Cryptography II

Encryption in practice

Why do we use Bitwise XOR?

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

Instead AND or OR?

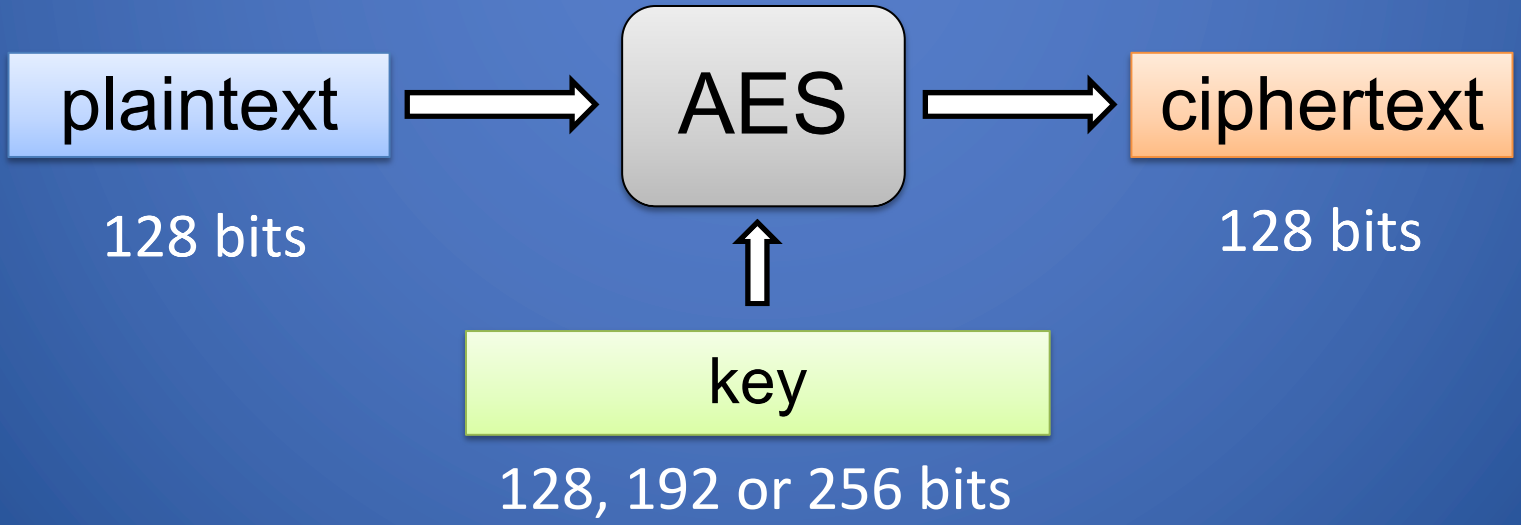
X	Y	$X \land Y$
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	$X \lor Y$
0	0	0
0	1	1
1	0	1
1	1	1

