Countdown

Class is starting now!

Cryptography II

Entropy

Block and Stream Ciphers

Public-Key Cryptography

Digital Signature

Entropy

Entropy

- Amount of uncertainty in a situation

- Fair Coin Flip
  - Maximum uncertainty
- Biased Coin Flip
  - More bias → Less uncertainty

Entropy

- Computers need a source of uncertainty (entropy) to generate random numbers.
  - Cryptographic keys.
  - Protocols that need coin flips.
- Which are sources of entropy in a computer?
  - Mouse and keyboard movements or thermal noise of processor.
  - Unix like operating systems use dev/random and dev/urandom as randomness collector

Random Number in Practice

- We need random numbers but...

  "Anyone who considers arithmetical methods of producing random numbers is, of course, in a state of sin." - John von Neumann

- Bootup state is predictable and entropy from the environment may be limited:
  - Temperature is relatively stable
  - Oftentimes the mouse/keyboard motions are predictable
- Routers often use network traffic
  - Eavesdroppers.
- Electromagnetic noise from an antenna outside of a building
- Radioactive decay of a 'pellet' of uranium
- Lava lamps…


Cloudflare company uses lava lamp has an entropy source
https://www.cloudflare.com/learning/ssl/lava-lamp-encryption/

Random Oracle Model

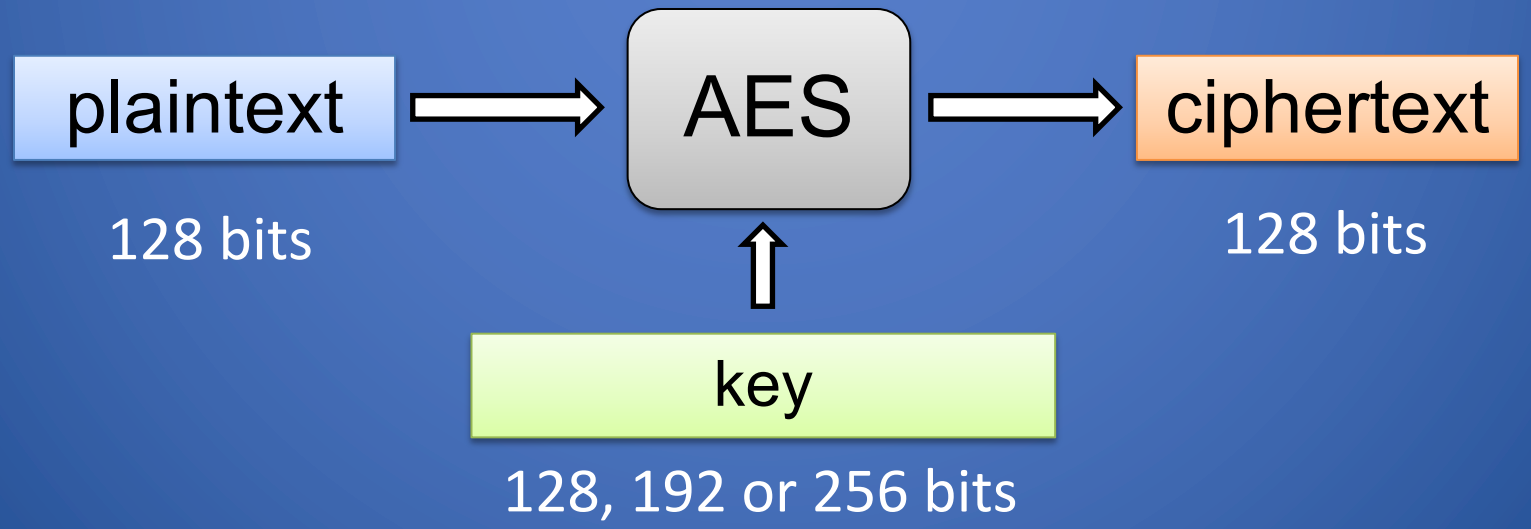
- A random oracle is a theoretical model (black box) that:
  - Answers to a query Q with a random number
  - Answers same way every time it receives same query Q

- This is often not an accurate representation of reality...

Block Ciphers

Block Cipher

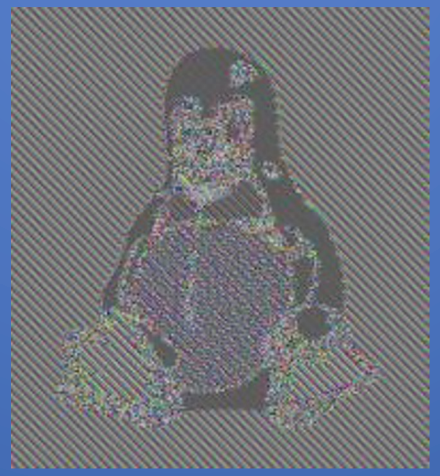
- A block cipher is a symmetric encryption scheme for messages (blocks) of a given fixed length
  - The length of the block is independent from the length of the key
- AES is a block cipher that operates on blocks of 128 bits (16 bytes)
  - AES supports keys of length 128, 192, and 256 bits



ECB Mode

- When plaintext is longer than block size, b
  - Partition plaintext P into sequence of m blocks P[0], …, P[m−1], where $n/b \le m < n/b + 1$
- Electronic Code Book (ECB) Mode
  - Assume n is multiple of b
  - Block P[i] encrypted into ciphertext block $C[i] = E_K(P[i])$

- Documents and images are not suitable for ECB
  - Zoom ECB case (2020)[1]
- ECB works well with random strings
- Encryption can be done in parallel

