Public Key Cryptography

Lecture Outline

Public key encryption

- Public key Cryptography
- RSA key Management
- Diffie-Hellman Key Exchange
- ElGamal Encryption Scheme

Public Key Cryptography

Symmetric Key Cryptography

Requires sender, receiver know shared secret key

Q1: How is the key distributed?

Q2: How to agree on key in first place (particularly if never "met")?

Public Key Cryptography

- Radically different approach [Diffie-Hellman76, RSA78]
- Sender, receiver do not share secret key
- Public encryption key known to all
- Private decryption key known only to receiver

Why Public Key Cryptography?

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption.

1. Key distribution: Key distribution under symmetric encryption requires either

a. That two communicants already share a key, which somehow has been distributed to them;
b. The use of a Key Distribution Center (KDC). This requirement negated the very essence of cryptography: the ability to maintain total secrecy over your own communication.

As Diffie put it [DIFF88], "*what good would it do after all to develop impenetrable cryptosystems, if their users were forced to share their keys with a KDC that could be compromised by either burglary or subpoena?*"

Why Public Key Cryptography?

2 Digital signatures. If the use of cryptography was to become widespread, not just in military situations but for commercial and private purposes, then electronic messages and documents would need the equivalent of signatures used in paper documents.

- That is- Is it possible to devise a method that would stipulate, to the satisfaction of all parties, that a digital message had been sent by a particular person?

Public Key Cryptography

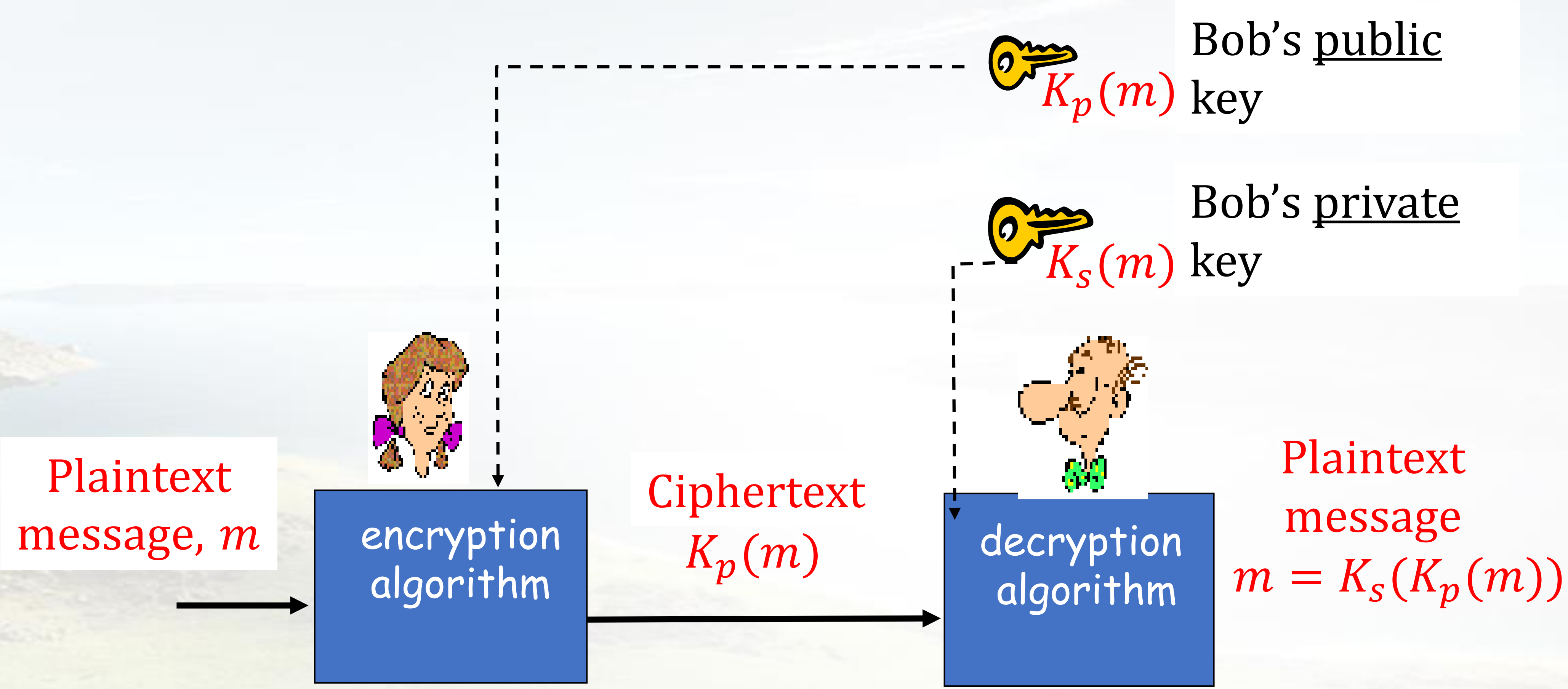
Public-key/asymmetric crypto involves use of two keys