Block Ciphers

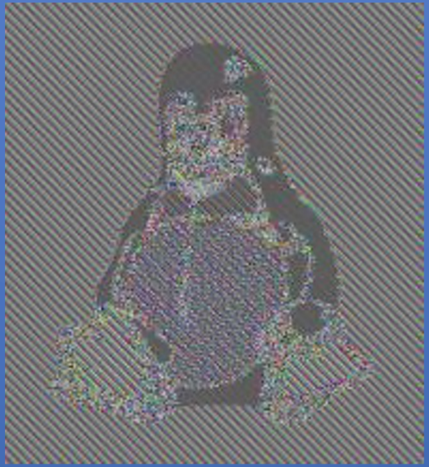
Block Cipher

- 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

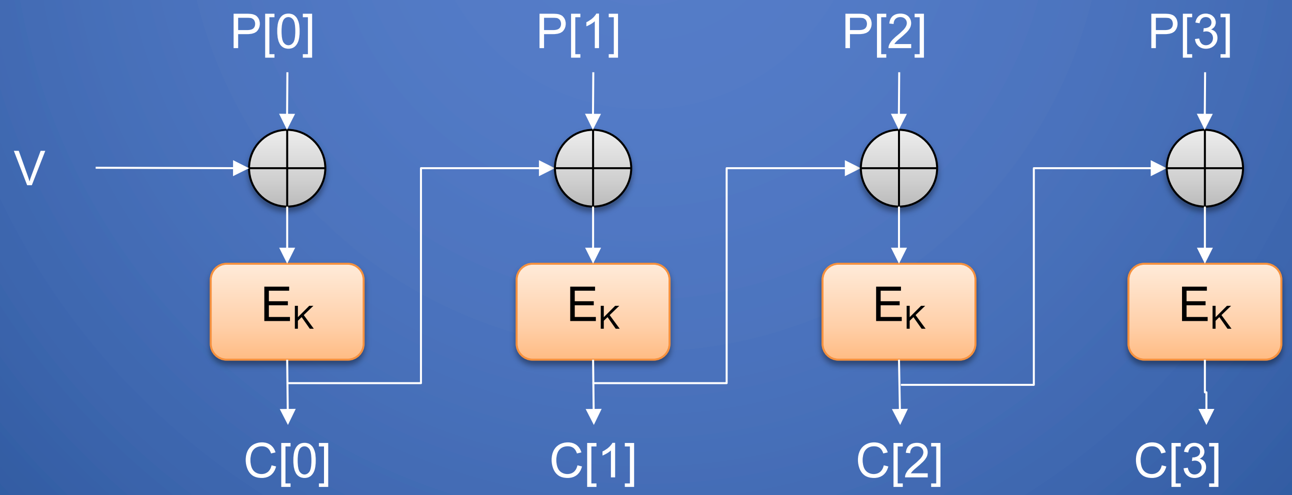


Source of images: Tux the Penguin created by Larry Ewing <lewing@isc.tamu.edu> with The GIMP and derived encrypted image by Lunkwill. Downloaded from https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

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

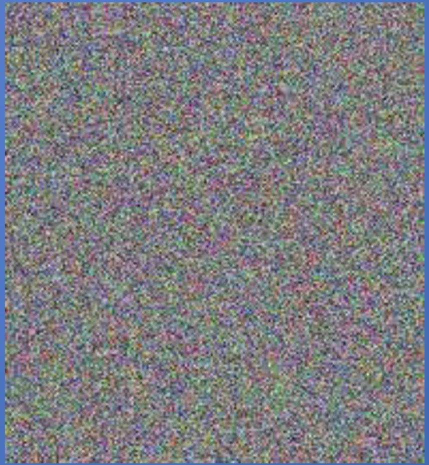
CBC Mode

- 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
  - To ensure that identical plaintexts encrypt to different ciphertexts, it's essential to use an initialization vector.



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
- Padding is the operation of filling the last block
- How to pad unambiguously the last block?
- We can use all zeros .
  - Problematic if the last character in the original block was already a zero

- Public-Key Cryptography Standards(PKCS) #7 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

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

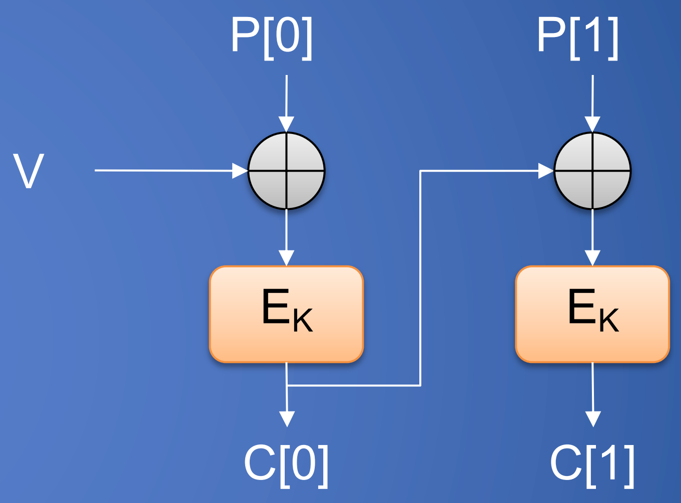
D. $E_k(P[0] \oplus V),\ \ E_k\left(P[1] \oplus E_k(P[0] \oplus V)\right)$

Clicker Question (1) - Answer

ANSWER: D

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

$$\begin{aligned} C &= C[0], C[1] \\ &= E_k(P[0] \oplus V),\ E_k(P[1] \oplus C[0]) \\ &= E_k(P[0] \oplus V),\ E_k\left(P[1] \oplus E_k(P[0] \oplus V)\right) \end{aligned}$$

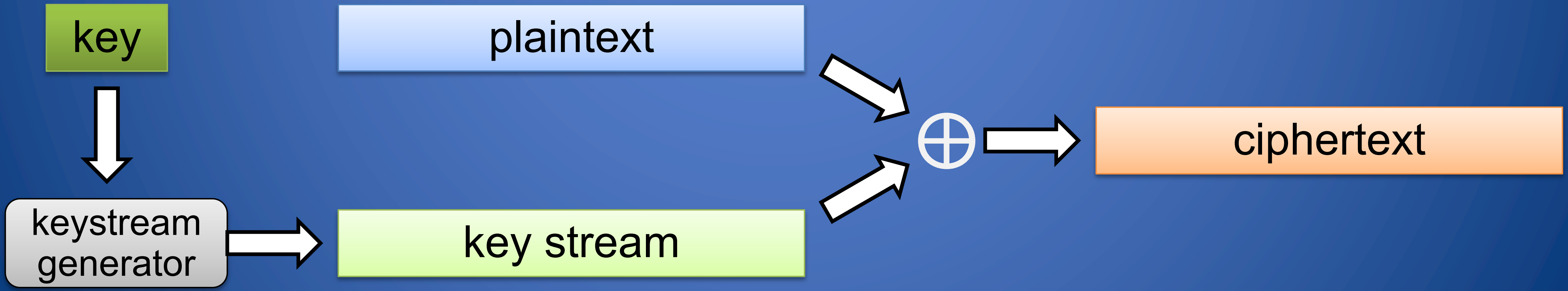


Stream Ciphers

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)
  - Works for packets sent over an unreliable channel
- Disadvantages
  - Key stream cannot be reused
  - Synchronization of plaintext/ciphertext with key stream
  - Sharing the key

