

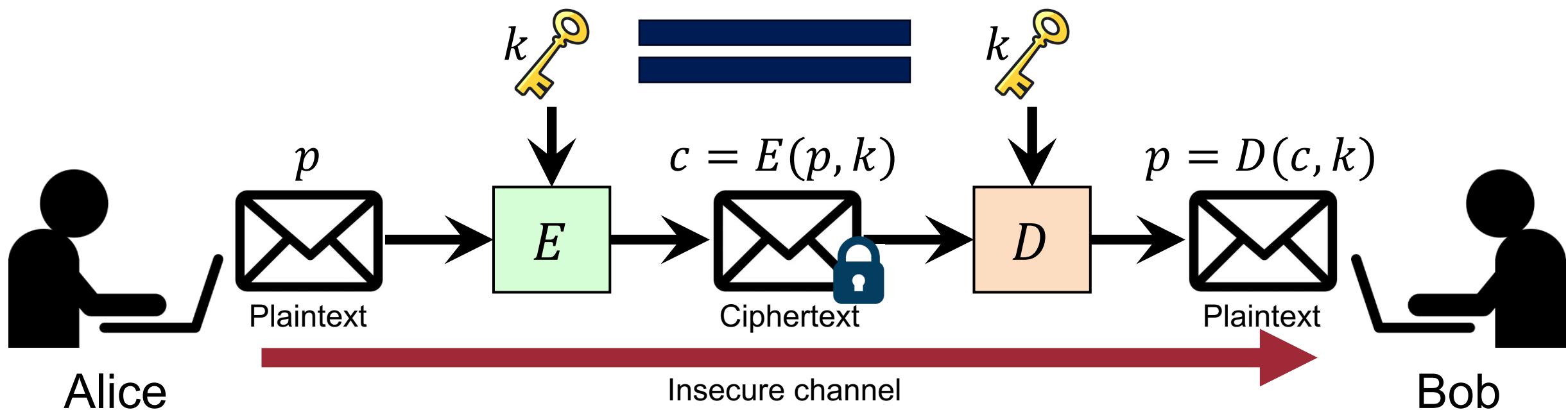
CSE467: Computer Security

6. Public-Key Infrastructure, Integrity

Seongil Wi

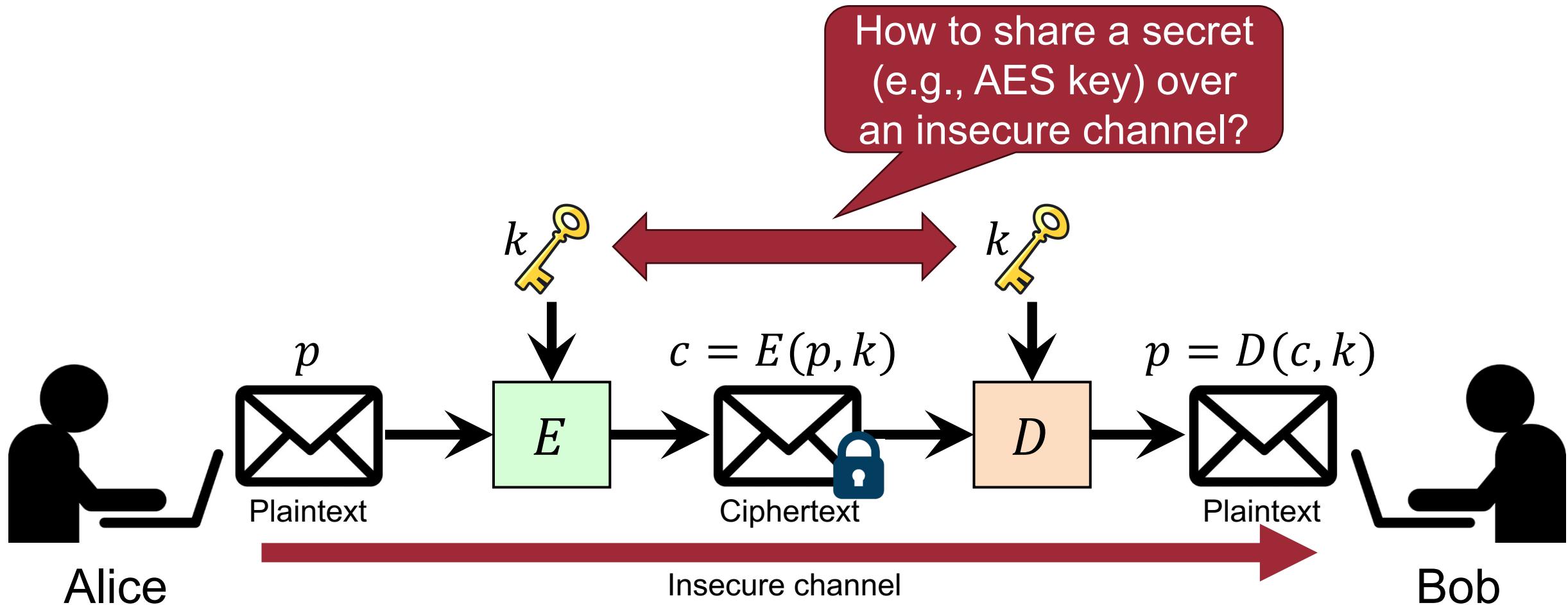
Recap: Symmetric-key Encryption

- **Symmetric:** the encryption and decryption keys are *the same*



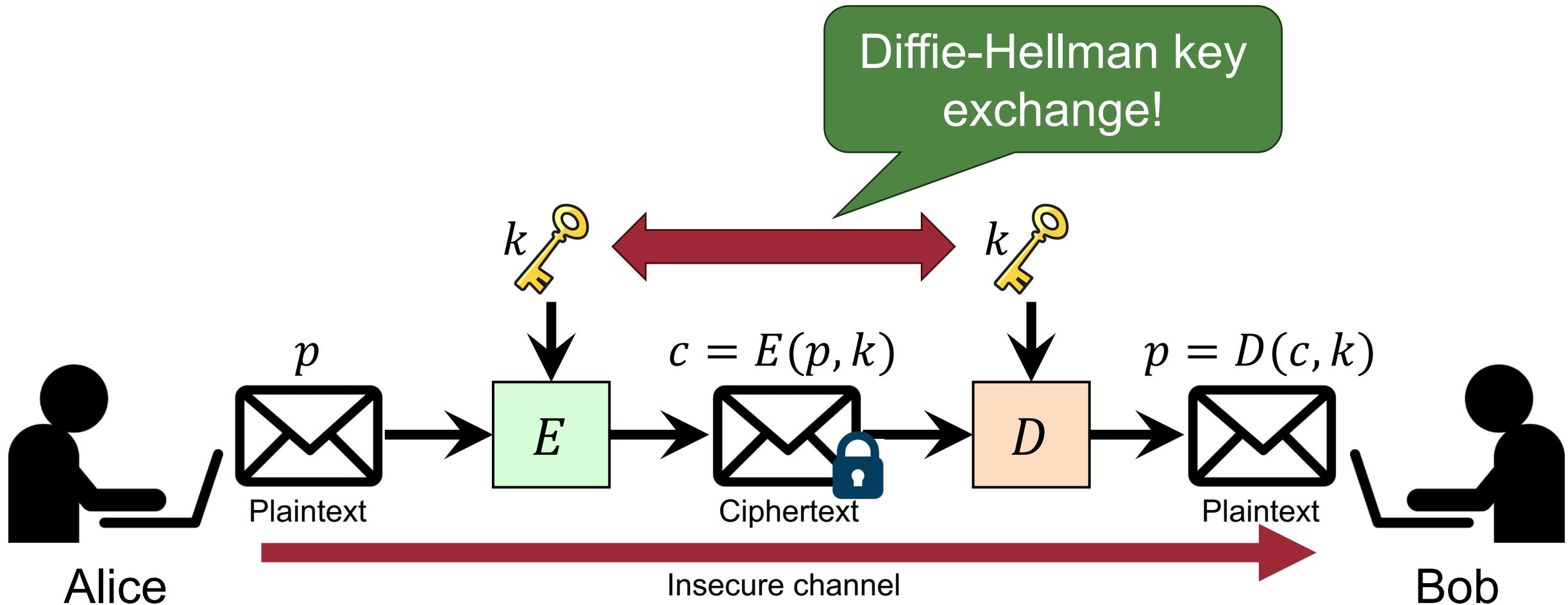
Recap: Symmetric-key Encryption

- **Symmetric:** the encryption and decryption keys are *the same*



Recap: Diffie-Hellman Key Exchange

- **Symmetric:** the encryption and decryption keys are *the same*



Recap: Diffie-Hellman Key Exchange



Symmetric key:



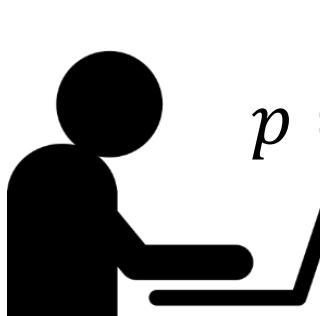
$$K = g^{ab} \bmod p$$



$$p = 23, g = 9$$

$$A = (g^a \bmod p) = 6$$

$$B = (g^b \bmod p) = 16$$



$$a = 4$$

$$p = 23, g = 9$$

$$A = (g^a \bmod p) = 6$$

$$B = (g^b \bmod p) = 16$$

Alice

Insecure channel

$$b = 3$$

$$p = 23, g = 9$$

$$A = (g^a \bmod p) = 6$$

$$B = (g^b \bmod p) = 16$$



Bob

Recap: Diffie-Hellman Key Exchange

Symmetric key:



$$K = g^{ab} \bmod p$$



$$\begin{aligned} p &= 23, g = 9 \\ A &= (g^a \bmod p) = 6 \\ B &= (g^b \bmod p) = 16 \end{aligned}$$

$$\begin{aligned} K &= (B^a \bmod p) = (g^{ab} \bmod p) \\ &= (16^4 \bmod 23) = 9 \end{aligned}$$


$$\begin{aligned} K &= (A^b \bmod p) = (g^{ab} \bmod p) \\ &= (6^3 \bmod 23) = 9 \end{aligned}$$




$$\begin{aligned} a &= 4 \\ p &= 23, g = 9 \end{aligned}$$

$$\begin{aligned} A &= (g^a \bmod p) = 6 \\ B &= (g^b \bmod p) = 16 \end{aligned}$$

Alice

Insecure channel



$$b = 3$$

$$\begin{aligned} p &= 23, g = 9 \\ A &= (g^a \bmod p) = 6 \\ B &= (g^b \bmod p) = 16 \end{aligned}$$

Bob

Recap: Security of the Diffie-Hellman Key Exchange

Symmetric key:



$$K = g^{ab} \bmod p$$



$$p = 23, g = 9$$

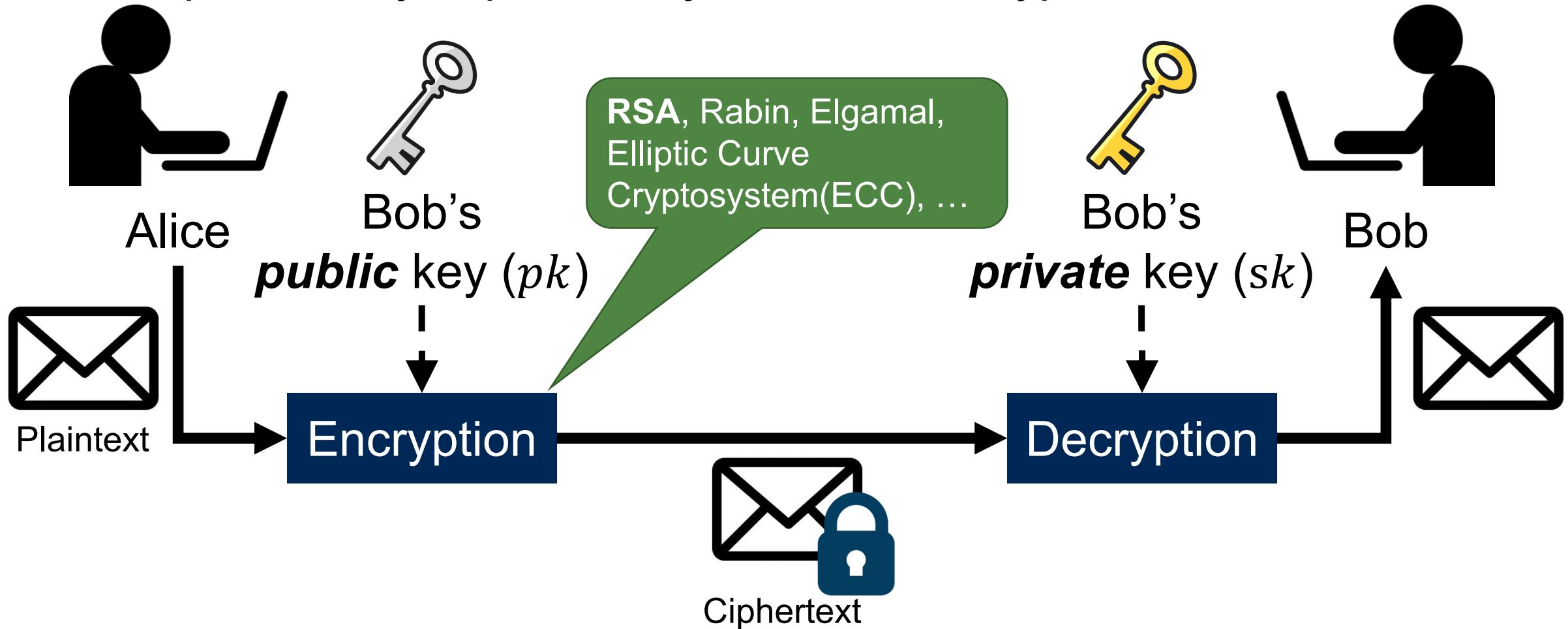
$$A = (g^a \bmod p) = 6$$

$$B = (g^b \bmod p) = 16$$

The attacker cannot efficiently
compute $(g^{ab} \bmod p)$
without knowing a and b

Recap: Asymmetric-key Cryptography

- pk : public key, widely disseminated, used for encryption
- sk : private key kept secretly, used for decryption

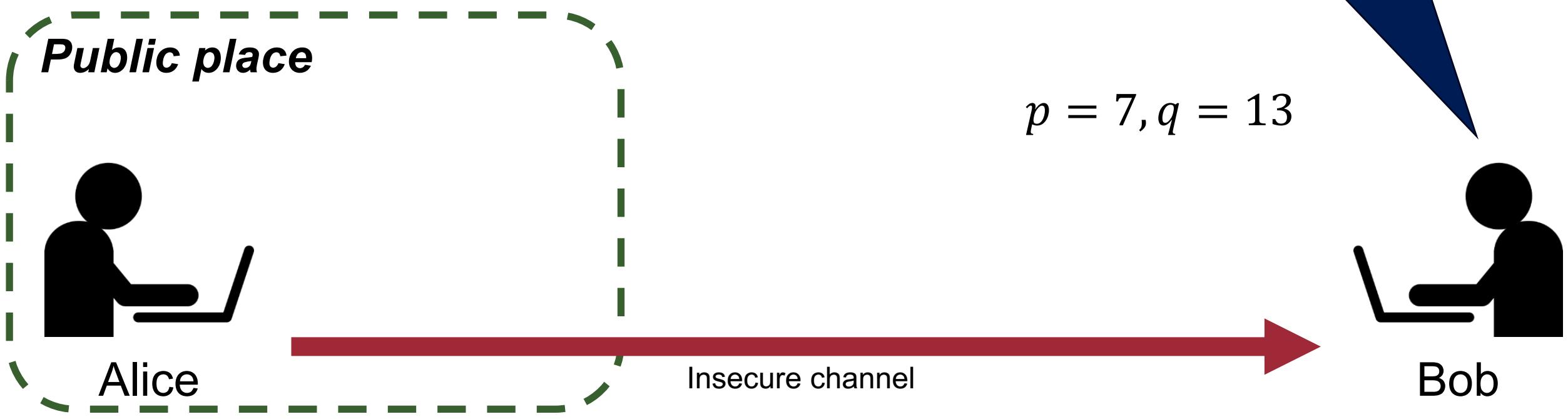


Recap: RSA Algorithm

9

Select two large
primes p and q

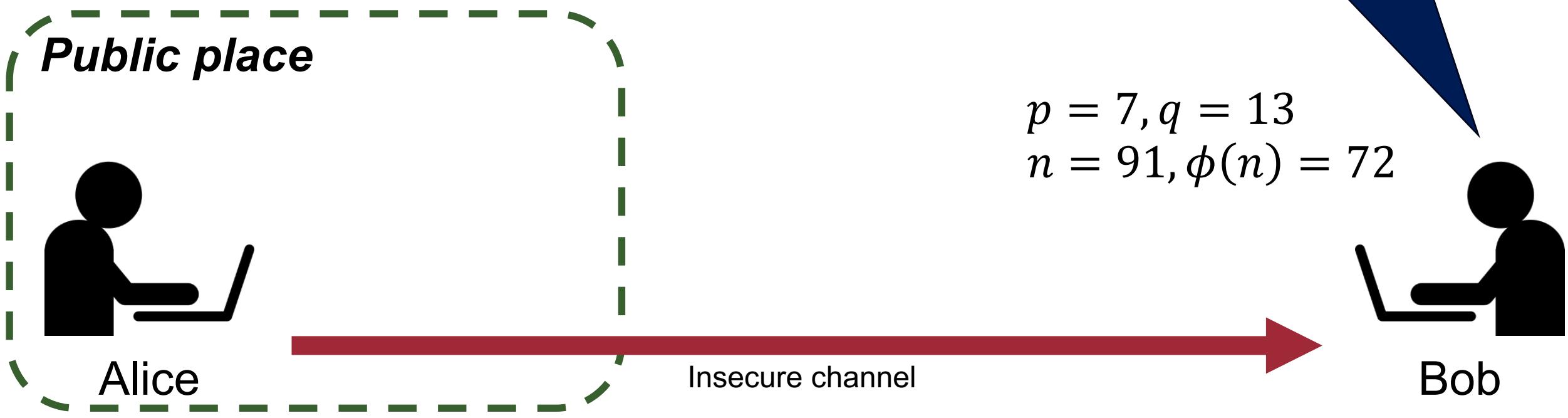
$$p = 7, q = 13$$



Recap: RSA Algorithm

10

Compute $n = pq$ and
 $\phi(n) = (p - 1)(q - 1)$



Recap: RSA Algorithm

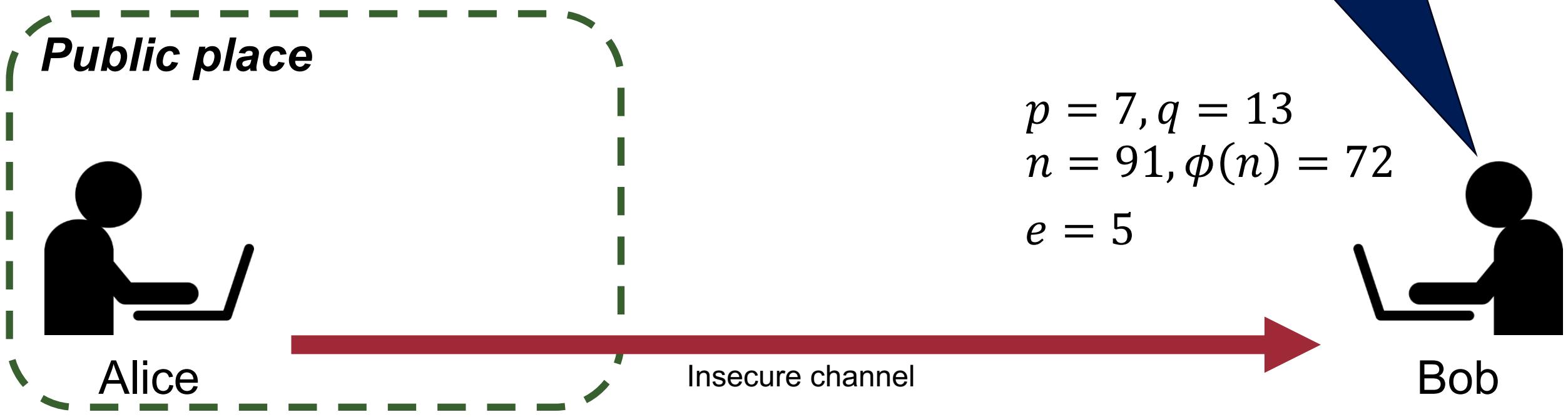
11

Choose e s.t.

- $1 < e < \phi(n)$ and
- $\gcd(\phi(n), e) = 1$

$$\begin{aligned} p &= 7, q = 13 \\ n &= 91, \phi(n) = 72 \end{aligned}$$

$$e = 5$$



Recap: RSA Algorithm

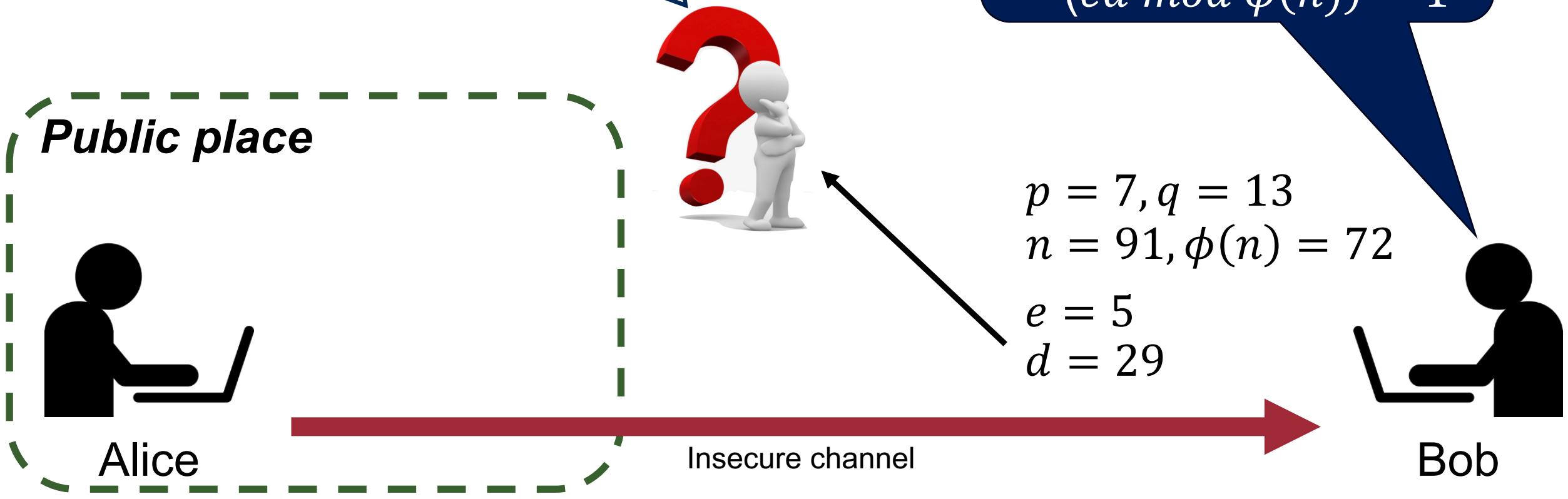
12

How to find d ?
→ Extended Euclidean Algorithm!

Choose d s.t.

