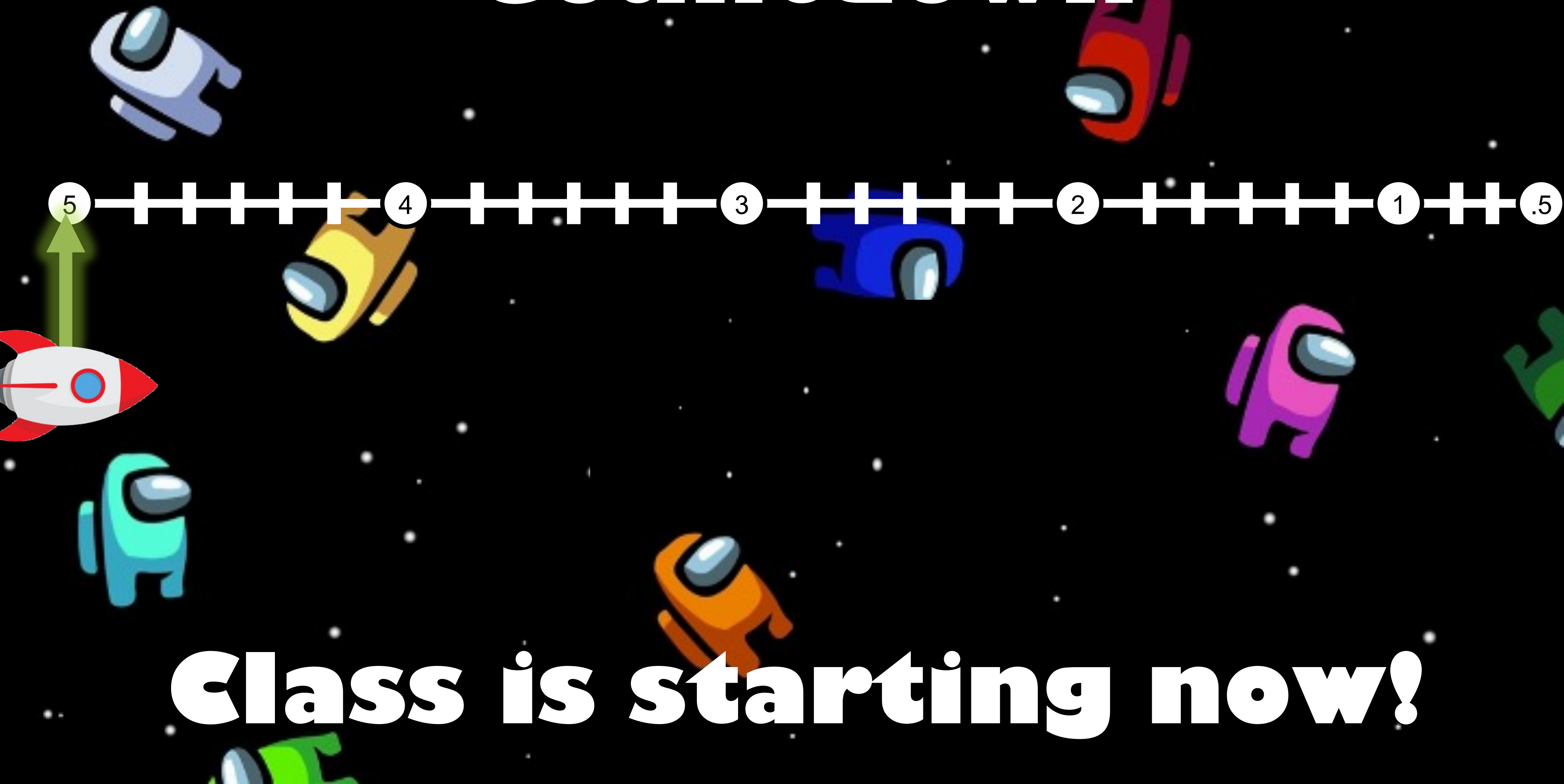
Countdown



Class is starting now!

