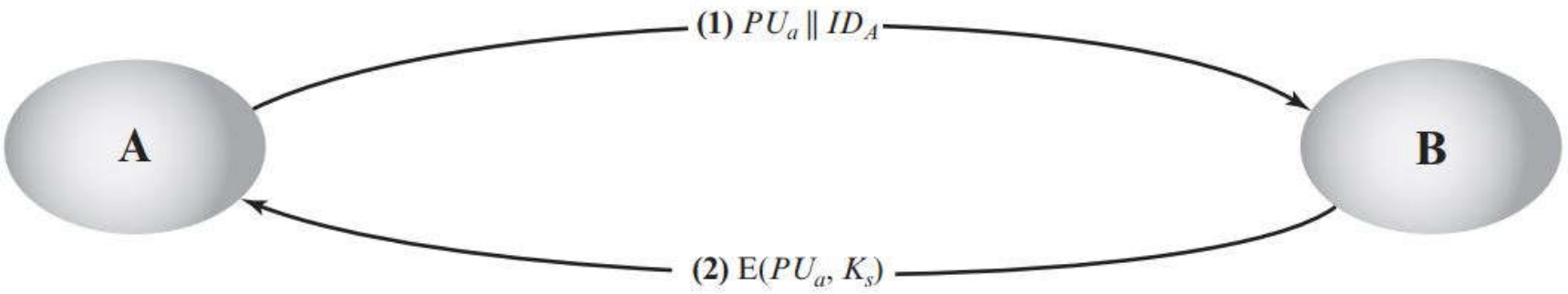
Public-Key Distribution of Secret Keys

- one of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution

Simple Secret Key Distribution

- proposed by Merkle in 1979
  - A generates a new temporary public key pair {PU_a, PR_a}
  - A sends B the public key PU_a and their identity, ID_a
  - B generates a session key K_s sends it to A encrypted using the supplied public key
  - A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s

- he protocol depicted is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message Such an attack is known as a **man-in-the-middle attack** .