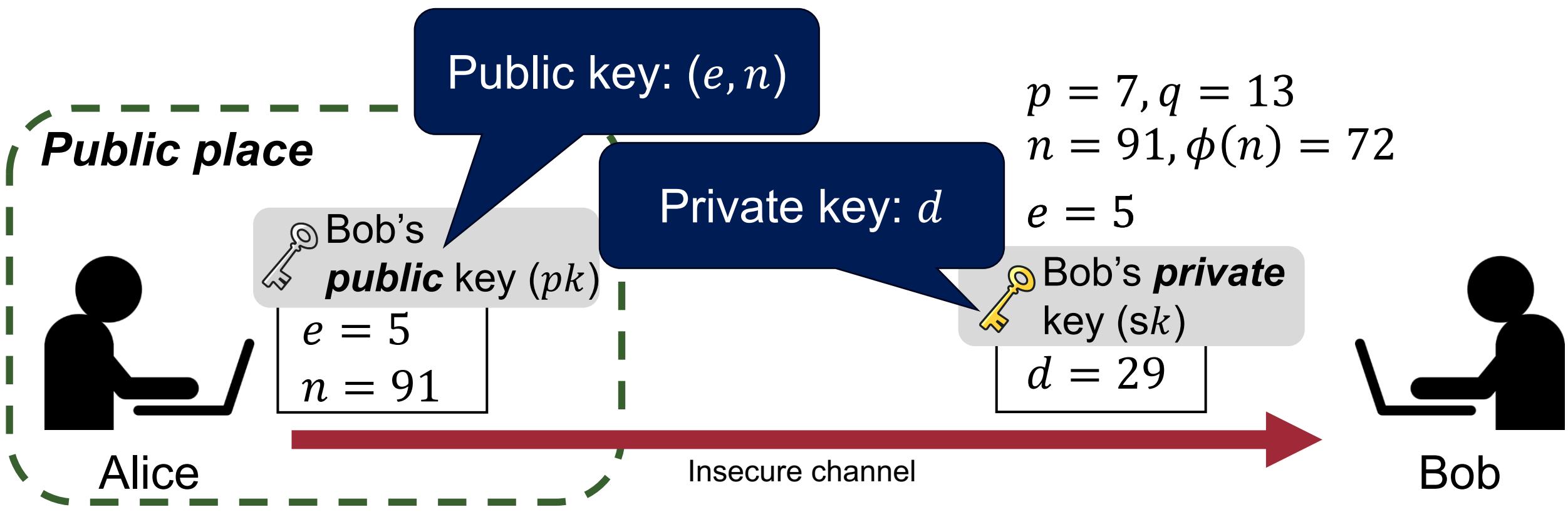
- $1 < d < \phi(n)$ and
- $(ed \bmod \phi(n)) = 1$

$$\begin{aligned} p &= 7, q = 13 \\ n &= 91, \phi(n) = 72 \\ e &= 5 \\ d &= 29 \end{aligned}$$



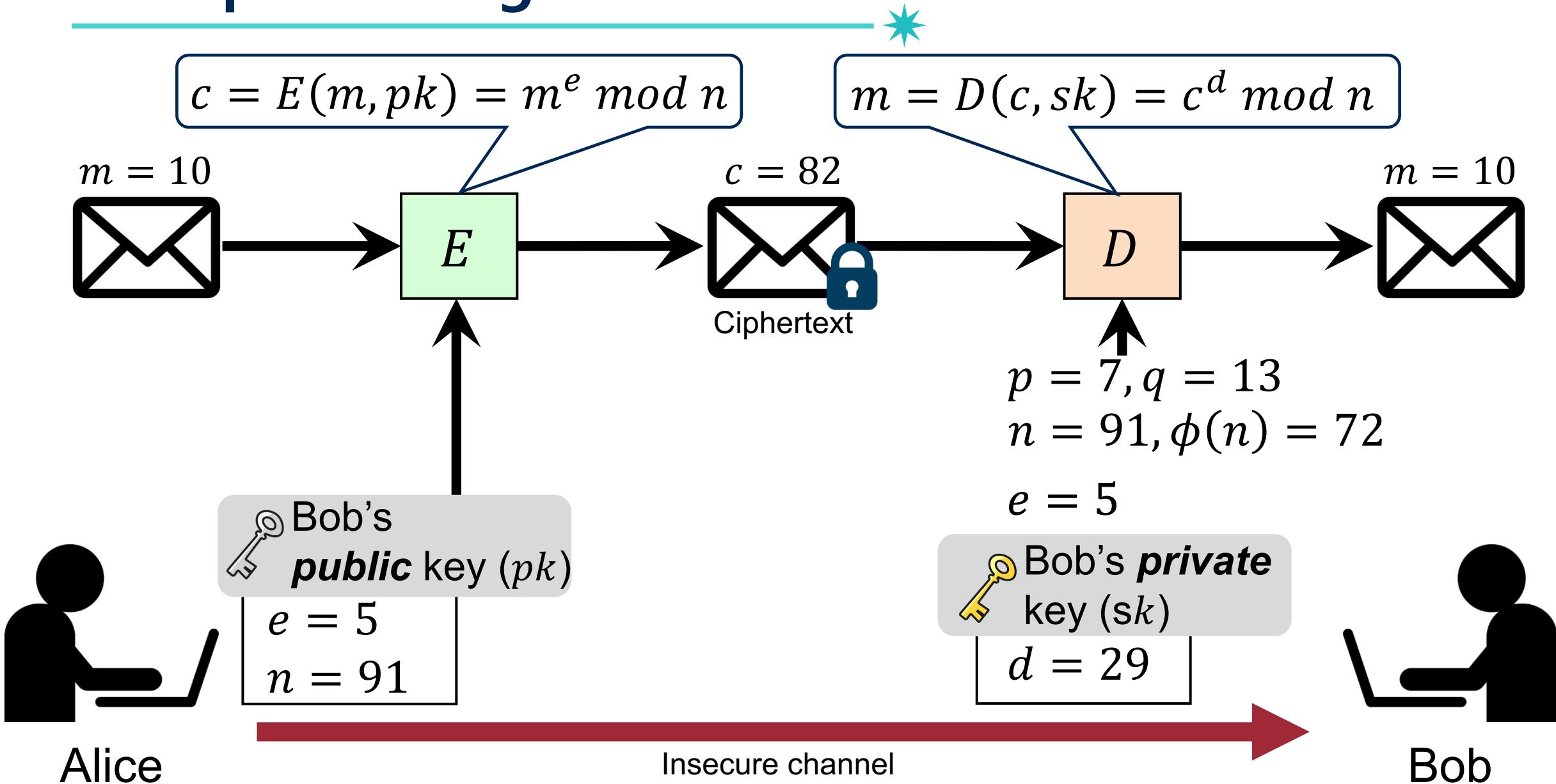
Recap: RSA Algorithm

13



Recap: RSA Algorithm

14



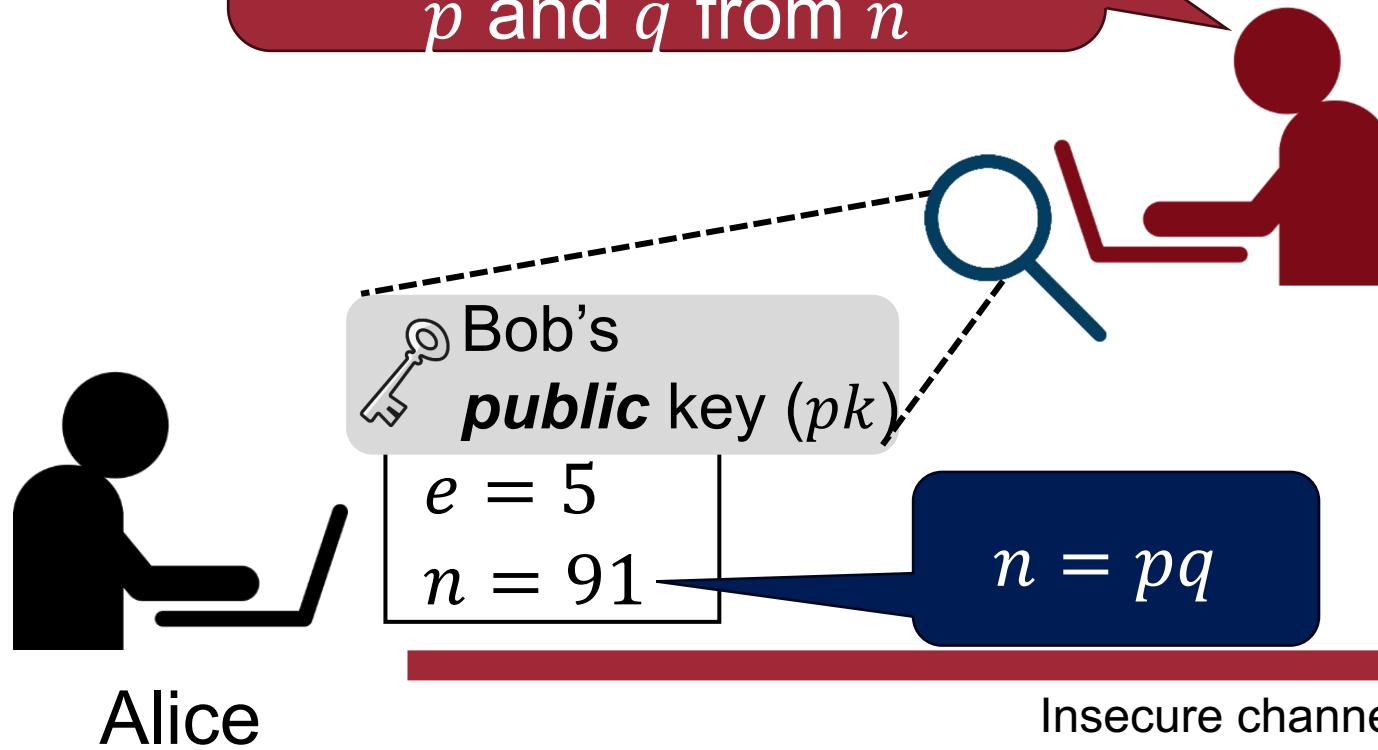
Recap: Security of the RSA Algorithm

15

$$c = E(m, pk) = m^e \bmod n$$

$$m = D(c, sk) = c^d \bmod n$$

The attacker cannot efficiently compute p and q from n



$$\begin{aligned} p &= 7, q = 13 \\ n &= 91, \phi(n) = 72 \end{aligned}$$

$$e = 5$$

A grey box labeled "Bob's **private** key (sk)" contains $d = 29$. This box is shown being sent from Bob to Alice over the same "Insecure channel" represented by the red arrow.



Recap: Digital Signature

16



Alice

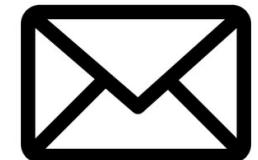


Bob's **private** key

Sign
(encryption)



Bob



Plaintext

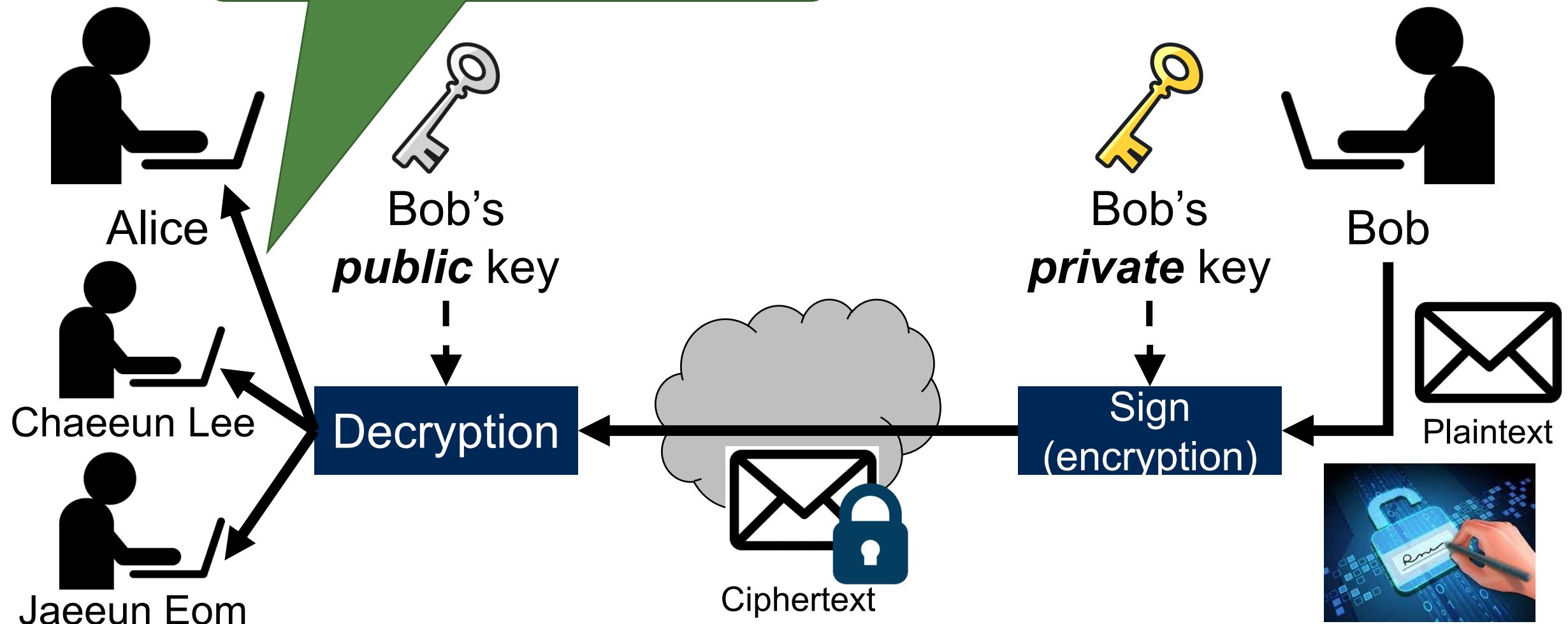


Ciphertext

Recap: Digital Signature

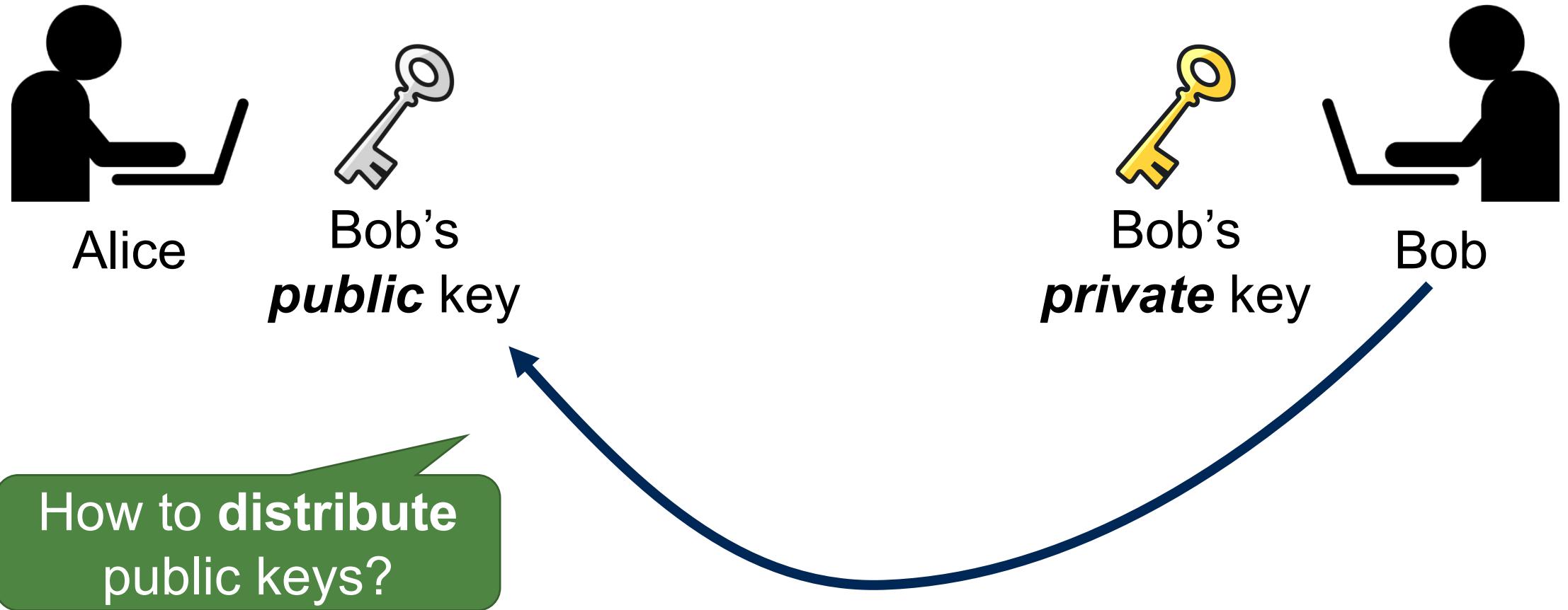
17

This message is from Bob
(authentication)



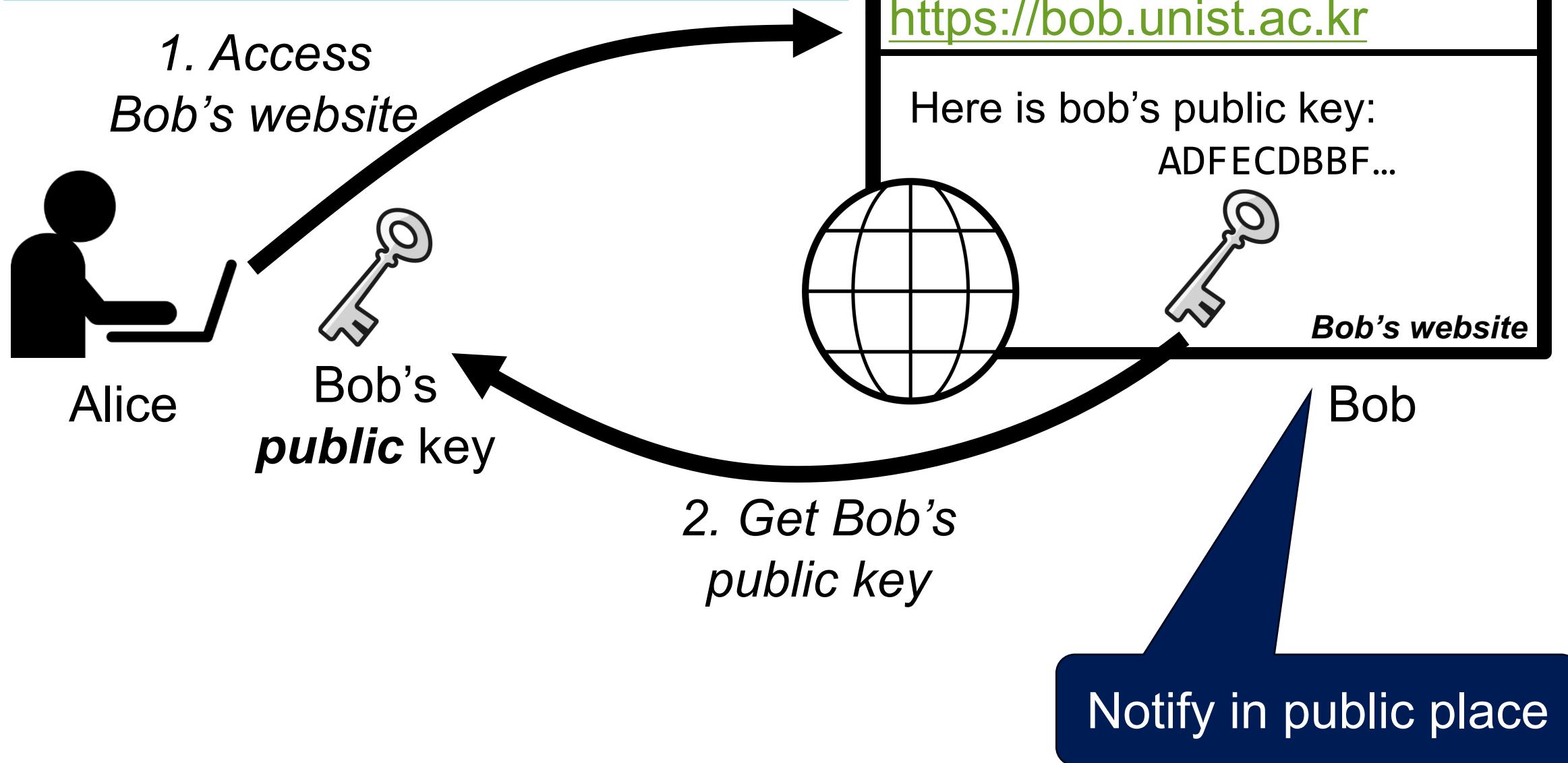
Distribution of Public Keys

18



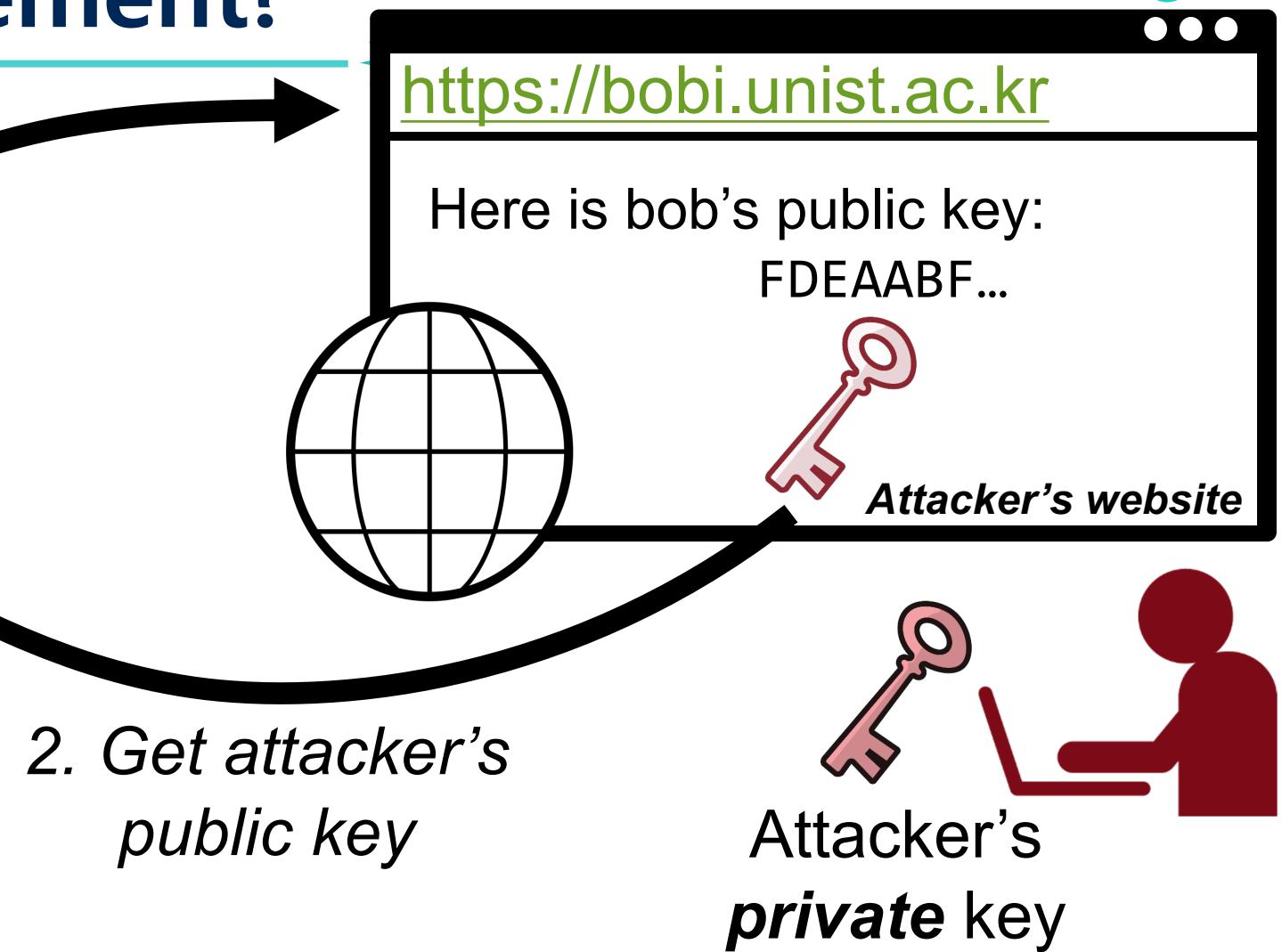
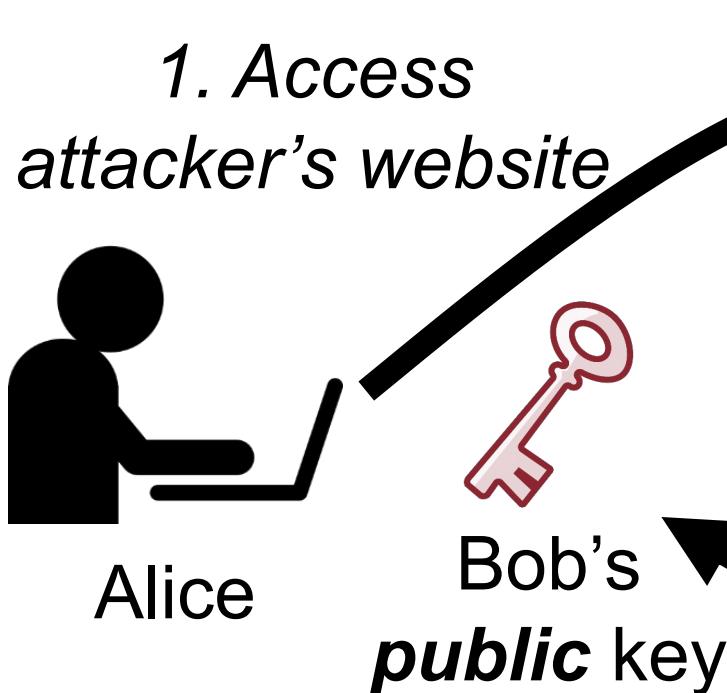
Public Announcement?

