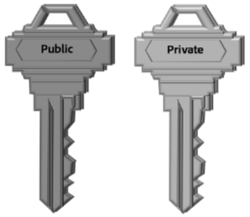
BAHRIA UNIVERSITY
Discovering Knowledge

Public-Key Cryptography and RSA

Data Encryption & Security (CEN-451)

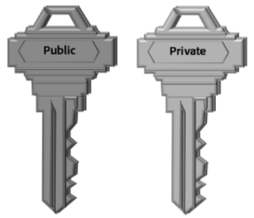
Spring 2025 (BSE-8A&B)

Overview



- Public-key cryptography is **asymmetric**, i.e. involving use of ***two separate keys***.
- There is nothing about **symmetric** or **asymmetric** encryption that makes one superior to another w.r.t. ***cryptanalysis***.
- Computational overhead of public-key encryption exists w.r.t. **key management** and **signature applications**.
- Some form of protocol is needed for key distribution, generally involving a **central agent**.

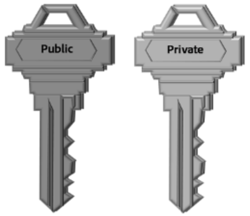
Terminologies



- **Asymmetric Keys:** two "related" keys, a **public** and **private key**, used to perform **complementary operations**, such as encryption and decryption or signature generation and verification.

- **Public Key Infrastructure (PKI):** set of policies, processes, server platforms, software and workstations used for administering **certificates** and **public-private** key pairs.

Public-Key Cryptosystems



- The concept of **public-key cryptography** evolved from an attempt to solve two of the most difficult problems in symmetric encryption:

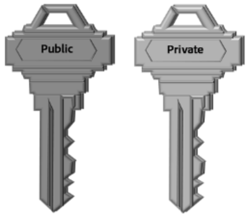
Key distribution

- How to have secure communications in general without having to trust a Key Distribution Center with your key

Digital signatures

- How to verify that a message comes from the claimed sender

Public-Key Cryptosystems (Cont.)

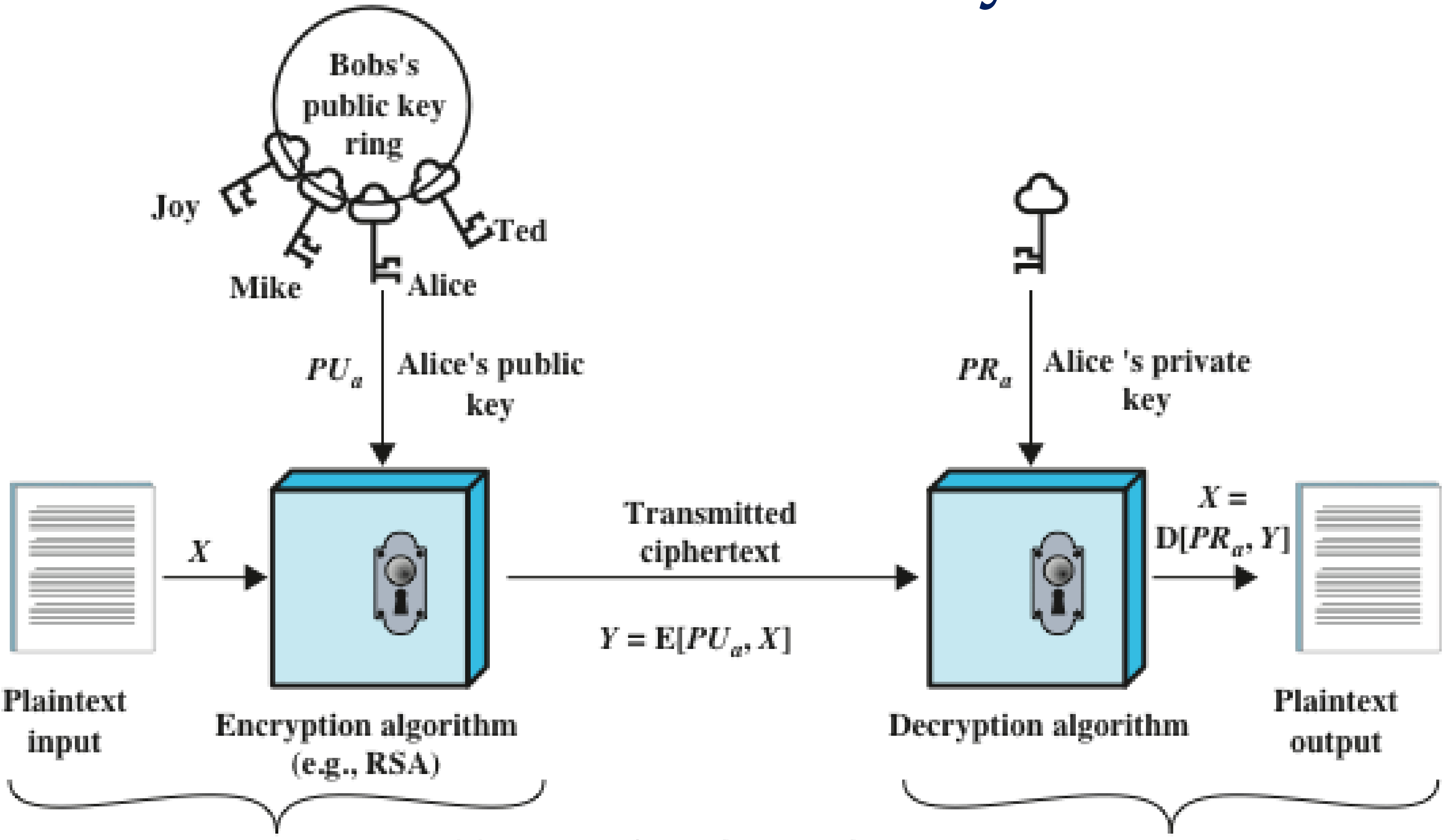


- Asymmetric algorithms rely on one key for encryption and a ***different but related*** key for decryption.
- These algorithms have the following general characteristic:

a. It is computationally infeasible to determine decryption key given only cryptographic algorithm and encryption key.