- **Public-key:** Known by anybody, and can be used to encrypt messages and verify signatures
- **Private-key:** Known only to recipient, used to decrypt messages and sign (create) signatures

Asymmetric because

- Can encrypt messages or verify signatures w/o ability to decrypt messages or create signatures

Public Key Cryptography



Public Key Cryptography

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.

1. It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
   - In addition, some algorithms, such as RSA, also exhibit the following characteristic.
2. Either of the two related keys can be used for encryption, with the other used for decryption.

Public Key Cryptography

A public-key encryption scheme has six ingredients

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

Public Key Cryptography

A public-key encryption scheme has six ingredients

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 scrambled 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 Cryptography

The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice’s public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice’s private key.

Public Key Cryptography

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 to Work: 1. One algorithm is used for encryption and 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).

Public Key Cryptography

Conventional Encryption	Public-Key Encryption
Needed for Security: 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	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 no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Public Key Cryptography

Applications for Public-Key Cryptosystems.

In broad terms, we can classify the use of **public-key cryptosystems** into three categories

1. **Encryption /decryption:** The sender encrypts a message with the recipient's public key.

2. **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

3. **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Public Key Cryptography

Requirements for Public-Key Cryptography

Conditions that cryptographic algorithms must fulfill

1. It is computationally easy for a party B to generate a pair (public key PU_b , private key PR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, $M,$ to generate the corresponding ciphertext: $C = E(PU_b, M)$

Public Key Cryptography

Conditions that cryptographic algorithms must fulfill

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message: $M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b,to determine the private key, PR_b.
5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext C, to recover the original message, M.

Public Key Cryptography

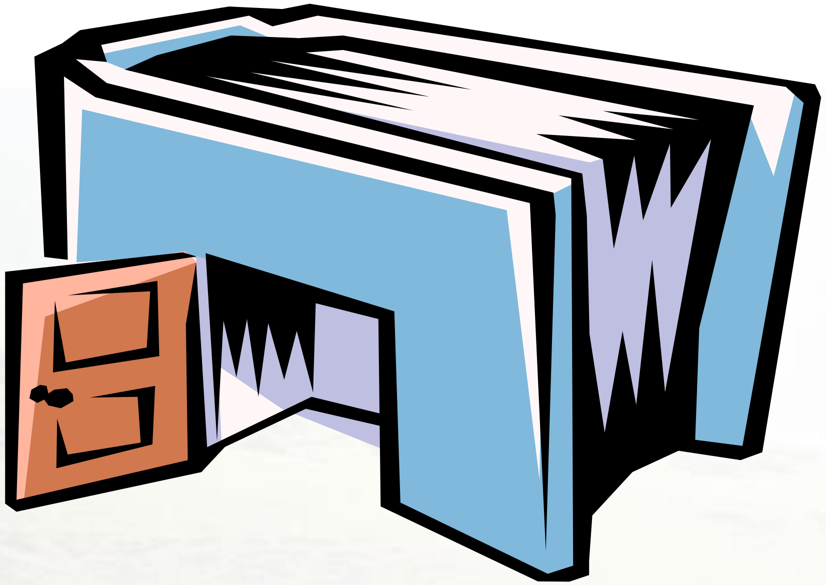
Requirements for Public-Key Cryptography