19



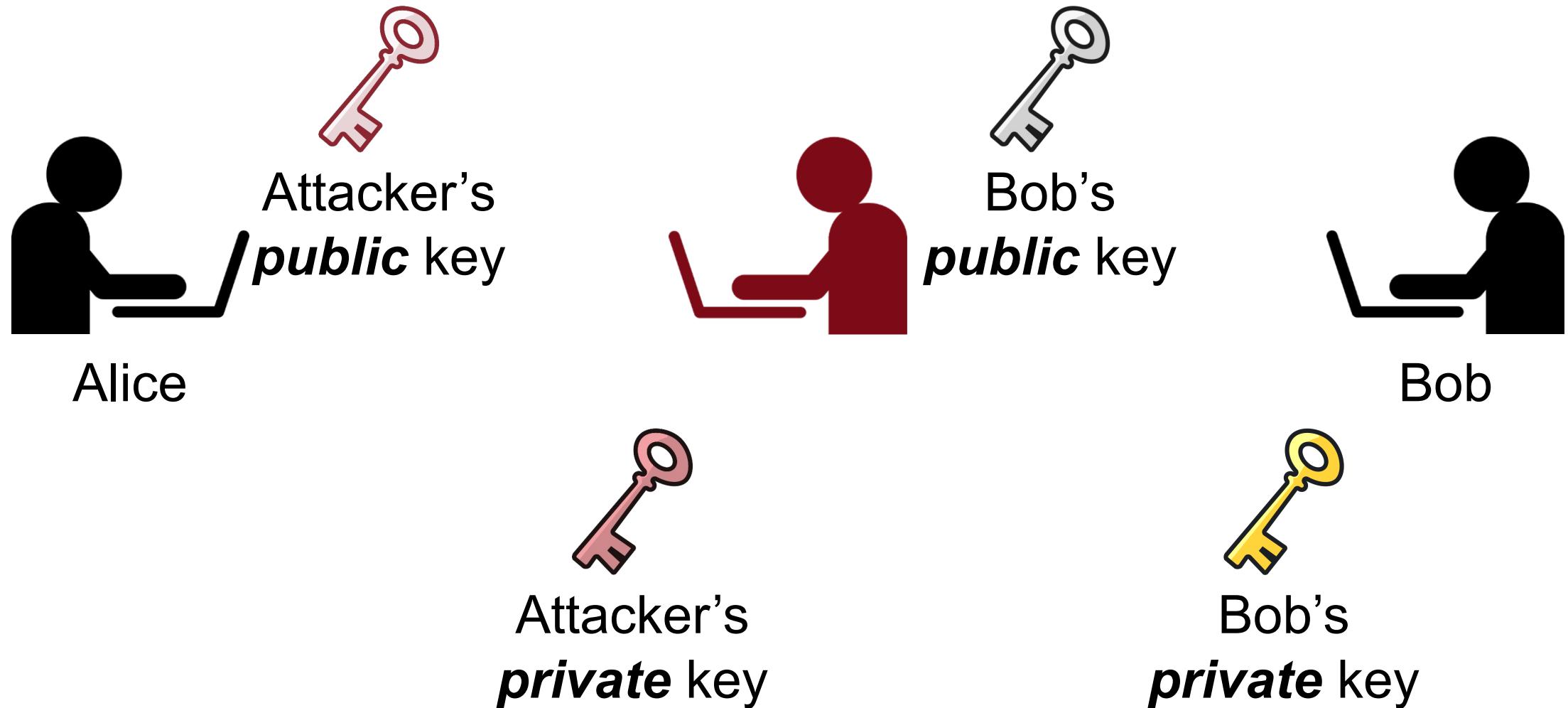
Public Announcement?

20



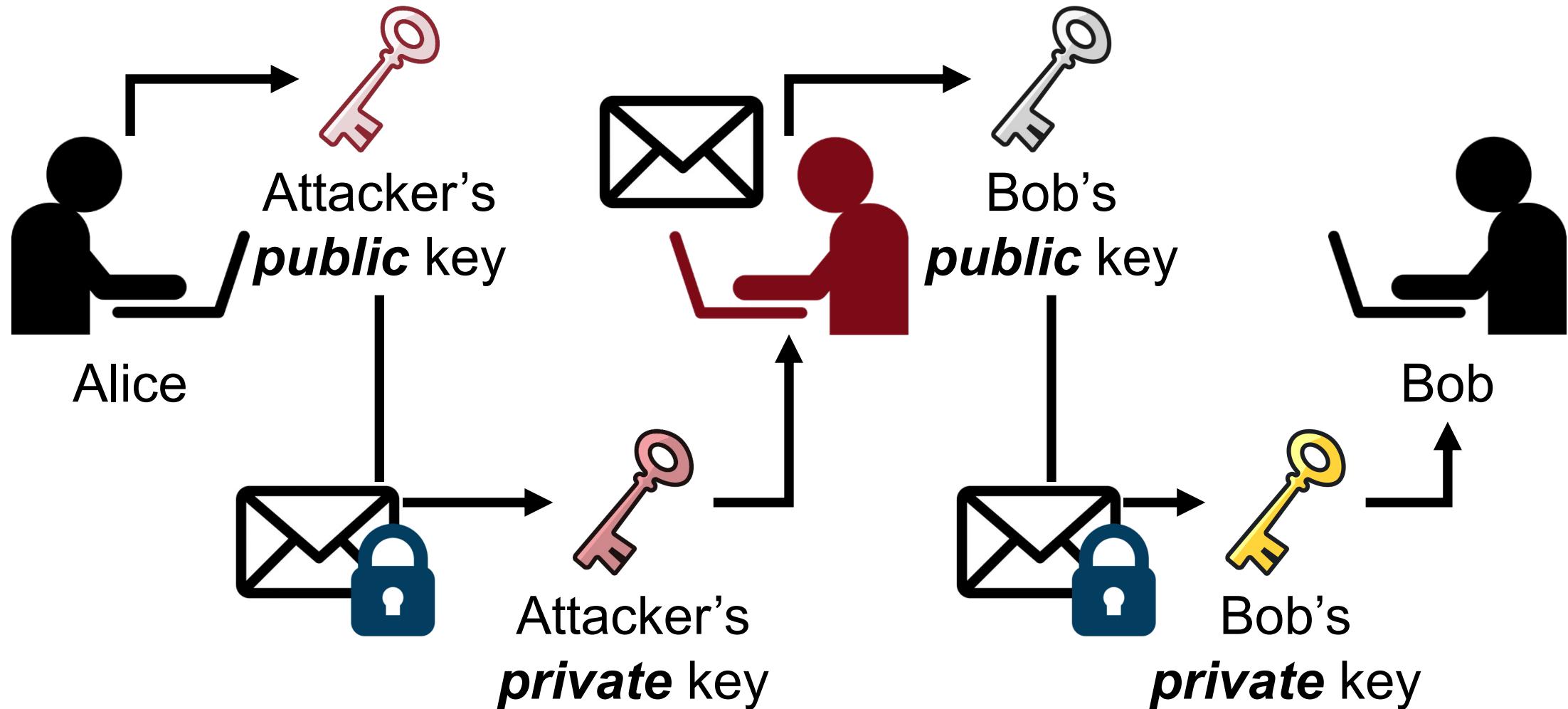
Man-in-the-Middle (MITM) Attack

21



Man-in-the-Middle (MITM) Attack

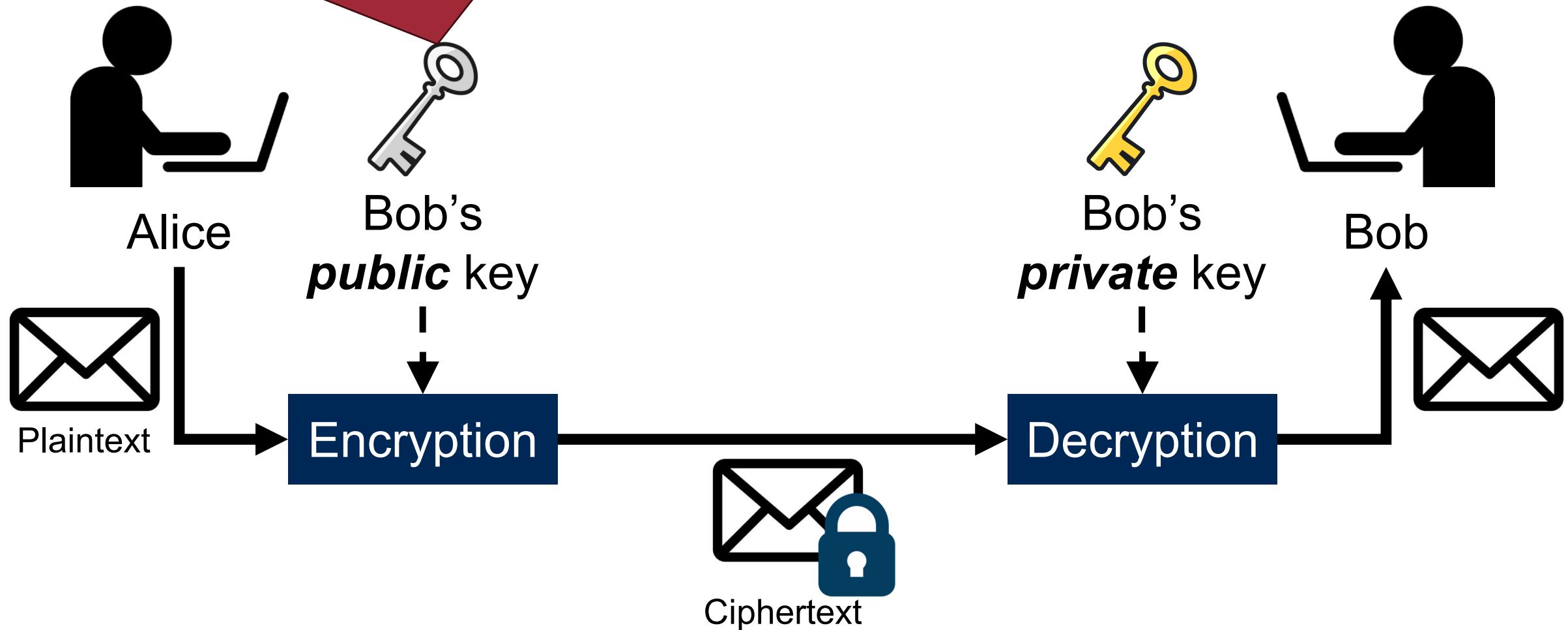
22



Motivation

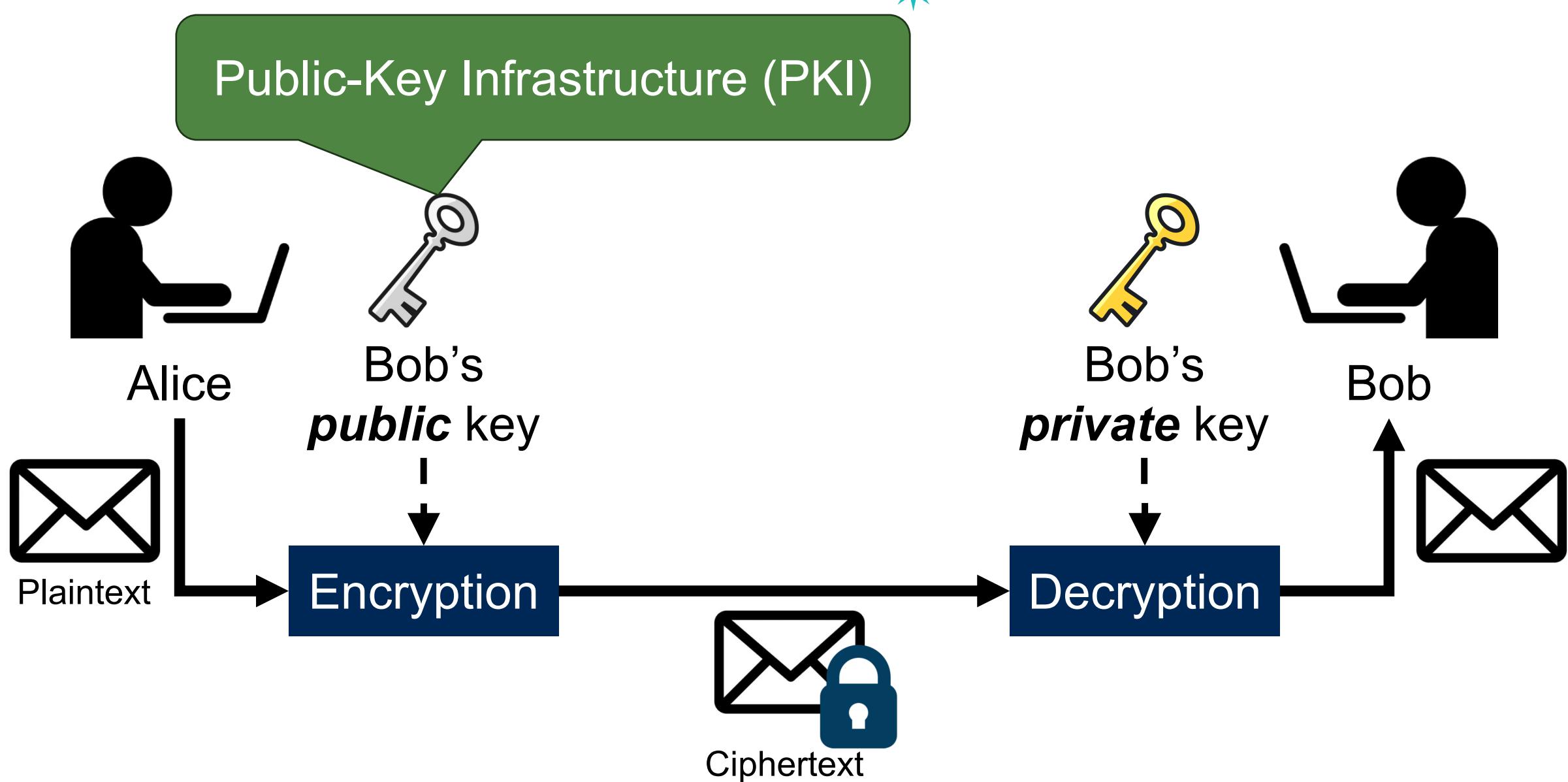
23

How can we trust that this
public key belongs to Bob?



Public-Key Infrastructure

24



Key Idea of Public-Key Infrastructure

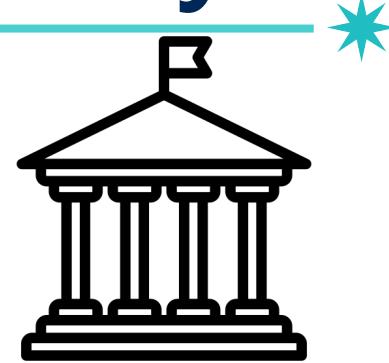
25



Alice



Alice



Certificate
Authority (CA)



Bob



Bob

Key Idea of Public-Key Infrastructure

26



Certificate
Authority (CA)



Alice

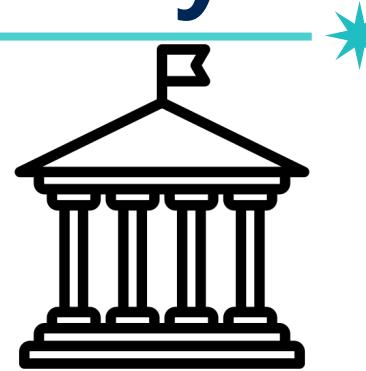
Trusted 3rd-party authority
(KISA, yesSign, Verisign ...)



Bob

Key Idea of Public-Key Infrastructure

27



Certificate
Authority (CA)



Alice

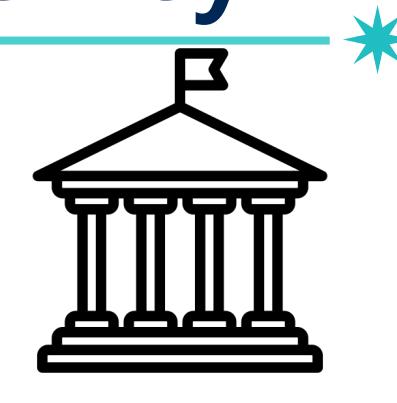
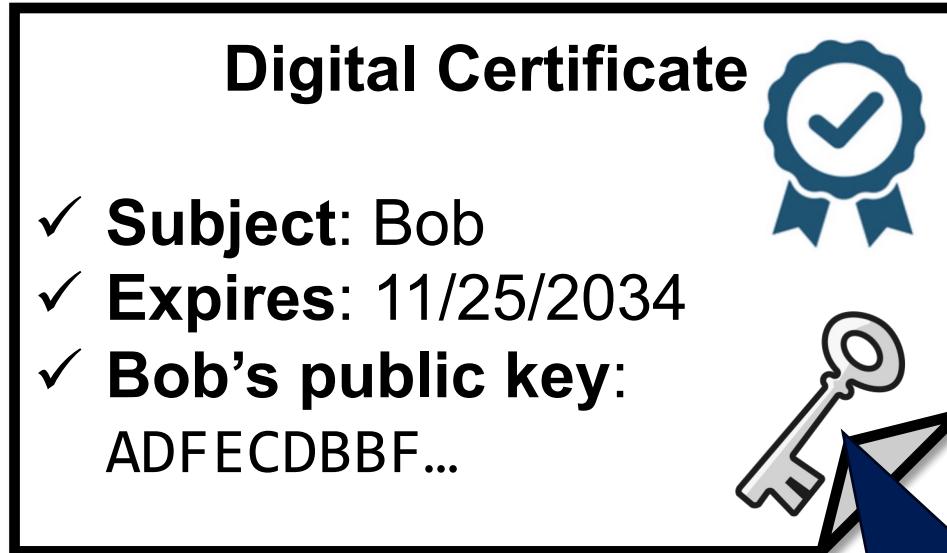
Manage, distribute, verify
public-keys



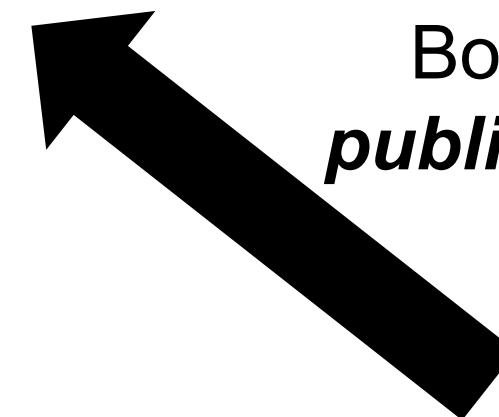
Bob

Key Idea of Public-Key Infrastructure

28



Certificate Authority (CA)