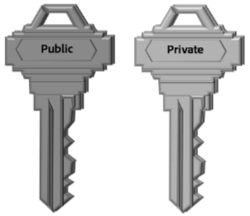
b. Either of the two related keys can be used for encryption, with the other used for decryption (as is the case in **RSA**).

Public-Key Cryptosystems: Confidentiality



(a) Encryption with public key

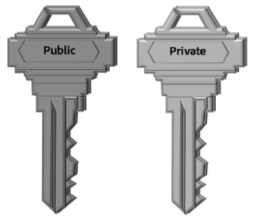
Public-Key Cryptosystems: Confidentiality (Cont.)



Example:

- If **Bob** wishes to send a confidential message to **Alice**, **Bob** encrypts the message using **Alice’s** public key.
- When **Alice** receives the message, she decrypts it using her private key, where no other recipient can decrypt the message because only **Alice** knows **Alice’s** private key.

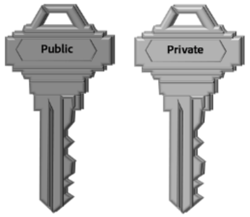
Public-Key Cryptosystems: Confidentiality (Cont.)



Mechanism in public-key cryptosystem:

- Each user generates a **pair of keys** to be used for encryption and decryption of messages.
- Each user places one of the two keys in a **public register** ***(this is the public key)***, while the companion key is kept private.
- All participants have access to public keys; hence each user has a collection of public keys obtained from others.

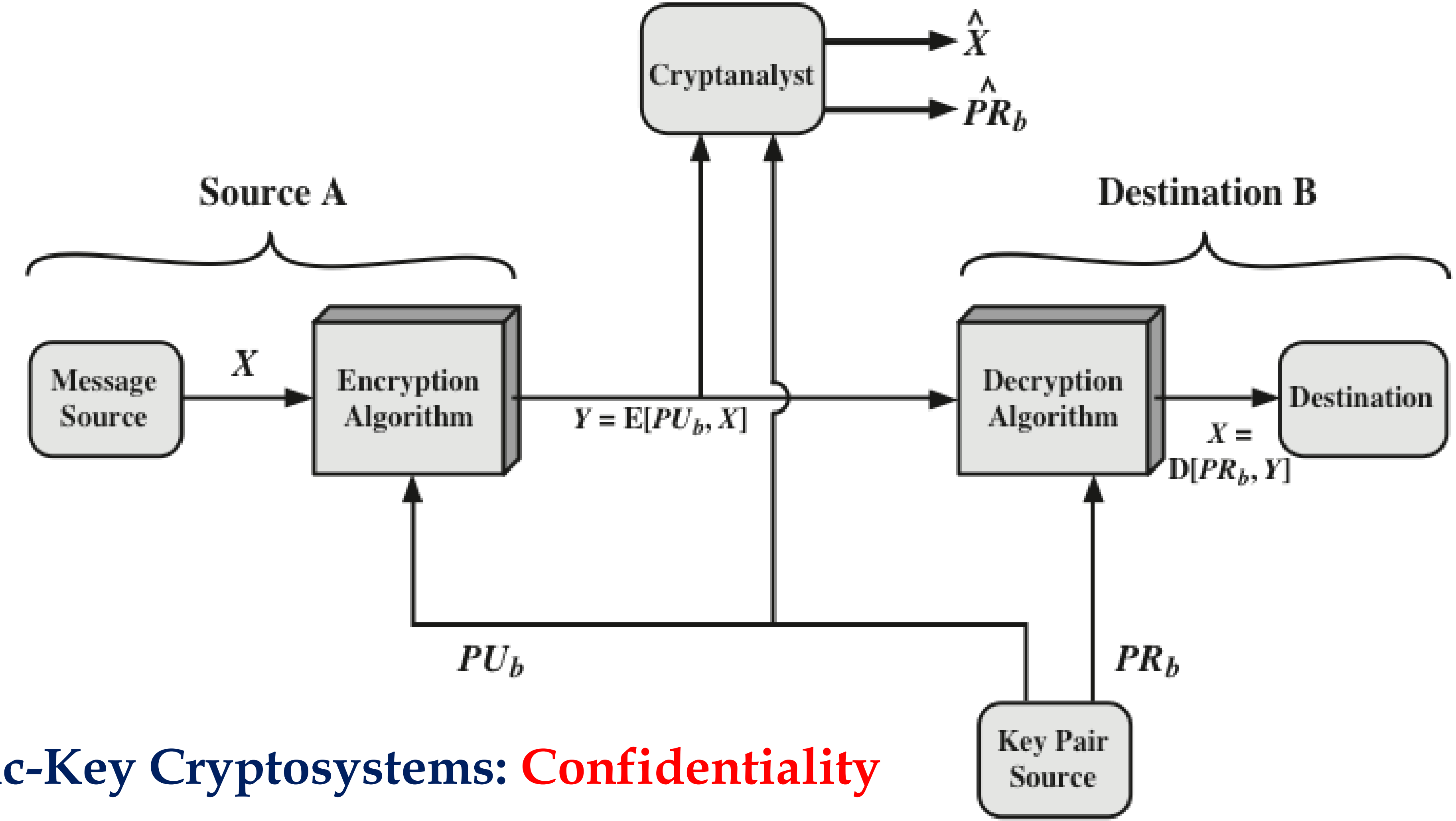
Public-Key Cryptosystems: Confidentiality (Cont.)



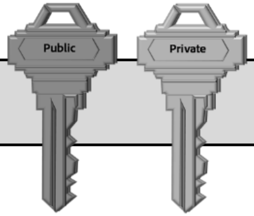
Mechanism in public-key cryptosystem (Cont.):

- Private keys are generated locally by each participant and therefore ***need never to be distributed***.
- At any time, a system can change its private key and publish the companion public key to replace its old public key.