The requirements boil down to the need for a trap-door one-way function

One way function: Is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy, whereas the calculation of the inverse is infeasible:

$Y = f(X)$ *easy*

$X = f^{-1}(Y)$ *infeasible*



Public Key Cryptography

Requirements for Public-Key Cryptography

If “one-way function” goes $c \leftarrow F(m, k)$, then public-key encryption is a “trap-door” function:

- Easy to compute $c \leftarrow F(m, k)$
- Hard to compute $m \leftarrow F^{-1}(c)$, without knowing k
- Easy to compute $m \leftarrow F^{-1}(c)$, by knowing k

Public Key Cryptography

Requirements for Public-Key Cryptography

A function f is one-way if

- Easy to compute $c \leftarrow F(m)$
- Hard to compute $m \leftarrow F^{-1}(m)$, without knowing k
- Easy to compute $m \leftarrow F^{-1}(m, k)$, by knowing k

If “one-way function” goes $c \leftarrow F(m)$, then public-key encryption is a “trap-door” function:

Public Key Cryptography

Relationship of One-Way Functions and Cryptography

- Secure encryption and MAC schemes imply/require the existence of one-way functions
- Given a one-way function, one can construct PRG, PRF, PRP
    - Thus one can construct secure encryption and MAC schemes
- One-way functions are foundation of modern cryptography

Public Key Cryptography

Public-Key Encryption Needs One-way Trapdoor Functions

Given a public-key crypto system,

- Alice has public key K
- E_K must be a one-way function, knowing $y = E_K(x)$, it should be difficult to find x
- However, E_K must not be one-way from Alice's perspective. The function E_K must have a trapdoor such that knowledge of the trapdoor enables one to invert it

PKI

RSA Cryptography

RSA

- Invented by Rivest, Shamir, and Adlema in 1978

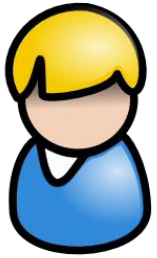
How RSA works

?

- The fundamental idea behind RSA is to try to construct a trap-door or one-way function on a set X.
- This is an invertible function $E : X \rightarrow X$ such that it is easy for Alice to compute E^{-1}, but extremely difficult for anybody else to do so.
- How does Alice makes a one-way function E on the set of integers *modulo* n.

RSA

Alice



1. Pick two large primes p and q, and let
$$n = pq$$
2. Compute
$$\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$$
3. Choose a random integer e with
$$1 < e < \varphi(n) \ and \ gcd(e, \varphi(n)) = 1$$
4. Use the ExEuclid algorithm to find a solution $x = d$ to the equation
$$ex \equiv 1(mod \varphi(n))$$
5. Define a function $E : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ by
$$E(m) = m^e \in \mathbb{Z}/n\mathbb{Z}$$

Note: e is the public key while d is the private key

RSA

Note

- ≈ Anybody can compute E fairly quickly
- ≈ Alice's public key is the pair of integers (n, e), which is just enough information for people to easily compute E.
- ≈ Alice knows a number d such that $ed \equiv 1\ (mod\ \varphi(n))$, so she can quickly compute E^{-1}.
- ≈ To send Alice a message, proceed as follows. Encode your message, in some way, as a sequence of numbers *modulo* n as

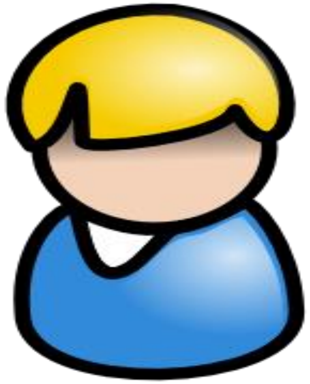
$$m_1, \dots, m_r \in \mathbb{Z}_n$$

- ≈ Then send to Alice,

$$E(m_1), \dots, E(m_r)$$

(Recall that $E(m) = c\ , \forall c \in \mathbb{Z}_n$)