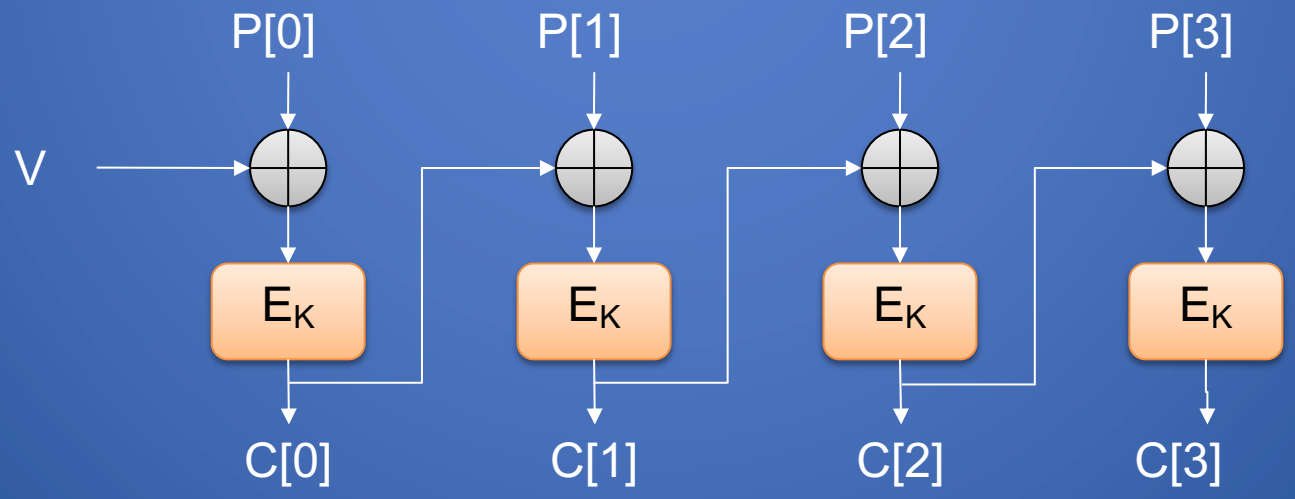


Source of images: Tux the Penguin created by Larry Ewing <lewing@isc.tamu.edu> with The GIMP and derived encrypted image by Lunkwill. Downloaded from https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

1: https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/

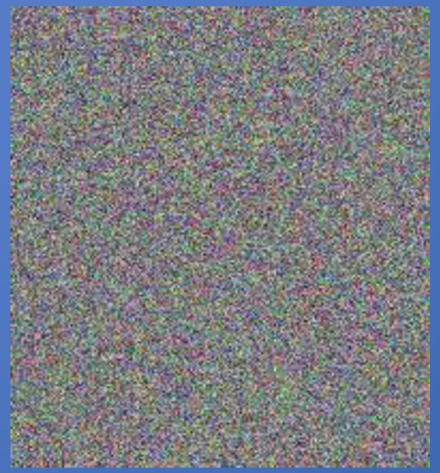
CBC Mode

- Cipher Block Chaining (CBC) Mode
  - Previous ciphertext block combined with current plaintext block
    $C[i] = E_K(C[i-1] \oplus P[i])$
  - C[−1] = V is a random block (initialization vector) sent encrypted during setup



CBC Mode Properties

- Works well with any input plaintext
- Requires the reliable transmission of all blocks
  - Not suitable for applications that allow packet losses
  - E.g., audio and video streaming



Source of images: Tux the Penguin created by Larry Ewing <lewing@isc.tamu.edu> with The GIMP and derived encrypted images by Lunkwill. Downloaded from https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Padding (PKCS #7)

- Block ciphers require the length n of the plaintext to be a multiple of the block size b
- How to pad unambiguously the last block?
- When the block size and plaintext length are a multiple of 8, a common padding method (PKCS #7) is a sequence of identical bytes, each indicating the length (in bytes) of the padding
- Public-Key Cryptography Standards(PKCS) an RSA lab standard

- Example for b = 128 (16 bytes)
  - Plaintext: “Bernardo” (7 bytes)
  - Padded plaintext: “Bernardo999999999” (16 bytes), where 9 denotes the number of bytes necessary for padding and not the character
- We need to always pad the last block, which may consist only of padding
- This padding method works provided the block size is at most 256 bytes (2,048 bits)

Clicker Question (1)

Eve has a diabolical plan to flood the CIT shower. She encrypts her 32-byte plan plaintext $P = P[0], P[1]$ using a 128-bit (16-byte) block cipher E_k with Cipher Block Chaining (CBC) mode.

How can she express the ciphertext in terms of the plaintext P, initialization vector V, and block cipher E_k? (assume no padding)?

A. $E_k(P[0], P[1])$

B. $E_k(P[0]), E_k(P[1])$

C. $E_k(P[0] \oplus V),\ \ E_k(P[1] \oplus V)$

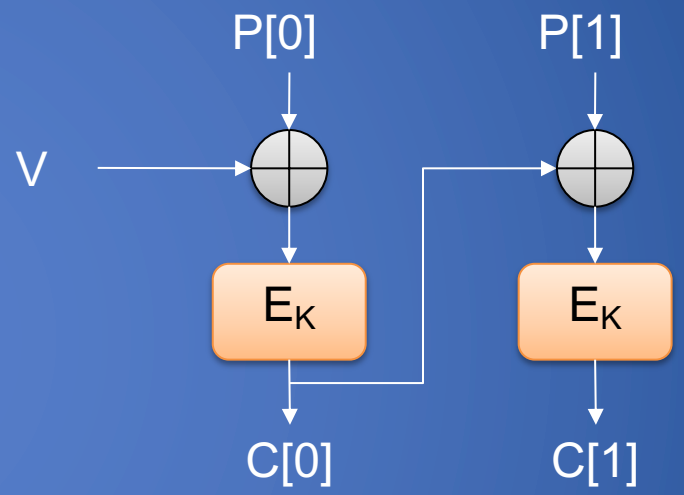
D. $E_k(P[0] \oplus V),\ \ E_k\left(P[1] \oplus E_k(P[0] \oplus V)\right)$