Public-Key Cryptosystems: Confidentiality (Cont.)

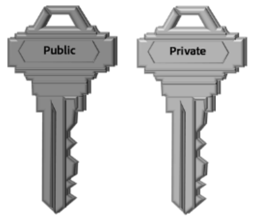


Public-Key Cryptosystems: Confidentiality



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

Public-Key Cryptosystems: Authentication

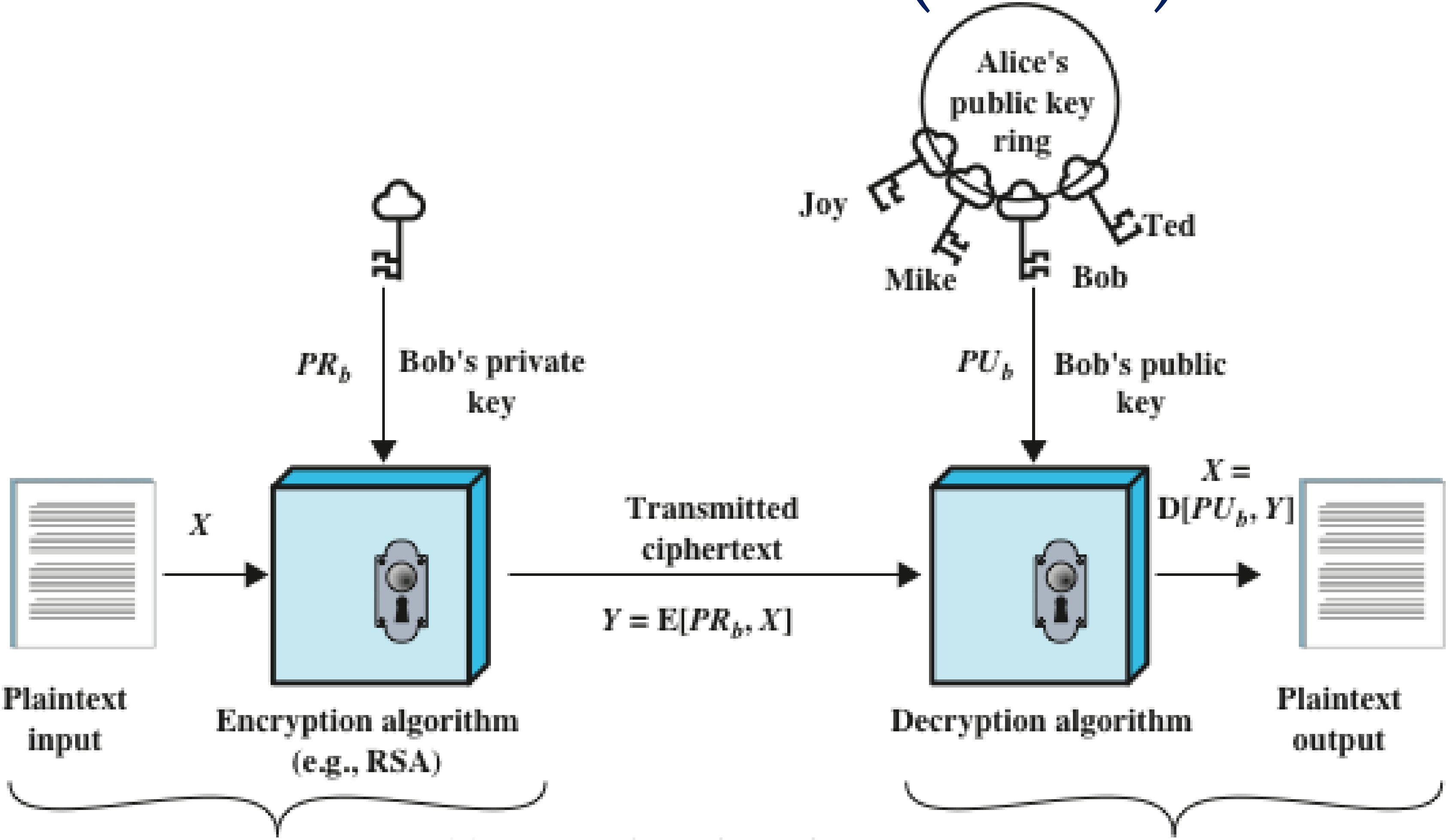


- Since either of the **two related keys** can be used for encryption with the other used for decryption, the public-key encryption can also be used to provide **authentication**.

Example:

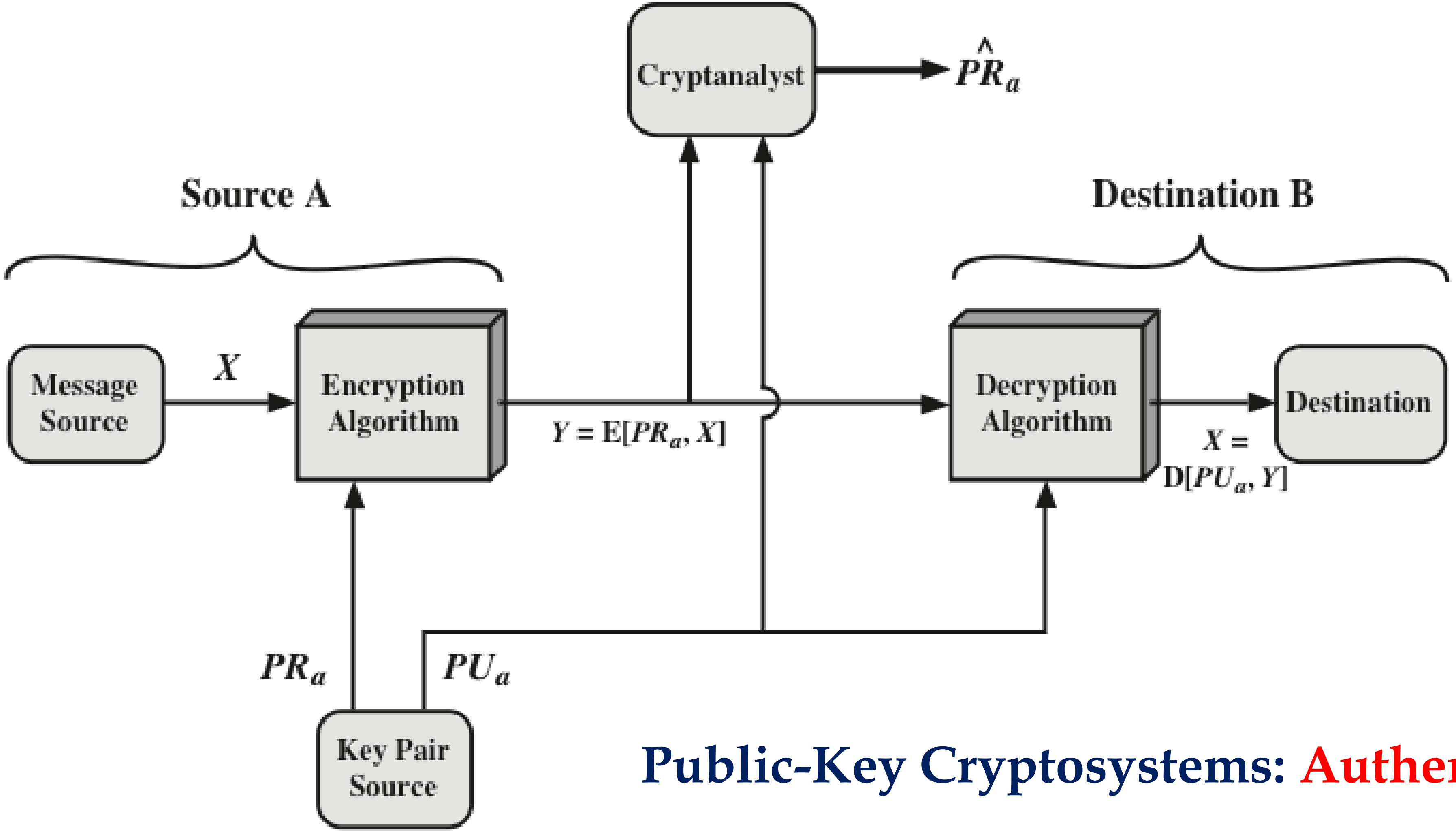
- **A** prepares a message to **B** and encrypts it using **A's** private key.
- **B** can decrypt the message using **A's** public key.
- Because the message was encrypted using **A's** private key, ***only A could have prepared the message not anyone else***.
- Hence, entire encrypted message serves as a **digital signature**.

Public-Key Cryptosystems: Authentication (Cont.)



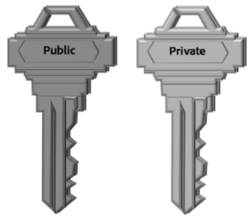
(b) Encryption with private key

Public-Key Cryptosystems: Authentication (Cont.)



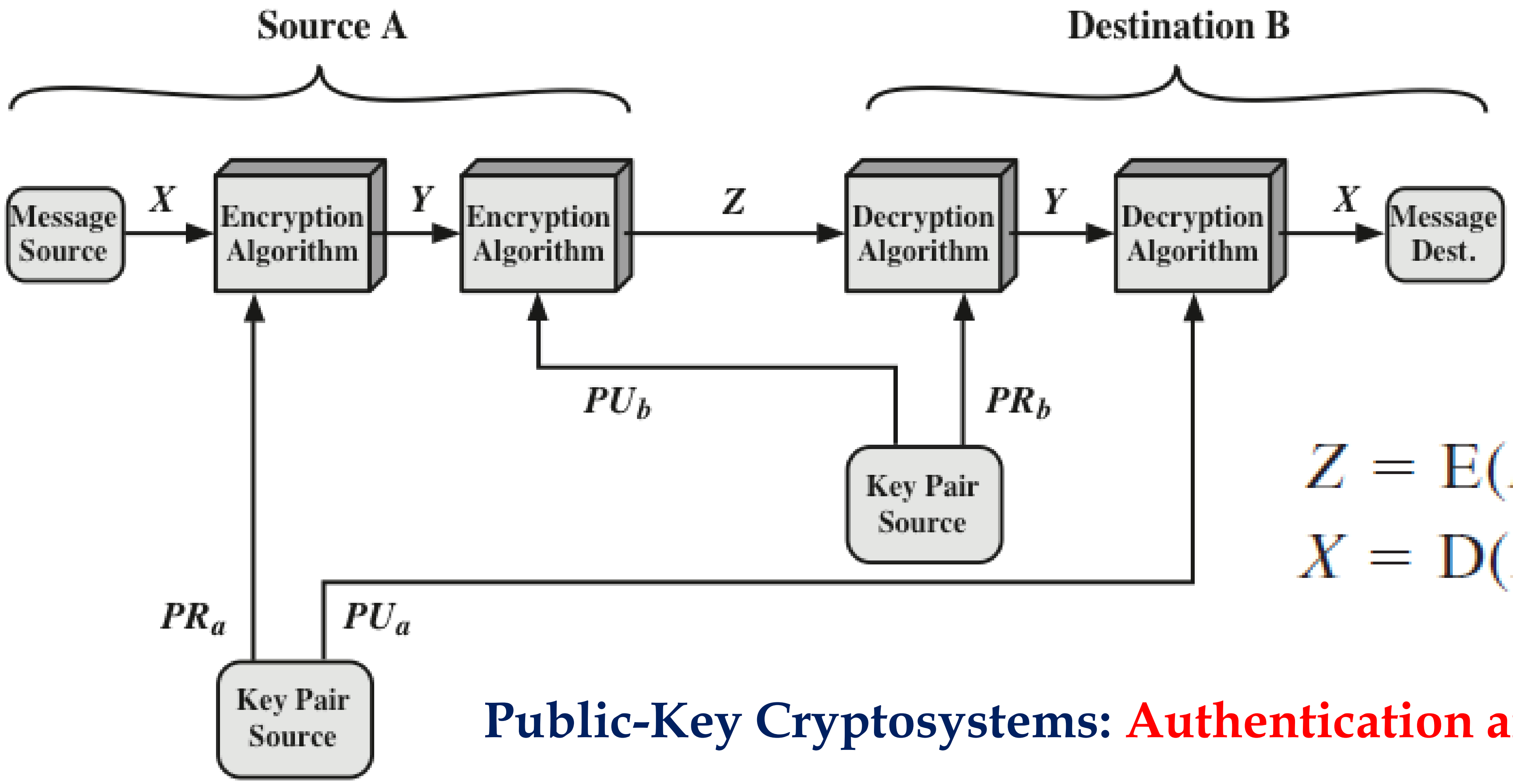
Public-Key Cryptosystems: Authentication

