Distributed and Edge Computing

Distributed Systems Security

Sharad K. Ghimire

Department of Electronics and Computer Engineering
Pulchowk Campus
Institute of Engineering
Tribhuvan University

Contents

Asymmetric Cryptography

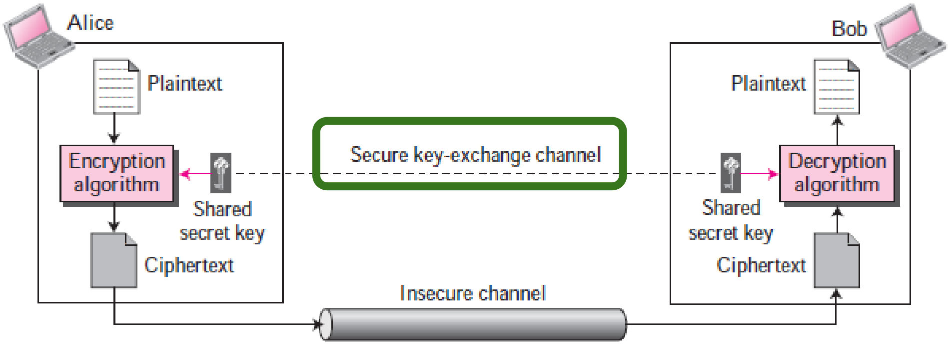
Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	2^{31} μs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} μs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} $\mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} $\mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} $\mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

Table: Average Time Required for Exhaustive Key Search

As AES allows key lengths of 128, 192, and 256 bits, what about time required for exhaustive key search?

Why is symmetric encryption not enough?

Why Asymmetric?



How to share the key securely?

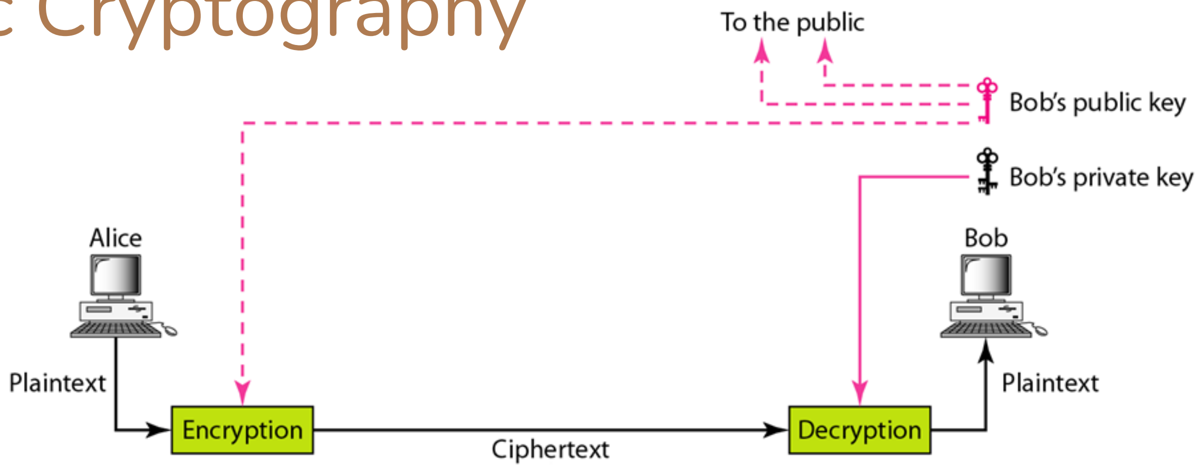
Symmetric encryption is universal technique for providing confidentiality for the transmitted data ⇒ single-key encryption

Symmetric encryption is very strong e.g. AES 256

Asymmetric Cryptography

Public-Key Cryptography

Asymmetric Cryptography



There are two keys: A **private key** and a **public key** ⇒ **Public Key Cryptography**

Private key is kept by the receiver and Public key is announced to the public

Asymmetric Cryptography

In the setting of private-key encryption, two parties agree on a secret key **k** which can be used (by either party) for both encryption and decryption

Public-key encryption is asymmetric in both these respects ⇒ specifically, one party (the receiver) generates a pair of keys called the **public key** and the **private key**

The public key is used by a sender to encrypt a message for the receiver; the receiver then uses the private key to decrypt the resulting ciphertext

The goal is to avoid the need for two parties to meet in advance to agree on key

Asymmetric Cryptography

Public-key, or asymmetric cryptography is one of the **greatest** revolution in the **history of cryptography**

Virtually all cryptographic systems have been based on the elementary tools of **substitution** and **permutation**

With the availability of computers, even more complex systems were devised, the most prominent of which was the Lucifer at IBM ⇒ Data Encryption Standard (**DES**) ⇒ still based on the basic tools of **substitution** and **permutation**

Asymmetric Cryptography

Public-key algorithms are based on **mathematical functions** rather than on **substitution** and **permutation**

Public-key cryptography is asymmetric, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key

The use of two keys has profound consequences in the areas of **confidentiality**, **key distribution**, **authentication** as well as **non-repudiation**

Symmetric & Asymmetric Cryptography

There are some common misconceptions concerning public-key encryption:

The public-key encryption is more secure from cryptanalysis than is symmetric encryption

- The security of any encryption scheme depends on the length of the key and the computational work involved in breaking a cipher
- There is nothing in principle about either symmetric or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis

Symmetric & Asymmetric Cryptography

The public-key encryption is a general-purpose technique that has made symmetric encryption obsolete:

- Due to computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that symmetric encryption will be abandoned
- The use of public-key cryptography is in key management and signature applications is almost universally accepted

Need of Both (Sym & Asym)
Either one is not eliminating another

The asymmetric key (public-key) cryptography does not eliminate the need for symmetric-key (secretkey) cryptography

The asymmetric-key cryptography, which uses mathematical functions for encryption and decryption, is much slower than symmetric-key cryptography, so for encipherment of large messages ⇒ symmetric-key cryptography is still needed

Need of Both (Sym & Asym) Either one is not eliminating another

On the other hand, the speed of symmetric-key cryptography does not eliminate the need for asymmetric-key cryptography

Asymmetric-key cryptography is still needed for authentication, digital signatures, and secret-key exchanges ⇒ to be able to use all aspects of security today, we need both symmetric-key and asymmetric-key cryptography ⇒ one complements the other

Both cryptography will exist in parallel and continue to serve the community

Need of Asymmetric Cryptography

Confidentiality: Only Bob can read Alice's message without sharing secret ⇒ used to share key of symmetric encryption for large message

Authenticity: Alice can digitally "sign" her message, so Bob knows that only Alice could have sent it ⇒ he also knows Tom couldn't have tampered with the message in transit

Non-repudiation: Alice can't deny she sent (or at least saw) the message contents later on

Public-Key Encryption Operation

Public Key Encryption - Operation

Public key cryptography ⇒ two keys: a **private key** and a **public key**

The public key is announced to the public, where as the private key is kept by the receiver because the private key should kept secret

The sender uses the public key of the receiver for encryption and the receiver uses his private key for the decryption

These algorithms have the important characteristic, i.e., it is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key