Cryptography III
Digital Signatures, MACs, IND-CPA

CS 166: Introduction to Computer Systems Security

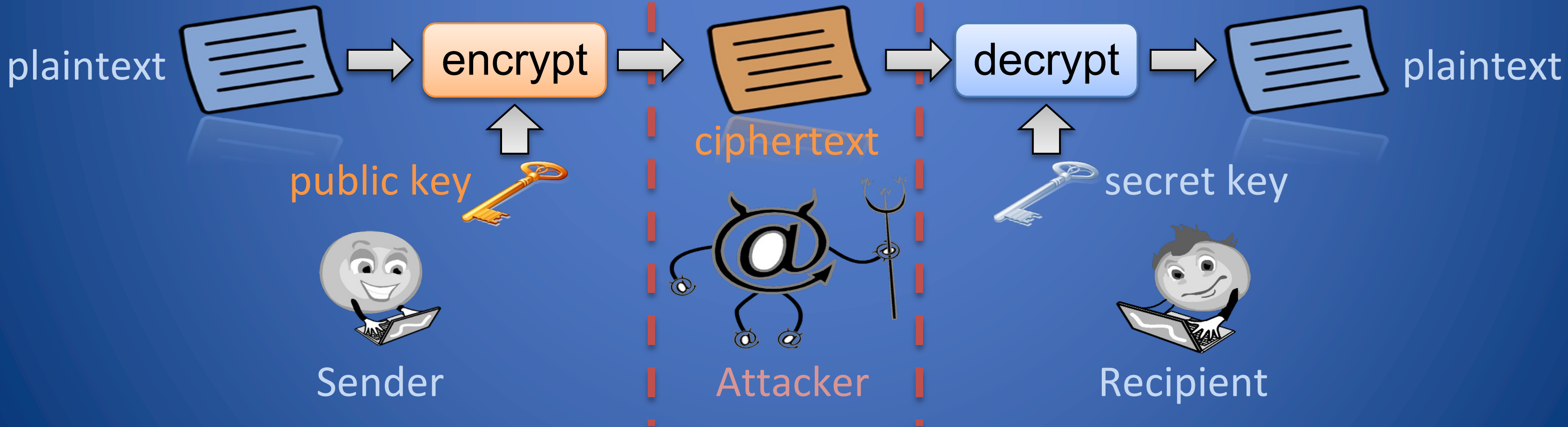
Public Key Cryptography (recap)

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



Public-Key Encryption in Formulas (recap)

- 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

Digital Signatures

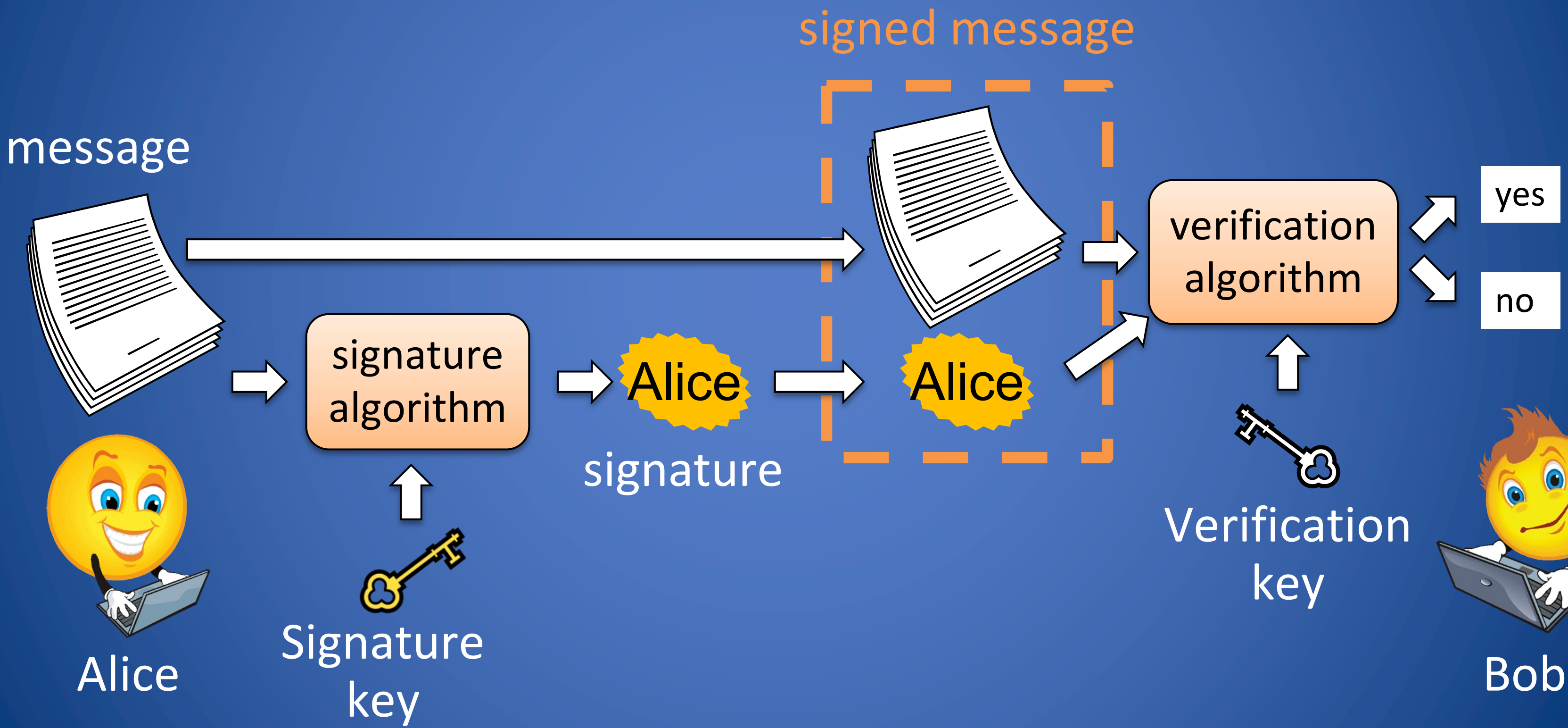
Signatures: from Ink to Digital

- 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



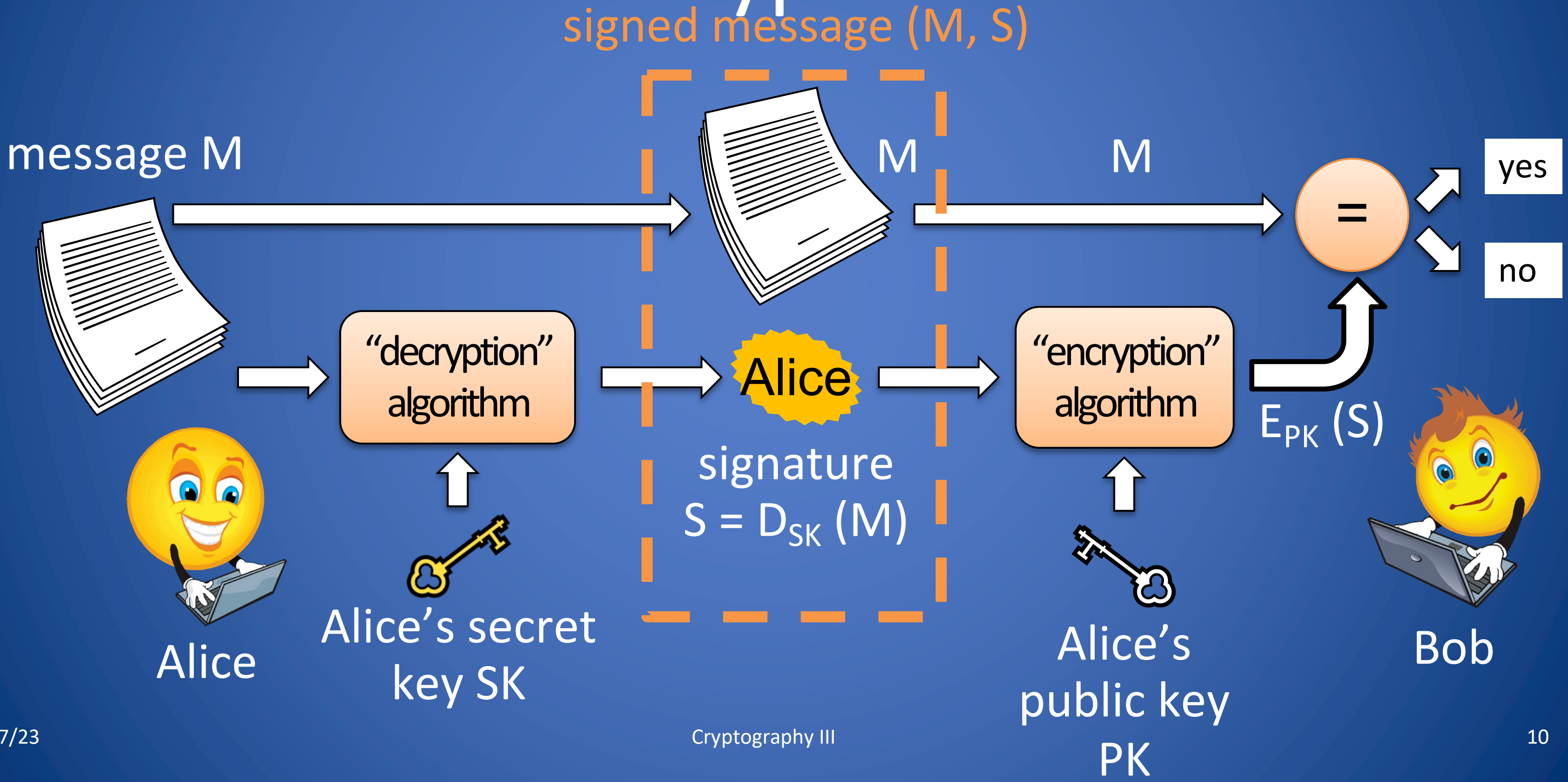
Goals for a Digital Signature

- 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



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 == $E_{PK}(S)$
) }
```

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 (1)