Public-Key Cryptosystems: Confidentiality and Authentication



- The encryption process, using the private key for encryption, does not provide **confidentiality**.
- The message being sent is safe from **authentication** issues but not from **eavesdropping**.
- There is no protection of **confidentiality** because any observer can decrypt the message by using the sender's public key.
- It is possible to provide both **authentication** and **confidentiality** by a **double use of the public-key scheme**.

Public-Key Cryptosystems: Confidentiality and Authentication (Cont.)

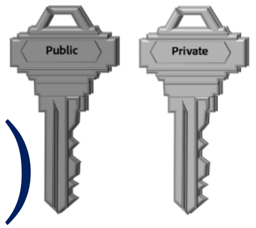


$$Z = E(PU_b, E(PR_a, X))$$

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

Public-Key Cryptosystems: Authentication and Confidentiality

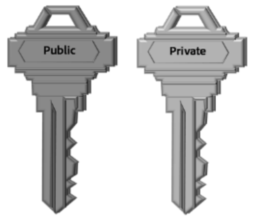
Public-Key Cryptosystems: Confidentiality and Authentication (Cont.)



Working mechanism:

- We begin by encrypting a message using **sender's private key**. Hence, providing **digital signature**.
- Next, we encrypt again using the **receiver's public key**. Hence, generating the **final ciphertext**. This **final ciphertext** can be decrypted only by the receiver who has the **matching private key**. Thus, **confidentiality** is achieved.
- The receiver decrypts the received **ciphertext** first by its **own private key**. Followed by decrypting the result with the **sender's public key**. By that, the **plaintext** is obtained.

Application of Public-Key Cryptosystems



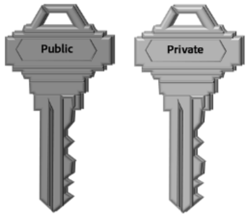
- Depending on the application, sender uses either the **sender's private key** or **receiver's public key** or **both**.
- Broadly, we can classify the use of public-key cryptosystems into three categories:

Encryption / Decryption
- **The sender encrypts a message with the recipient's public key**

Digital signature
- **The sender "signs" a message with its private key**

Key exchange
- **Two sides cooperate to exchange a session key, which is a secret key for symmetric encryption**

Application of Public-Key Cryptosystems (Cont.)

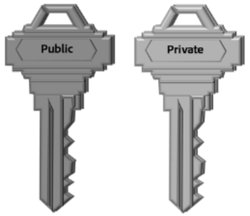


- Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications.
- Table below indicates the applications supported by the algorithms.

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

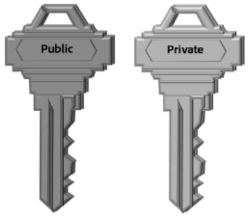
(Rivest–Shamir–Adleman)
RSA

RSA Overview



- **RSA** scheme is the most widely accepted and implemented "general-purpose" approach to public-key encryption.
- **RSA** is a cipher in which **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**.
- So, n is less than 2^{1024}.

RSA Algorithm

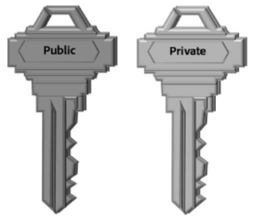


- **RSA** makes use of **exponential** expressions.
- For some plaintext block ***M*** and ciphertext block ***C***, encryption and decryption are of the following form:

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

- Sender knows value of ***e*** and "only" receiver knows value of ***d***.
- However, both sender and receiver must know the value of ***n***.
- **RSA** is a public key encryption algorithm with the following:
  - ***Public key of PU = {e, n}*** ***Private key of PR = {d, n}***

RSA Algorithm (Cont.)



- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

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

RSA Algorithm (Cont.)

- The relationship $M^{ed} \bmod n = M$ holds if e and d are multiplicative inverses modulo **Φ(n)**, where **Φ(n)** is the **Euler totient function**.

- The relationship between **e** and **d** can be expressed as:

$$ed \bmod \Phi(n) = 1$$

$$ed \equiv 1 \bmod \Phi(n)$$

- According to rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\Phi(n)$. Equivalently, $\gcd(\Phi(n), d) = 1$.

RSA Algorithm (Cont.)

- RSA algorithm is based on a fact that finding factors of large composite numbers is difficult ***when the integers are prime numbers***.

Following are required in RSA algorithm:

- p, q are two prime numbers *(private, chosen)*
- $n = pq$ *(public, calculated)*
- $\Phi(n) = \Phi(pq) = (p - 1)(q - 1)$.
- e, such that $\gcd(\Phi(n), e) = 1$, *(public, chosen)*
  where $1 < e < \Phi(n)$
- $d \equiv e^{-1} \bmod \Phi(n)$ *(private, calculated)*

Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
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	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

RSA Example

Example of RSA algorithm:

- Select two prime numbers, $p = 17$ and $q = 11$
- Calculate $n = pq = 17 \times 11 = 187$
- Calculate $\Phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$
- Select e such that e is relatively prime to $\Phi(n) = 160$ and $1 < e < \Phi(n)$. We choose $e = 7$
- Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$
- Value of $d = 23$ *(calculated using extended Euclid's algorithm)*
- Resulting keys are $PU = \{7, 187\}$ and $PR = \{23, 187\}$

RSA Example (Cont.)