Public Key Encryption - Operation

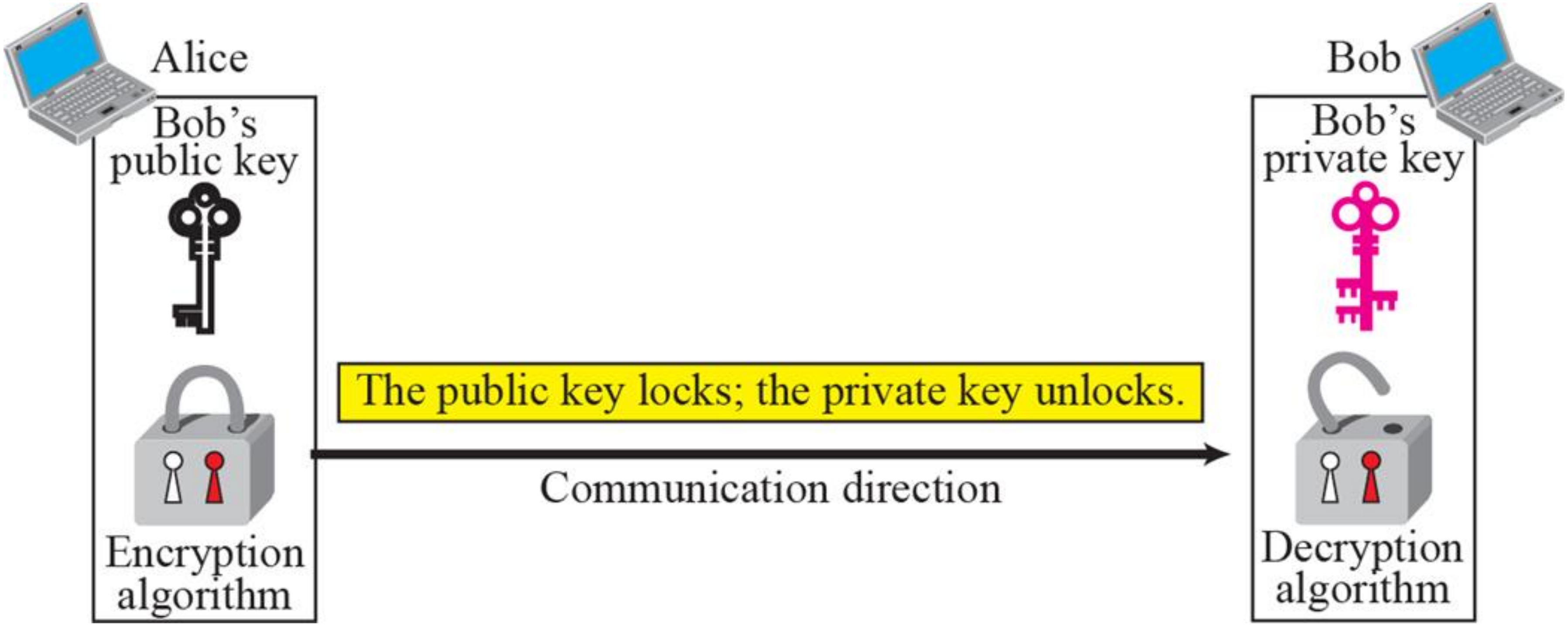
Public key is used for encryption

Private key is used for decryption

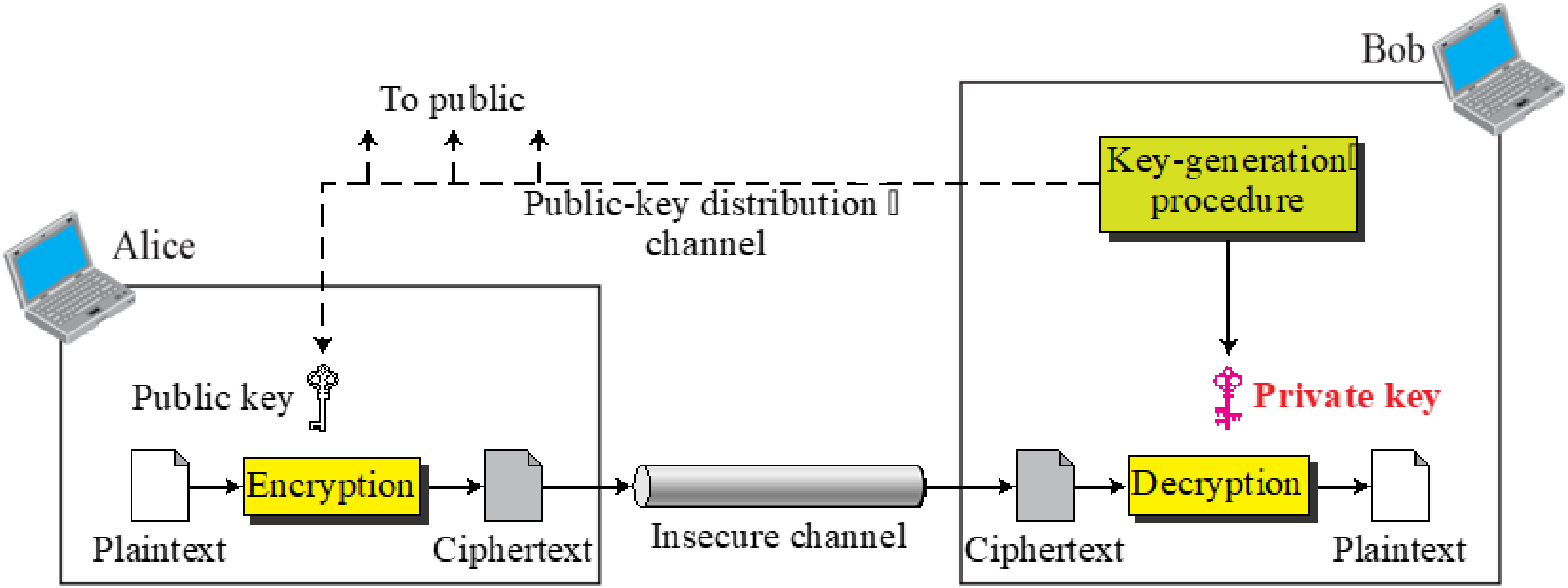
Infeasible to determine decryption key given encryption key and algorithm

Steps:

- User generates pair of keys
- User places one key in public domain
- To send a message to user, encrypt using public key
- User decrypts using private key



Locking and unlocking in asymmetric-key cryptosystem



General idea of asymmetric-key cryptosystem

Diffie-Hellman Algorithm

Diffie-Hellman Algorithm

Originally designed for key exchange

Two parties create a symmetric session key to exchange data without having to remember or store key for further use

No need to meet to agree on the key

Common key exchange can be done through public channel such as Internet

The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages

The algorithm itself is limited to the exchange of secret values

Diffie-Hellman Algorithm - Steps:

Alice chooses a large random number **x** and calculates $\mathbf{R_1 = g^x \bmod p}$

Bob chooses another large random number **y** and calculates $\mathbf{R_2 = g^y \bmod p}$

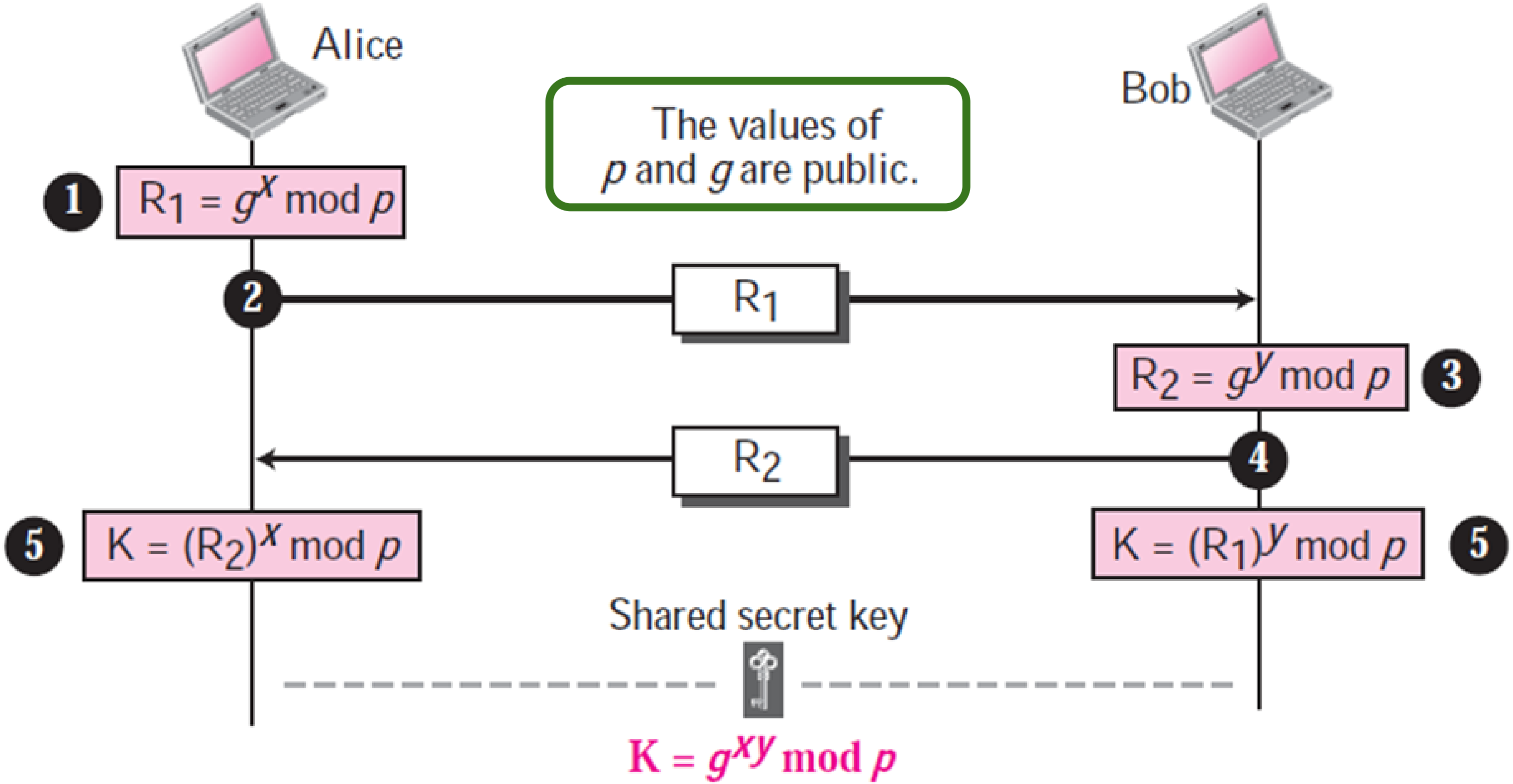
Alice sends $\mathbf{R_1}$ to Bob (but Alice does not send the value of x)