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 (1) - 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 (2)

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 (2) - 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.

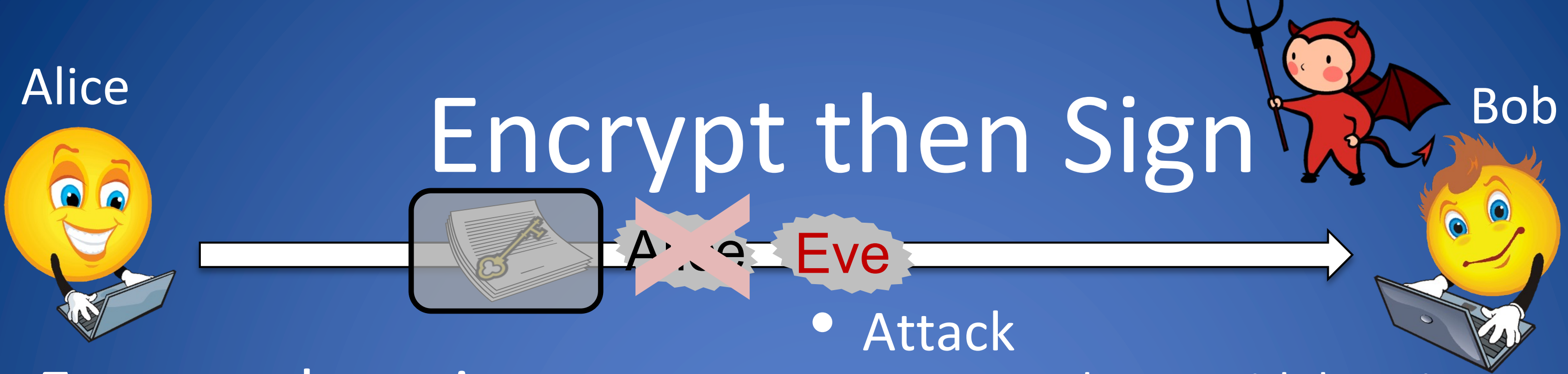
Send a message securely

Alice
Bob
Alice

- 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

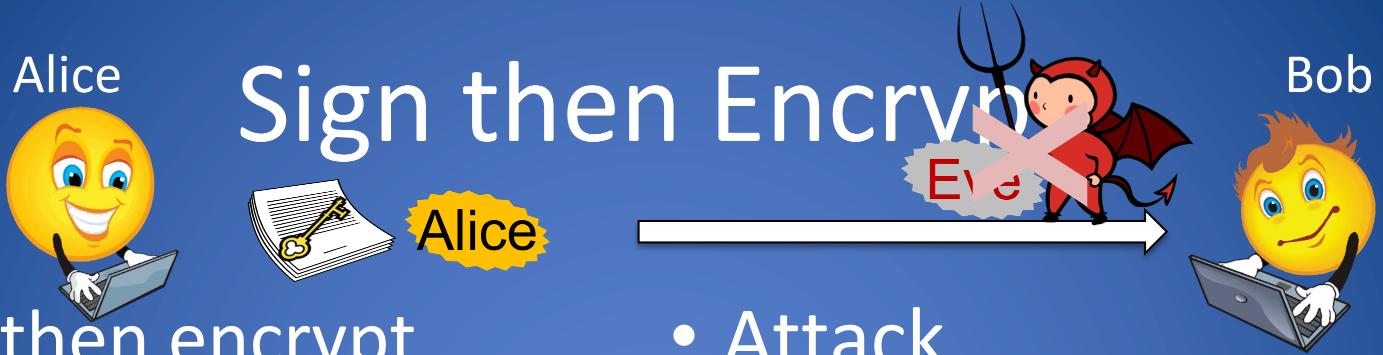


- Encrypt then sign
  - Alice encrypts $C = E((M, PK_{Bob}))$
  - Alice signs $C_S=(C, SK_{Alice})$
  - Alice sends C_S to Bob
  - Bob verifies $C=(C_S, PK_{Alice})$ and decrypts C to (C_S, SK_{Bob})

- 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

Message Authentication Code

MAC

- Similar to Digital Signature, but *symmetric*
  - Therefore does not provide nonrepudiation