Example of RSA algorithm (Cont.):

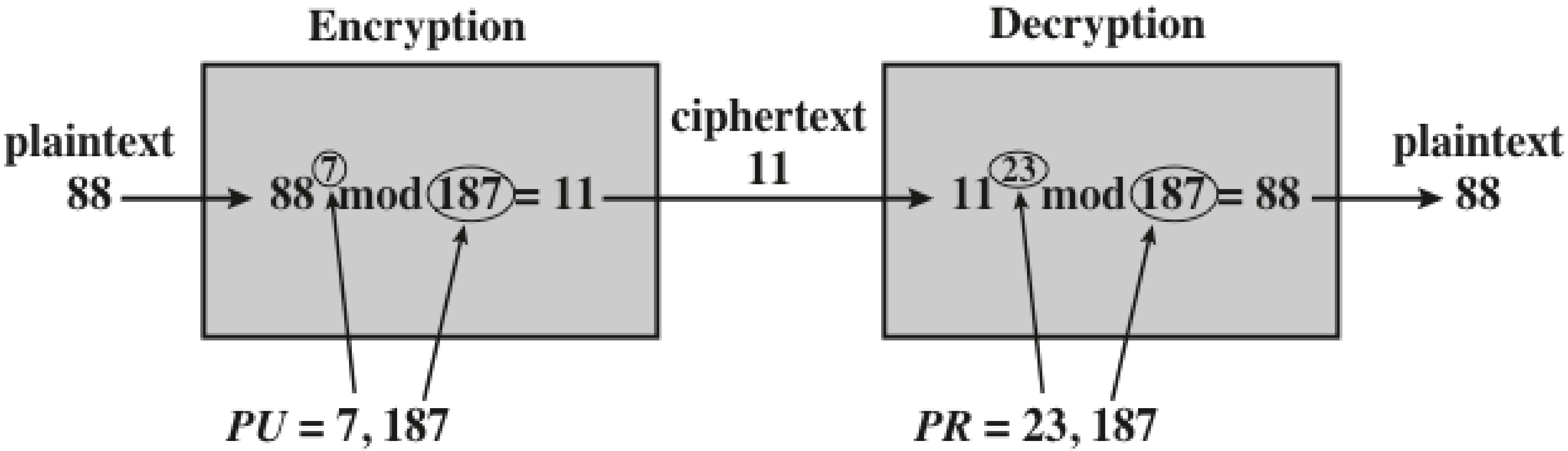
- Use the generated keys for a plaintext input of $M = 88$
- For encryption, we calculate $C = 88^7 \bmod 187$
- By exploiting properties of modular arithmetic, we have:
  - $88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$
  - $88^1 \bmod 187 = 88$
  - $88^2 \bmod 187 = 7744 \bmod 187 = 77$
  - $88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$
  - So, $88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$
  - So, **C = 11**

RSA Example (Cont.)

Example of RSA algorithm (Cont.):

- For decryption, we calculate $M = 11^{23} \bmod 187$:
  - $11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$
  - $11^1 \bmod 187 = 11$
  - $11^2 \bmod 187 = 121$
  - $11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$
  - $11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$
  - $11^{23} \bmod 187 = (11 * 121 * 55 * 33 * 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$
  - So, **M = 88**

RSA Example (Cont.)



RSA Practice

- **Example:** while using the RSA algorithm, show the process to encrypt and decrypt a letter with ASCII value of 32 *(i.e. space character)*. Given that: p = 3, q = 11 and e = 17

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Diffie-Hellman Key Exchange

Diffie-Hellman Overview

- The main purpose of Diffie-Hellman algorithm is to enable two users to ***securely exchange a "key"***.
- The shared ***"key"*** can then be used for subsequent ***symmetric encryption***.
- Diffie-Hellman algorithm is limited to the exchange of ***secret values (i.e. keys)***.
- For its effectiveness, the algorithm depends on the difficulty of computing **discrete logarithms**.

Primitive Root

- A **primitive root** of a prime number $\boldsymbol{p}$ is one whose powers **modulo** $\boldsymbol{p}$ generate all the integers from (**1 to** $\boldsymbol{p-1}$).
- If $\boldsymbol{a}$ is a **primitive root** of the prime number $\boldsymbol{p}$, then the numbers below are ***distinct*** and consist of integers from **1 to** $\boldsymbol{p-1}$ in some permutation.

$$a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p$$

Key Exchange Algorithm

- There are **two** ***publicly known numbers***, i.e. a **prime number** q and an integer α that is a **primitive root** of q.
- To create a shared *"secret key"* between Alice and Bob, the following are adopted:
  a. Alice independently selects a random integer $X_A < q$
  b. Bob independently selects a random integer $X_B < q$
  c. Alice computes $Y_A = \alpha^{X_A} \bmod q$
  d. Bob computes $Y_B = \alpha^{X_B} \bmod q$