Secret Key Distribution with Confidentiality and Authentication

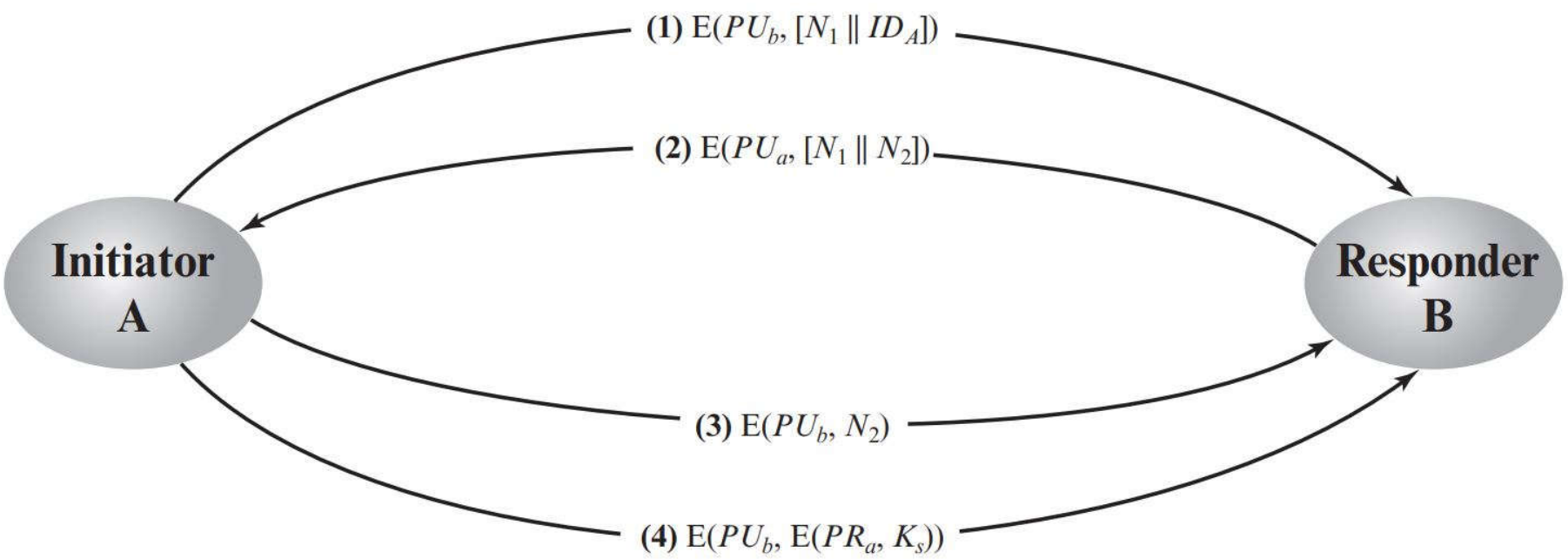


Figure: Public-Key Distribution of Secret Keys

Diffie-Hellman Key Exchange

- First public-key type scheme proposed by Diffie & Hellman in 1976 along with the exposition of public key concepts.
- It is a practical method for **public exchange of a secret key** and that can be used **subsequent encryption of messages**.
- It is used in a number of commercial products.
- The Diffie–Hellman algorithm depends for its effectiveness on the **difficulty of computing discrete logarithms**.
- For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$b \equiv a^i \pmod{p} \quad \textit{where } 0 \leq r \leq (p-1)$$

- This exponent i is referred to as the **discrete logarithm** of the number b for the base $a \pmod{p}$. We denote this value as $\text{dlog}_{a,p}(b)$

Diffie-Hellman Key Exchange: Algorithm

- For this scheme, there are two publicly known numbers:
- a prime number q and an integer α that is a primitive root of q. Suppose the users A and B wish to create a shared key.
- User A generates a random integer x_A such that:
  - chooses a secret key (number): $x_A < q$
  - compute their **public key**: $y_A = \alpha^{x_A} \bmod q$
- User B generates their key as
  - chooses a secret key (number): $x_B < q$
  - compute their **public key**: $y_B = \alpha^{x_B} \bmod q$
- each user makes public that key y_A, y_B and keeps the X value private

Diffie-Hellman Key Exchange: Algorithm

- User A compute the key as:

$$K = y_B^{x_A} \bmod q$$

- User B computes the key as :

$$K = y_A^{x_B} \bmod q$$

- K is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys attacker needs an x, must solve discrete log