Alice

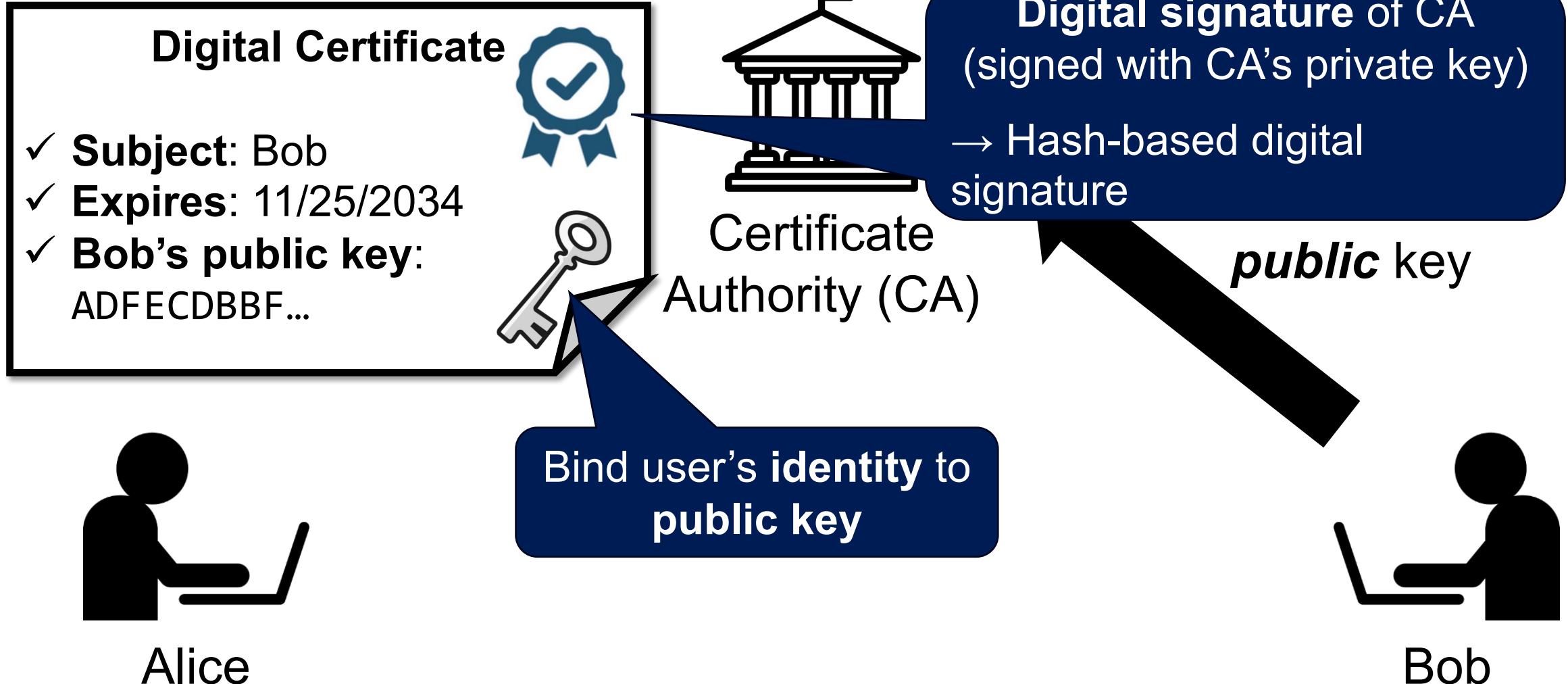
Bind user's **identity** to
public key



Bob

Key Idea of Public-Key Infrastructure

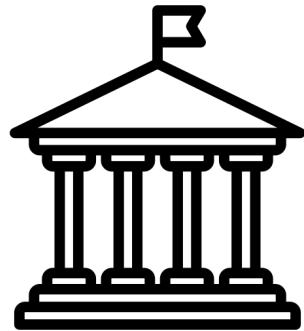
29



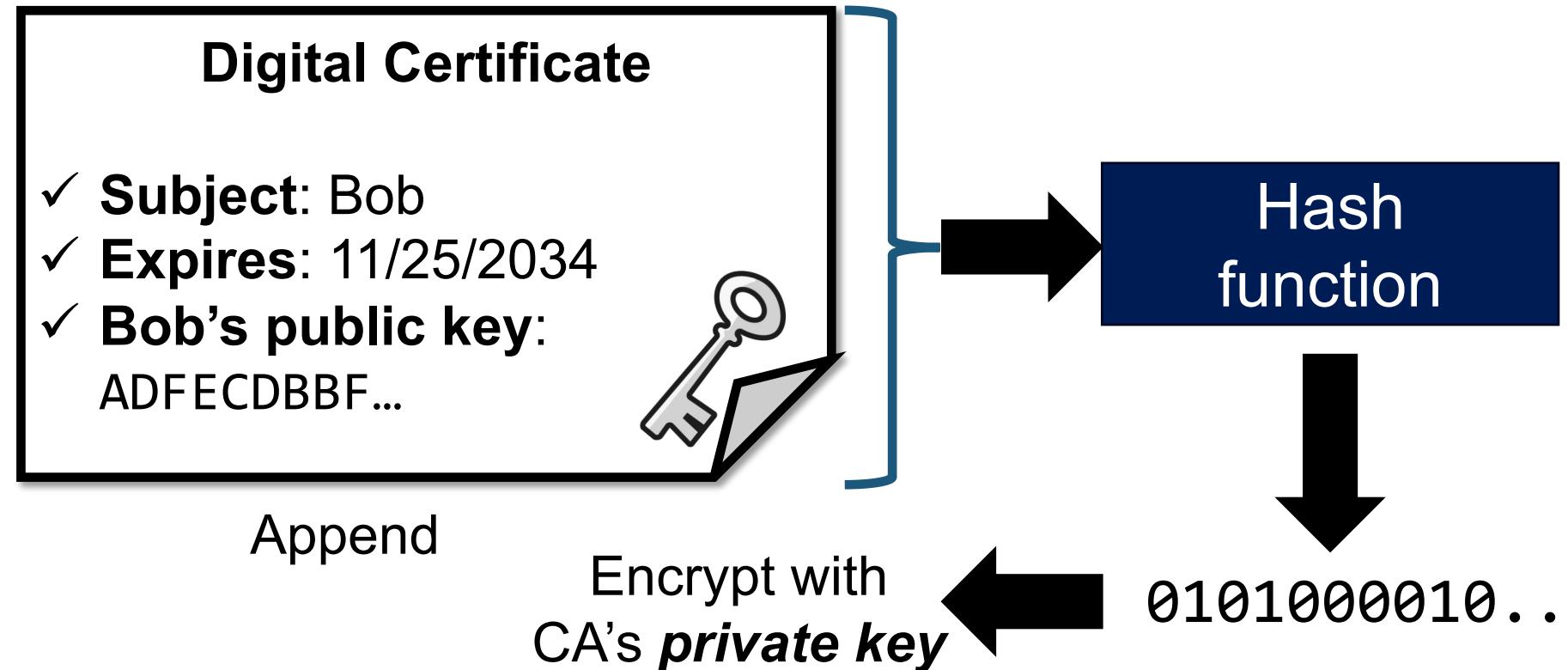
Hash-based Digital Signature in PKI

30

Signing



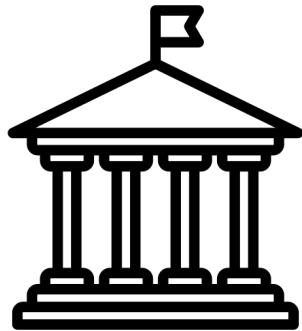
Certificate
Authority (CA)



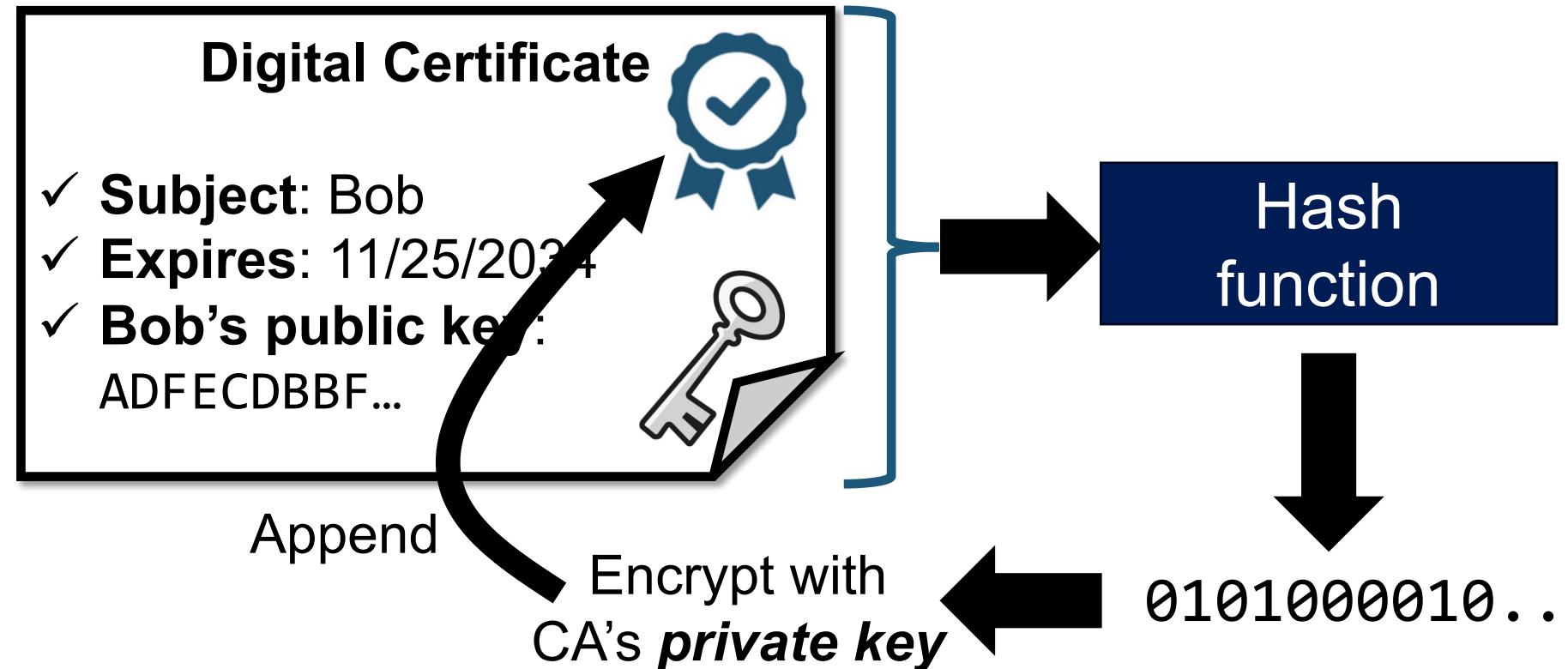
Hash-based Digital Signature in PKI

31

Signing

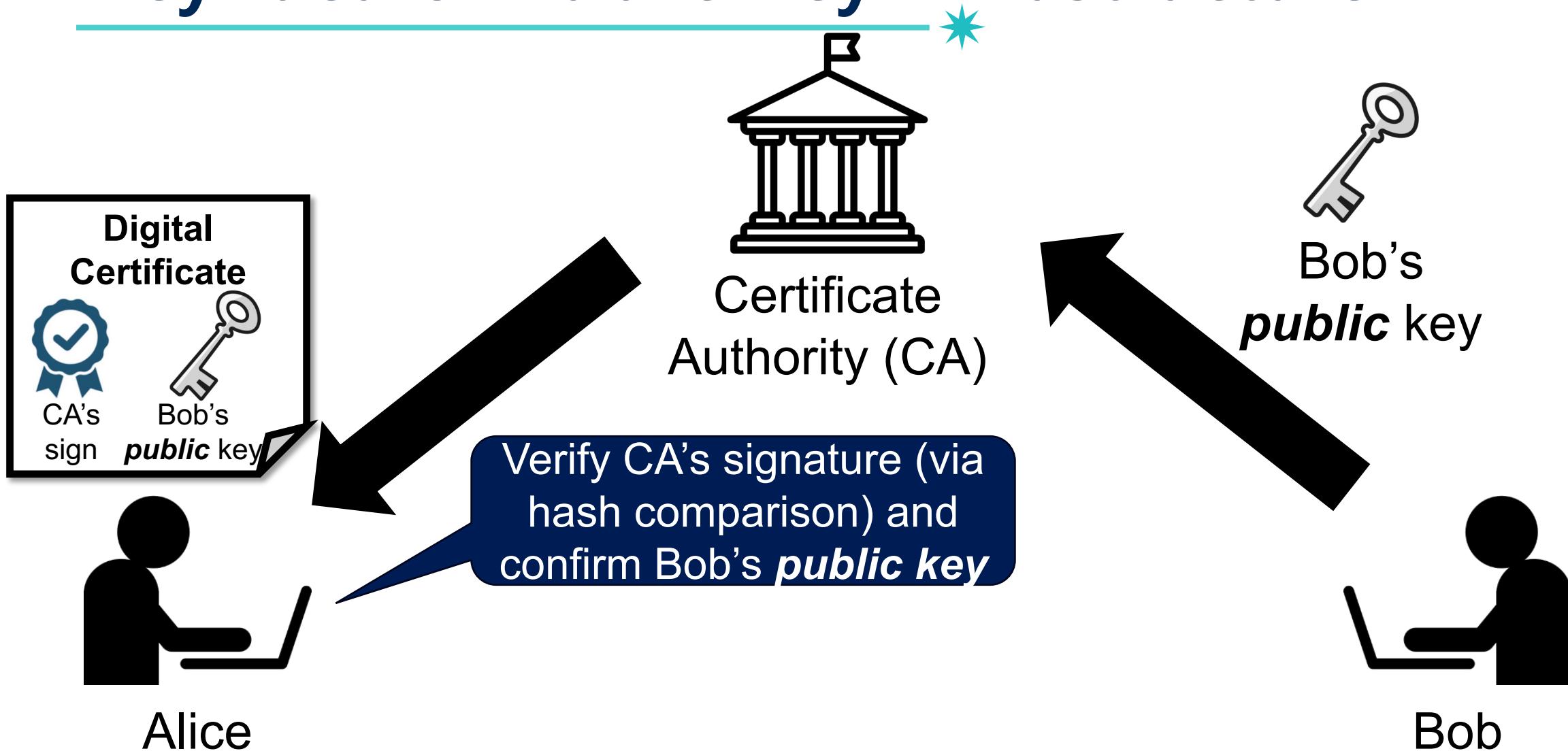


Certificate
Authority (CA)