- Provides a guarantee that a message came from a *certain sender* and *has not been changed*

MAC Properties

- Unforgeability
  - Even after seeing many MAC-message pairs, an attacker cannot produce a valid MAC for a new message
- Integrity
  - If the MAC or the message is altered, the recipient can detect it

Implementing MACs

Block Ciphers

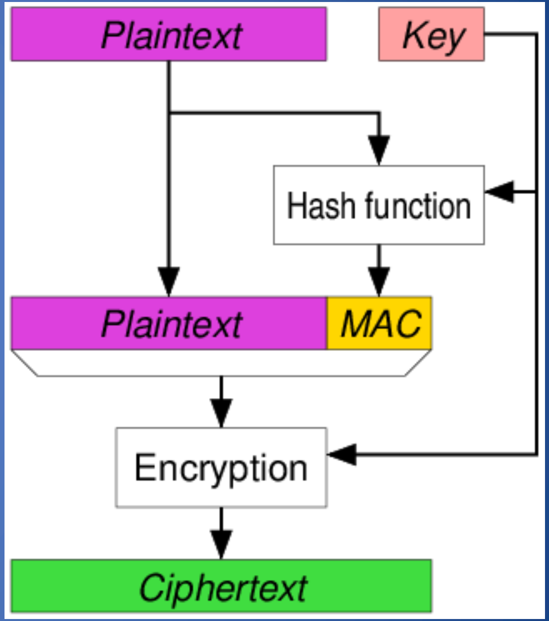
- CBC-MAC
  - Using a block cipher in CBC mode, encrypt a message and use the last cipher block as a MAC
  - *Requires some tweaks!* You must fix the IV and you must prepend each message with its length

Cryptographic Hash Functions

- HMAC
  - Use hash function and a shared secret
  - Theoretical construction:
    - *H(M||K)*
  - In practice:
    - Length extension attacks require padding schemes