- ≈ When Alice receives $E(m_i)$, she finds each m_i by using the fact that $E^{-1}(c) = m^{de}$

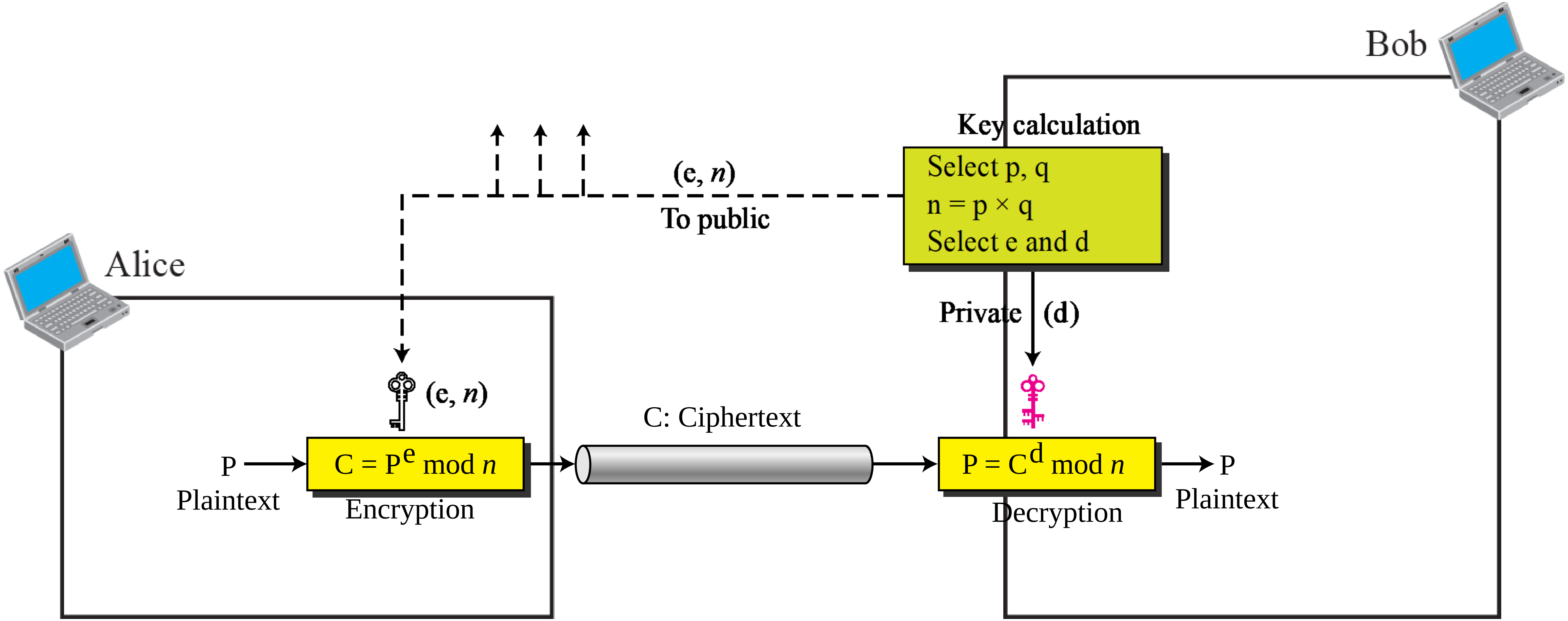


Alice: How correct is the RSA decryption?

Thus to decrypt $E(m_i)$, Alice computes

$$(E(m_i))^d = (m_i)^{de} = m_i$$

Encryption, decryption, and key Generation in RSA



Example 1

- Let Bob choose 7 and 11 as p and q and calculate $n = 7 \times 11 = 77$.
- The value of $\varphi(n) = (7-1)(11-1)$, *or* 60. If he *chooses* e to be 13, then d is 37. Note that $e \times d \bmod 60 = 1$.
- Now imagine that Alice wants to send the plaintext 5 to Bob.
- She uses the public exponent 13 to encrypt 5.
- Note: This system is not safe because p and q are small.