Clicker Question (1) - Answer

ANSWER: D

$$C[0] = E_k(P[0] \oplus V)$$
$$C[1] = E_k(P[1] \oplus C[0])$$

$$C = C[0], C[1]$$
$$= E_k(P[0] \oplus V),\ E_k(P[1] \oplus C[0])$$
$$= E_k(P[0] \oplus V),\ E_k\left(P[1] \oplus E_k(P[0] \oplus V)\right)$$



Stream Ciphers

One-Time Pad (Recap)

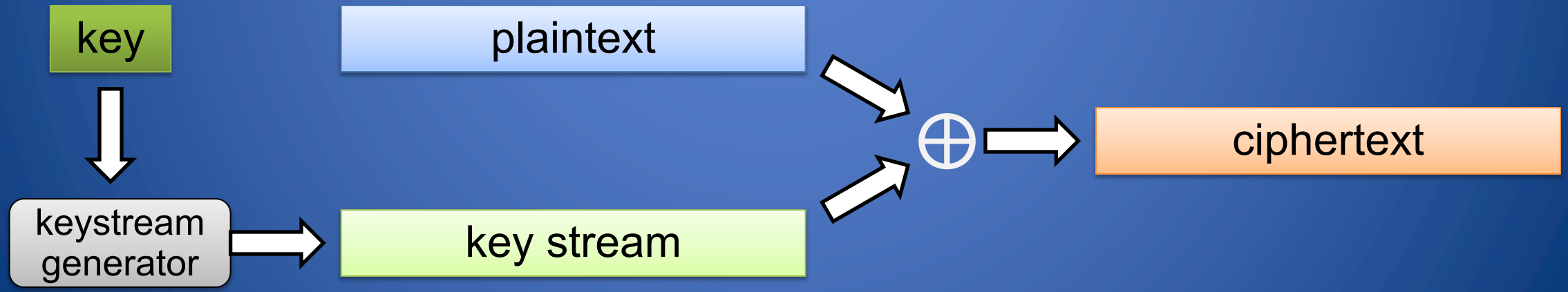
- Key K
  - Sequence of random bits
  - Same length as plaintext
- Encryption
  - $C = K \oplus P$
  - Example
    - P = 01101001
    - K = 10110010
    - C = 11011011
- Decryption
  - $P = K \oplus C$

- Advantages
  - Each bit of the ciphertext is random
  - Fully secure if key used only once
- Disadvantages
  - Key as large as plaintext
    - Difficult to generate and share
  - Key cannot be reused

Stream Cipher

- Key stream
  - Pseudo-random bit sequence generated from a secret key K

    $S_K = S_K[0], S_K[1], S_K[2], \ldots$
  - Generated on-demand, one bit (or block) at the time
- Stream cipher
  - XOR the plaintext with the key stream $C[i] = S_K[i] \oplus P[i]$



Stream Cipher

- Advantages
  - Fixed-length secret key
  - Plaintext can have arbitrary length (e.g., media stream)
  - Incremental encryption and decryption
  - Works for packets sent over an unreliable channel
- Disadvantages
  - Key stream cannot be reused

Key Stream Generation

- Block cipher in counter mode
  - Use a block cipher E_K with block size b
  - e.g., AES (block size is 128)
  - The secret key is a pair (K, t), where K a is key and t is a counter with b bits
  - The key stream is the concatenation of ciphertexts

    $E_K(t), E_K(t + 1), E_K(t + 2), \ldots$

- Advantages
  - Simplicity
  - Speed
- Disadvantages
  - Very long key streams can be distinguished from random

Attacks on Stream Ciphers

- Repetition attack
  - Stream reuse yields XOR of plaintexts
  - Cryptanalysis can recover the original plaintexts
- Replacement attack
  - The attacker knows a certain portion of the plaintext P
  - P = A B C, where the attacker knows B
  - From the ciphertext of P, the attacker can derive the ciphertext of Q = A D C, where D is an arbitrary message chosen by the attacker

Initialization Vector

- Goal
  - Avoid sharing a new secret key for each stream encryption
- Solution
  - Use a two-part key (U, V)
  - Part U is fixed
  - Part V is transmitted together with the ciphertext
  - V is called initialization vector

- Setup
  - Alice and Bob share secret U
- Encryption
  - Alice picks V and creates key
    $$K = (U, V)$$
  - Alice creates stream ciphertext C and sends (V, C)
- Decryption
  - Bob reconstructs key K = (U, V)
  - Bob decrypts the message

Clicker Question (2)

- Which of the following is NOT true regarding block and stream ciphers?
  A. A block cipher operates on a fixed size plaintext, while a stream cipher can operate on any length plaintext
  B. Block ciphers are more secure than stream ciphers
  C. Both block and stream ciphers use symmetric keys, meaning there is a shared secret key used for both encryption and decryption
  D. Initialization vectors can be used with BOTH block and stream ciphers

Clicker Question (2) - Answer

- Which of the following is NOT true regarding block and stream ciphers?
  A. A block cipher operates on a fixed size plaintext, while a stream cipher can operate on any length plaintext
  B. Block ciphers are more secure than stream ciphers
  C. Both block and stream ciphers use symmetric keys, meaning there is a shared secret key used for both encryption and decryption
  D. Initialization vectors can be used with BOTH block and stream ciphers

Break!!!!!

60 60 60 60 60

Class is starting now!