    - RFC 2104

MAC then Encrypt (MtE)

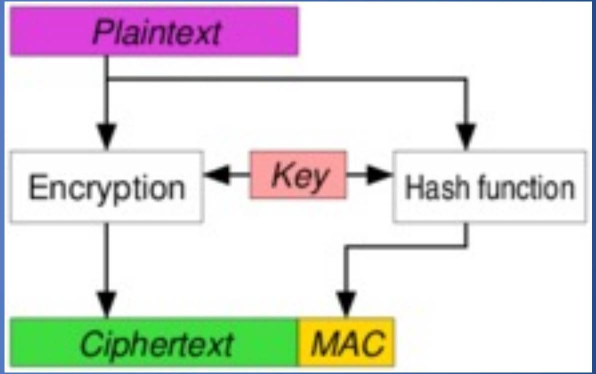
- E(*Message* || MAC(*Message*))
- Was used by TLS (although with special padding schemes)
- Does not provide integrity of ciphertext, only plaintext
- **Not** proven to be secure in general case (some exceptions like TLS)



https://upload.wikimedia.org/wikipedia/commons/a/ac/Authenticated_Encryption_MtE.png

MAC and Encrypt (M&E)

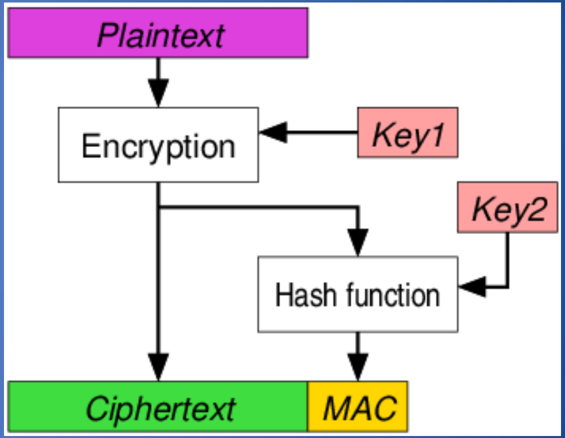
- (E(*Message*), MAC(*Message*))
- Can leak message equality even if E() does not
  - Unless you use the Key in counter mode
- Does not provide integrity of ciphertext, only plaintext
- Not proven to be secure (but again, some variants are in SSH)



https://upload.wikimedia.org/wikipedia/commons/a/a5/Authenticated_Encryption_EaM.png

Encrypt then MAC (EtM)

- (E(*Message*), MAC(E(*Message*)))
- Integrity guarantee on ***both*** ciphertext and plaintext
- Generally recommended order of operations
- Proposed to replace MtE in TLS (RFC 7366) and used in IPSEC
- You should use EtM! (We will see more in last project…)



https://upload.wikimedia.org/wikipedia/commons/b/b9/Authenticated_Encryption_EtM.png

Break!!!!!

60 60 60 60 60

Class is starting now!

Formalizing Encryption Security

Adversary Models

- Alice and Bob are sending encrypted messages to each other
  - Adversary Eve can eavesdrop on those messages
  - …and maybe do other things as well
- Security goal: protect confidentiality w.r.t. Eve
  - Useful to formalize: What are Eve's capabilities as an adversary?