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

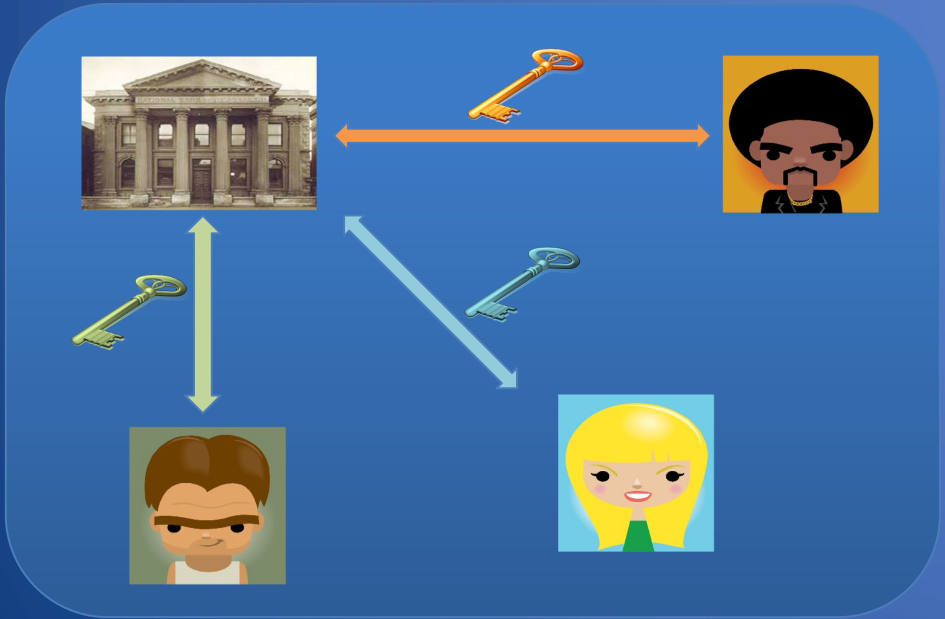
Clicker Question (2) - Answer

- 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

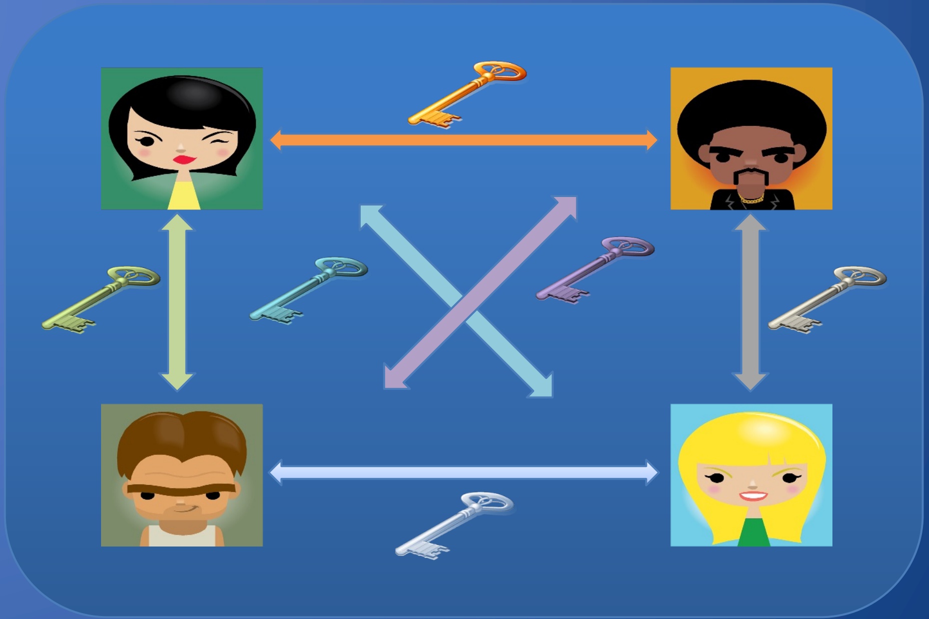
Public Key Cryptography

The problem of Key Distribution

- In symmetric Encryption distinct keys need to be set up for each pair of communicating users