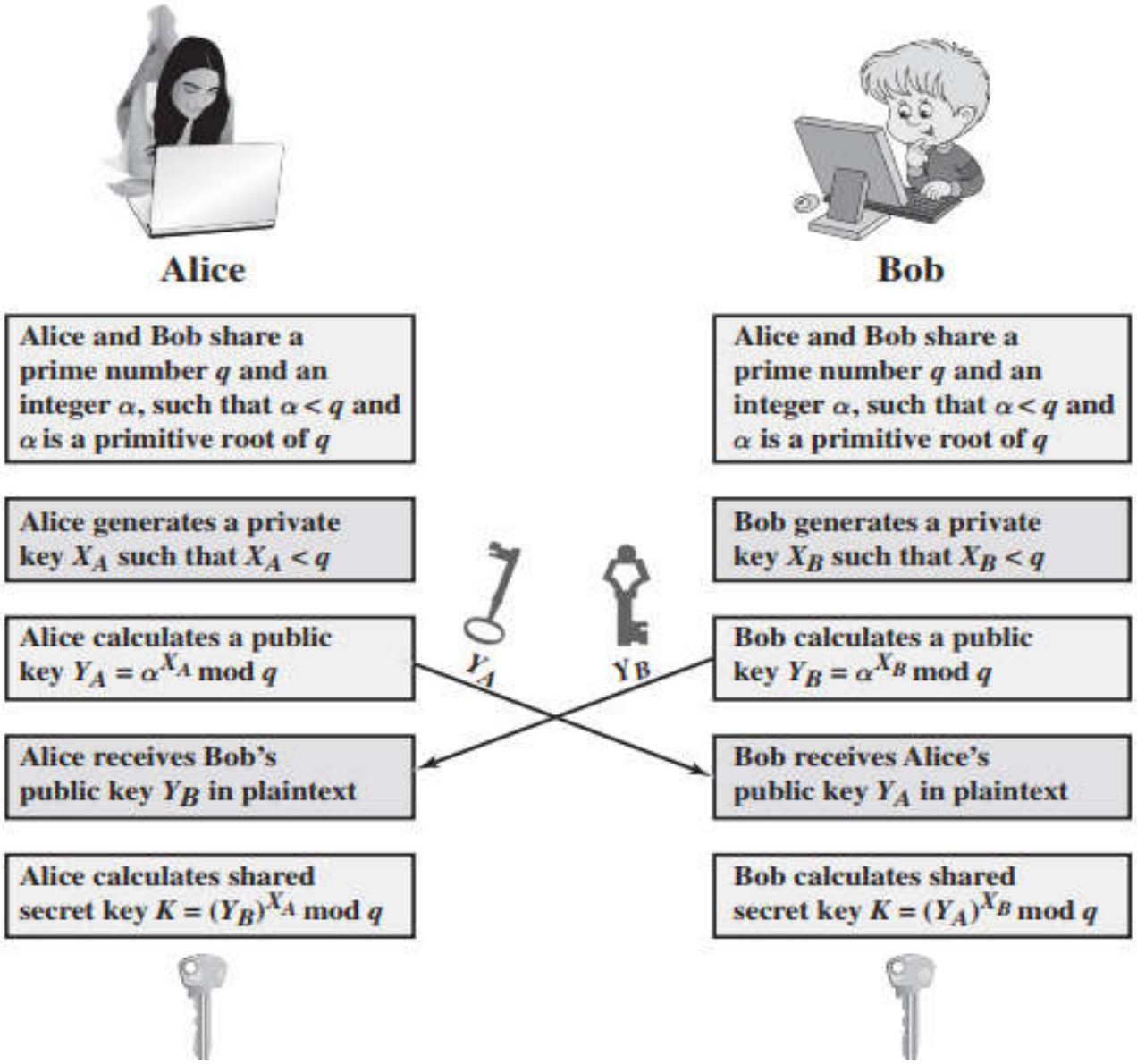


Figure: The Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange: Algorithm

- These two calculations produce identical results:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

by the rules of modular arithmetic

Diffie-Hellman Key Exchange

- Now consider an adversary who can observe the key exchange and wishes to determine the secret key K.
- Because X_A and X_B are private, an adversary only has the following ingredients to work with: q, α, Y_A, and Y_B.
- Thus, the adversary is forced to take a discrete logarithm to determine the key. For example, to determine the private key of user B, an adversary must compute

$$X_B = \text{dlog}_{\alpha,q}(Y_B)$$

- The adversary can then calculate the key *K* in the same manner as user B calculates it. That is, the adversary can calculate *K* as

$$K = y_A^{x_B} \bmod q$$

The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Global Public Elements

q	Prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Figure: The Diffie-Hellman Key Exchange Algorithm

Diffie-Hellman Key Exchange: Example

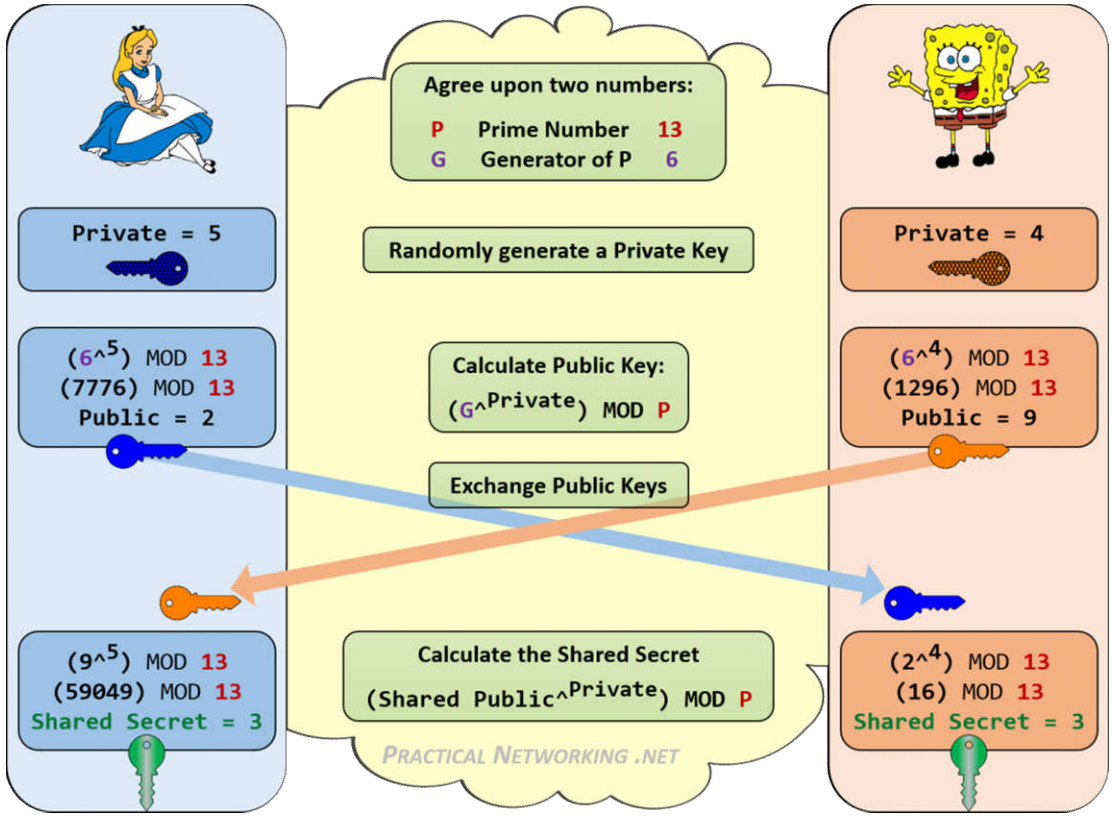
- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $\alpha=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute respective public keys:
  - $y_A = 3^{97} \bmod 353 = 40$ (Alice)
  - $y_B = 3^{233} \bmod 353 = 248$ (Bob)
- compute shared session key K as:
  - $K = y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)
  - $K = y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