Plaintext: 5

$C = 5^{13} = 26 \bmod 77$

Ciphertext: 26

Ciphertext: 26

$P = 26^{37} = 5 \bmod 77$

Plaintext: 5

Example 2

- Let Bob choose 17 and 11 as p and q and $e = 7$: Compute the rest using RSA
- Encrypt a message M=88

Example 3

Realistic example calculated with a computer.

1. Choose a 512-bit p and q, calculate n and $\varphi(n)$,
2. Choose e and calculate d.
3. Finally, show the results of encryption and decryption.

The integer p is a 159-digit number.

p =	96130345313583504574191581280615427909309845594996215822583150879647940455056470638491257160180347503120986666064924201918087806674210960633542199266612O9

The integer q is a 160-digit number.

q =	1206019195723144691827679420445089600155592505463703393606179832173148214848376465921538945320917522527322683010712069560460251388714552496900035966004561 7

Example 2

The modulus $n = p \times q$. It has 309 digits.

$\boldsymbol{n} =$	115935041739676149688925098646158875237714573754541447754855261376 147885408326350817276878815968325168468849300625485764111250162414 552339182927162507656772727460097082714127730434960500556347274566 628060099924037102991424472292215772798531727033839381334692684137 327622000966676671831831088373420823444370953

$\varphi(n) = (p - 1)(q - 1)$ has 309 digits.

$\phi(n) =$	115935041739676149688925098646158875237714573754541447754855261376 147885408326350817276878815968325168468849300625485764111250162414 552339182927162507656751054233608492916752034482627988117554787657 013923444405716989581728196098226361075467211864612171359107358640 614008885170265377277264467341066243857664128

Example 2

Bob chooses $e = 35535$ (the ideal is 65537). He then finds d.

e =	35535
d =	58008302860037763936093661289677917594669062089650962180422866111380593852822358731706286910030021710859044338402170729869087600611530620252495988444804756824096624708148581713046324064407770483313401085094738529564507193677406119732655742423721761767462077637164207600337085333288532144708859551366702948 31

Alice wants to send the message “THIS IS A TEST”, which can be changed to a numeric value using the 00−26 encoding scheme (26 is the space character).

P =	1907081826081826002619041819

Example 2

The ciphertext calculated by Alice is $C = P^e$, which is

C =	4753091236462268272063655506105451809423717960704917165232392430544529606131993285666178434183591141511974112520056829797945717360361012782188478927415660904800235071907152771859149751884658886321011483541033616578984679683867637337657774656250792805211481418440481418443081277305900469287424855916646 2108656

Bob can recover the plaintext from the ciphertext using $P = C^d$, which is

P =	1907081826081826002619041819

The recovered plaintext is “THIS IS A TEST” after decoding.

Security of RSA

Four possible approaches to attacking the RSA algorithm are

1. Brute force: This involves trying all possible private keys.
2. Mathematical attacks: Equivalent in effort to factoring the product of two primes. Intractable problem of integer factorization
3. Timing attacks: These depend on the running time of the decryption algorithm.
4. Chosen ciphertext attacks: Exploits properties of the RSA algorithm.

Diffie-Hellman Key Exchange

The Diffie-Hellman protocol (1984)

- Fix a finite cyclic group G (e.g $G = (Z_p)^*$) of order n
- Fix a generator g in G (i.e. $G = \{1, g, g^2, g^3, \dots, g^{n-1}\}$)

Alice

choose random **a** in {1,…,n}

$$A = g^a$$

$$\longrightarrow$$

Bob

choose random **b** in {1,…,n}

$$B = g^b$$

$$\longleftarrow$$

$$\boldsymbol{A_k} = \left(g^b\right)^a$$

$$\boldsymbol{B_k} = \left(g^a\right)^b$$

The ElGamal Performance

Example

- For example, let us start with the prime field GF(19); that is, $p = 19 \text{ . } It\ has\ K$ primitive roots {2, 3, 10, 13, 14, 15}. We choose $g\ =?.$

Alice

Bob

ElGamal Encryption

ElGamal: Converting to public key encryption (1977)