Bob sends $\mathbf{R_2}$ to Alice (but Bob does not send the value of y)

Alice calculates $\mathbf{K = (R_2)^x \bmod p}$

Bob also calculates $\mathbf{K = (R_1)^y \bmod p}$

The symmetric key for the session is $\mathbf{K = g^{xy} \bmod p}$



Diffie-Hellman Key Exchange, Shared Key is: $K = g^{xy} \bmod p$

Diffie-Hellman Method Example

Assume ***g = 7*** *and* ***p = 23****. The steps are as follows:*

Alice chooses $x = 3$ *and calculates* $R_1 = 7^3 \bmod 23 = 21$

Bob chooses $y = 6$ *and calculates* $R_2 = 7^6 \bmod 23 = 4$

Alice sends the number 21 to Bob

Bob sends the number 4 to Alice

Alice calculates the symmetric key $K = 4^3 \bmod 23 = 18$

Bob calculates the symmetric key $K = 21^6 \bmod 23 = 18$

The value of K is the same for both Alice and Bob

$$g^{xy} \bmod p = 7^{18} \bmod 23 = \mathbf{18}$$

Diffie-Hellman Key Exchange

Two parties got a common secret key, without passing common secret across the public channel

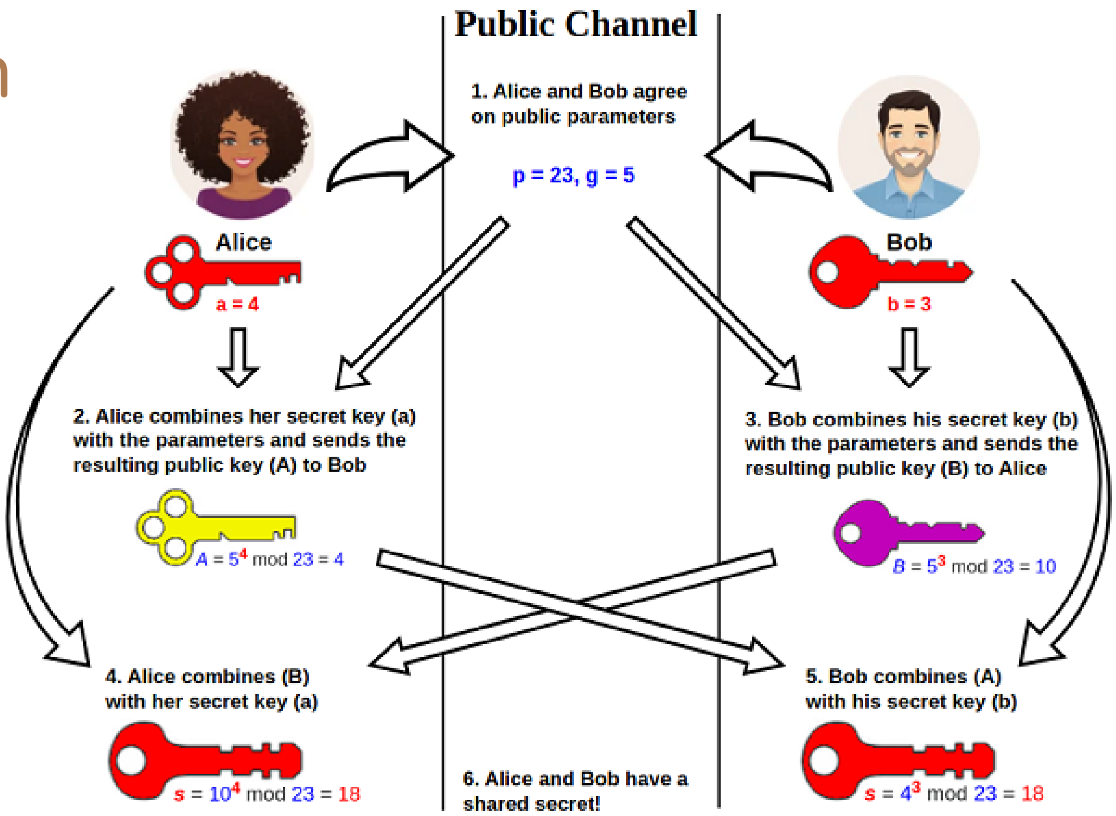
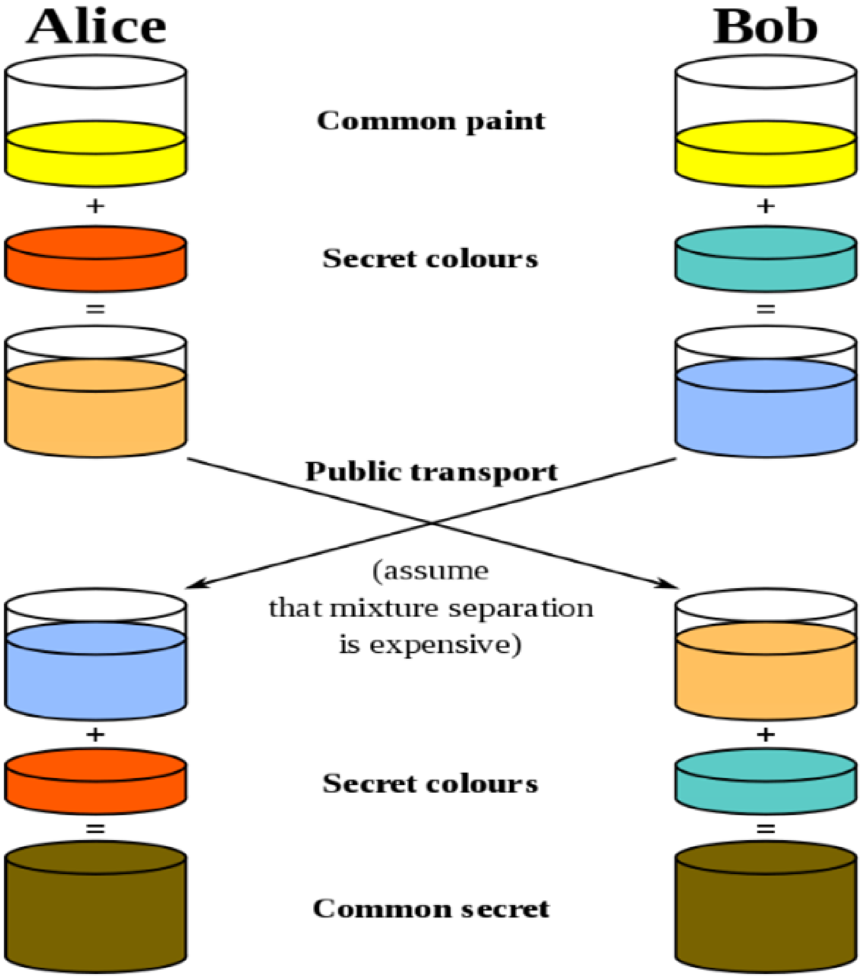


Illustration of Diffie-Hellman key exchange



Realistic Example

Let us create a random integer of 512 bits (the ideal is 1024 bits). The integer p is a 159-digit number. We also choose g, x, and y as:

p	764624298563493572182493765955030507476338096726949748923573772860925 235666660755423637423309661180033338106194730130950414738700999178043 6548785807987581
g	2
x	557
y	273