We assume an attacker would have available the following information:
$q = 353, \alpha=3, y_A = 40, Y_B = 248$

it would be possible by brute force to determine the secret key 160

In particular, an attacker E can determine the common key by discovering a solution to the equation $3^a \bmod 353 = 40$ or the equation $3^b \bmod 353 = 248$.
The brute-force approach is to calculate powers of 3 modulo 353, stopping when the result equals either 40 or 248.
The desired answer is reached with the exponent value of 97, which provides $3^{97} \bmod 353 = 40$.
With larger numbers, the problem becomes impractical.

Diffie-Hellman Key Exchange: Example



Diffie-Hellman Key Exchange: Man-in-Middle Attack

- The Diffie-Helman Key Exchange protocol depicted in figure is insecure against a Man-in-Middle Attack; Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows:

1. Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2} and then computing the corresponding public keys Y_{D1} and Y_{D2}.
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates:

$$K2 = y_A^{X_{D2}} \bmod q$$

4. Bob receives Y_{D1} and calculates : $K1 = y_{D1}^{X_B} \bmod q$
5. Bob transmits Y_B to Alice.
6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates: $K1 = y_B^{X_{D1}} \bmod q$.
7. Alice receives Y_{D2} and calculate : $K_2 = y_{D2}^{X_A} \bmod q$.

Diffie-Hellman Key Exchange: Man-in-Middle Attack

- At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key *K*1 and Alice and Darth share secret key *K*2.
- All future communication between Bob and Alice is compromised in the following way.
  1. Alice sends an encrypted message *M*: E(*K*2, *M*).
  2. Darth intercepts the encrypted message and decrypts it to recover *M*.
  3. Darth sends Bob *E*(*K*1, *M*) or E(*K*1, *M*′), where *M*′ is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.
- The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates.