- Quadratic number of keys for pairwise communication



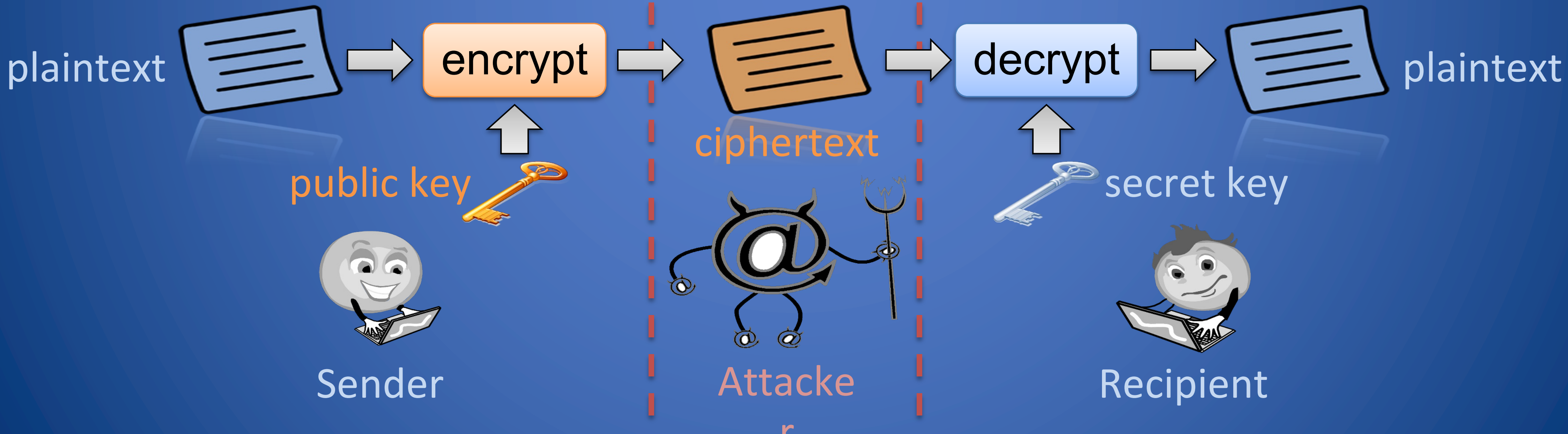
Public Key Cryptography

Key pair

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

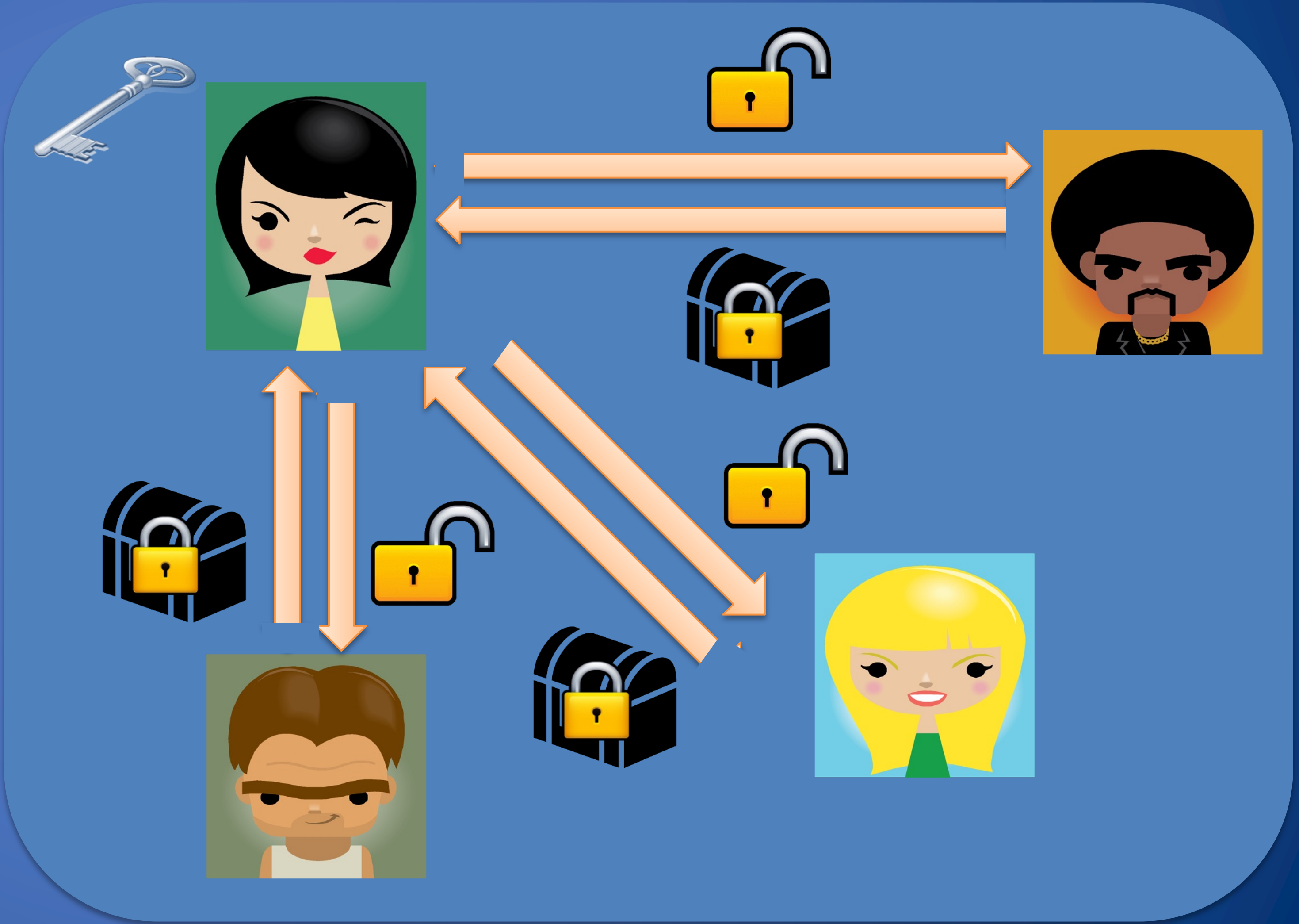
Protocol

- 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

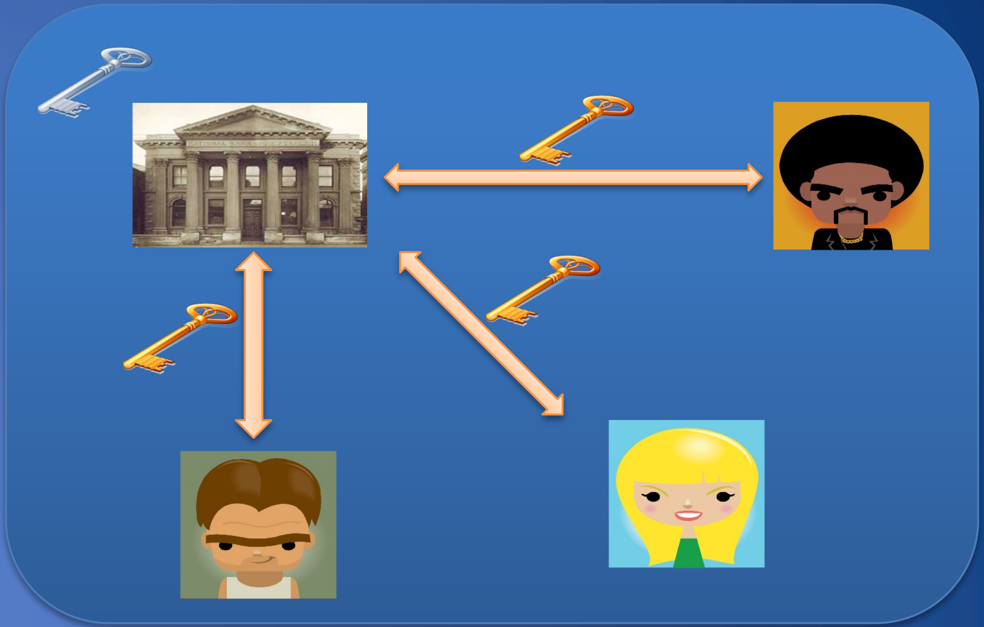


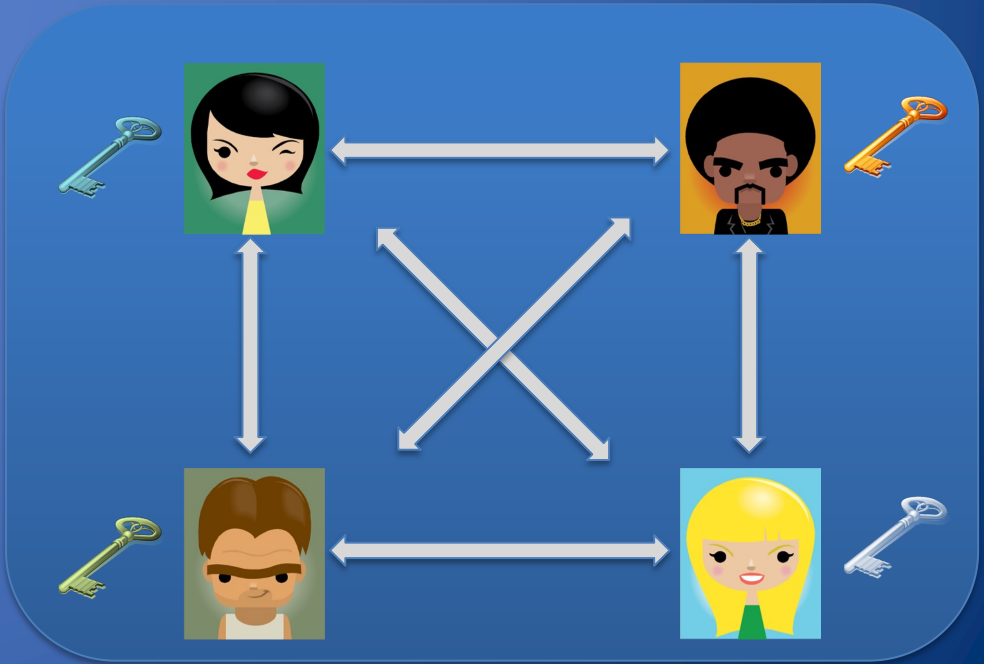
Public-Key Encryption in Formulas

- 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

- 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

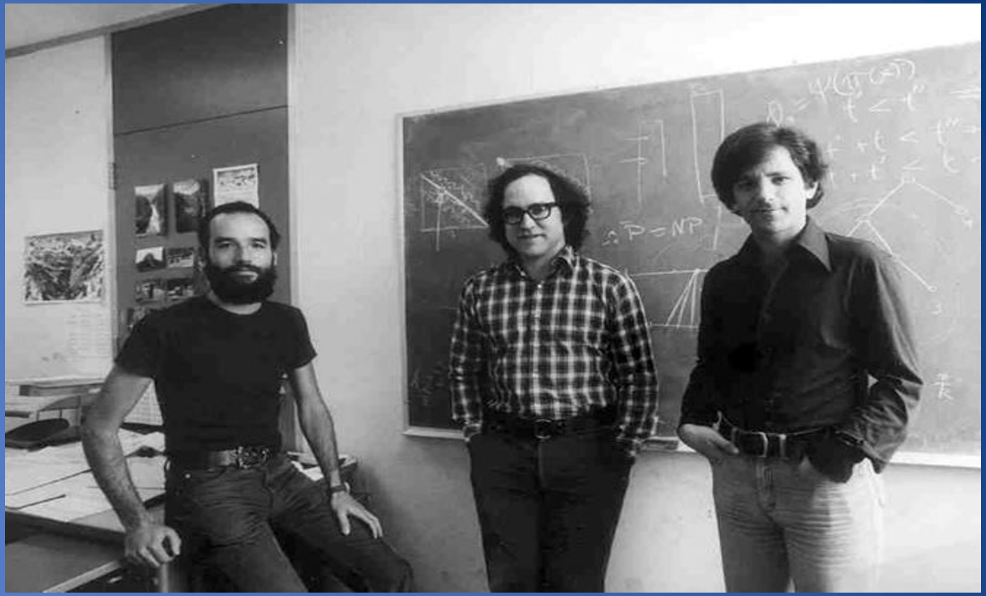


Image used with permission from Ron Rivest and Len Adleman

Image source: The Royal Society

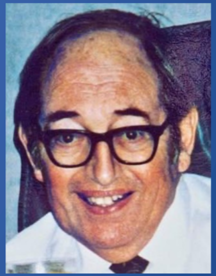


Image source: The Telegraph

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.