- Fix a finite cyclic group G (e.g $G = (Z_p)^*$) of order n
- Fix a generator g in G (i.e. $G = \{1, g, g^2, g^3, \dots, g^{n-1}\}$)

Treated as a public key

Alice

Bob

Choose random **a** in $\{1, \dots, n\}$

Choose random **b** in $\{1, \dots, n\}$

$$A = g^a$$

[Compute $g^{ab} = A^b$,
derive symmetric key k,
encrypt message m with k]

The ElGamal System : A Modern Approach

- G: finite cyclic group of order n
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H: G^2 \longrightarrow K$ a hash function

We construct a pub-key enc. system (Gen, E, D):

- Key generation Gen:
  - Choose random generator g in G and random a in Z_n
  - Output $sk = a\ , pk = (g, h = g^a)$

The ElGamal System : A Modern Approach

$\underline{\mathbf{E(pk = (g, h), m)}}$

$b \leftarrow Z_n$

$u \leftarrow g^b$

$k \leftarrow h^b = g^{ab}$

$v \leftarrow E(k, m) = km \bmod n$

$output \quad (u, v)$

$\underline{\boldsymbol{D\left(sk = a, (u, v)\right)}}$

$k \leftarrow u^a$

$m \leftarrow D_s(k, v) = k^{-1}v$

$output \quad m$

The ElGamal Performance

$\underline{\boldsymbol{E(pk = (g, h), m)}}$:

$b \leftarrow Zn \qquad u \leftarrow g^b$,

$v \leftarrow h^b$,

$\underline{\boldsymbol{D(sk = a, (u, c))}}$:

$v \leftarrow u^a$

Encryption: 2 exp. (fixed basis)

- Can pre-compute $[\, \mathrm{g}^{(2^i)}, \mathrm{h}^{(2^i)} for \quad i = 1, \ldots, \log_2 n \,]$
- 3x speed-up (or more)

Decryption: 1 exp. (variable basis)

The ElGamal Performance

Example

- For example, let us start with the prime field GF(19); that is, $q = 19$. *It has K* primitive roots {2, 3, 10, 13, 14, 15}. We choose $\alpha = 10$.

Alice generates a key pair as follows:

- Alice chooses $x = 5$
- Then $\mathrm{Y}_A = 10^5 mod\ 19 = 3$
- Alice's private key is 5; Alice's pubic key is {19, 10, 3}.

The ElGamal Encryption and Decryption

Encryption

- Suppose Bob wants to send the message with the value *M 17.*

Bob chooses $k = 6.$

- Then K=$(Y_A)^k \; mod \; 19 = 729 \; mod \; 19 = 7$
- So $u = \alpha^k mod \; 19 = 10^6 mod \; 19 = 11$
- v $= KMmod \; 19 = 7 \times 11 \; mod \; 19 = 5$
- Bob sends the ciphertext $(u, v) = (11{,}5).$

Decryption

Alice calculates $K = u^x mod \; 19 = 11^5 mod \; 19 = 7 \;\; to \; recover \; key \; K.$

Then in K^{-1} in GF(19) is $7^{-1} mod \; 19 = -8 = 11$

Finally $M = 11 \times 5 \; mod \; 19 = 17$

The ElGamal Performance

Try with $\underline{\boldsymbol{\alpha = 13, and\ m < 19}}$

The ElGamal Security

- Computational Diffie-Hellman Assumption
- G: finite cyclic group of order n
- Comp. DH (CDH) assumption holds in G if: $g, g^a, g^b \nRightarrow g^{ab}$

For all efficient algorithms A:

$$\Pr\left[A\left(g, g^a, g^b\right) = g^{ab}\right] < \epsilon \; (\mathit{negligible})$$

where $g \longleftarrow \{generators\ of\ G\}, a, b \longleftarrow Zn$

Some Words of Wisdom

V

Be careful when using crypto:

- A tremendous tool, but if incorrectly implemented: system will work, but may be easily attacked

Make sure to have others review your designs and code

Don't invent your own ciphers or modes