Public Key Cryptography

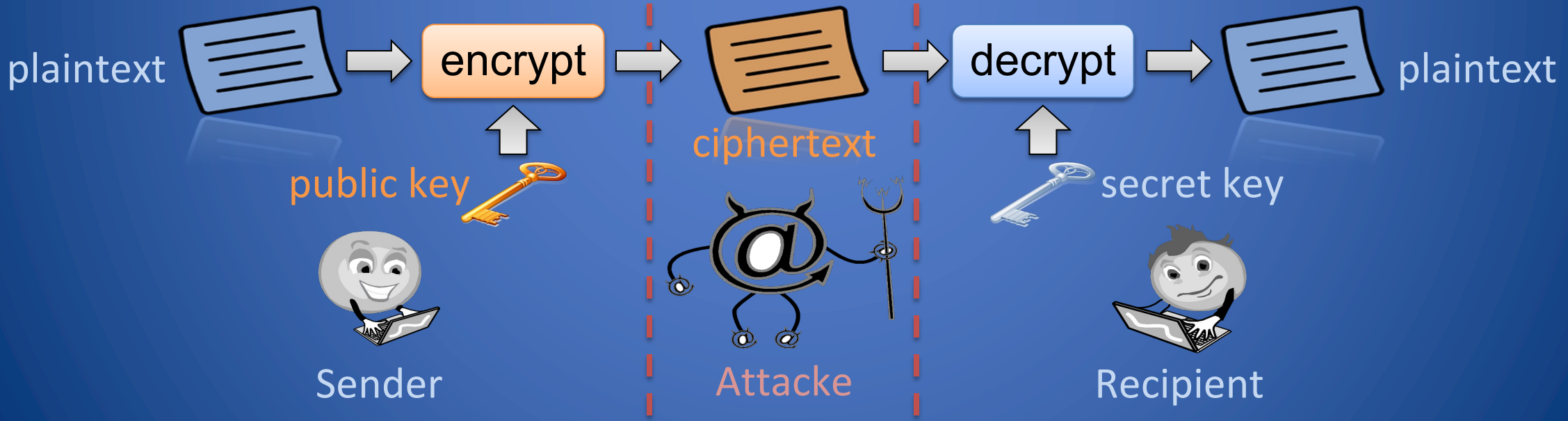
Public Key Cryptography

Key pair

- Public key: shared with everyone
- Secret key: kept secret, hard to derive from the public key

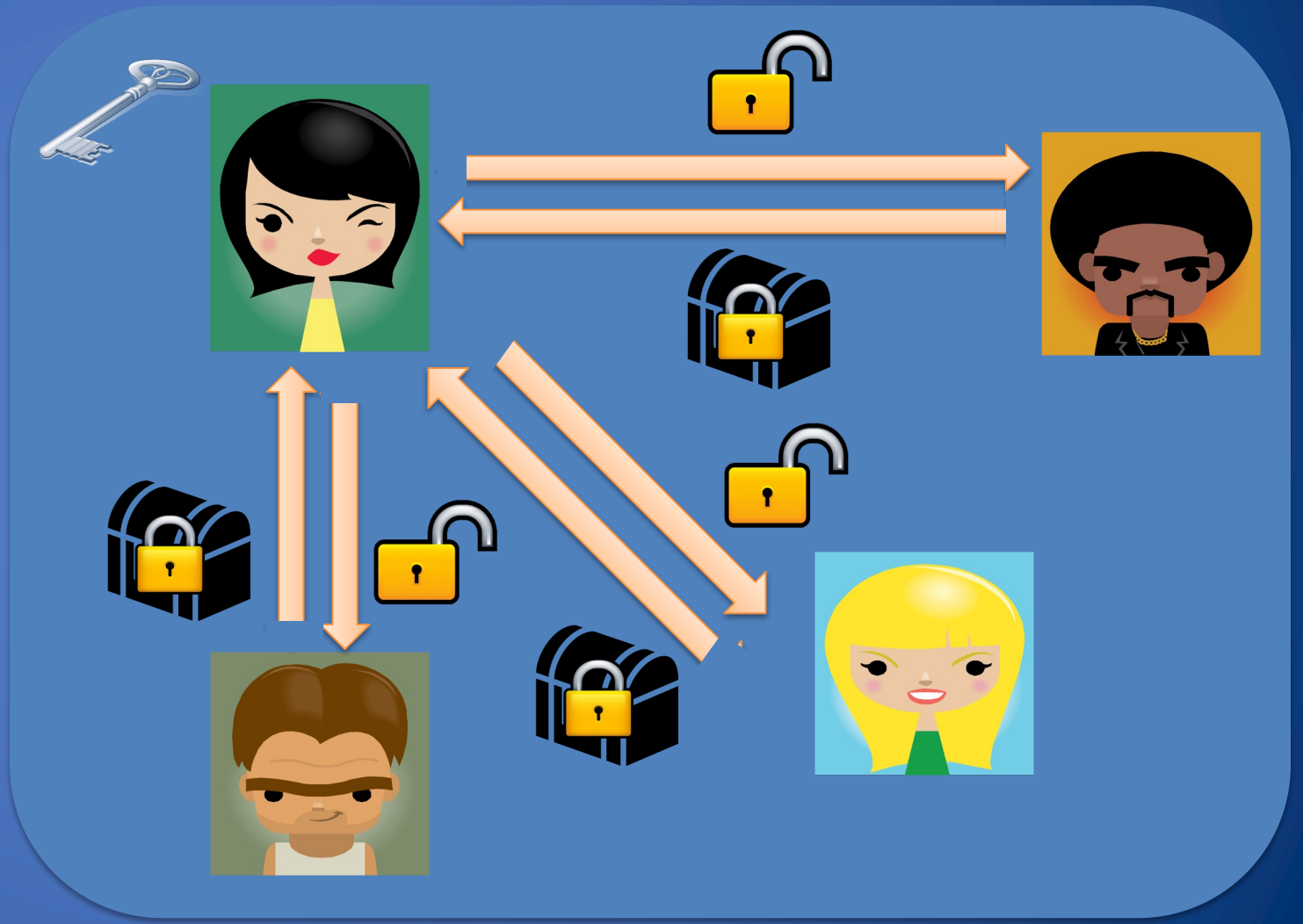
Protocol

- Sender encrypts using recipient's public key
- Recipient decrypts using its secret key



Intuition

- Alice
  - Buys padlock
  - Keeps key
  - Sends open padlock to Bob
- Bob
  - Locks message with Alice's padlock
  - Sends locked message to Alice
- Alice
  - Opens padlock with key
- No keys are exchanged
- Alice can share identical open padlocks with multiple people