Key Idea of Public-Key Infrastructure

32



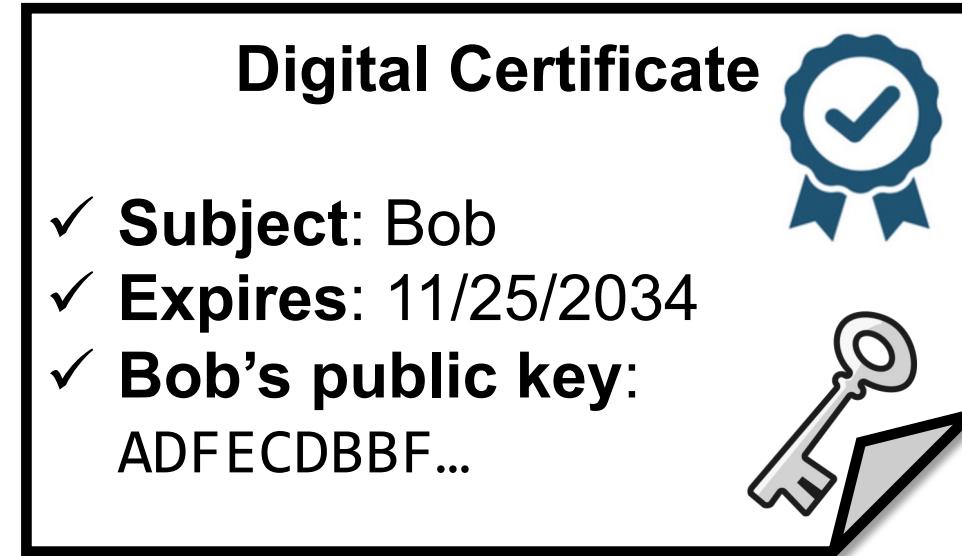
Hash-based Digital Signature in PKI

33

Verification



Alice



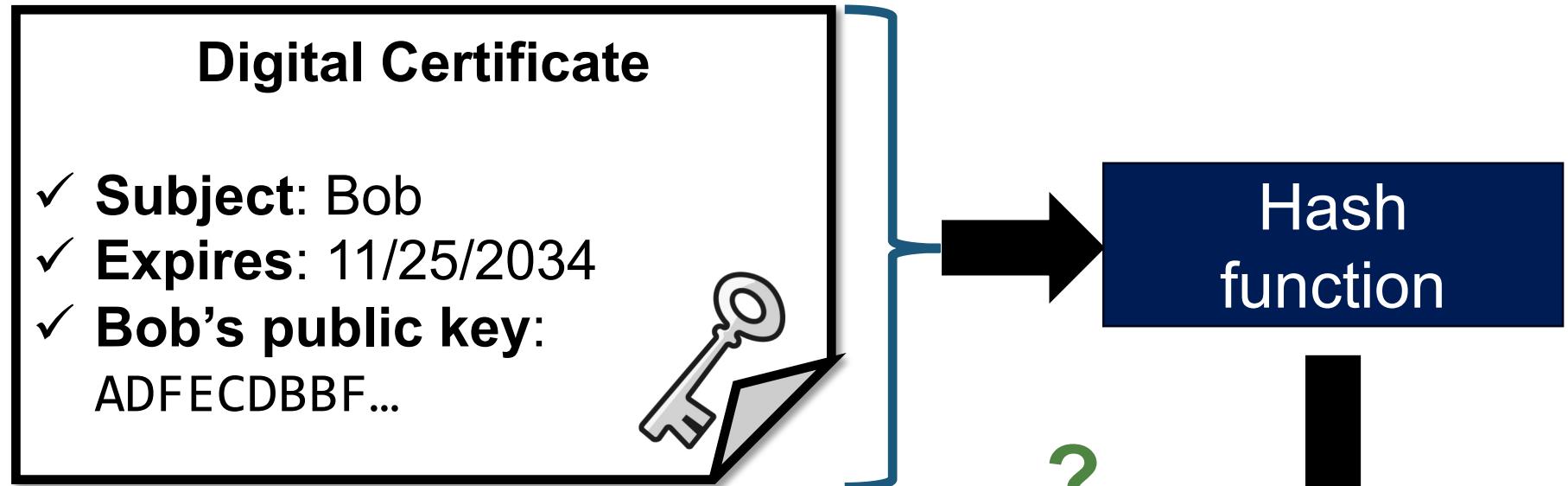
Hash-based Digital Signature in PKI

34

Verification



Alice

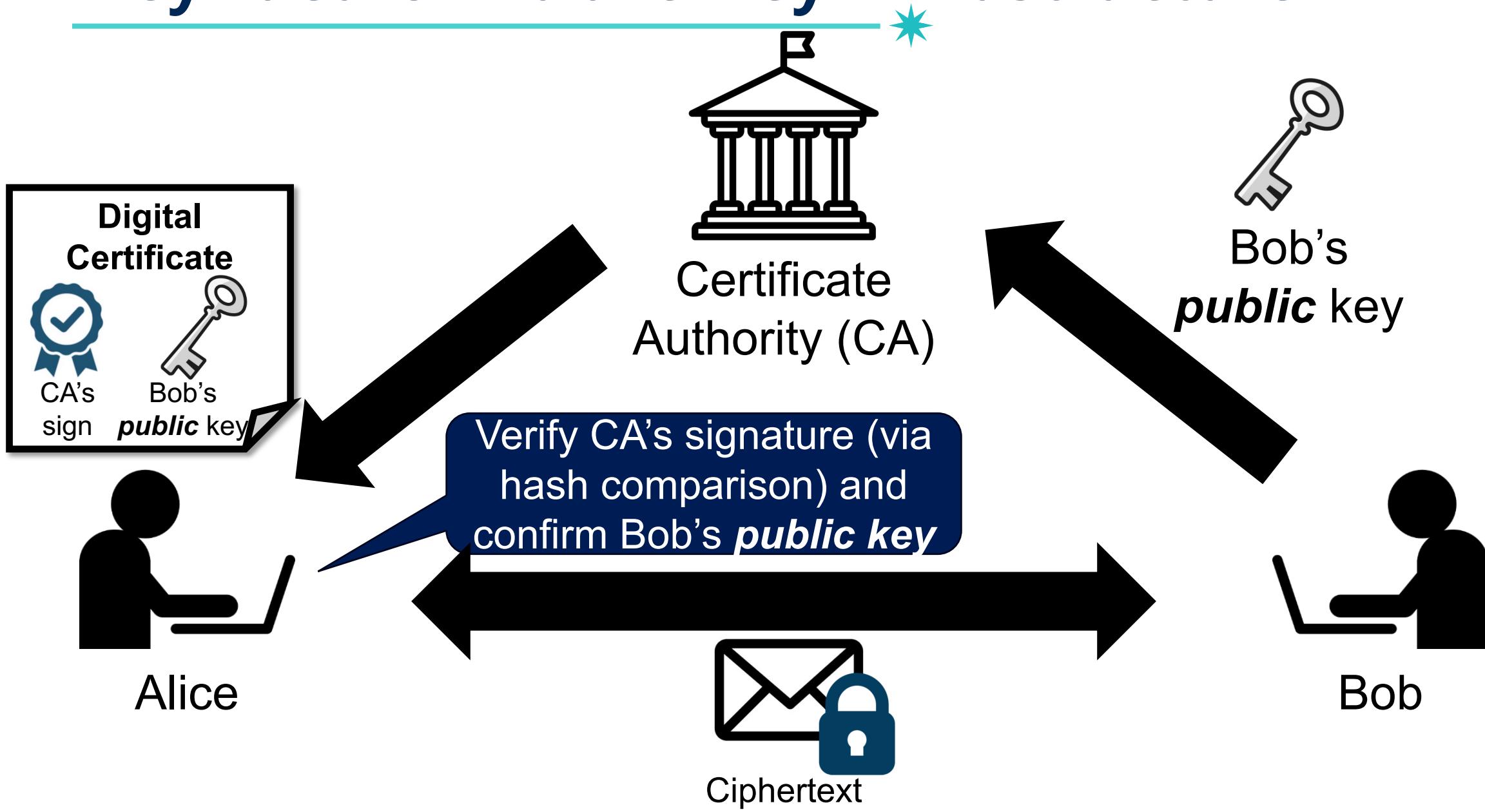


Decrypt with
CA's *public key*

1. Confirm Bob's public key
2. Integrity check

Key Idea of Public-Key Infrastructure

35



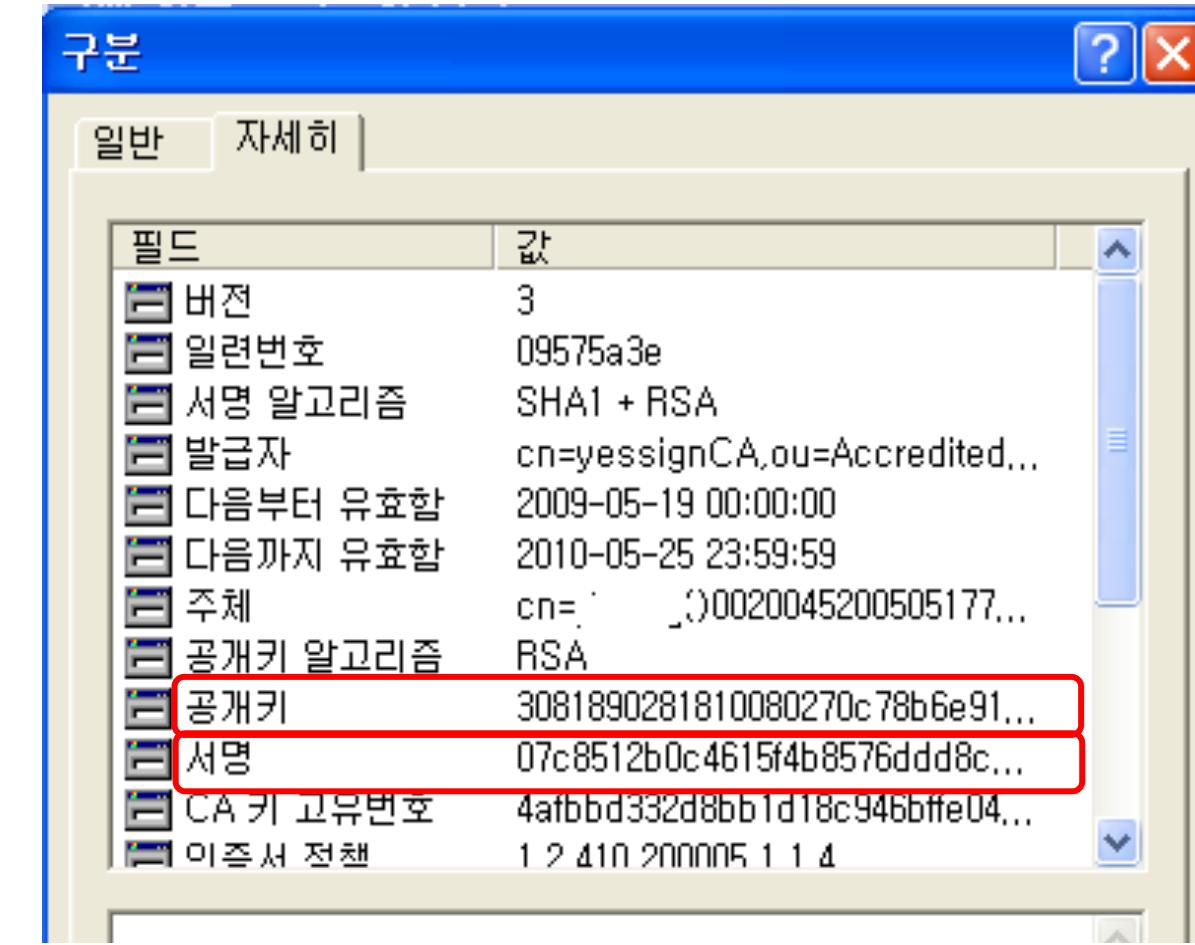
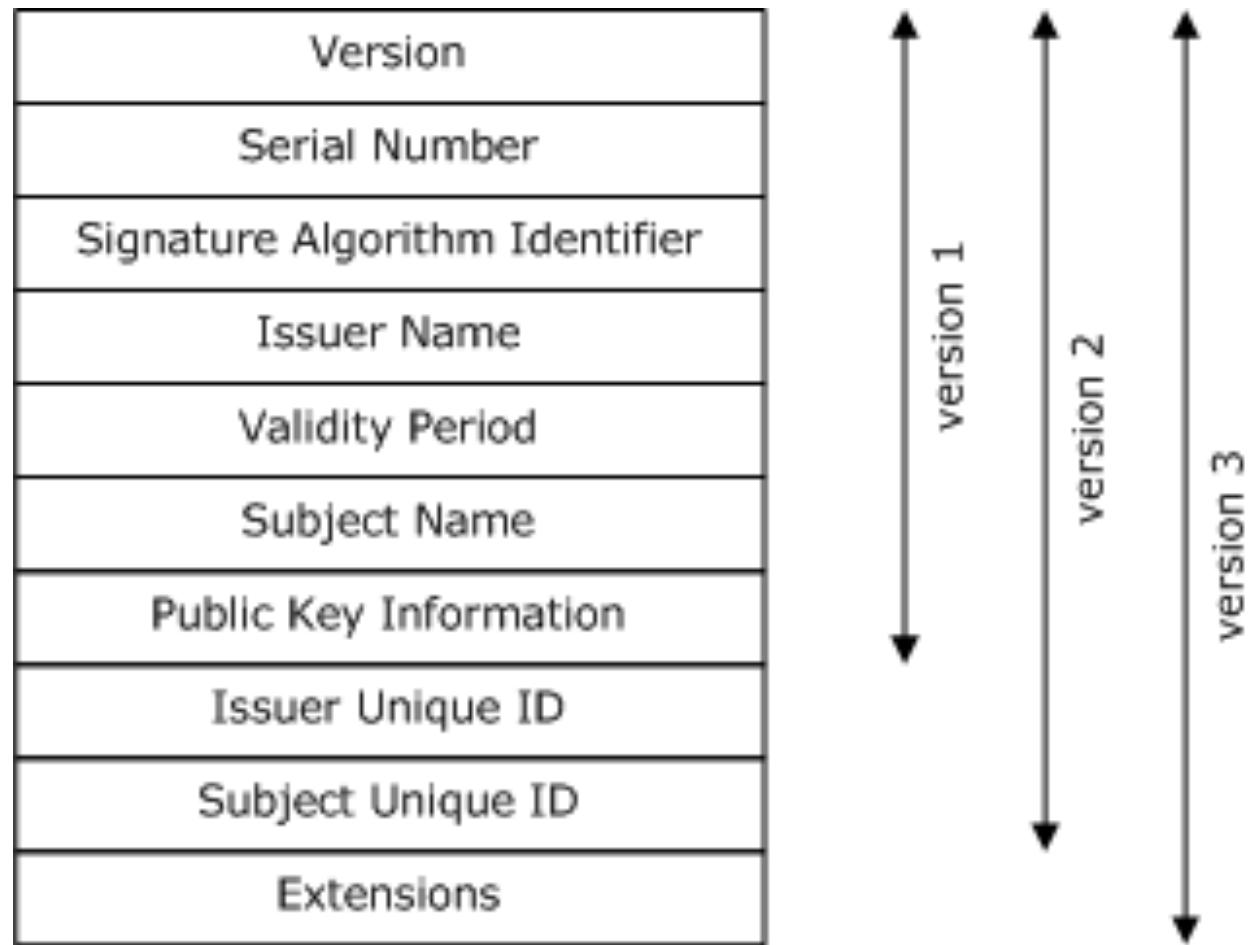
Public-Key Infrastructure (PKI)

36

- The set of processes required to create, manage, distribute, use, store, and revoke **digital certificates and public-keys**
- Two important components
 - **Certificate Authority (CA)**: a **trusted party**, responsible for verifying the identity of users, and then bind the verified identity to a public keys
 - **Digital Certificates**: a document certifying that the public key included inside does belong to the identity described in the document
 - X.509 standard

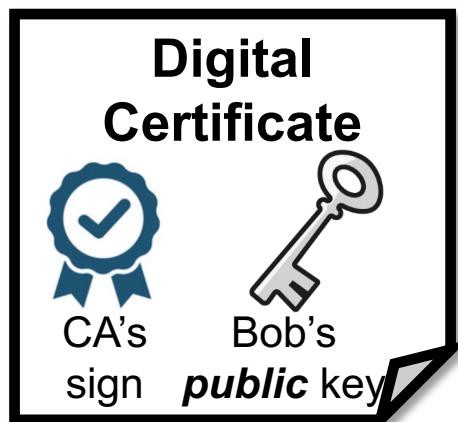
X.509 Certificate

37

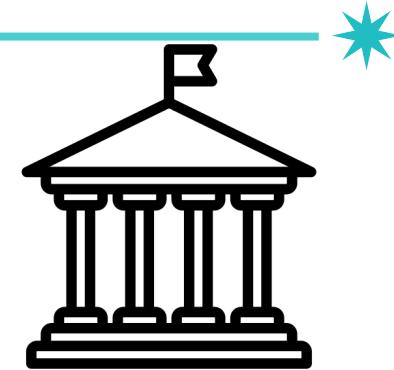


Chain of Trust

38



Alice



Certificate Authority (CA)

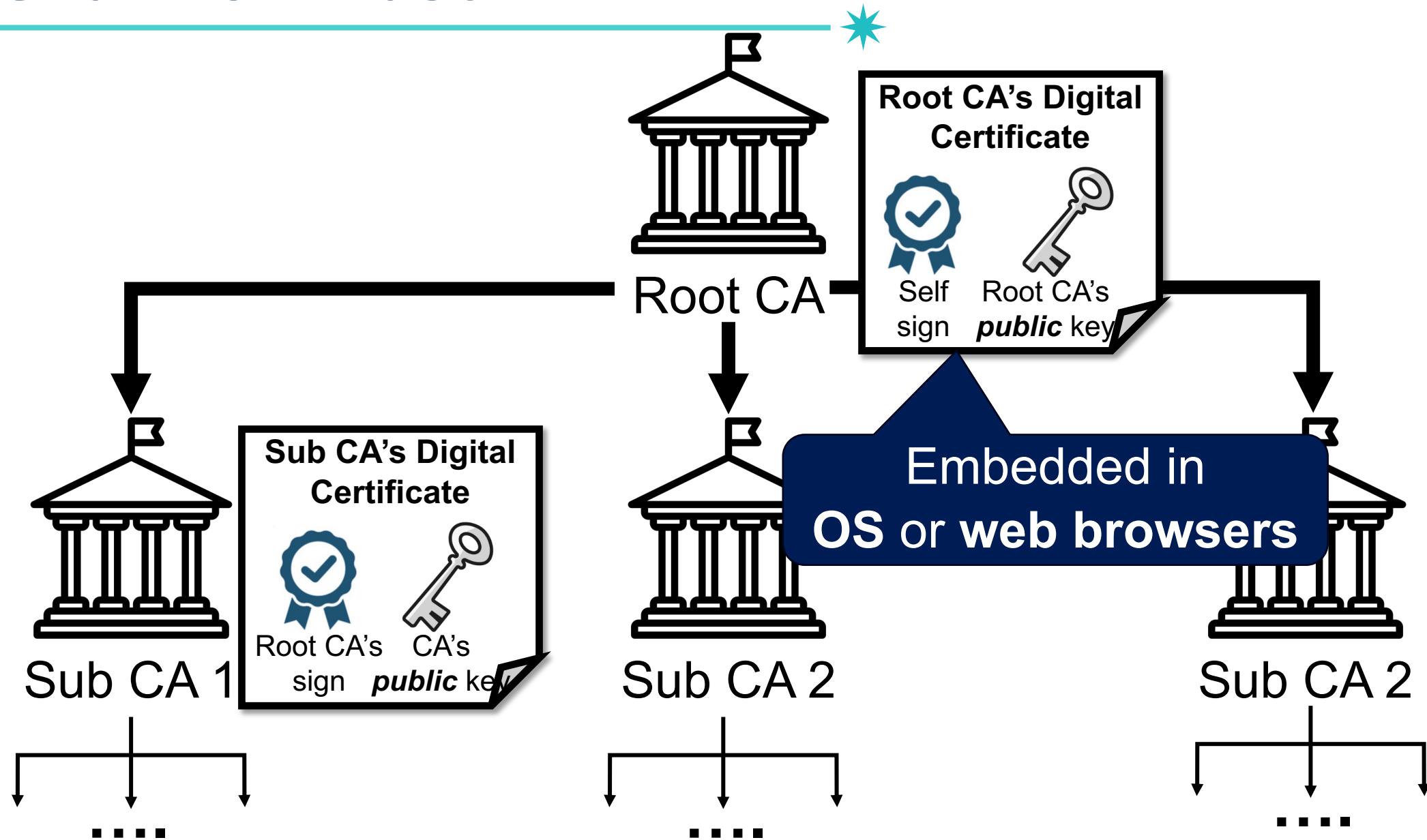


Recursive concern:

How can we trust that the public key belongs to CA?

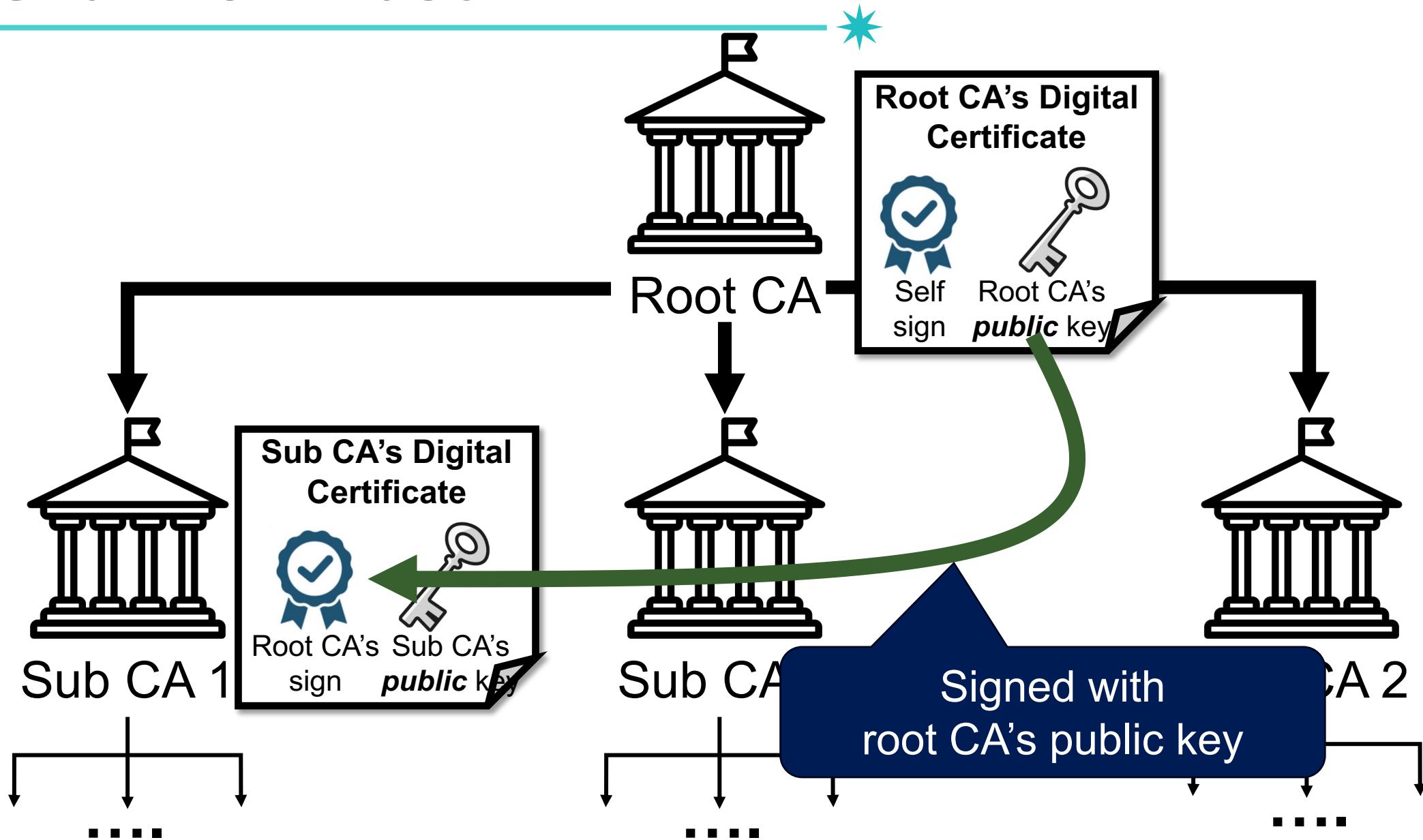
Chain of Trust

39



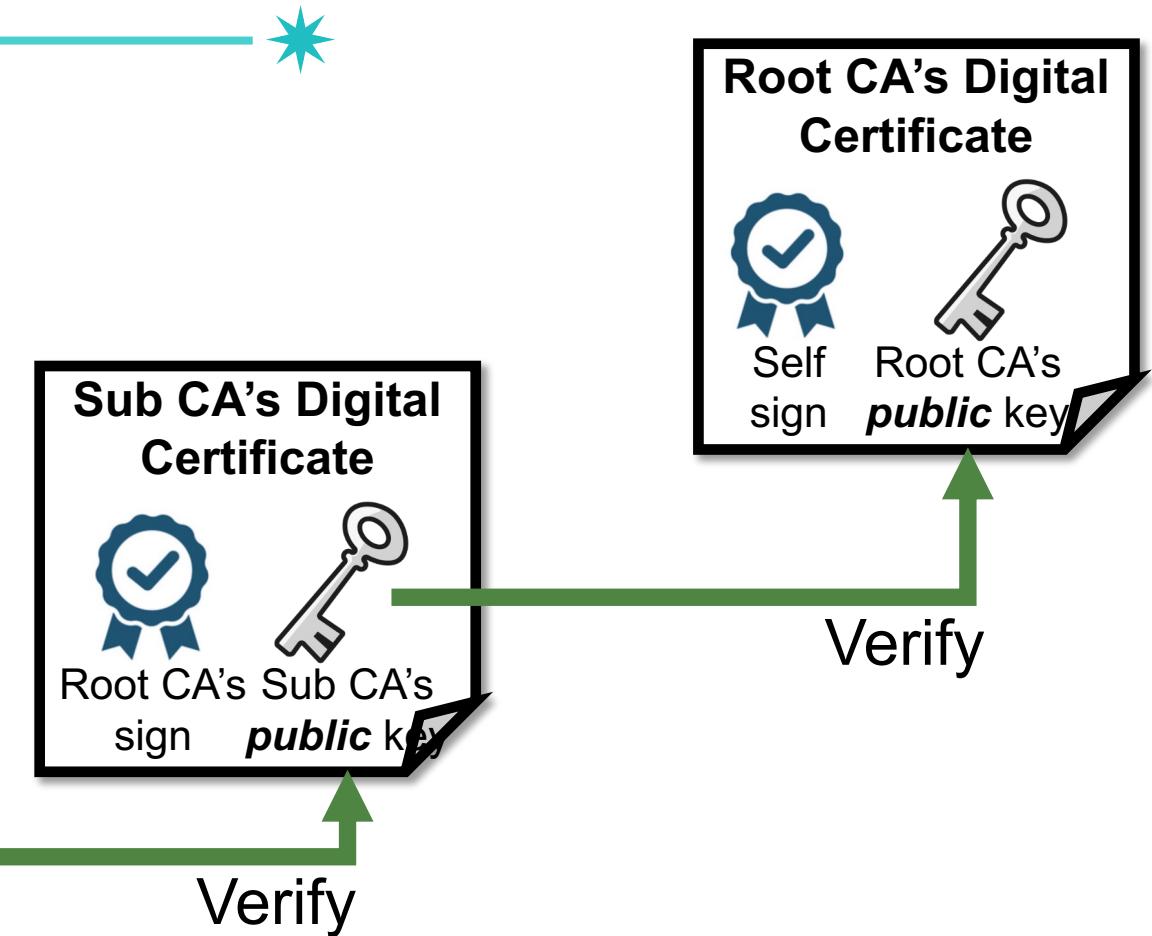
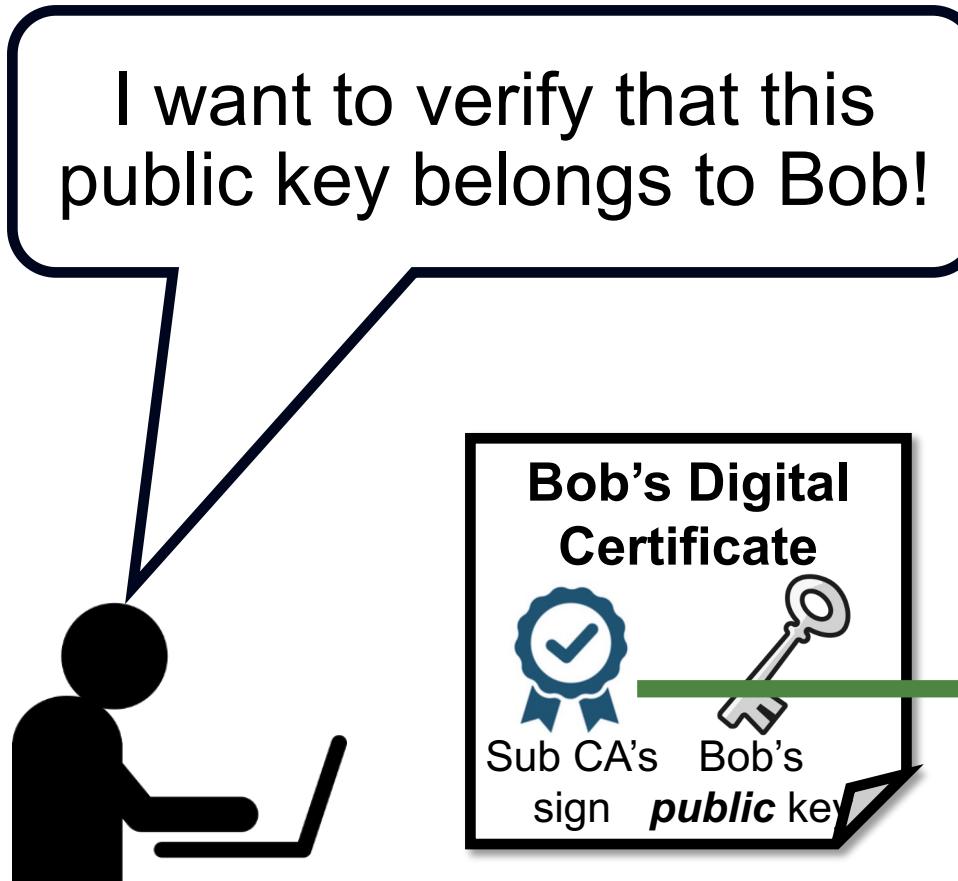
Chain of Trust

40



Chain of Trust

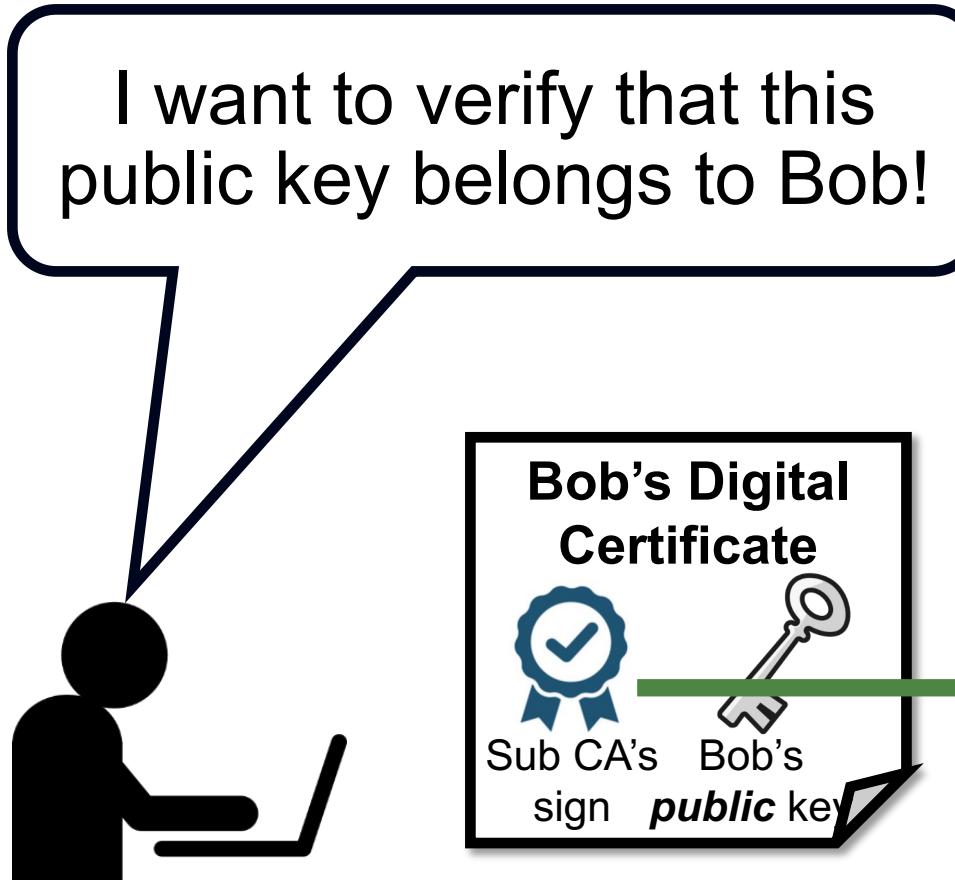
41



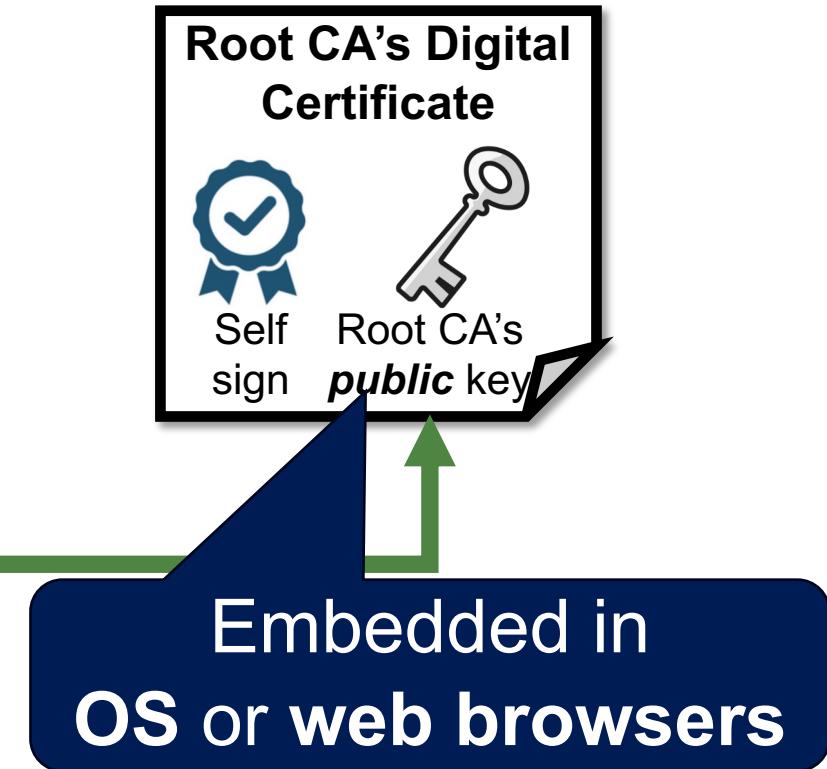
Alice

Chain of Trust

42

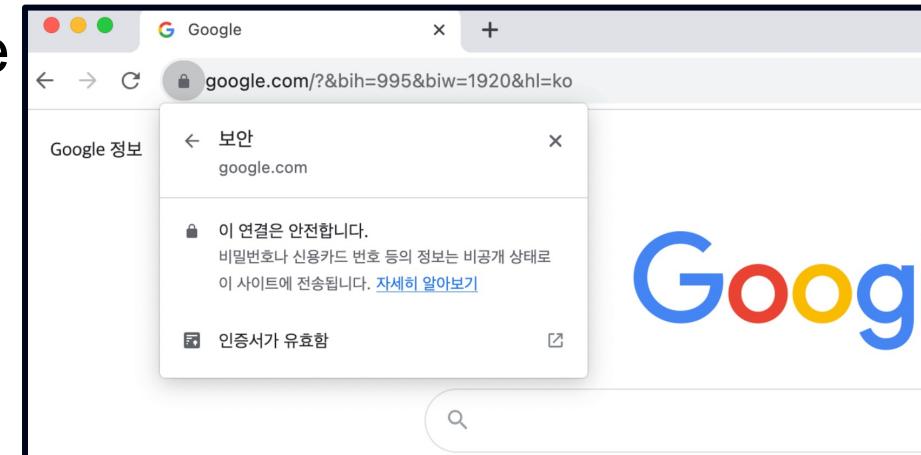


Verify



Certificate Authority and Root CA

- Users need some “Root” keys to start with
 - Root CA
 - Embedded in OS or web browsers
 - (Example #1) Root CAs for iOS: <https://support.apple.com/en-us/HT208125>
 - (Example #2) Chrome