(Weaker) Adversary Models

1. Ciphertext-only
   - Eve sees all ciphertexts, but has no / vague information about the underlying plaintext
2. Known plaintext
   - Eve also knows part of / format of plaintext messages
   - How could this happen?
     - All of your internet requests start with the same header
     - Sending a order CSV in the same format every week
     - You text “hi” to people when you first start texting them
   - Open design principle

(Stronger) Adversary Models

3. Chosen plaintext
   - Eve is able to encrypt plaintexts of Eve's choosing and see the resulting ciphertexts
   - How can this happen?
     - Eve sends Alice email spoofed from Alice's boss saying "Please securely forward this to Bob"
     - Public key cryptography
     - Your dorm room at the CREWMATE ACADEMY has a router that you can send plaintexts to…
4. Chosen ciphertext
   - Eve chooses ciphertexts and Alice reveals some info about the decryption
   - Mostly not covered too much in course…unless you're a CS1620/CS2660 student ☺

Formalization

- How do we show that our schemes are secure against these different kinds of attacker models?
- Intuitive definition: "No adversary can reconstruct plaintext M from ciphertext C"
  - This isn't sufficient—what if adversary can tell first letter of M, but nothing else?
    - Satisfies the definition, but still a broken scheme
    - Adversary could still reconstruct other parts of M based on what they know about its format
  - Need something stronger than this

- *Goal*: Cryptosystem should not leak *any* information about M
  - Idea: No adversary should be able to distinguish between two messages based on their encryption
- We model "security" of encryption schemes as a game
  - Played between a challenger (with access to the encryption algorithm and the secret key) and an adversary

IND-CPA

- ”Indistinguishability under Chosen Plaintext Attack”
- Adversary has polynomially-bounded access to an encryption oracle
  - If an adversary has access to this kind of oracle, we say they are an “IND-CPA adversary”)

If adversary guessed correct i, then adversary wins.

If adversary’s probability of winning the game is equal to ½, then our scheme is “IND-CPA secure” (why ½?)

Challenger **Adversary**

Setup Phase

Generate a key k = KeyGen()

Query Phase

$c = Enc_k(m)$

$\xleftarrow{m}$

$\xrightarrow{c}$

Repeat as many times as desired polynomially

Challenge Phase

$\xleftarrow{m_1, m_2 \text{ (of equal length) with } m_1 \neq m_2}$

Randomly pick $m_i \in \{m_1, m_2\}$

$c' = Enc_k(m_i)$

$\xrightarrow{c'}$

$\xleftarrow{\text{Output guess if } i = 1 \text{ or } i = 2}$

Clicker Questions

3) Is the Caesar cipher cryptosystem IND-CPA secure?

A. Yes

B. No

4) Is the one-time-pad cryptosystem IND-CPA secure?

A. Yes

B. No

5) Is the encryption function

$$Enc_k(m) = 1$$

IND-CPA secure?

A. Yes

B. No

Clicker Question (3)

ANSWER: B (No)

What's the adversary's strategy in the IND-CPA game against Caesar?

- Setup phase: Not necessary
- Challenge phase: Send plaintexts "AB" and "AA"
  - If output is in the form "XY" (where X =! Y), then output "AB"
  - Otherwise, output must be in form "XX"; then output "AA"

Three Clicker Questions

4) Is the one-time-pad cryptosystem IND-CPA secure?

A. Yes

B. No

5) Is the encryption function

$$Enc_k(m) = 1$$

IND-CPA secure?

A. Yes

B. No

Clicker Question (4)

ANSWER: B (No)

What's the adversary's strategy in the IND-CPA game against OTP?