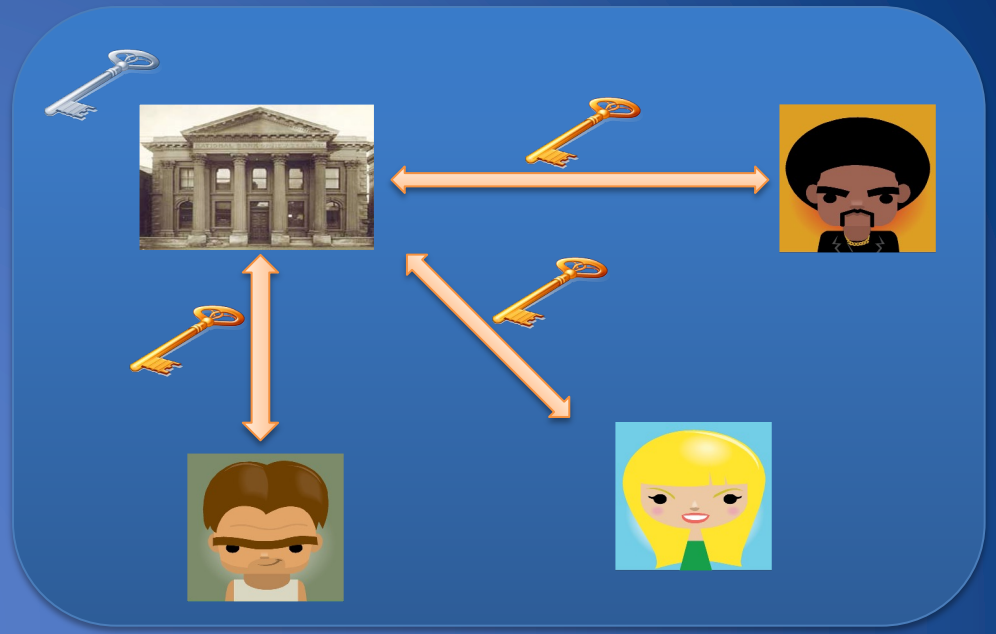


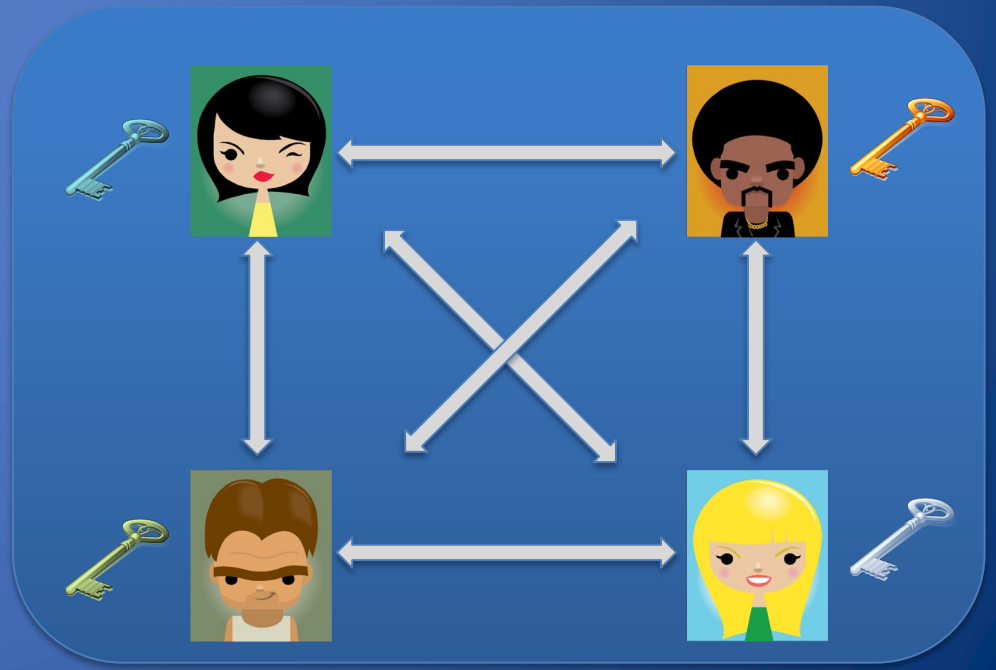
Public-Key Encryption in Formulas

- Notation
  - PK: public key of recipient
  - SK: secret key of recipient
  - M: plaintext
  - C: ciphertext
- Encryption
  - $C = E_{PK}(M)$
  - The sender encrypts the plaintext with the public key of the recipient
- Decryption
  - $M = D_{SK}(C)$
  - The recipient decrypts the ciphertext with their private key
- Properties
  - Anyone can encrypt a message since the recipient openly shares the public key
  - Only the recipient can decrypt the message since the private key is kept secret
  - It should be unfeasible to derive the secret key from the public key

Properties

- Advantages
  - A single public-secret key pair allows receiving confidential messages from multiple parties
- Disadvantages
  - Conceptually complex
  - Slower performance than symmetric cryptography





RSA

- Most widely used public key cryptosystem today
- Developed by Rivest, Shamir, and Adleman (1978)
- RSA patent expired in 2000
- 2048-bit (or longer) keys recommended
- Much slower than AES
- Typically used to encrypt an AES symmetric key
- In 1973, Clifford Cocks and James Ellis developed an equivalent system at GCHQ, declassified in 1997