Asymmetric Crypto Systems

Algorithm	Year	Message Size Limitations	Performance Tradeoffs
Diffie-Hellman	1976	No direct use in encryption	Fast key exchange
RSA	1977	Limited by key size (e.g. 2048 or 4096 bits)	High security but slower, for larger key sizes.
Elliptic Curve Cryptography (ECC)	1985	Smaller key sizes compared to RSA (e.g., 256 bits)	Faster and more efficient with smaller keys, but implementation can be complex.
ElGamal	1985	Limited by key size, similar to RSA	Offers high security but less efficient than RSA
DSA (Digital Signature Algorithm)	1991	Primarily used for digital signatures, not encryption	Faster for signing but slower for verification compared to RSA.

Symmetric vs. Asymmetric

Characteristic	Symmetric Encryption	Asymmetric Encryption
Use Cases	Ideal for encrypting large amounts of data	Often used for secure key exchange, digital signatures, and short strings
Key Length	Shorter key lengths are often sufficient (e.g., 128 or 256 bits).	Requires longer key lengths to maintain security (e.g., 2048 bits or more).
Computational Resources	Requires less computational power.	More computationally intensive.
Security	Security depends on the secrecy of the key. Key distribution is an issue.	Provides a higher level of security, especially for key distribution.
Scalability	Less scalable due to the need for key management for each pair of users.	More scalable in environments with numerous users.
Data in transit and at rest	Commonly used for encrypting data in transit and data at rest	Used for exchanging the secure key for symmetric encryption.

BREAK!