- Setup phase: Send messages m_1, m_2 to get c_1, c_2
- Challenge phase: Send plaintexts m_1, m_2; challenger returns c_i
  - If $c_i \oplus c_1 = 0$, then output c_1
  - Otherwise, it must be that $c_i \oplus c_2 = 0$, so output c_2
  - Why does this work?

Three Clicker Questions

5) Is the encryption function

$$Enc_k(m) = 1$$

IND-CPA secure?

A. Yes
B. No

Clicker Question (5)

ANSWER: A (Yes)

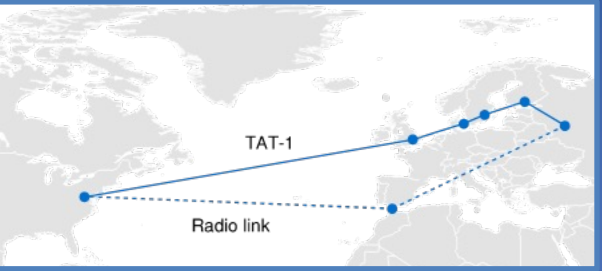
But it’s not “correct”…

We also care about *correctness*—i.e. that we can actually decrypt a given encryption.

Summary

- Digital signature
  - Authenticity, Unforgeability, Nonrepudiation, Integrity
- Message Authentication Codes
  - CBC-MAC and HMAC
- Formalizing Encryption Security
  - IND-CPA model

CRYPTO IN PRACTICE

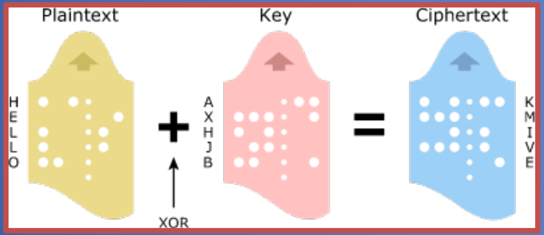


Washington-Moscow Hotline

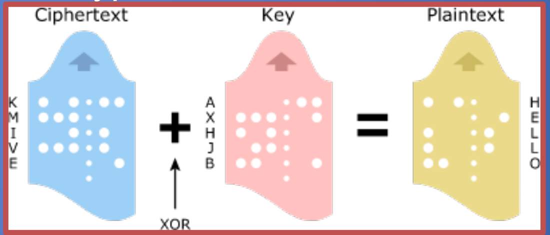
One-Time Tape (OTT)

- Teleprinter version of OTP (Vernam Cipher)

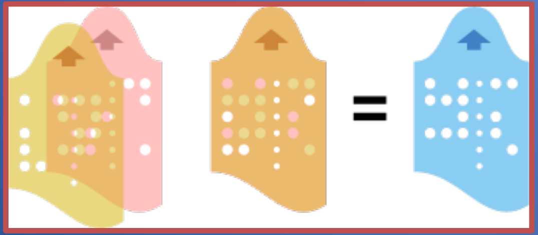
Encryption

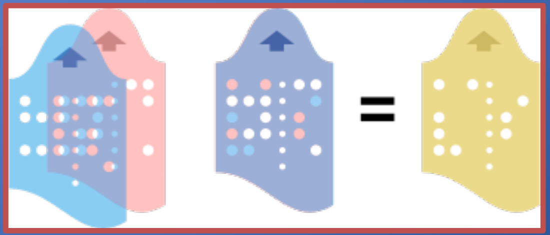


Decryption



Mixing





Electronic Teleprinter Cryptographic Regenerative Repeater Mixer (ETCRRM)

source: https://www.cryptomuseum.com/crypto/hotline/index.htm

Cybersecurity mindset

- The first message from USA to Moscow 8/30/1963
  - Which characteristics…

```
THE QUICK BROWN FOX
JUMPED OVER THE LAZY
DOG'S BACK 1234567890
```

  - a line that contains all letters of the alphabet (a pangram) and numbers, so they tested that every possible characters worked