R_1	844920284205665505216172947491035094143433698520012660862863631067673 619959280828586700802131859290945140217500319973312945836083821943065 966020157955354
R_2	435262838709200379470747114895581627636389116262115557975123379218566 310011435718208390040181876486841753831165342691630263421106721508589 625520128859414 3
K	155638000664522290596225827523270765273218046944423678520320400146406 500887936651204257426776608327911017153038674561252221315161097658420 0 120408643361774 0

Showing the values of R1, R2, and K

Analysis of Diffie-Hellman

The secret key between Alice and Bob as made of three parts: g, x, and y

The first part is public; everyone knows 1/3 of the key; g is a public value

The other two parts must be added by Alice and Bob; each of them add one part ⇒ Alice adds x as the second part for Bob; Bob adds y as the second part for Alice

When Alice receives the 2/3 completed key from Bob, she adds the last part, her x, to complete the key

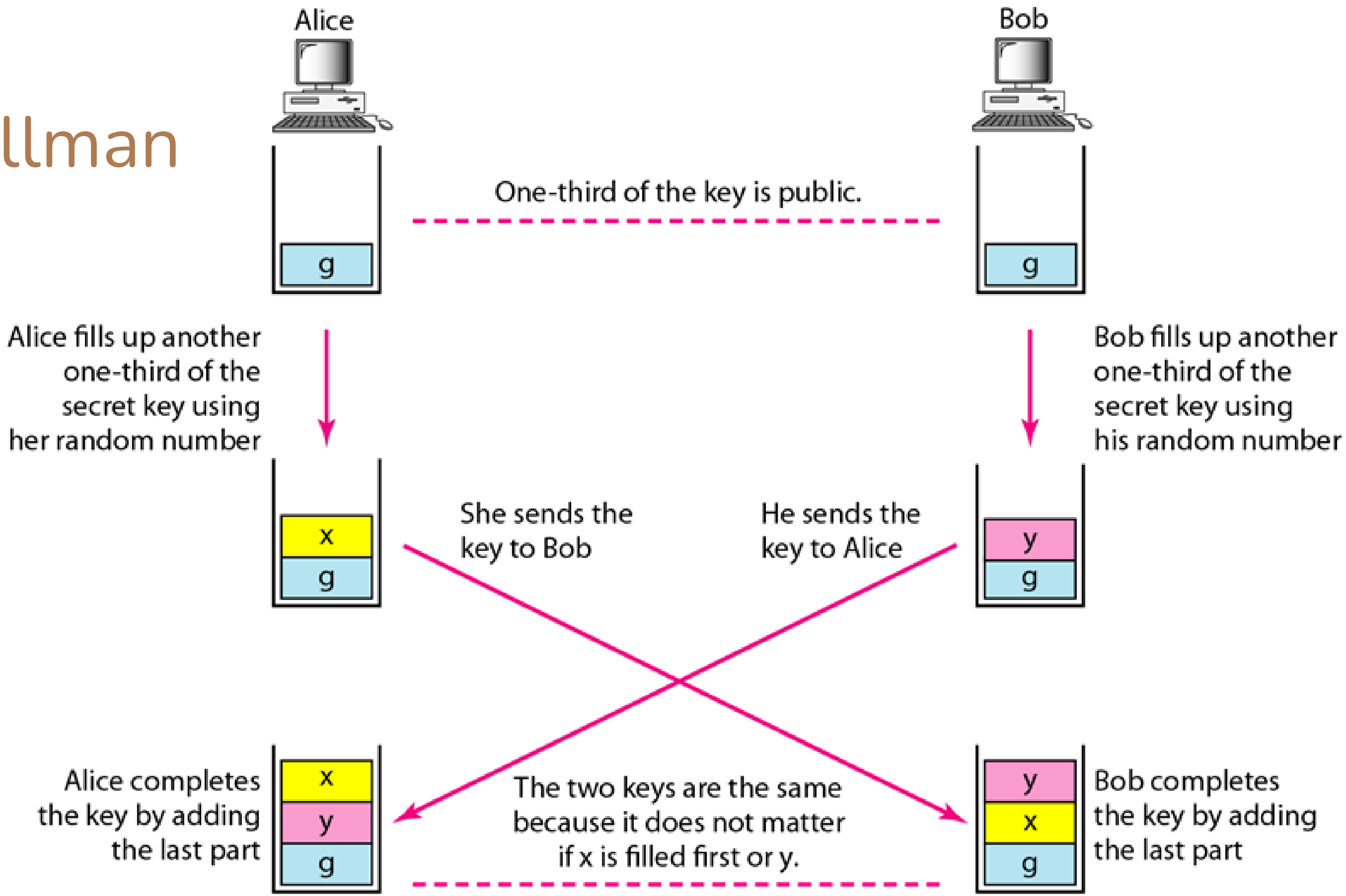
Analysis of Diffie-Hellman

When Bob receives the 2/3-completed key from Alice, he adds the last part, his y, to complete the key

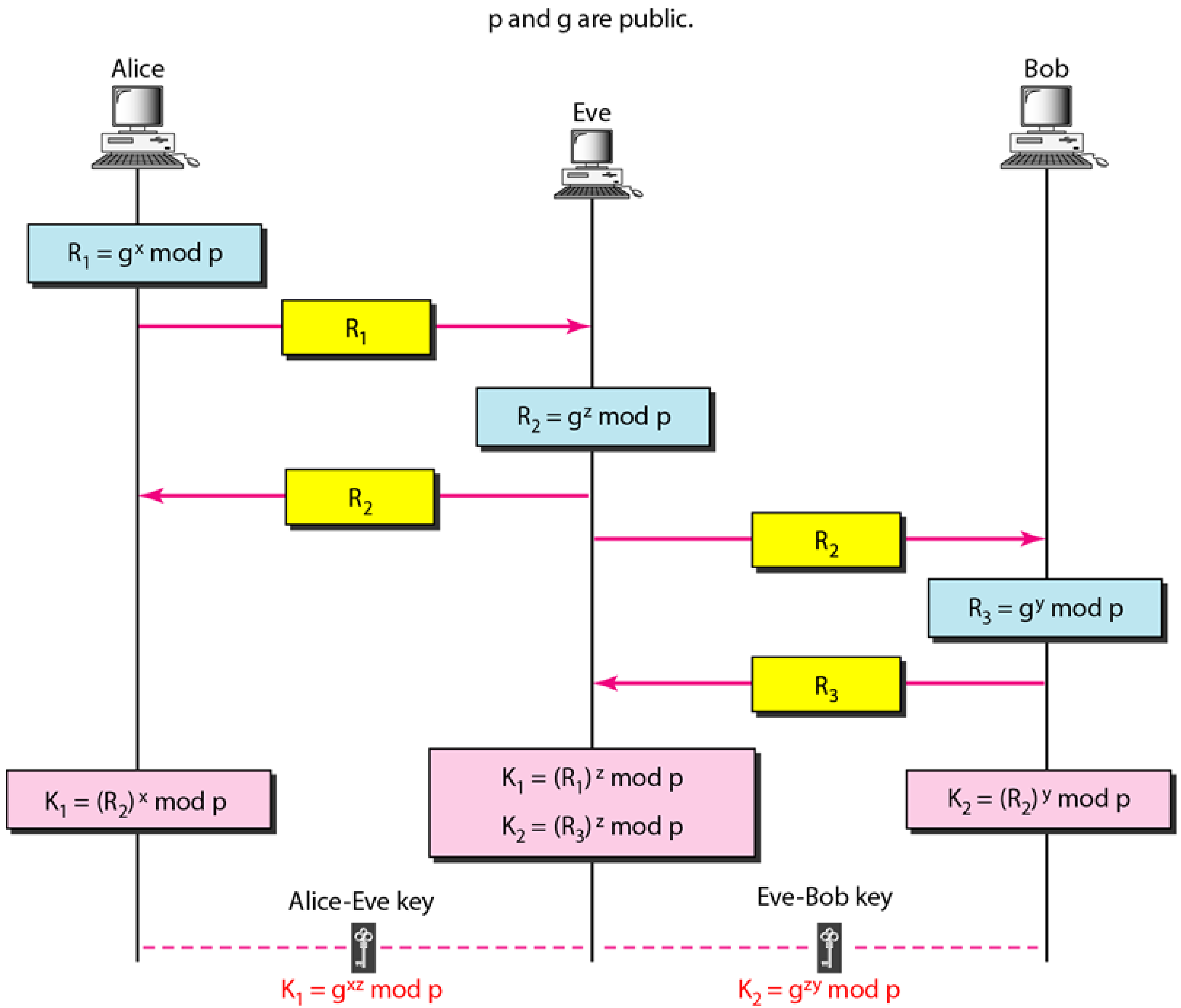
Here the key in Alice's hand consists of g, y, and x and the key in Bob's hand also consists of g, x, and y, these two keys are the same because $\mathbf{g^{xy} = g^{yx}}$