Formalizing Encryption Security

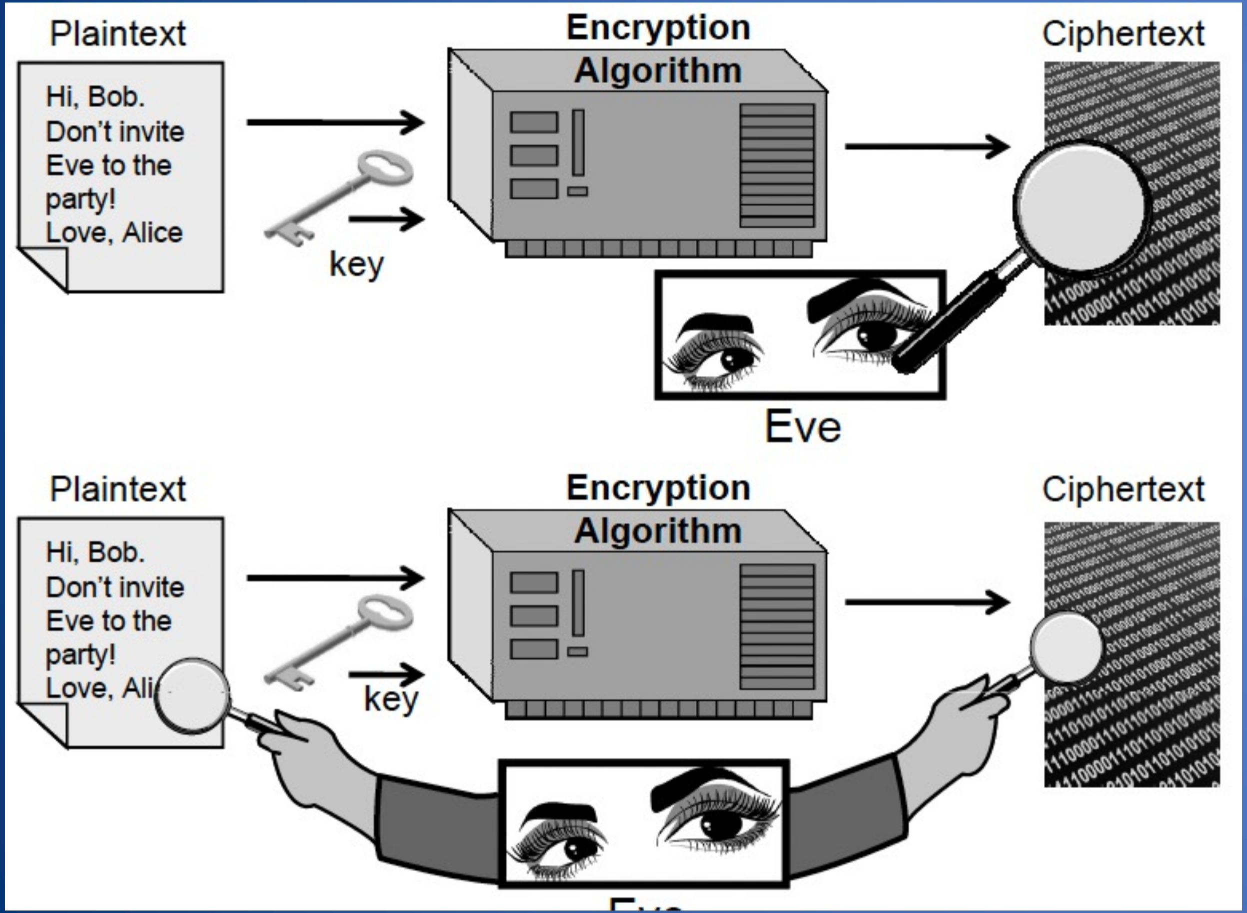
Adversary Models

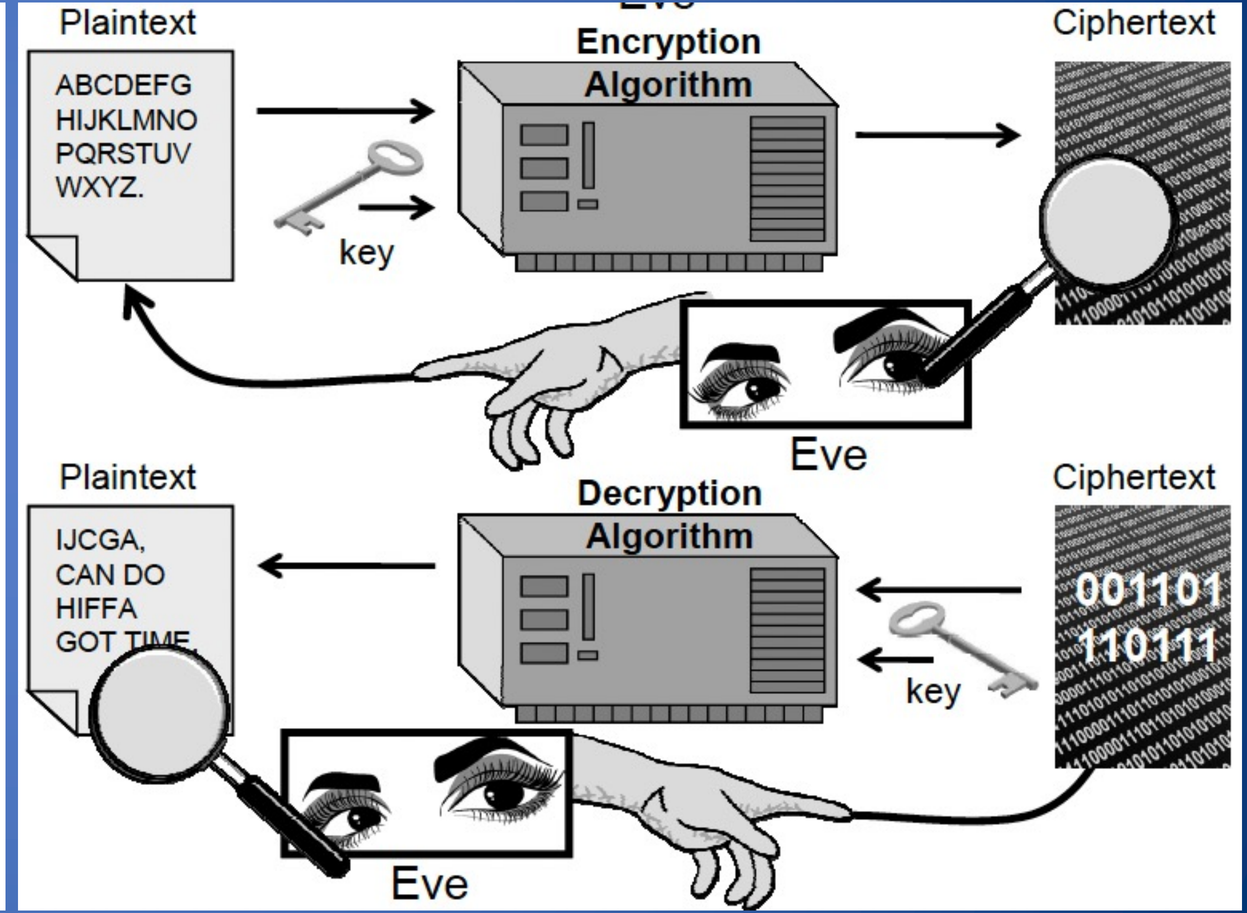
- Alice and Bob are sending encrypted messages to each
  - Adversary Eve can eavesdrop on those messages
  - …and maybe do other things as well
- Security goal: protect confidentiality w.r.t. Eve
  - Useful to formalize: What are Eve's capabilities as an adversary?

Threat modeling

Ciphertext-only

Chosen plaintext


Plaintext
Hi, Bob. Don't invite Eve to the party! Love, Alice
key
Encryption Algorithm
Ciphertext
Eve
Plaintext
Hi, Bob. Don't invite Eve to the party! Love, Ali
key
Encryption Algorithm
Ciphertext



Known plaintext

Chosen ciphertext

(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

$$c = Enc_k(m) \xleftarrow{m} \qquad \xrightarrow{c}$$

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

$$c' = Enc_k(m_i) \xrightarrow{c'}$$

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

Taking a step back: Integrity

Telephone game

- Who knows the telephone game?
- What is the goal?
- Integrity
  - Ensuring that data is not altered or tampered with during storage, transmission, or processing.
  - This means that the data received at the destination is exactly the same as the data sent by the source.

Hash Functions

- A hash function transforms
  - an input message or file of arbitrary length
  - into a fixed-length output value (e.g., 256 bits) called hash value
- A collision occurs when two distinct messages have the same hash value