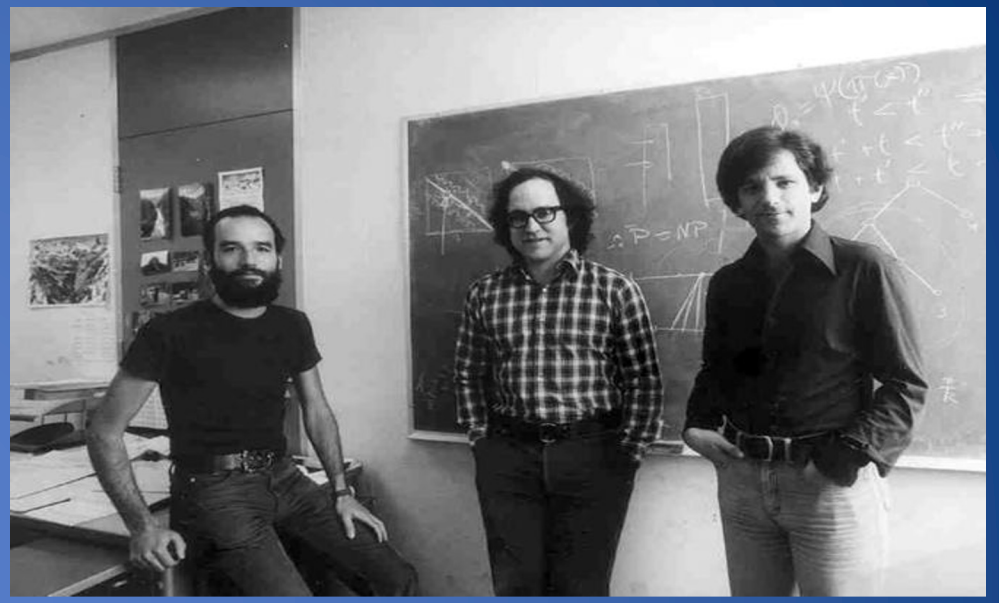


Image used with permission from Ron Rivest and Len Adleman

Image source: The Royal Society

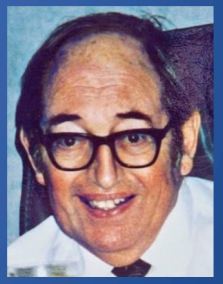


Image source: The Telegraph

Clicker Question (3)

Which of the following is NOT true regarding the RSA public-key cryptosystem?

A. The sender encrypts the plaintext using the recipient's public key, and the recipient decrypts the ciphertext using their secret key
B. Often used to encrypt an AES symmetric key for further secure communication
C. Slower in performance than the AES symmetric cryptosystem
D. Requires a different public-secret key pair for each distinct party from whom you wish to receive confidential messages

Clicker Question (3) - Answer

ANSWER: D

Requires a different public-secret key pair for each distinct party from whom you wish to receive confidential messages [Wrong!]

One person can use the same public-secret key pair to communicate with multiple parties—they share the same public key with each person that they are communicating with, and decrypt every message with their own secret key. This is an advantage over symmetric encryption, which requires a distinct key for every pair of users.

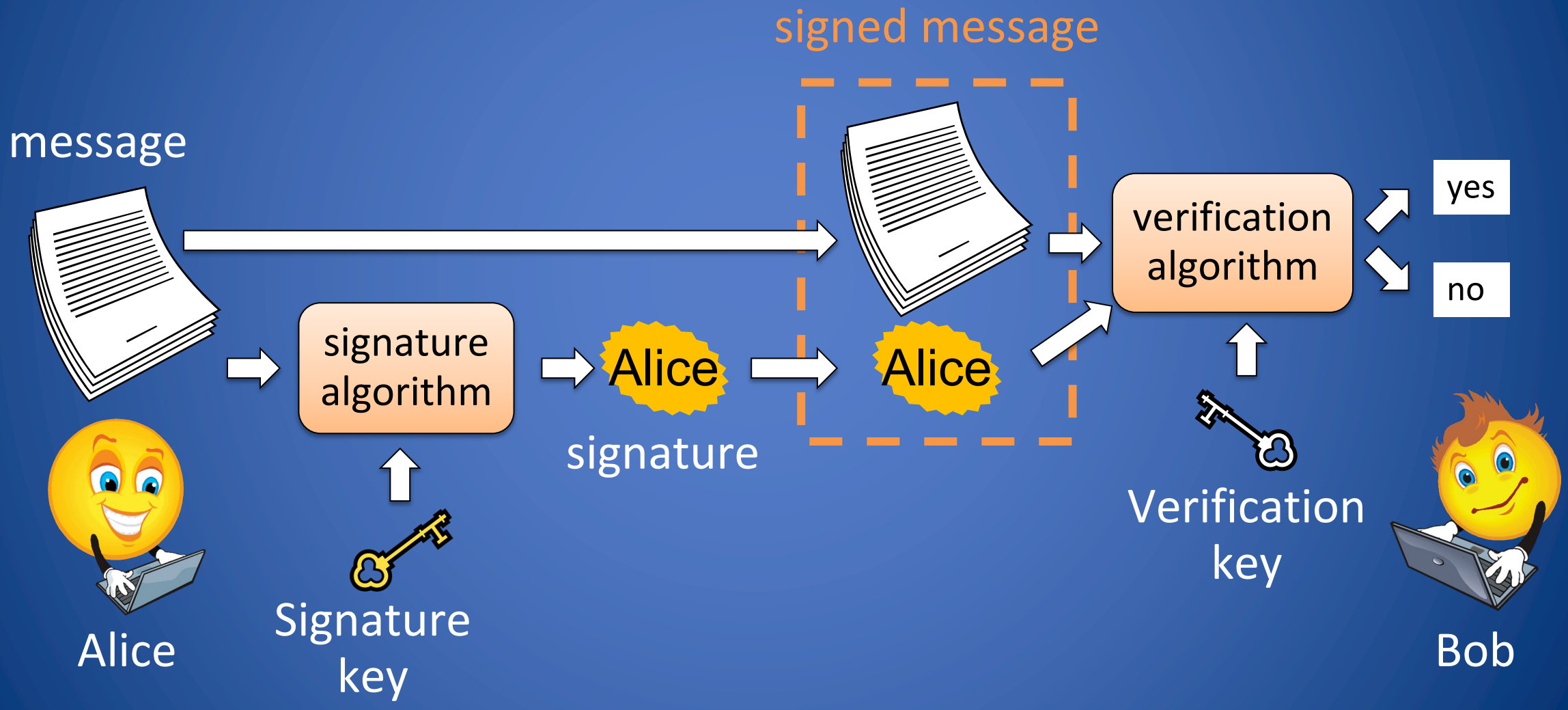
Digital Signatures

Signatures: from Ink to Digital

- Signature in the real world
  - Contracts
  - Checks
  - Job offers
  - Affidavits
- Digital signatures are a matter of both computer security and law
- ESIGN Act (2000 US)
- eIDAS Regulation (2014 EU)
- Technological failures can have legal consequences

What is a Digital Signature?