Although the two keys are the same, Alice cannot find the value y used by Bob because the calculation is done in modulo p; Alice receives $\mathbf{g^{y} \bmod p}$ from Bob, not $\mathbf{g^{y}}$

Diffie-Hellman Idea



Man-in-the-Middle Attack



RSA (Rivest-Shamir-Adleman) Algorithm

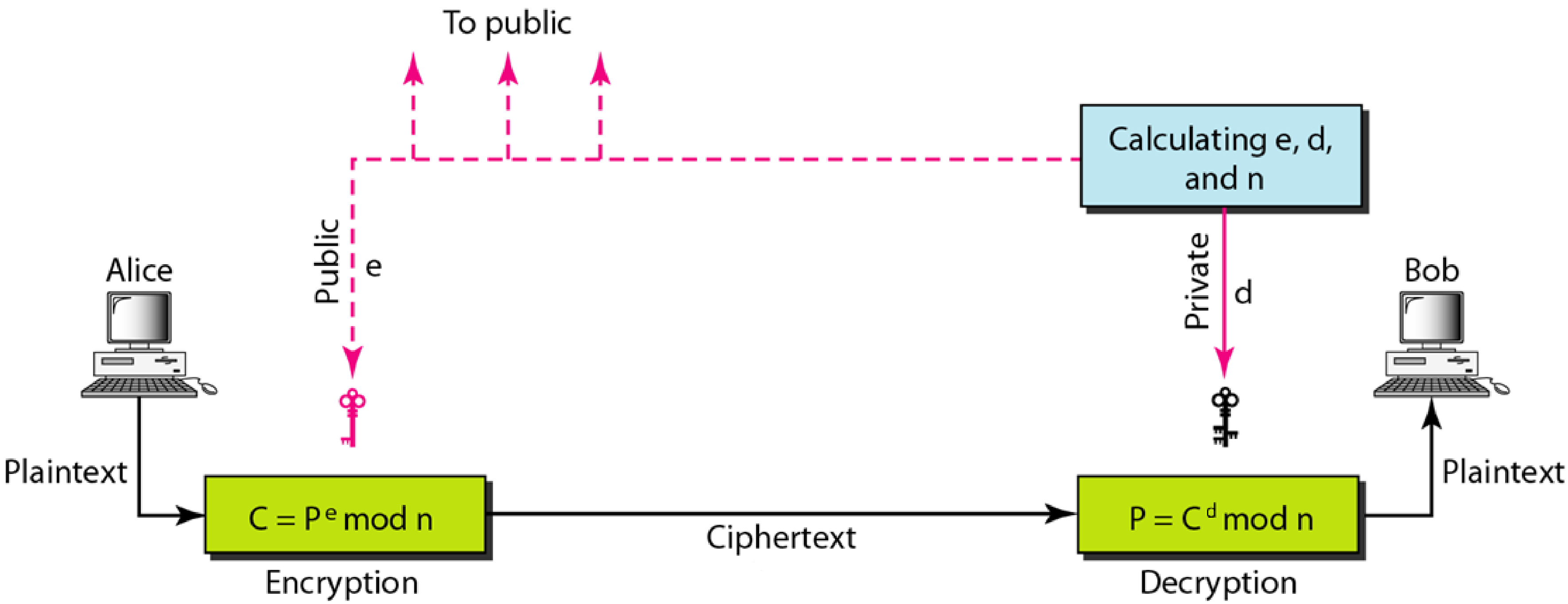
RSA (Rivest-Shamir-Adleman) Algorithm

The most common public-key algorithm used by modern computers to encrypt and decrypt messages

An asymmetric cryptographic algorithm also know as public key cryptographic algorithm

Named from its inventors **Rivest**, **Shamir**, and **Adleman**, who publicly described the algorithm in 1977

RSA Algorithm



RSA Algorithm

Public Key: e, n
Private Key: d, n

Choose two different large random **prime numbers p** and **q**

Calculate **$n = pq$** ⇒ n is the modulus for the public key and the private keys

Calculate the **$\phi(n) = (p-1)(q-1)$**

Choose an integer e, such that **$1 < e < \phi(n)$**, such that e and $\phi(n)$ share no factor other than 1, i.e. **$\gcd(e, \phi(n)) = 1$** ⇒ **e is announced as public key** exponent

Compute d such that **$de \bmod \phi(n) = 1$** ⇒ **d is kept as the private key** exponent

In RSA (**e, n**) is public; and the integer **d** is private

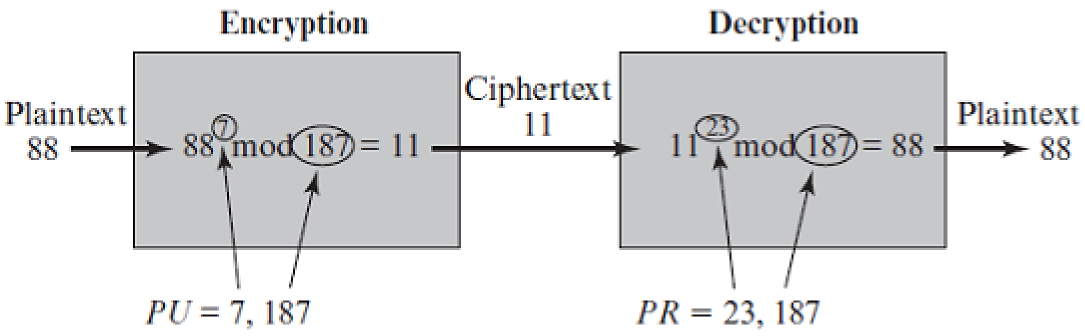
RSA Encryption and Decryption

Encryption

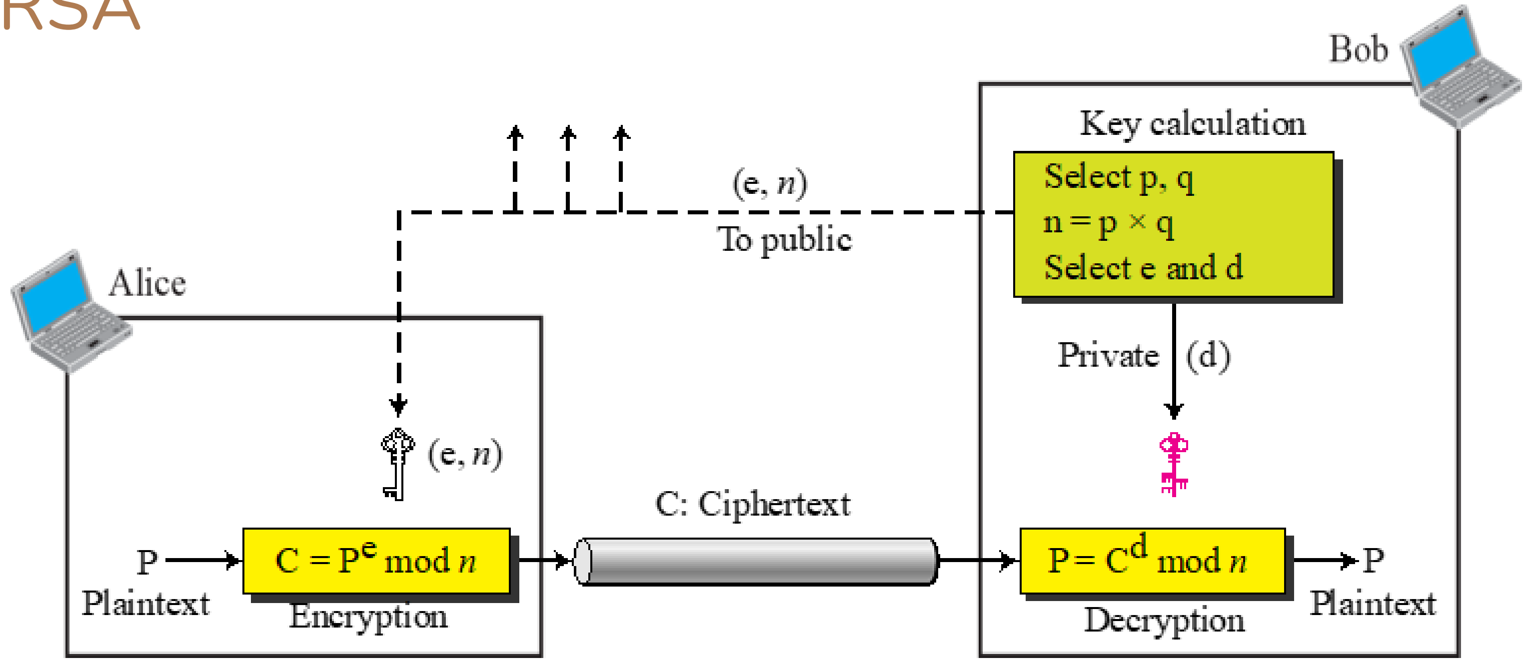
$C = P^e \bmod n$

Decryption

$P = C^d \bmod n$



Encryption, decryption, and key Generation in RSA



RSA Example
Key Generation

Now, the public key (e, n) = (5, 35)