- An example chain of CAs assuring the shinhanca.com:



DigiCert High Assurance EV Root CA



↳ DigiCert SHA2 Extended Validation Server CA



↳ www.shinhanca.com

The Core Functionalities of CA



1. Verify the subject

- Ensure that the person applying for the certificate either owns or represents the identity in the subject field

2. Signing digital certificates

- CA generates a digital signature for the certificate **using its private key**
- Once the signature is applied, the certificate cannot be modified
- Signatures can be verified by anyone with the CA's public key

Digital Certificate



- Let's get paypal's certificates

```
$ openssl s_client -showcerts -connect www.paypal.com:443 </dev/null
```

```
-----BEGIN CERTIFICATE-----
MIIEWTCCBkGgAwIBAgIQLNQEFQ30N5KOSAFavbCfzANBgkqhkiG9w0BAQsFADB3
MQswCQYDVQQGEwJVUzEdMBsGA1UEChMUU3ltYW50ZWMgQ29ycG9yYXRpb24xHzAd
... (omitted) ...
GN/QMQ3a55rjwNQnA3s2WWuHGPaE/jMG17iiL2O/hUdIvLE9+wA+fWrey5//74x1
NeQitYiySDIepHGnng==
-----END CERTIFICATE-----
```

- Save the above data to paypal.pem, and use the following command decode it (see next slide)

```
$ openssl x509 -in paypal.pem -text -noout
```

Example of X.509 Certificate (1st Part)

46

The CA's identity
(Symantec)

The owner of the
certificate
(paypal)

```
Certificate:  
Data:  
    Serial Number:  
        2c:d1:95:10:54:37:d0:de:4a:39:20:05:6a:f6:c2:7f  
    Signature Algorithm: sha256WithRSAEncryption  
    Issuer: C=US, O=Symantec Corporation, OU=Symantec Trust Network,  
        CN=Symantec Class 3 EV SSL CA - G3  
    Validity  
        Not Before: Feb 2 00:00:00 2016 GMT  
        Not After : Oct 30 23:59:59 2017 GMT  
    Subject: 1.3.6.1.4.1.311.60.2.1.3=US/  
        1.3.6.1.4.1.311.60.2.1.2=Delaware/  
        businessCategory=Private Organization/  
        serialNumber=3014267, C=US/  
        postalCode=95131-2021, ST=California,  
        L=San Jose/street=2211 N 1st St,  
        O=PayPal, Inc., OU=CDN Support, CN=www.paypal.com
```

Example of X.509 Certificate (2nd Part)

47

Public key {

Subject Public Key Info:

 Public Key Algorithm: rsaEncryption

 Public-Key: (2048 bit)

 Modulus:

 00:da:43:c8:b3:a6:33:5d:83:c0:63:14:47:fd:6b:22:bd:
 bf:4e:a7:43:11:55:eb:20:8b:e4:61:13:ee:de:fe:c6:e2:
 ... (omitted) ...
 7a:15:00:c5:01:69:b5:10:16:a5:85:f8:fd:07:84:9a:c9:
 Exponent: 65537 (0x10001)

Signature Algorithm: sha256WithRSAEncryption
4b:a9:64:20:cc:77:0b:30:ab:69:50:d3:7f:de:dc:7c:e2:fb:93:84:fd:
78:a7:06:e8:14:03:99:c0:e4:4a:ef:c3:5d:15:2a:81:a1:b9:ff:dc:3a:
... (omitted) ...
fb:00:3e:7d:6a:de:cb:9f:ff:ef:8c:65:35:e4:22:b5:88:b2:48:32:1e:

CA's signature {

Encryption vs Integrity



- “Encryption hides message contents and thus adversary cannot modify the encrypted message” [T / F]?

In many cases, message integrity is equally (or more) important

Recap: Integrity



- Information has not been altered in an unauthorized way
- How to ensure the integrity of computer systems?

Cryptographic
hash function
(e.g., SHA256)

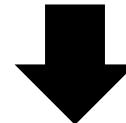
Name	Last modified	Size	Description
Parent Directory		-	
SHA256SUMS	2022-08-11 11:07	202	
SHA256SUMS.gpg	2022-08-11 11:07	833	
ubuntu-22.04.1-desktop-amd64.iso	2022-08-10 16:21	3.6G	Desktop image for 64-bit PC (AMD64) computers (standard download)

Cryptographic Hash Functions

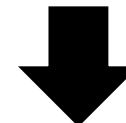


- Condense arbitrary message to fixed size (512 bit...)
- (important!) No **key** for input
- Usually assume hash function is public (e.g., MD5, SHA-512, etc.)

```
01010100100010101000001000010101011010100100001010100  
0001000010101011010100100111110010010001000010101011011  
111010001010100011111010010101001010010101011111...
```



Hash function



```
01011010101001011111111
```

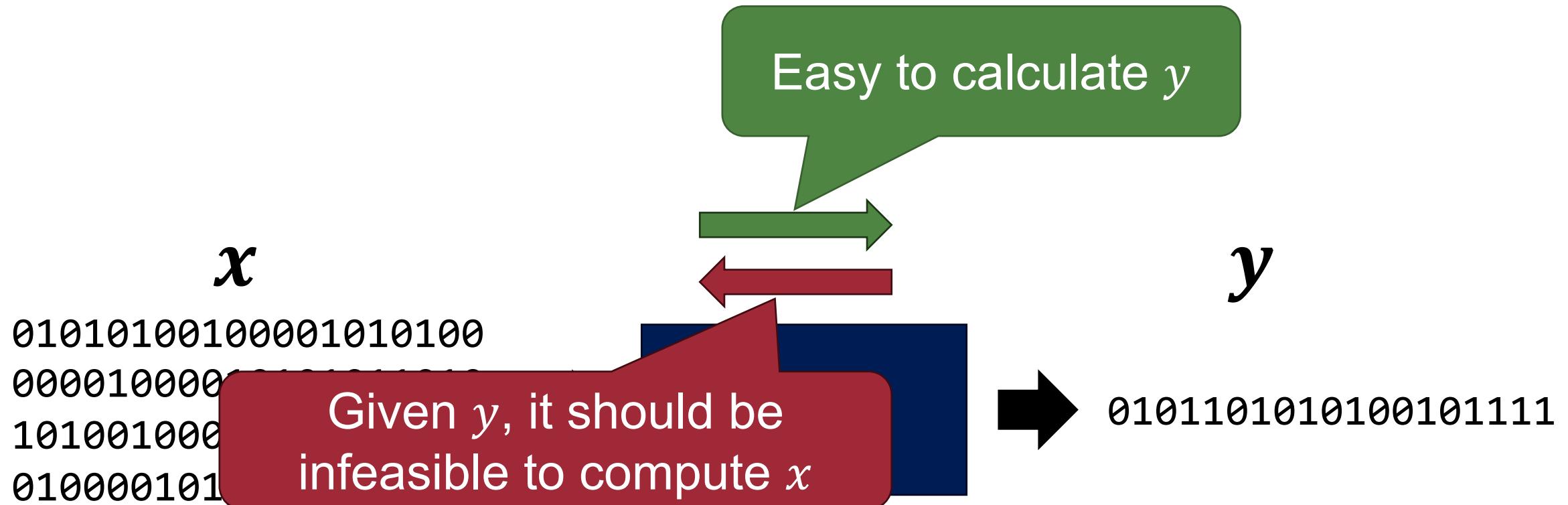
Hash Function Requirements



- 1. Preimage resistant**
- 2. Second preimage resistant**
- 3. Collision resistant**
- 4. Efficiency:** It is relatively easy to compute for any given input.

Property #1: Preimage Resistant

- Given y , computationally infeasible to find x such that $H(x) = y$
 - So-called one-way property



Property #1: Preimage Resistant



- Given y , computationally infeasible to find x such that $H(x) = y$
 - So-called one-way property
- Example:
 - Factoring: $H(x_1, x_2) = x_1 \times x_2$ where x_1, x_2 are prime numbers
 - Discrete logarithm: $H(x) = kx \text{ mod } p$



Application: Password Storage



- Goal: store ID and password pairs to authenticate users
- Bad approach: store ID and password pairs in plaintext to a DB

ID	Password
Batyrkulov	1234abcd
Heesoo	verysecure
Yujin	1234abcd

Application: Hash-based Password Storage

55

- Hashing passwords

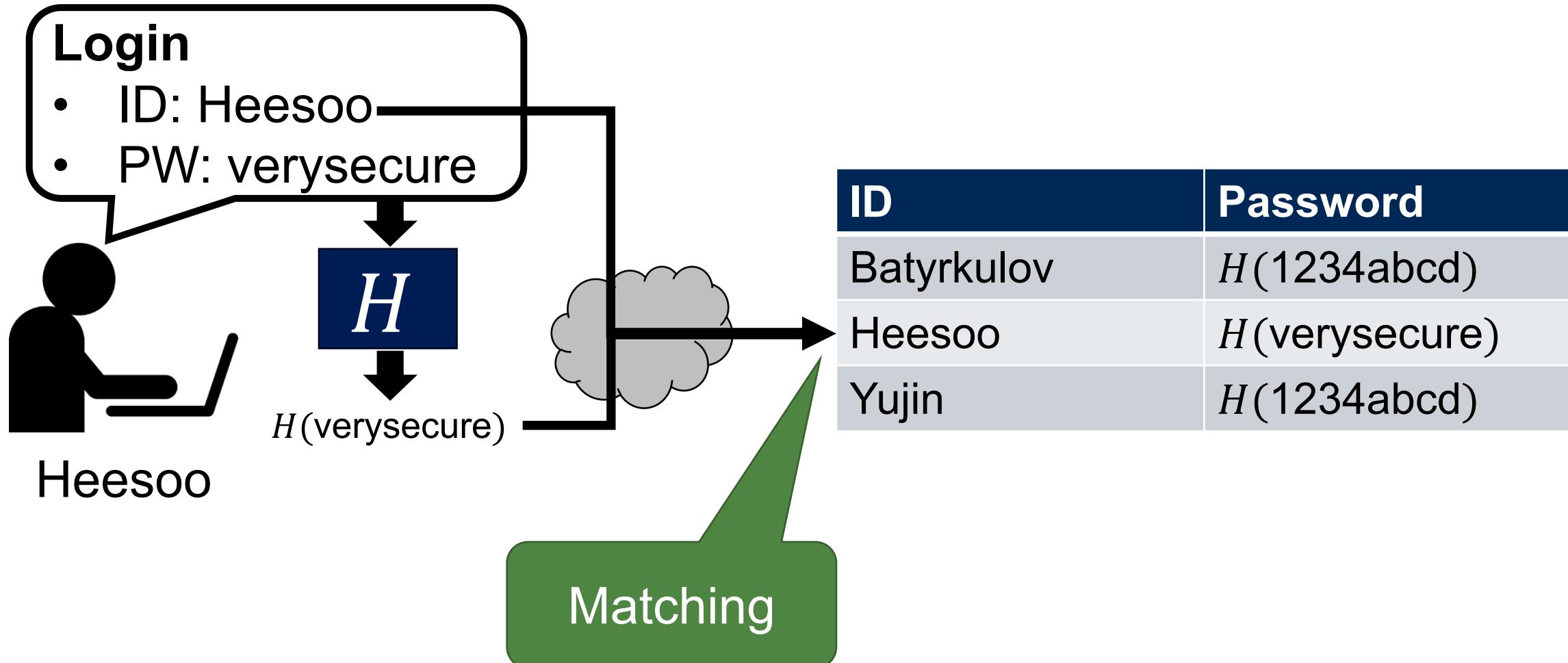
ID	Password
Batyrkulov	$H(1234abcd)$
Heesoo	$H(verysecure)$
Yujin	$I(1234abcd)$

The attacker is not
able to calculate
“verysecure”

Application: Hash-based Password Storage

56

- Hashing passwords



Application: Hash-based Password Storage

57

- Hashing passwords
- BTW, why do we need strong password requirements?

ID	Password
Batyrkulov	$H(1234abcd)$
Heesoo	$H(verysecure)$
Yujin	$H(1234abcd)$

Same password →
Same hash value

Application: Salted Hash



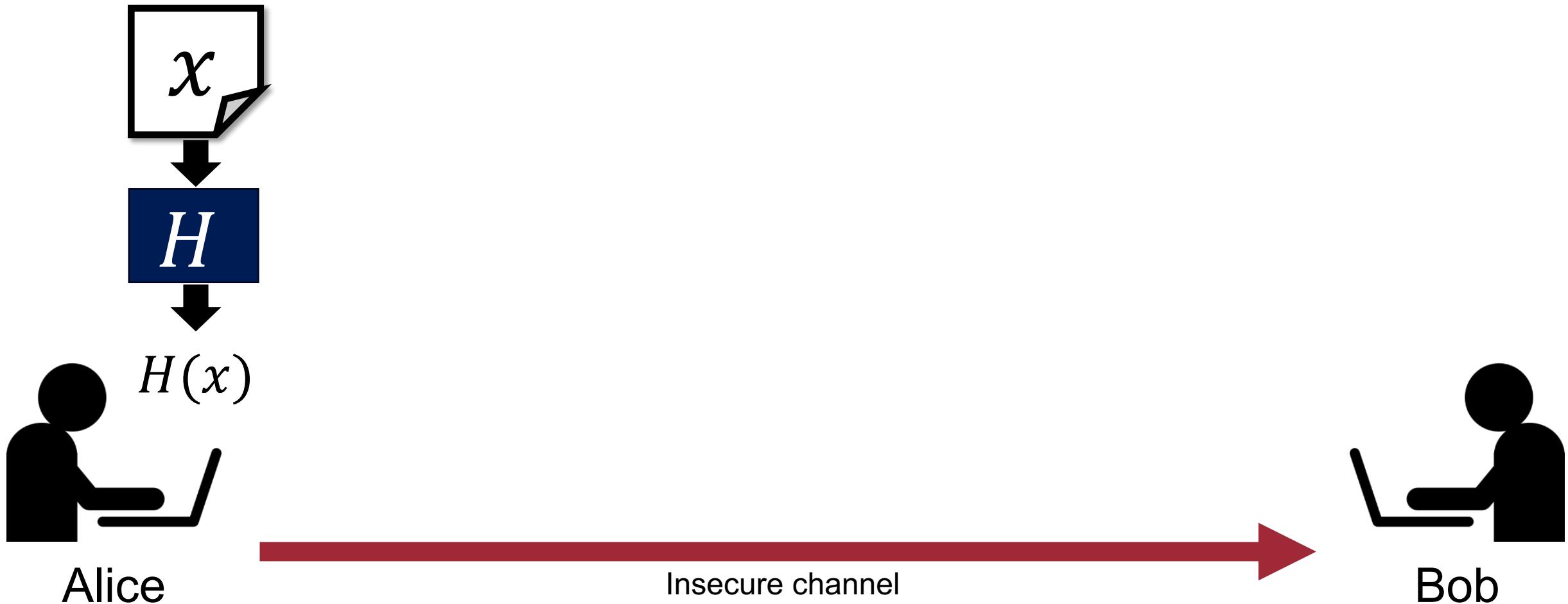
- Hashing passwords
- BTW, why do we need strong password requirements?
=> **Salted Hash**: use a randomly generated number (a salt) to make a hash.

ID	Salt	Password
Batyrkulov	23	$H(1234abcd, 23)$
Heesoo	51	$H(verysecure, 51)$
Yujin	97	$H(1234abcd, 97)$

Property #2: Second Preimage Resistance

59

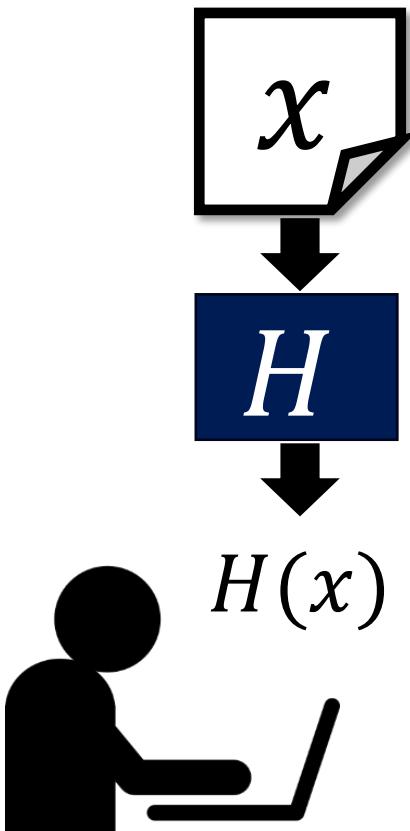
- Given x , computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



Property #2: Second Preimage Resistance

60

- Given x ,
 $H(x) = H(z)$



Ubuntu 22.04.1 LTS (Jammy Jellyfish)

$x \neq z$ and

A full list of available files, including BitTorrent files, can be found below.
If you need help burning these images to disk, see the [Image Burning Guide](#).

Name	Last modified	Size	Description
Parent Directory		-	
SHA256SUMS	2022-08-11 11:07	20	
SHA256SUMS.gpg	2022-08-11 11:07	835	
ubuntu-22.04.1-desktop-amd64.iso	2022-08-10 16:21	3.6G	Desktop image for computers (starters)

Expect a message
with hash value $H(x)$

Insecure channel



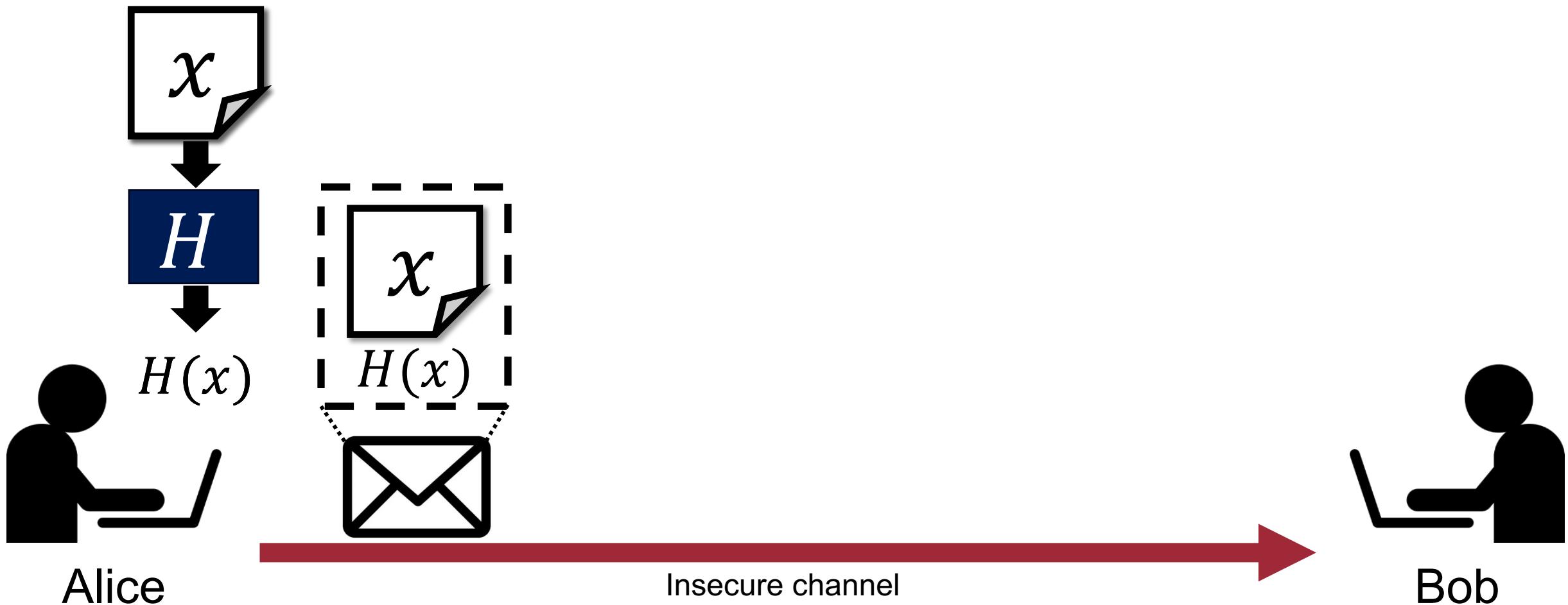
Alice

Bob

Property #2: Second Preimage Resistance

61

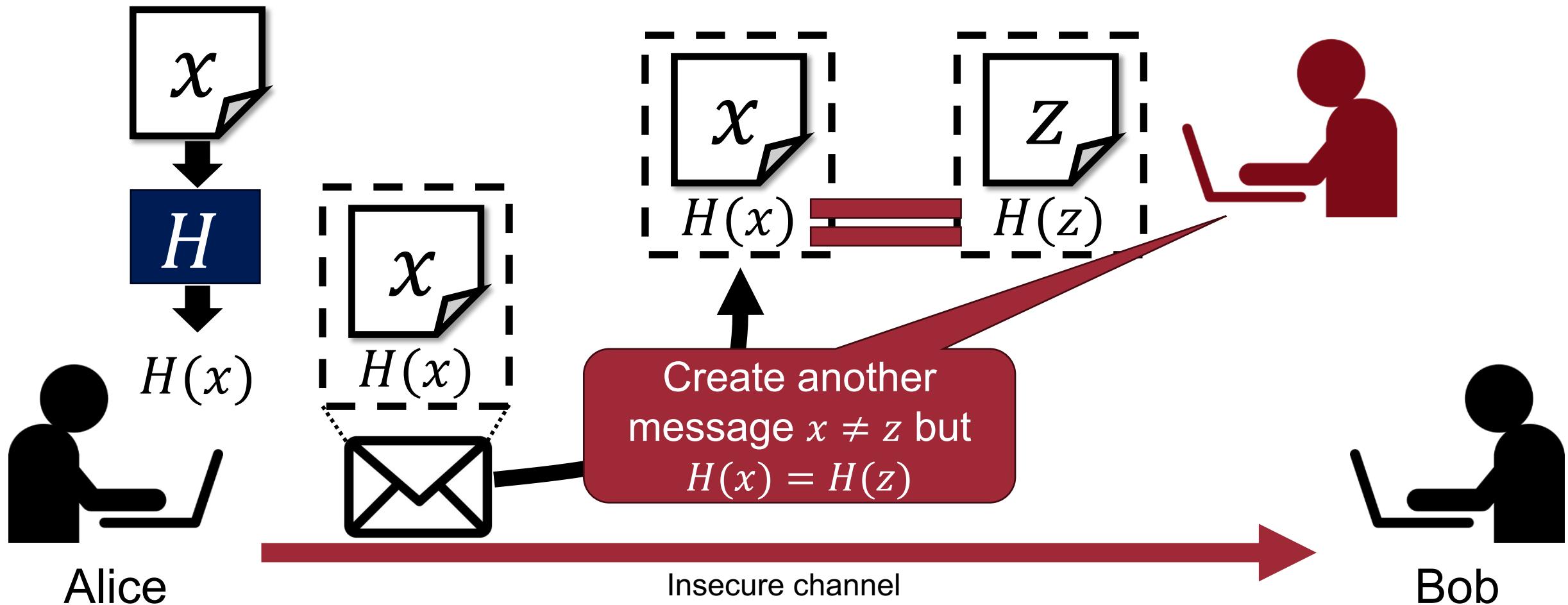
- Given x , computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