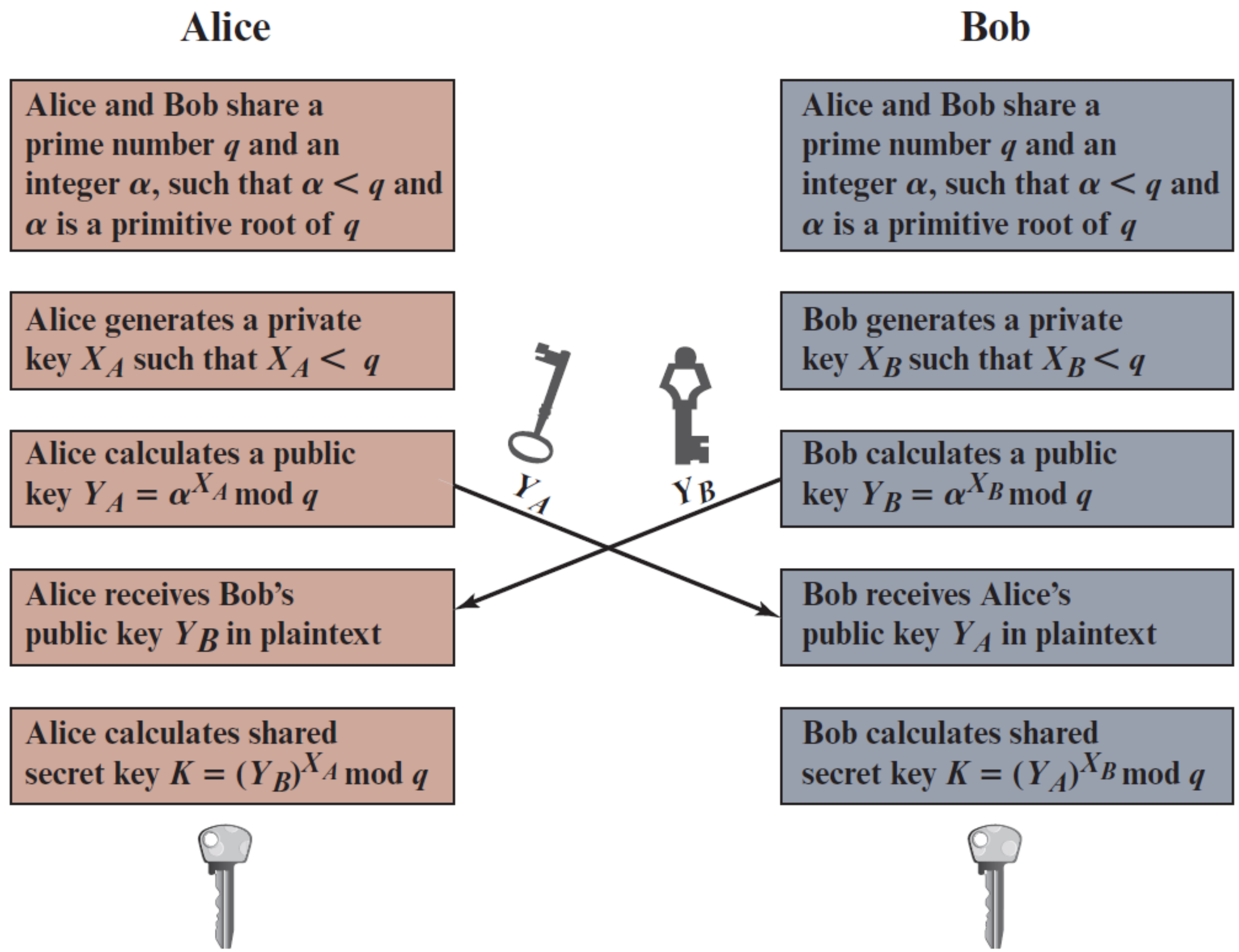
Key Exchange Algorithm (Cont.)

- Each side keeps the *X* value **"private"** and makes the *Y* value available **"publicly"** to the other side.
- Hence, X_A is Alice's **private key** and Y_A is Alice's corresponding **public key** → Alice (X_A , Y_A).
- Whereas, X_B is Bob's **private key** and Y_B is Bob's corresponding **public key** → Alice (X_B , Y_B).

Key Exchange Algorithm (Cont.)

- Finally, the **"secret key"** is computed at each side as follows:
  1. Alice computes the key as $\boldsymbol{K = (Y_B)^{X_A} \bmod q}$
  2. Bob computes the key as $\boldsymbol{K = (Y_A)^{X_B} \bmod q}$
- The above two calculations would produce **identical results**!

Key Exchange Algorithm (Cont.)



Diffie-Hellman Example

- **Q)** Given the prime number q = **353** and a primitive root a = 3. Calculate the **secret key** if Alice and Bob select private keys X_A = **97** and X_B = **233**, respectively.

- **A)** Alice computes $Y_A = \alpha^{X_A} \bmod q$
  - $Y_A = 3^{97} \bmod 353$ → ***In Book (397 mod 353 = 40)!***
  - $Y_A = [(3^{20} \bmod 353) \times (3^{20} \bmod 353) \times (3^{20} \bmod 353) \times (3^{20} \bmod 353) \times (3^{17} \bmod 353)] \bmod 353$
  - $Y_A = [73 \times 73 \times 73 \times 73 \times 55] \bmod 353$
  - $Y_A = 40$

Diffie-Hellman Example (Cont.)

- Bob computes $Y_B = \alpha^{X_B} \textit{ mod } q$
  - $Y_B = 3^{233} \textit{ mod } 353$ → ***In Book (3233 mod 353 = 248)!***
  - $Y_B = [(3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{20} \text{ mod } 353) \times (3^{13} \text{ mod } 353)] \text{ mod } 353$
  - $Y_B = [73 \times 73 \times 73 \times 73 \times 73 \times 73 \times 73 \times 73 \times 73 \times 73 \times 73 \times 175] \text{ mod } 353$
  - $Y_B = [2{,}073{,}071{,}593 \times 2{,}073{,}071{,}593 \times 12775] \text{ mod } 353$

Diffie-Hellman Example (Cont.)

- Bob computes $Y_B = \alpha^{X_B} \; mod \; q$ **(Cont.)**
  - $Y_B = [(2{,}073{,}071{,}593 \bmod 353) \times (2{,}073{,}071{,}593 \bmod 353) \times (12775 \bmod 353)] \bmod 353$
  - $Y_B = [21 \times 21 \times 67] \bmod 353$
  - $Y_B = [29547] \bmod 353$
  - $Y_B = 248$

Diffie-Hellman Example (Cont.)