And the private key (d, n) = (29, 35)

Choose two prime numbers, p and q ⇒ let p = 5 and q = 7

Compute $n = p * q \Rightarrow n = 5 * 7 = 35$

Calculate the $\phi(n) = (p - 1) * (q - 1) \Rightarrow \phi(35) = (5-1) * (7-1) = 24$

Choose an integer e, such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1 \Rightarrow$ let e = 5

Compute the modular multiplicative inverse of e, which is $d = e^{-1} \pmod{\phi(n)}$

$\Rightarrow d = 29$ (since $5 * 29 = 1 \pmod{24}$)

RSA Example: Encryption and Decryption

Let us represent the letters a to z by numbers from 01 to 26

As plaintext message is **j**, the corresponding numerical plaintext is P = 10 and with encryption key e = 5

$C = P^e \pmod{n}$

$C = 10^5 \pmod{35} =$ **22** (cipher text)

Receiver uses the decryption key d = 29

The ciphertext message C = 22 is decrypted as:

$M = C^d \pmod{n}$

$M = 22^{29} \pmod{35} = 10$

While decoding 10 ⇒ **j** same with that of plaintext (P)

RSA Example Key Generation

Let, two random prime numbers **p = 5** and **q = 11**

Now, **n** = 5 × 11 = **55**

Then **ϕ** = (p-1) × (q-1) = 4 × 10 = **40**

Let us choose a prime value **e = 7**

Since, the decryption key d must be the multiplicative inverse of e modulo ϕ

e × d mod ϕ = 1, As the value 23 satisfies the requirement, so **d = 23**

Exercise

RSA Keys

Public: **e, n ⇒ 7, 55**

Private: **d, n ⇒ 23, 55**

Encrypt the message "CRYPTO" using RSA

Also decrypt the encrypted message to recover the plaintext

Encoding message in numeric values (using 1 to 26 for A to Z):

C	⇒	3
R	⇒	18
Y	⇒	25
P	⇒	16
T	⇒	20
O	⇒	15

Encryption

$C = P^e \pmod{n}$ with **e, n ⇒ 7, 55**

$3^7 \bmod 55 = 42$

$18^7 \bmod 55 = 17$

$25^7 \bmod 55 = 20$

$16^7 \bmod 55 = 36$

$20^7 \bmod 55 = 15$

$15^7 \bmod 55 = 5$

$M = C^d \pmod{n}$ with **d, n ⇒ 23, 55**

$42^{23} \bmod 55 = 3$

$17^{23} \bmod 55 = 18$

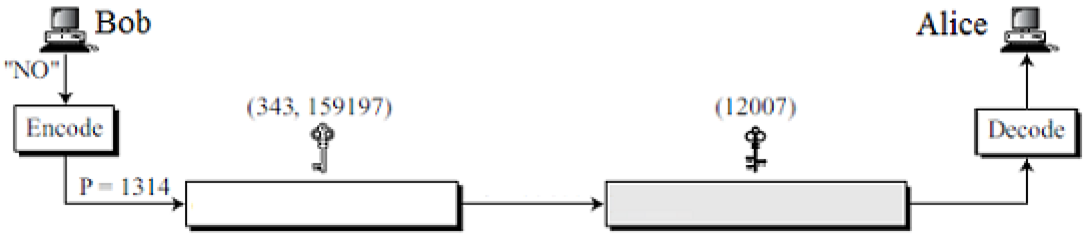
$20^{23} \bmod 55 = 25$

$36^{23} \bmod 55 = 16$

$15^{23} \bmod 55 = 20$

$5^{23} \bmod 55 = 15$

More Practical Example



Alice creates a pair of keys by choosing **p** = **397** and **q** = **401**

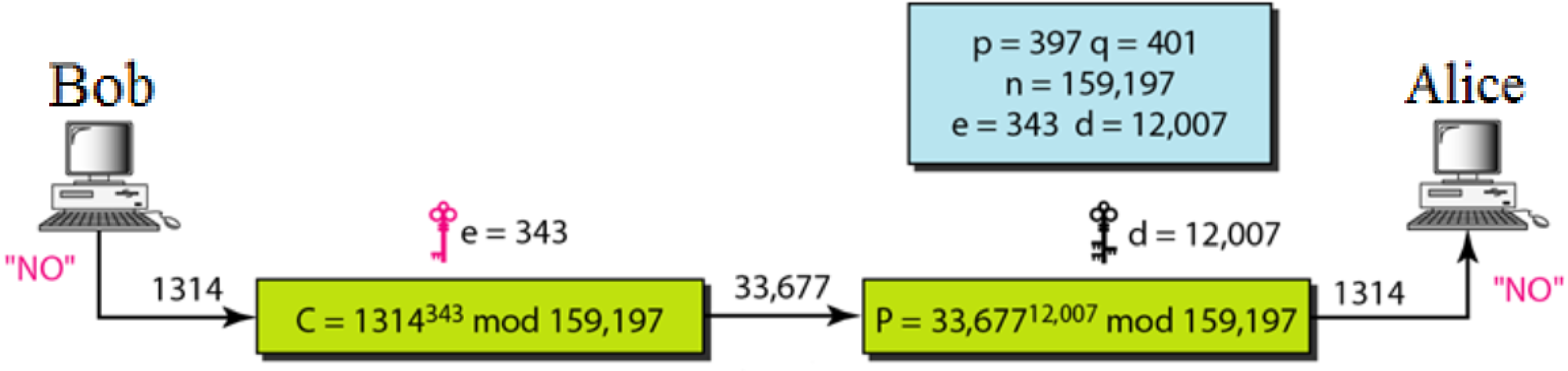
Then **n** = **159,197** and **ϕ(n)** = **396 · 400** = **158,400**

She then chooses **e** = **343** and **d** = **12,007**

Now Bob wants to send a message "**NO**" to Alice by knowing e and n

Bob changes each character to a number (from 00 to 25) with each character coded as two digits

He then concatenates the two coded characters and gets a four-digit number; plaintext as **1314**



Bob then uses e and n to encrypt the message

The ciphertext becomes 1314343 = 33,677 mod 159,197

Alice receives the message 33,677 and uses the decryption key d to decipher it as 33,67712007 = 1314 mod 159,197

Alice then decodes 1314 as the message “NO”

A Realistic Example of RSA

A Realistic Example of RSA

Randomly chose an integer of 512 bits

p = 96130345313583504574191581280615427909309845594996215822583150879647940
45505647063849125716018034750312098666606492420191808780667421096063354
219926661209

The integer q is a 160-digit number

q = 12060191957231446918276794204450896001555925054637033936061798321731482
14848376465921538945320917522527322683010712069560460251388714552496900
0359660045617

$$
\begin{aligned}
\mathbf{n} = {} & 1159350417396761496889250986461588752377145737545414477548552613761478 8\\
& 540832635081727687881596832516846884930062548576411125016241455233918 29\\
& 271625076567727274600970827141277304349605005563472745666280600999240 37\\
& 102991424472292215772798531727033839381334692684137327622000966676671 83\\
& 1831088373420823444370953
\end{aligned}
$$

While calculating n, it becomes 309 digits

Similarly ϕ becomes 309 digits as

$$
\begin{aligned}
\phi = {} & 1159350417396761496889250986461588752377145737545414477548552613761478 8\\
& 540832635081727687881596832516846884930062548576411125016241455233918 29\\
& 271625076567510542336084929167520344826279881175547876570139234444057 16\\
& 989581728196098226361075467211864612171359107358640614008885170265377 27\\
& 72644673410662438576641 28
\end{aligned}
$$