- Alice wants to send a message and prove that it comes from her



Goals for a Digital Signature

- Authenticity
  - Binds an identity (signer) to a message
  - Provides assurance of the signer
- Unforgeability
  - An attacker cannot forge a signature for a different identity
- Nonrepudiation
  - Signer cannot deny having signed the message
- Integrity
  - An attacker cannot take a signature by Alice for a message and create a signature by Alice for a different message

Digital Signatures in practice

- Use symmetric key encryption
  - Requires previous secure communication
  - Only works with single recipient
- Can we use public key encryption?

Digital Signature with Public-Key Encryption

- In a public-key cryptosystem (e.g., RSA), we can often reverse the order of encryption and decryption

$$E_{PK}\,(D_{SK}\,(M)) = M$$

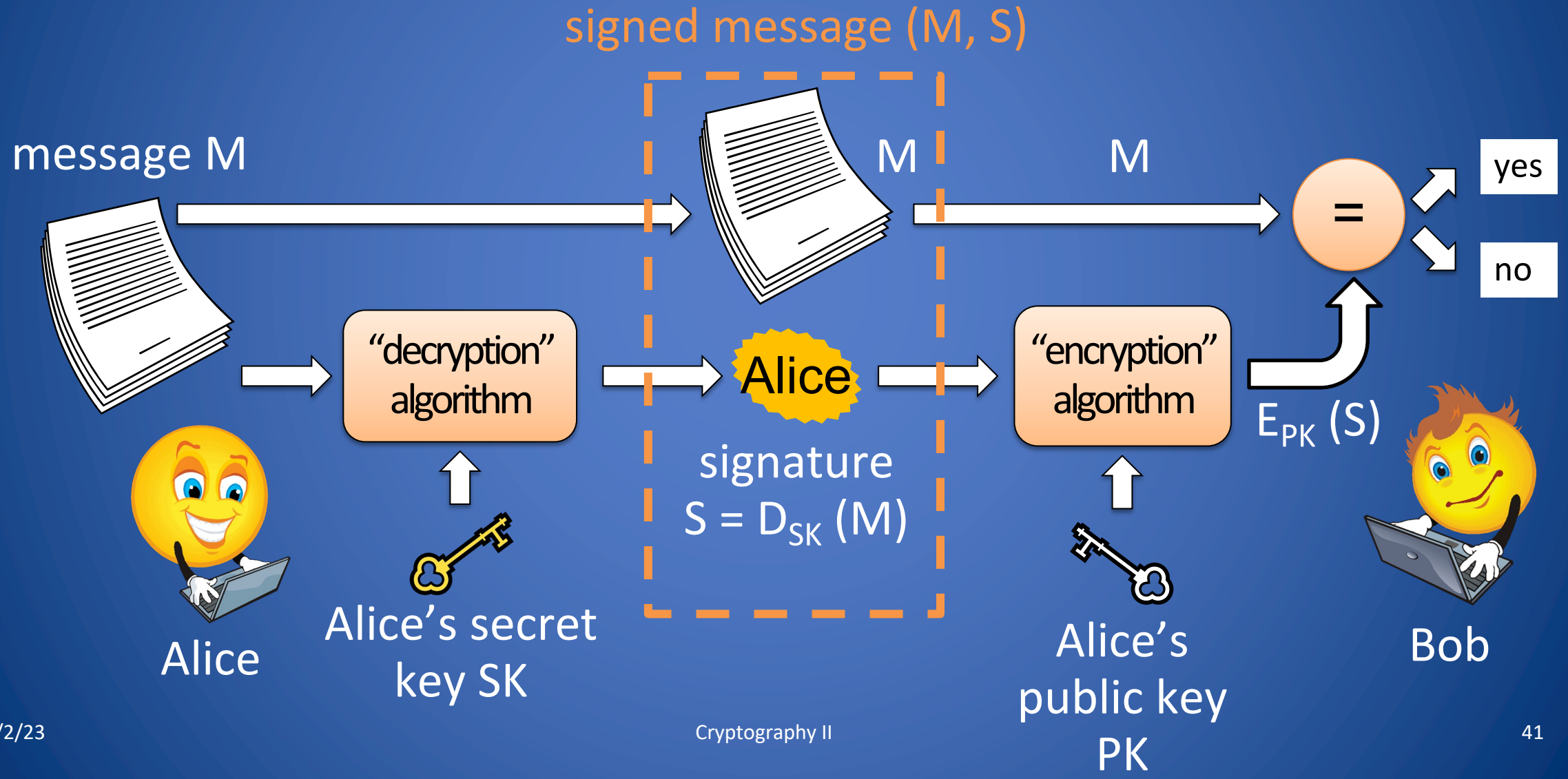
- Alice "decrypts" plaintext message M with the secret key and obtains a digital signature on M

```
sign(M, SK) {
  return S = D_SK (M) }
```

- Knowing Alice's public key, PK, can verify the validity of signature S on M
- Bob "encrypts" signature S with PK, and
- Checks if it the result is message M

```
verify(M, S, PK) {
    return (M == E_PK (S) )
}
```

Digital Signature with Public-Key Encryption



Signing Hashes

- Basic method for public-key digital signatures
  - Signature as long as the message
  - Slow public-key encryption/decryption
- Preferred method
  - Sign a cryptographic hash of the message
  - Hash is short and fast to compute

- Sign

$$S = D_{SK}(h(M))$$

- Verify

$$h(M) == E_{PK}(S)$$

- Security of signing hash
  - Security of digital signature
  - Collision resistance of hash function

Clicker Question (4)

Alice wants to increase the efficiency of her public-key digital signature system by signing a cryptographic hash of each message instead of the message itself. Given the decryption function D, secret key SK, and message M, how can we represent Alice's digital signature S on the hash of the message?