- After Alice and Bob exchange **public keys**, each can compute the common **“secret key”**:
- Alice computes $\boldsymbol{K = (Y_B)^{X_A} \ mod \ q}$

  = 248 ^{97} mod 353 = 24897 mod 353 = 160.
- Bob computes $\boldsymbol{K = (Y_A)^{X_B} \ mod \ q}$

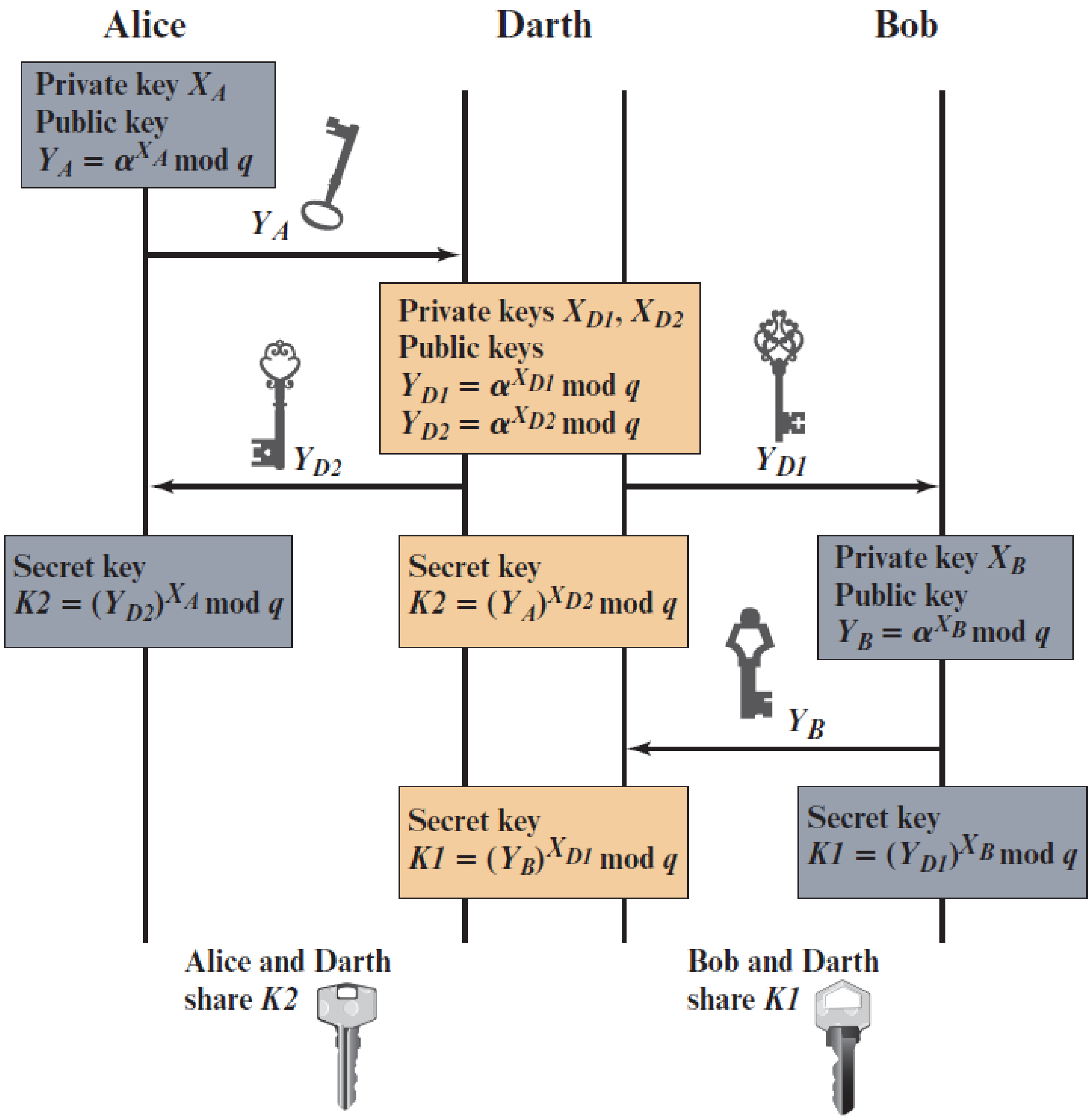
  = 40 ^{233} mod 353 = 40233 mod 353 = 160.
- Hence, the common **secret key = 160**.

Diffie-Hellman: Brute Force Attack

- For small values, e.g. for prime number ***q*** and primitive root ***a***, it would be possible to determine the **secret key** by **brute force**.
- **E.g.**, an attacker can determine the **secret key** by discovering a solution to 3^a mod 353 = 40 ***OR*** 3^b mod 353 = 248, through **brute-force**.
- For the case of ***"3^a mod 353 = 40"***, the attacker would stop the search process when ***a*** = 97, which provides 3^{97} mod 353 = 40.
- However, **brute force** would becomes **impractical** when considering larger numbers.

Diffie-Hellman: MiTM Attack

- The protocol behind Diffie-Hellman Key Exchange is insecure against a **man-in-the-middle (MiTM)** attack.
- 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**.


Alice
Darth
Bob
Private key X_A
Public key
$Y_A = \alpha^{X_A} \bmod q$
Y_A
Private keys X_{D1}, X_{D2}
Public keys
$Y_{D1} = \alpha^{X_{D1}} \bmod q$
$Y_{D2} = \alpha^{X_{D2}} \bmod q$
Y_{D2}
Y_{D1}
Secret key
$K2 = (Y_{D2})^{X_A} \bmod q$
Secret key
$K2 = (Y_A)^{X_{D2}} \bmod q$
Private key X_B
Public key
$Y_B = \alpha^{X_B} \bmod q$
Y_B
Secret key
$K1 = (Y_B)^{X_{D1}} \bmod q$
Secret key
$K1 = (Y_{D1})^{X_B} \bmod q$
Alice and Darth share $K2$
Bob and Darth share $K1$

Thank You!