Diffie-Hellman Key Exchange: Man-in-Middle Attack

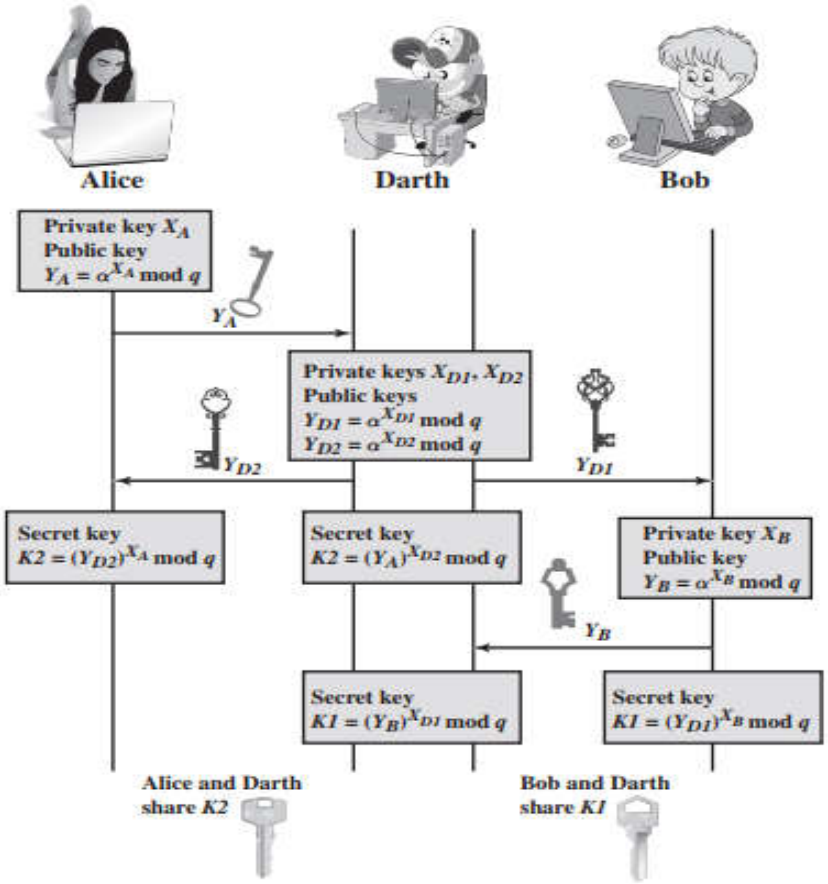


Figure: Man-in-Middle Attack

Elgamal Cryptographic System

- In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique.
- The Elgamal cryptosystem is used in some form in a number of standards including the Digital Signature Standard (DSS) and the S/MIME e-mail standard.
- This algorithm is based on the difficulty of finding discrete logarithm in a cyclic group.
- As with Diffie-Hellman, the global elements of Elgamal are a prime number $\mathbf{q}$ and $\boldsymbol{\alpha}$, which is a primitive root of $\mathbf{q}$.

ElGamal Cryptographic System

- The steps in ElGamal Algorithm are summarized as follows:
  1. Alice generates a public/private key pair;
  2. Bob encrypts using Alice's public key; and
  3. Alice decrypts using private key.

ElGamal Cryptographic System

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

ElGamal Cryptographic System

Encryption by Bob with Alice's Public Key

Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key

Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

ElGamal Cryptographic System

- **Demonstration why the ElGamal scheme works:**

$K = (Y_A)^k \bmod q$	K is defined during the encryption process
$K = (\alpha^{X_A} \bmod q)^k \bmod q$	substitute using $Y_A = \alpha^{X_A} \bmod q$
$K = \alpha^{kX_A} \bmod q$	by the rules of modular arithmetic
$K = (C_1)^{X_A} \bmod q$	substitute using $C_1 = \alpha^k \bmod q$

Next, using K, we recover the plaintext as

$$C_2 = KM \bmod q$$

$$(C_2K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

ElGamal Cryptographic System: Example

- let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15} We choose $\alpha = 10$. Alice generates a key pair as follows:

1. Alice choose $X_A = 5$.
2. Then $Y_A = \alpha^{X_A} \bmod q = 10^5 \bmod 19 = 3$.
3. Alice's private key is 5 and Alice public key is $\{q, \alpha, Y_A\} = \{19, 10, 3\}$.

   Suppose Bob wants to send the message with the value $M = 17$. Then, **for encryption**:
4. Bob choose $k = 6$.
5. Then $K = Y_A^k \bmod q = 3^6 \bmod 19 = 7^{29} \bmod 19 = 7$.
6. So,

   $C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$

   $C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$
7. Bob sends the ciphertext $(C_1, C_2) = (11, 5)$

For decryption

1. Alice calculates $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$
2. Then K^{-1} in GF(19) is $7^{-1} \bmod 19 = 11$
3. Finally, $M = (C_2 K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$

ElGamal Cryptographic System

- If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of k should be used for each block. If k is used for more than one block, knowledge of one block M_1 of the message enables the user to compute other blocks as follows. Let

$$C_{1,1} = \alpha^k \bmod q; \quad C_{2,1} = = KM_1 \bmod q$$
$$C_{1,2} = \alpha^k \bmod q; \quad C_{2,2} = = LM_2 \bmod q$$

- Then,

$$C_{2,1} = KM_1 \bmod q; \quad C_{2,1} = M_1 \bmod q$$
$$C_{2,2} = KM_2 \bmod q; \quad C_{2,1} = M_2 \bmod q$$

- If M1 is known, them M2 is easily computed as

$$M_2 = (C_{2,1})^{-1} C_{2,2} M_1 \bmod q$$

Security of ElGamal

- The security of Elgamal is based on the difficulty of computing discrete logarithms.
  - To recover A's private key, an adversary would have to compute
    $$X_A = \text{dlog}_{a,q}(Y_A).$$
  - Alternatively, to recover the one-time key K, an adversary would have to determine the random number k, and this would require computing the discrete logarithm
    $$k = \text{dlog}_{a,q}(C_1)$$
- These calculations are regarded as infeasible if p is at least 300 decimal digits and $q - 1$ has at least one "large" prime factor.

References

- W. Stallings, *Cryptography and Network Security: Principles and Practice*
- B. A. Forouzan, *Cryptography & Network Security*, Tata Mc Graw Hill.