$\mathbf{e} = 35535$

$$\mathbf{d} = \begin{array}{l}580083028600377639360936612896779175946690620896509621804228661113805938 52 \\ 8223587317062869100300217108590443384021707298690876006115306202524959884 4 \\ 4804756824096624708148581713046324064407770483313401085094738529564507193 6 \\ 7740611973265574242372176176746207763716420760033708533328853214470885955 1 \\ 36670294831\end{array}$$

By choosing e = 35,535 the value of d is computed

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

$$\mathbf{P} = 1907081826081826002619041819$$

$$
\begin{aligned}
\mathbf{C} = {} & 4753091236462268272063655506105451809423717960704917165232392430544529 \\
& 6061319932856661784341835911415119741125200568297979457173603610127821 \\
& 8847892741566090480023507190715277185914975188465888632101148354103361 \\
& 6578984679683867637337657774656250792805211481418440481418443081277305 \\
& 9004692874248559166462108656
\end{aligned}
$$

$$\mathbf{P} = 1907081826081826002619041819$$

Asymmetric vs. Symmetric Cryptography

Key Length for Encryption

112 Bits of 3DES	----------------	2048 Bits of RSA
128 Bits of AES-128	----------------	3072 Bits of RSA
192 Bits of AES-192	----------------	7680 Bits of RSA

Need of Both

Asymmetric key cryptography does not eliminate the need for **symmetric-key** cryptography; Both are essential because one complements the other

Asymmetric-key cryptography, is much slower than symmetric key cryptography

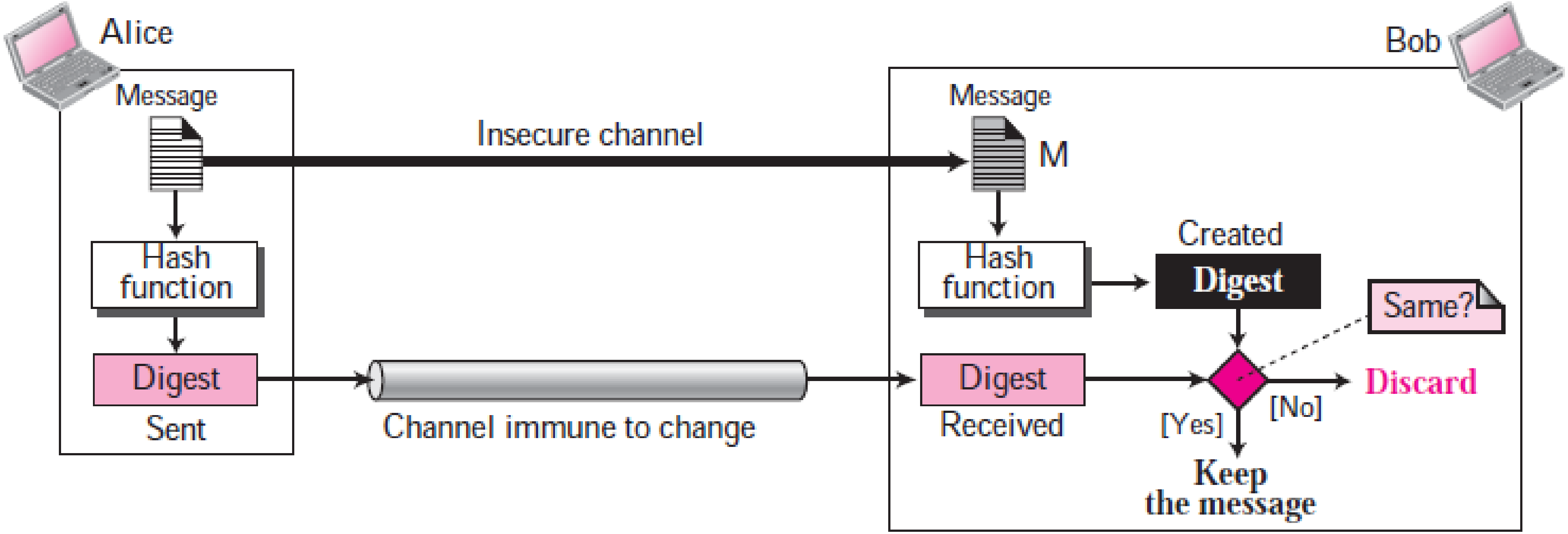
For encipherment of large messages, symmetric-key cryptography is still needed

On the other hand, the speed of symmetric-key cryptography does not eliminate the need for asymmetric-key cryptography

Asymmetric-key cryptography is still needed for authentication, digital signatures, and secret-key exchanges

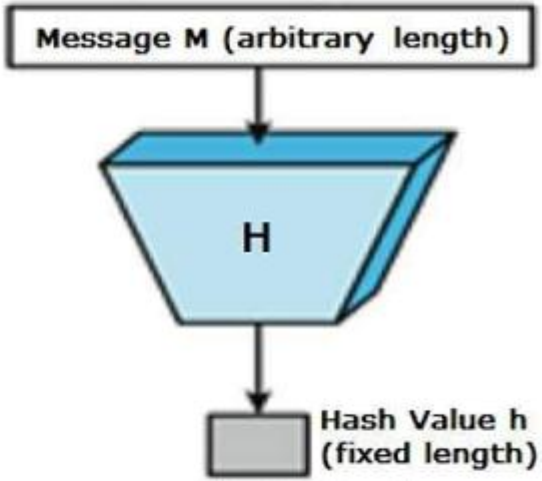
Message Integrity

Message and Message Digest



The message digest needs to be safe from change.

Hash Functions



Take a message of arbitrary length

Creates a message digest of fixed length

Message size can be varied but the digest output is of fixed size

Several hash algorithms were designed by Ron Rivest: MD2, MD4, MD5

Secure Hash Algorithm (SHA) is a standard developed by the NIST

Different versions of SHA: SHA0, SHA1, SHA2, SHA3

Hash Function Properties

Can be applicable for any sized message M

Produces fixed-length output h

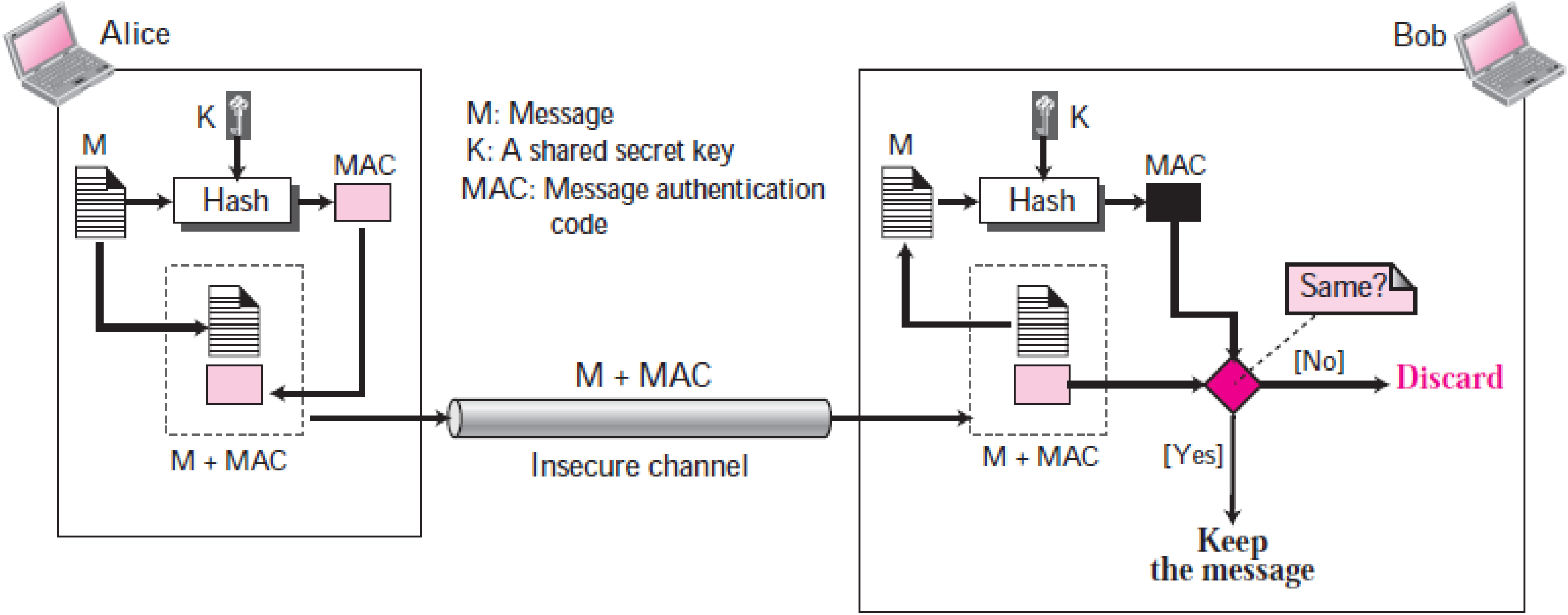
Easy to compute h = H(M) for any message M