A. $S = D_{SK}(M)$

B. $S = D_{SK}(h(M))$

C. $S = (h(M), D_{SK}(M))$

D. $S = h(D_{SK}(M))$

Clicker Question (4) - Answer

Alice wants to increase the efficiency of her public-key digital signature system by signing a cryptographic hash of each message instead of the message itself. Given the decryption function D, secret key SK, and message M, how can we represent Alice's digital signature S on the hash of the message?

A. $S = D_{SK}(M)$

B. $S = D_{SK}(h(M))$

C. $S = (h(M), D_{SK}(M))$

D. $S = h(D_{SK}(M))$

Clicker Question (5)

Bob wants to send Alice an encrypted message. He found Alice's profile online, and it lists her public key, PK. How can Bob verify that this is really Alice's public key?

A. Check whether E_{PK} (D_{PK} (M)) = M
B. Use PK to encrypt message *M* = "If you can decrypt this message, reply with password **MySecretPassword**" and send it to the profile. Check whether you get the correct password back.
C. Send a request to the profile asking for a message digitally signed with the secret key corresponding to PK. Check whether the signature is valid.
D. None of the above

Clicker Question (5) - Answer

Bob wants to send Alice an encrypted message. He found Alice's profile online, and it lists her public key. How can Bob verify that this is really Alice's public key?

ANSWER: D. None of the above.

Bob cannot use method A since he does not have the private key. Also, it's unclear what message M would be in this method.

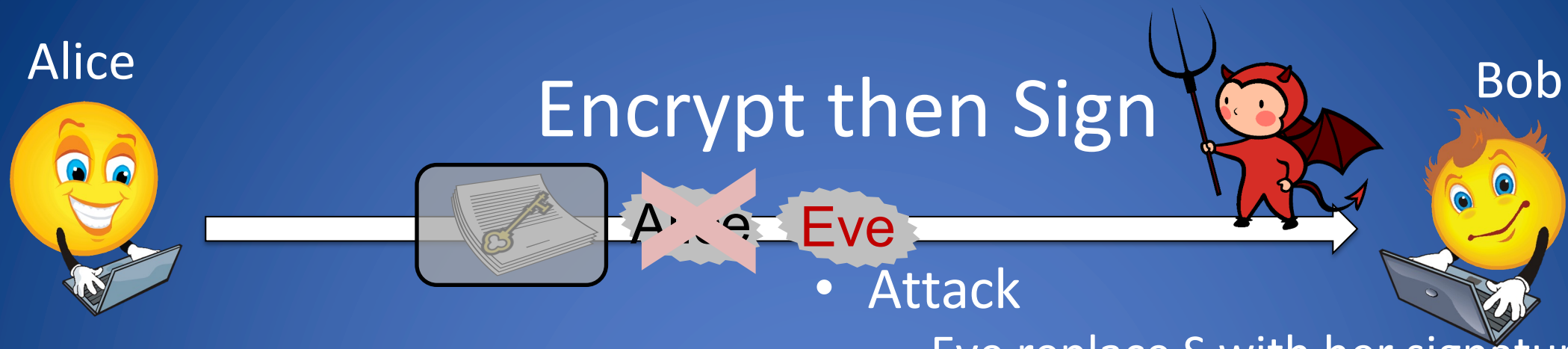
Methods B and C assure Bob that he is interacting with a party who has possession of the private key corresponding to the posted public key. However, they do not prove this party is Alice.

Send a message securely

- Alice wants to send a message that only Bob can read and that only she can have sent.
- Requirements
  - Confidentiality of all communication
  - Bob understands he is communicating with Alice
- Message M needs to be encrypted and digitally signed

- Active adversary, Eve
  - Can eavesdrop and modify messages
- Eve knows:
  - PK_{Alice}
  - PK_{Bob}

Encrypt then Sign



- Encrypt then sign
  - Alice encrypts $C = E((M, PK_{Bob}))$
  - Alice signs $C_S=(C, SK_{Alice})$
  - Alice sends C_S to Bob
  - Bob verifies $C=(C_s, PK_{Alice})$ and decrypts C to (C_s, SK_{Bob})

- Attack
  - Eve replace S with her signature S' on $C_{S'}$ and forwards (C, S') to Bob
  - Bob now thinks he is communicating with Eve
  - Eve can then forward Bob's response (intended for Eve) to Alice

This is a subtle risk but it could be dangerous

- during a transaction
- Authentication protocol
- …

Sign then Encrypt

- Sign then encrypt
  - Alice signs $M_S=(M, SK_{Alice})$
  - Alice encrypts $C = E((M_S, PK_{Bob}))$
  - Alice sends C to Bob
  - Bob decrypts C to (M_s, SK_{Bob}) and verifies $M=(M_s, PK_{Alice})$
- Attack
  - Eve does not know SK_{Bob}
    - She can not read M
  - Eve does not know SK_{Alice}
    - She can not tamper M

This is the correct order

Relying on Public Keys

- The verifier of a signature must be assured that the public key corresponds to the correct party
- The signer should not be able to deny the association with the public key
- Public keys usually are stored in browsers or in OS
- A trusted party could keep and publish pairs (identity, public key)
  - Government?
  - Private organizations?
- What if the private key is compromised?
  - Need for key revocation mechanism

Summary

- Entropy
- Block ciphers
- Stream ciphers
- Public Key Cryptography
- Digital signature