Property #2: Second Preimage Resistance

62

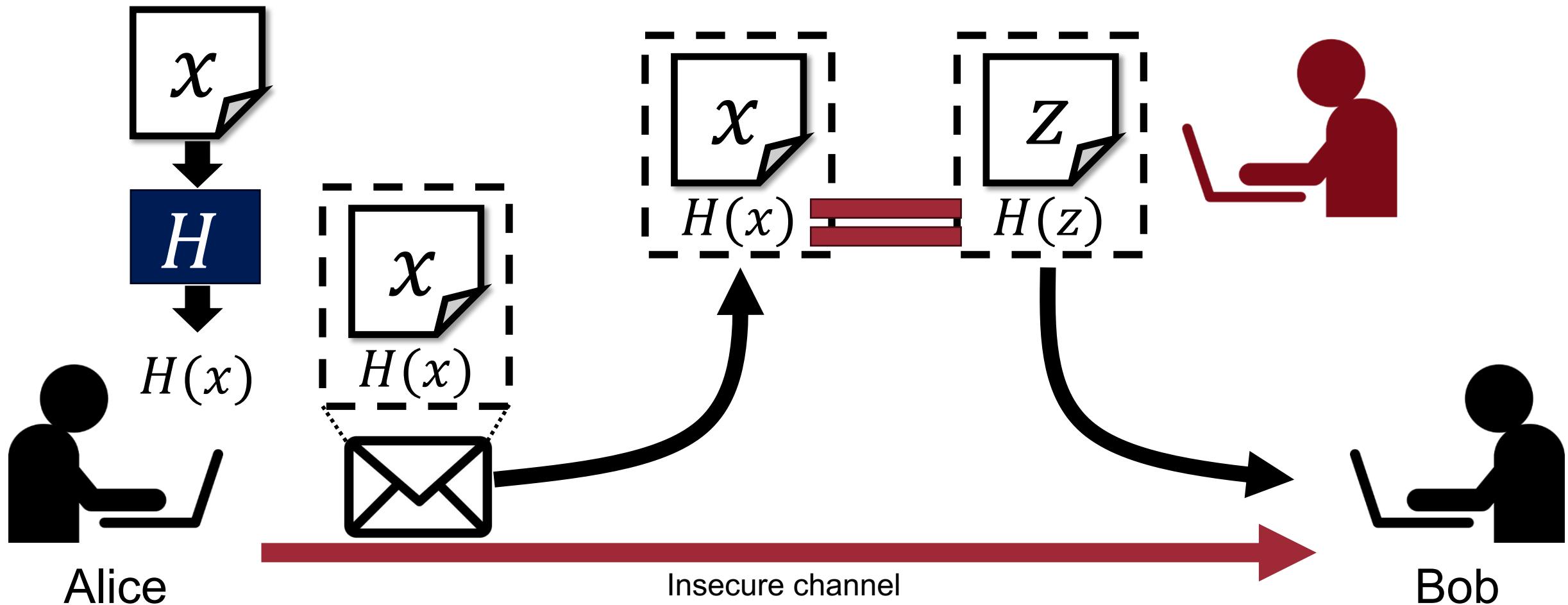
- Given x , computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



Property #2: Second Preimage Resistance

63

- Given x , computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



Property #2: Second Preimage Resistant

64

- Given x , computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$
- Example: integrity of software distribution, fingerprinting (e.g., virus, deduplication)

Property #3: Collision Resistant

65

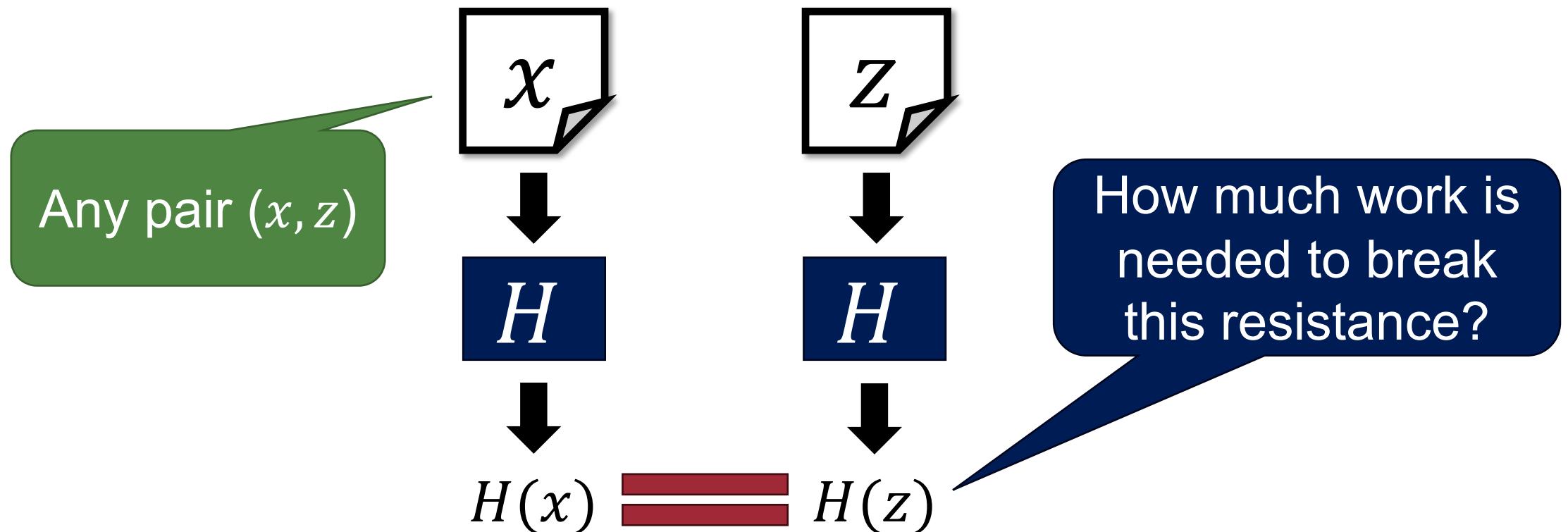
- Computationally infeasible to find **any pair** (x, z) such that $x \neq z$ and $H(x) = H(z)$



Property #3: Collision Resistant

66

- Computationally infeasible to find **any pair** (x, z) such that $x \neq z$ and $H(x) = H(z)$



Birthday Paradox

67



How many people must be in a group, such that there is more than 50% probability that at least two of them have the same birthday?

=> 23 people

Birthday Paradox

68

Probability that in a set of n random people, at least two will share a birthday

- Find n such that $p(n) \geq 0.5$
 - # of people in the group: n
 - A year has 365 days
- $p(n) = 1 - \bar{p}(n)$

Probability that all n people have different birthdays

Birthday Paradox

69

Probability that in a set of n random people, at least two will share a birthday

- Find n such that $p(n) \geq 0.5$
 - # of people in the group: n
 - A year has 365 days
- $p(n) = 1 - \bar{p}(n)$
- If $n = 3$,

Probability that all n people have different birthdays

$$P(3) = 1 - \left(\frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \right) = 1 - 0.9917 = 0.0083$$

Birthday Paradox

70

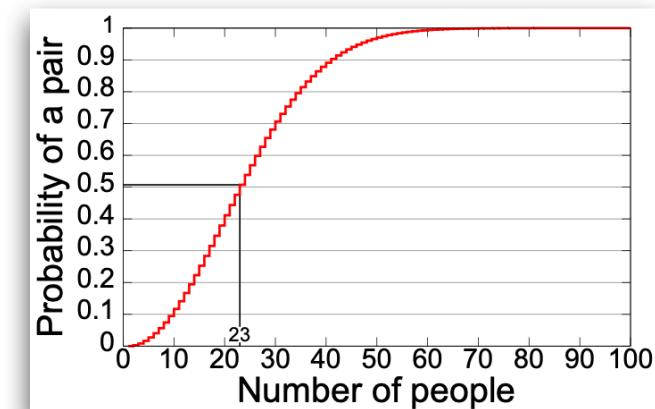
Probability that in a set of n random people, at least two will share a birthday

- Find n such that $p(n) \geq 0.5$
 - # of people in the group: n
 - A year has 365 days
- $p(n) = 1 - \bar{p}(n)$
- If $n = 3$,

$$P(3) = 1 - \left(\frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \right) = 1 - 0.9917 = 0.0083$$

$$\begin{aligned}\bar{p}(n) &= 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right) \\ &= \frac{365 \times 364 \times \cdots \times (365 - n + 1)}{365^n}\end{aligned}$$

Probability that all n people have different birthdays



Birthday Paradox

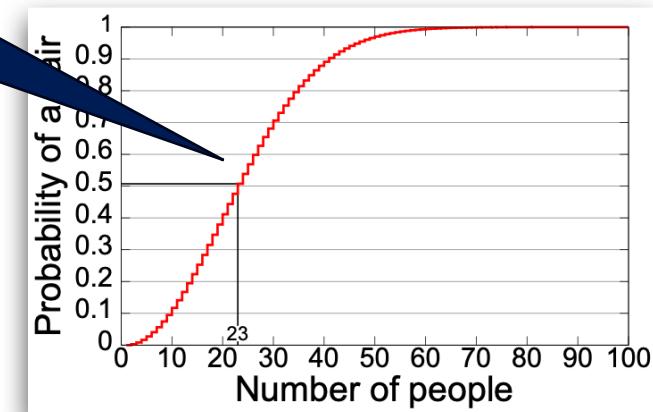
- Find n such that $p(n) \geq 0.5$
 - # of people in the group: n
 - A year has 365 days
- $p(n) = 1 - \bar{p}(n)$
- If $n = 3$,

P

If we put $n = 23$, the probability
is 50.7%

$$\begin{aligned}\bar{p}(n) &= 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right) \\ &= \frac{365 \times 364 \times \cdots \times (365 - n + 1)}{365^n}\end{aligned}$$

$- 0.9917 = 0.0083$



Property #3: Collision Resistant



- Computationally infeasible to find **any pair** (x, z) such that $x \neq z$ and $H(x) = H(z)$
- Birthday attack: If we have an m bit hash value, $2^{m/2}$ **work** is needed to break collision resistant (not 2^m , birthday paradox)
 - To ensure security against 2^n attacks, the hash output length must be $2n$ -bits.

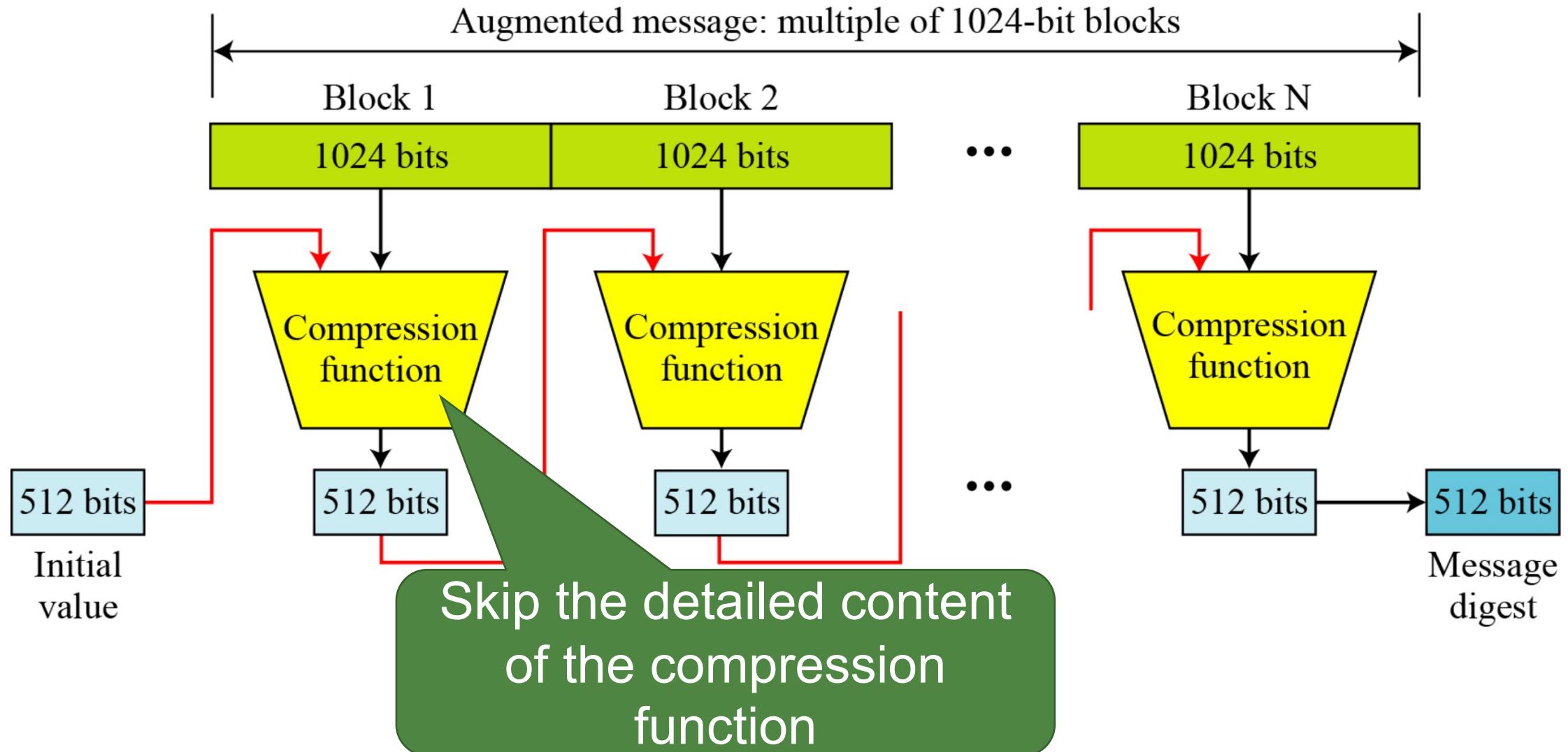
Hash Function Standards

73

- MD5
 - Pairs of collisions reported
 - Still used for simple data diffing
- SHA-1
 - Pairs of collisions reported
 - Broken
- SHA-256, SHA-384, SHA-512 (message digest size)

SHA-512 Overview

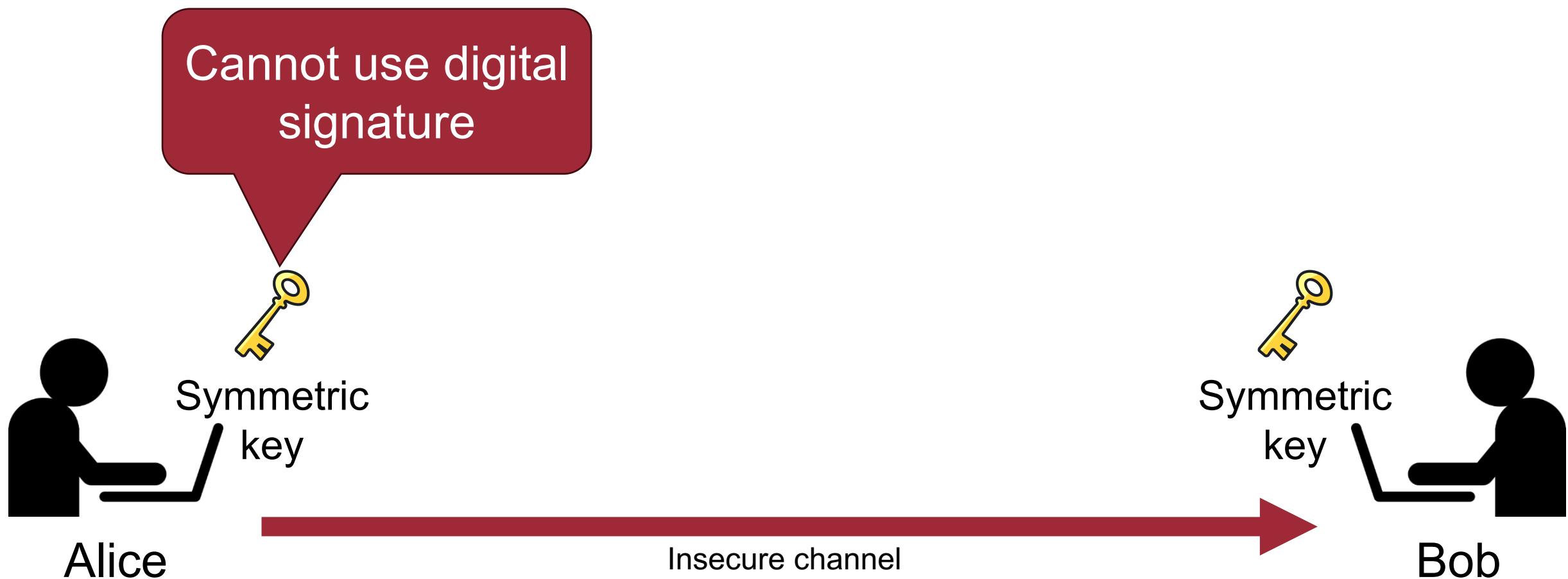
74



Message Authentication Codes (MAC)

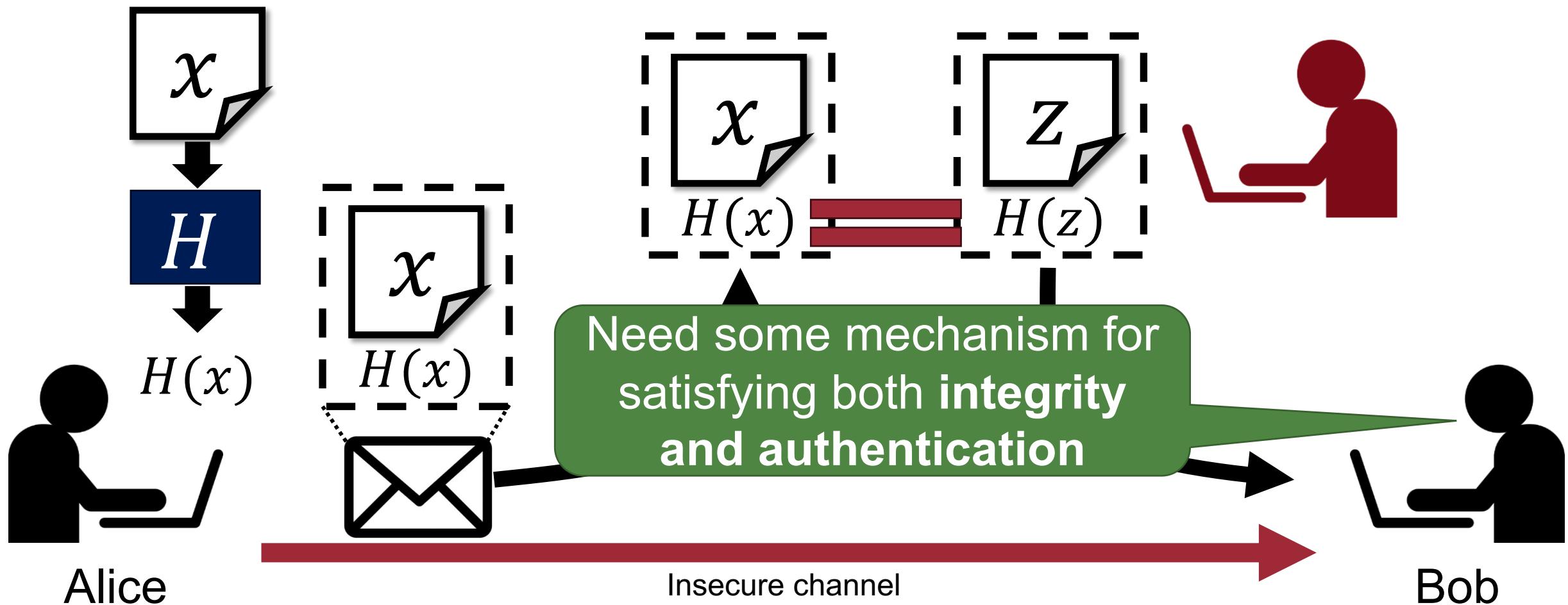
75

- “Cryptographic checksum” to ensure the **integrity** of the message and the data origin **authentication** (in symmetric-key cryptography)