One-way property:- Given h, it is infeasible to find x s.t. H(x) = h

Collision resistance:- Given x, it is infeasible to find a y s.t. H(y) = H(x)

Similarly, It is infeasible to find any x,y s.t. H(y) = H(x)

Message Authentication Code (MAC)



Requirements for MAC

Given a message and a MAC, it should be infeasible to find another message with same MAC

MAC should depend equally on all bits of the message

MAC Properties

A MAC is a cryptographic checksum

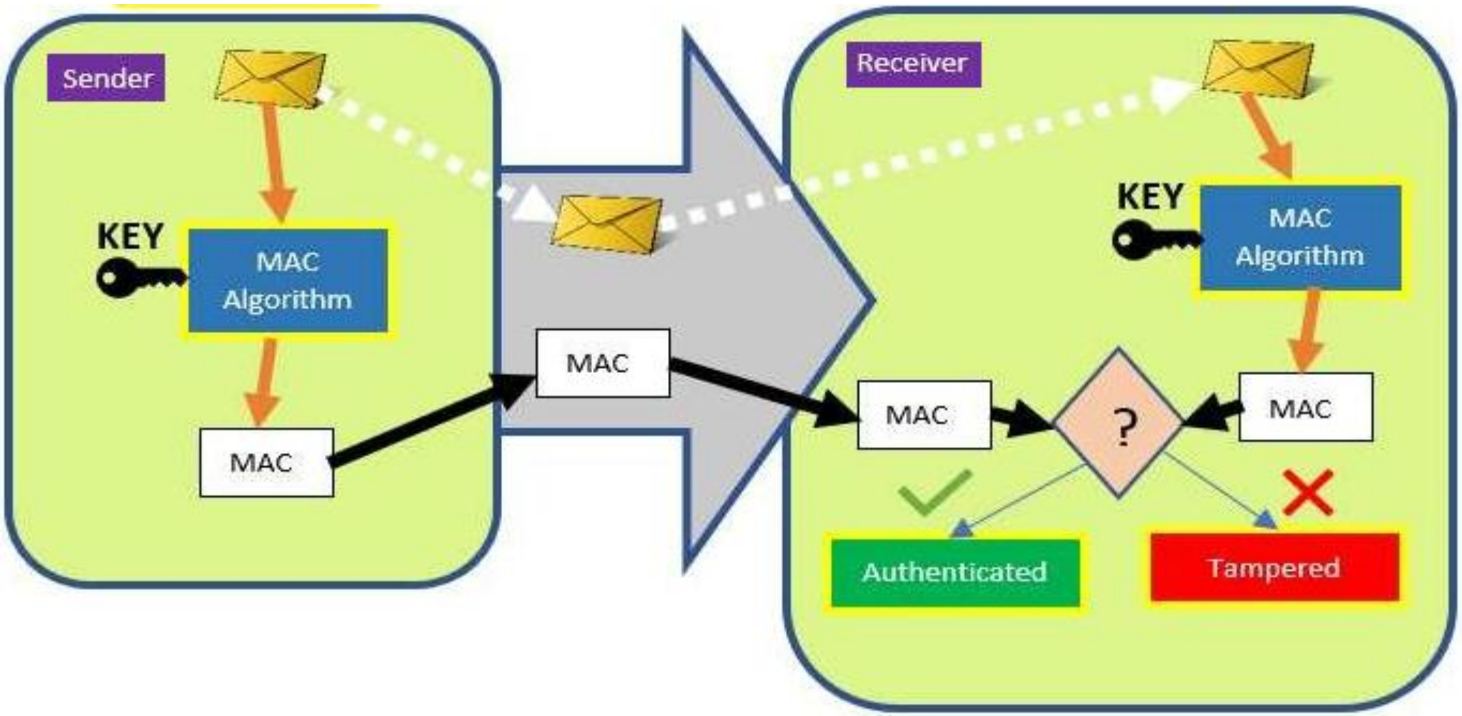
- $MAC = C_K(M)$

Can accept variable-length message M

Using a secret key K to a fixed-sized authenticator

Potentially many messages may have same MAC but finding these needs to be very difficult

Message Authentication using MAC



Digital Signatures

Message authentication

Data Integrity

Non-repudiation

Digital Signature

Required Conditions:

- Receiver can verify claimed identity of sender
- Sender cannot later repudiate contents of message

Digital Signature

A standard element of most cryptographic protocol suites, and are commonly used for software distribution, financial transactions and in other cases where it is important to detect forgery or tampering

Often used to implement electronic signatures, which include any electronic data that carries the intent of a signature

Employ asymmetric cryptography

Public-Key Signatures

Sender encrypts message with private key

Receiver decrypts with sender's public key

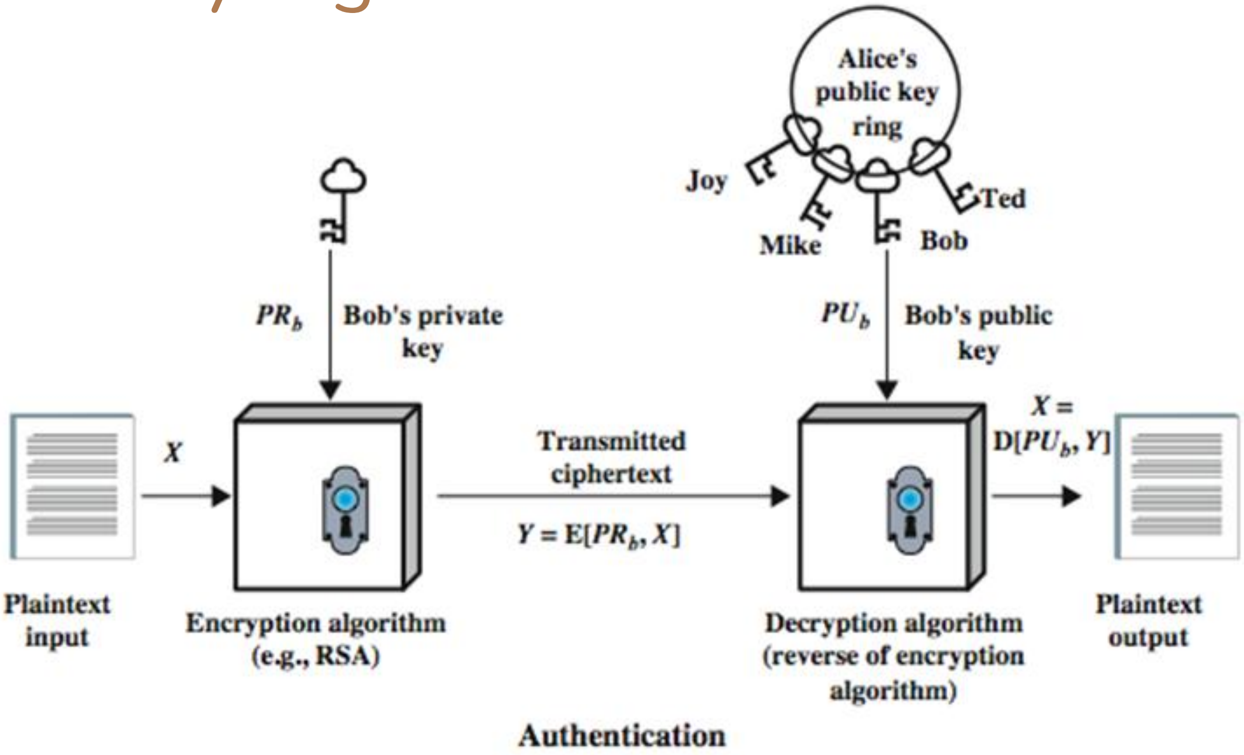
Authenticates sender

Does not give privacy of data

More efficient to sign authenticator

- A secure hash of message
- Send signed hash with message

Public-Key Signatures



Digital Signature