Recap: Second Preimage Resistant

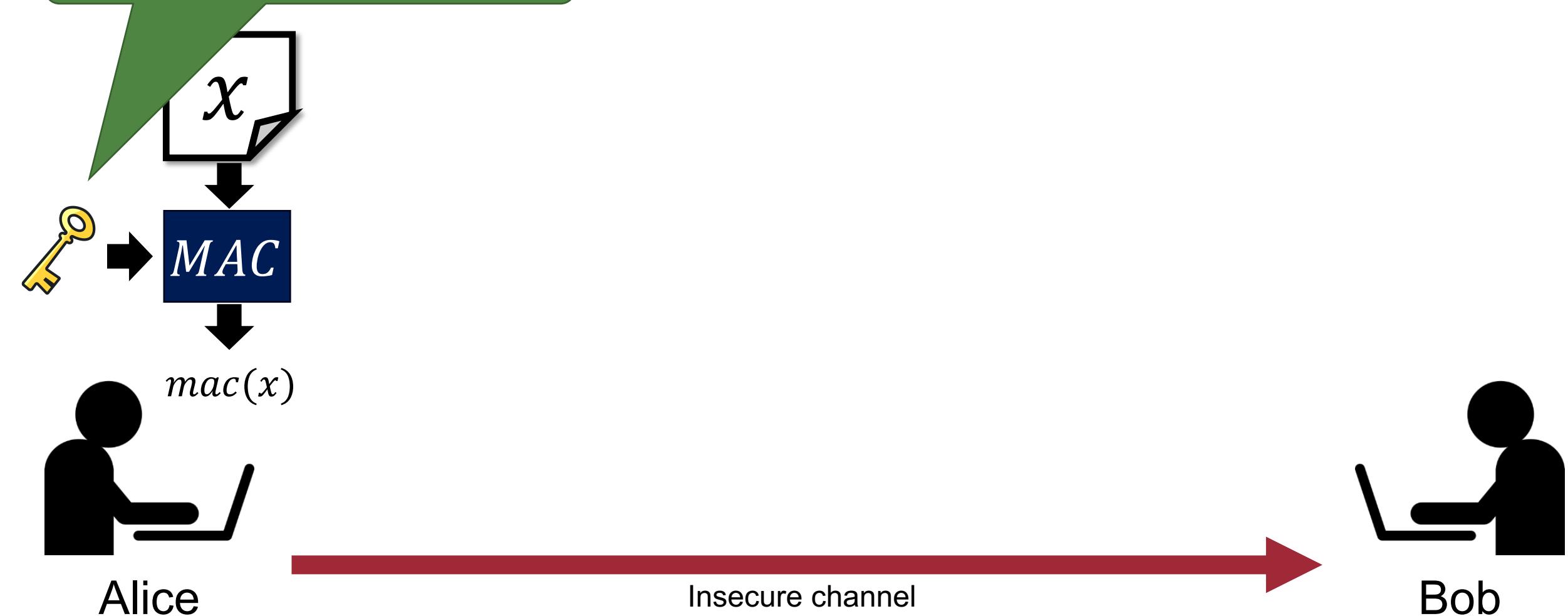
76



Message Authentication Codes (MAC)

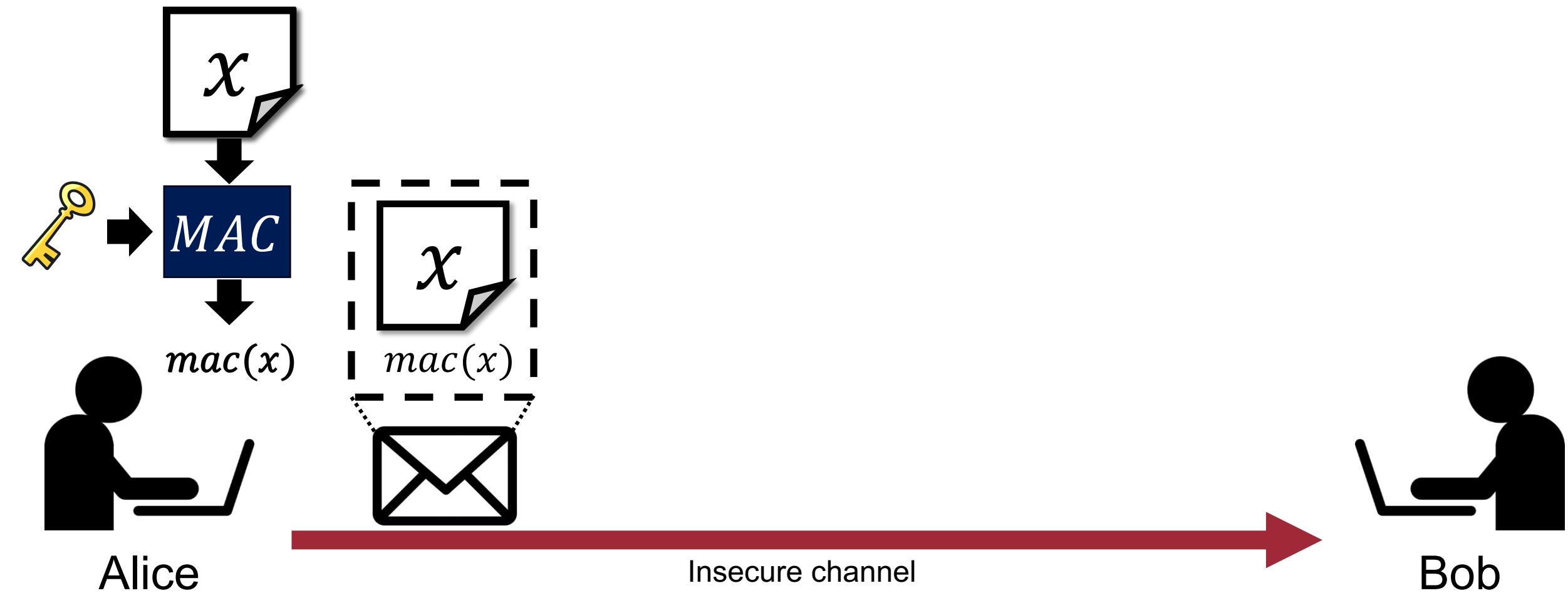
77

Use the symmetric key!



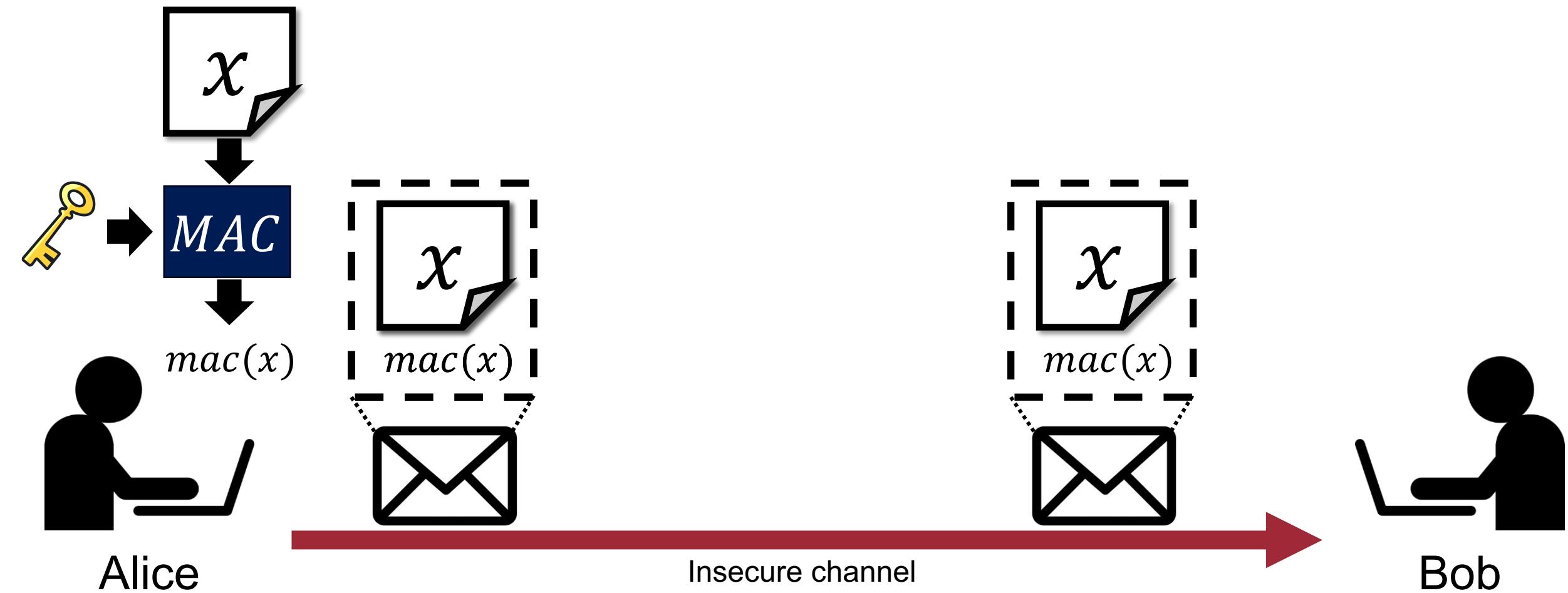
Message Authentication Codes (MAC)

78



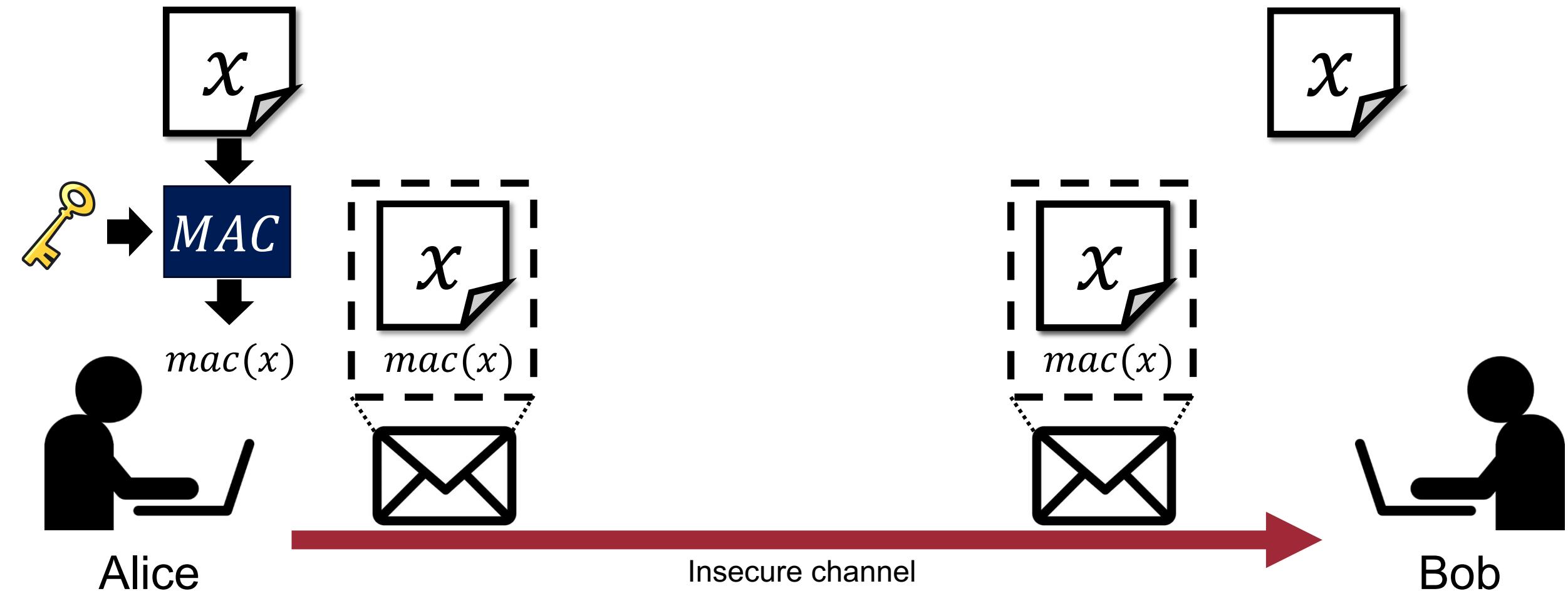
Message Authentication Codes (MAC)

79



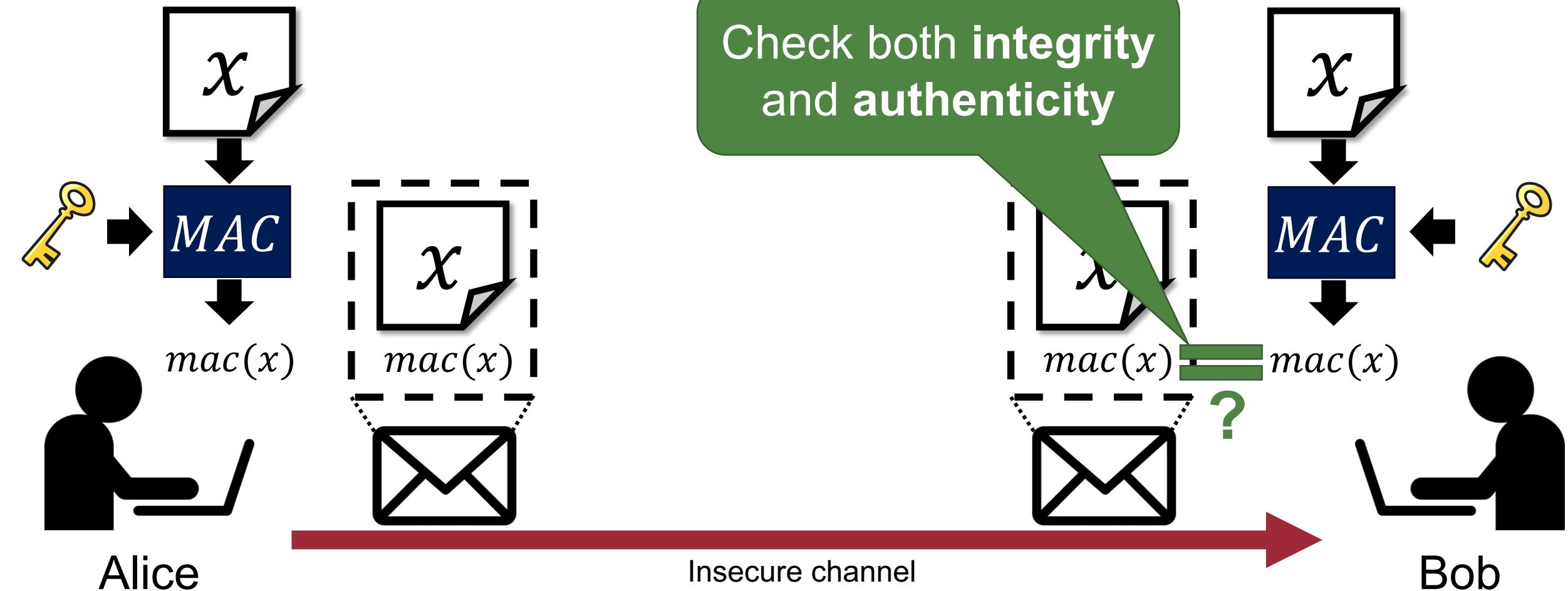
Message Authentication Codes (MAC)

80



Message Authentication Codes (MAC)

81



Message Authentication Codes (MAC)

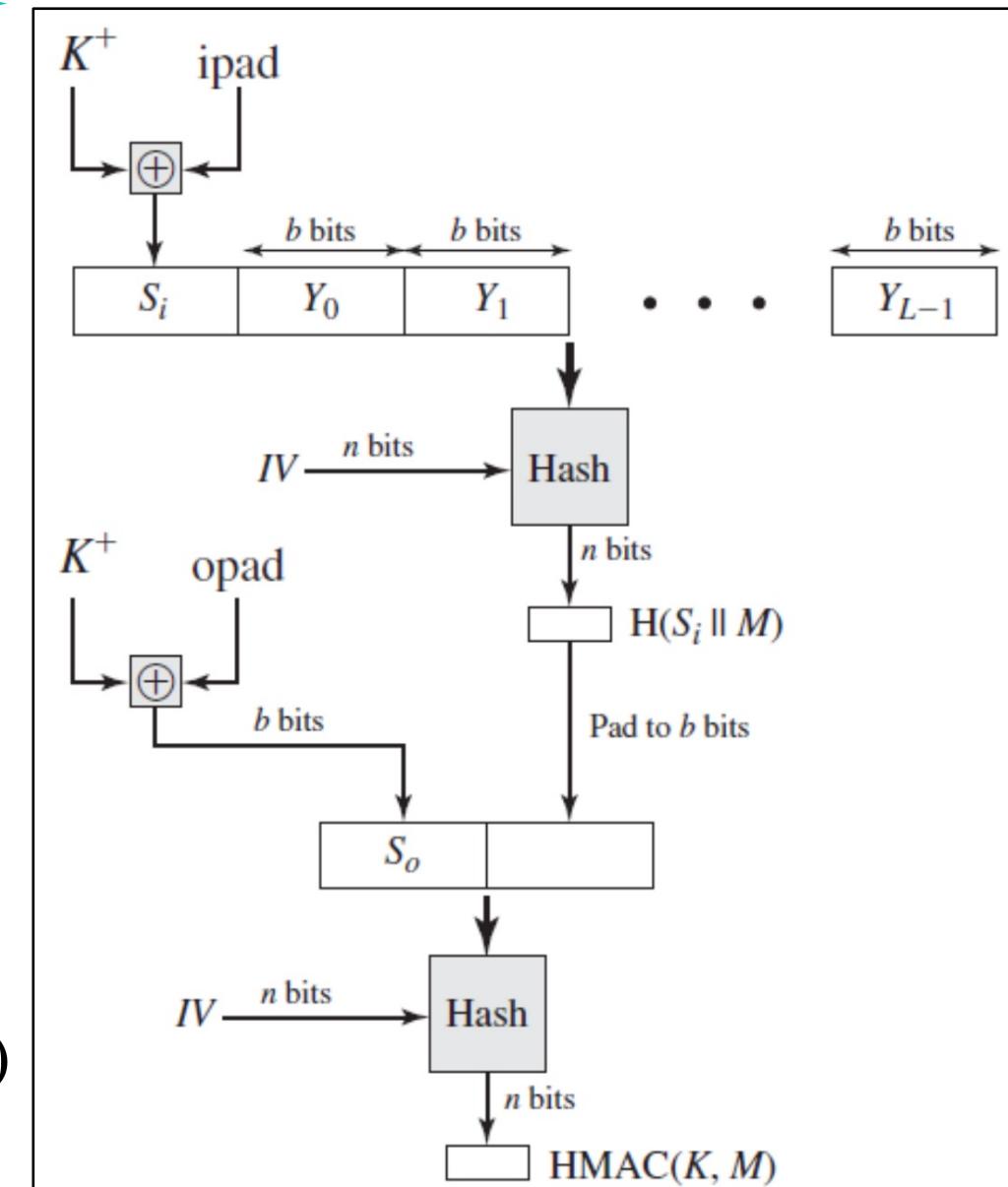
82

- “Cryptographic checksum” to ensure the **integrity** of the message and the data origin **authentication** (in symmetric-key cryptography)
- CBC-MAC, CMAC, OMAC, HMAC, ...

MAC Algorithm Example: HMAC

- For your information ☺

$$HMAC_K(m) = H((K' \oplus opad) || H((K' \oplus ipad) || m))$$



Holy Grail of Cryptography

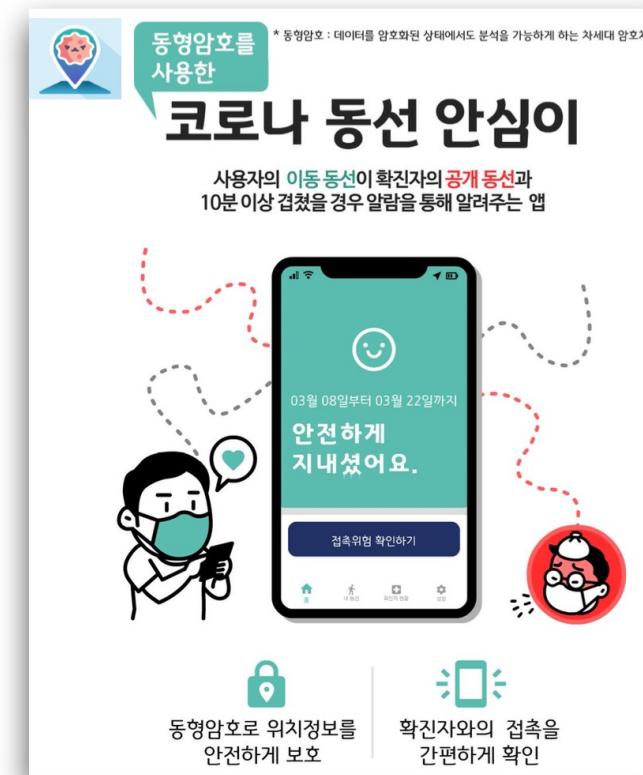
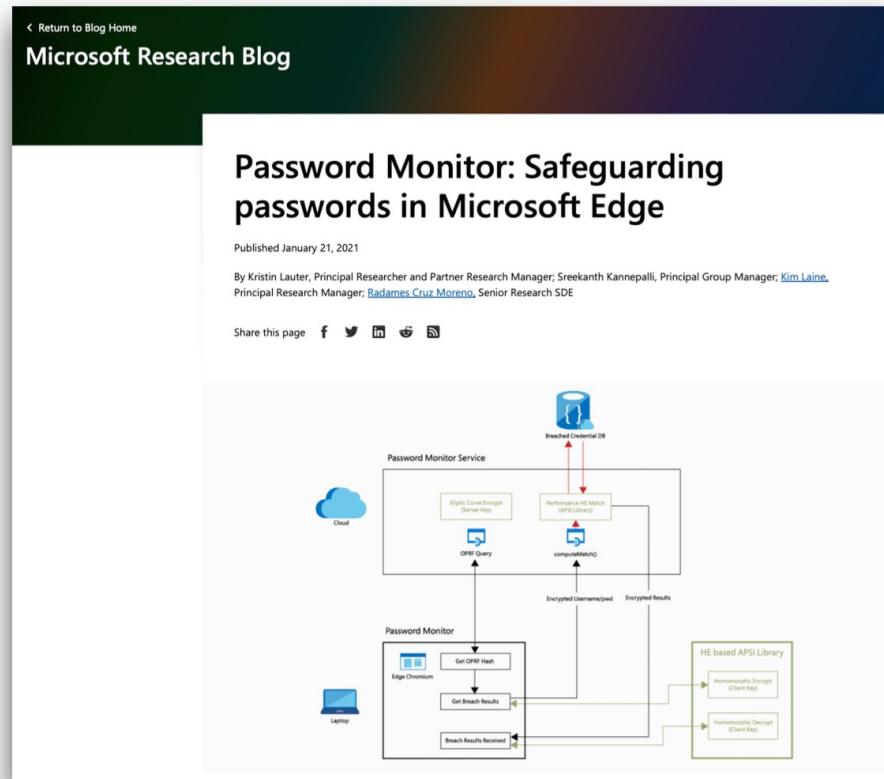
85

- Is it possible to provide a secure public service?
 - i.e., computations on encrypted data
- Example
 - Average GPA in the class with encrypted individual GPAs
 - Covid-19 alert with encrypted location information
 - Election with encrypted votes
- Necessary property: **homomorphism**
 - $Dec(c1 \oplus c2) = Dec(c1) \oplus Dec(c2)$

Homomorphic Encryption (동형 암호)

86

- Allows computations on encrypted data
- “*A Fully Homomorphic Encryption Scheme*”, C. Gentry, 2009
- Applications:



A Simplified Symmetric Homomorphic Encryption

87

- **Plaintext space:** $\{0,1\}$
- **Secret key:** p
- **Random numbers:** q and ϵ
- **Encryption:** $Enc(m) = m + pq + 2\epsilon$
- **Decryption:** $Dec(c) = (c \bmod p) \bmod 2$
- **Homomorphism**
 - $Dec(Enc(m1) + Enc(m2)) = Dec(Enc(m1 + m2)) = m1 + m2$
 - $Dec(Enc(m1) \times Enc(m2)) = Dec(Enc(m1 \times m2)) = m1 \times m2$

Summary



- Public-Key Infrastructure
 - Certificate Authority (CA)
 - Digital Certificate
 - Chain of trust
- Cryptographic Hash Functions
 - Preimage resistant
 - Second preimage resistant
 - Collision resistant
- Message Authentication Codes (MAC)
 - Check both integrity and authenticity for symmetric key environment
- Homomorphic Encryption

Question?