  - Inevitable because there are more inputs than outputs
  - If two hashes are different, the inputs are different
  - The converse is not true

Cryptographic Hash Functions

- Short output
  - The hash value has small fixed length (e.g., 256 or 512 bits)
- One-way
  - It is hard to find a message with a given hash value
- Collision resistance
  - Given a message, it is hard to find a different message with the same hash value

Cryptographic Hash Functions

- Cryptographic hash function
  - Hash function with special properties
  - Not all properties always required
  - Public function, no secrets
- Only feasible attack to break a property is brute-force search
  - Length of hash value should be at least 256 bits (32 bytes)
- One-way
  - Given a hash value x, it is hard to find a plaintext P such that h(P) = x
- Weak collision resistance
  - Given a plaintext P, it is hard to find a plaintext Q such that h(Q) = h(P)
- Strong collision resistance
  - It is hard to find a pair of plaintexts P and Q such that h(Q) = h(P)
  - Birthday Paradox

Hashing People to Birthdays

- Define the birthday hash function as the mapping of a person to the month and date of birth (e.g., August 15)
  - 366 possible hash values
- Birthday paradox…
  - Suppose there are N students in a classroom
  - To be sure that at least two students have the same birthday N must be at least 367
  - How many people have the same birthday in this classroom? (DEMO)

Demo: Birthday survey

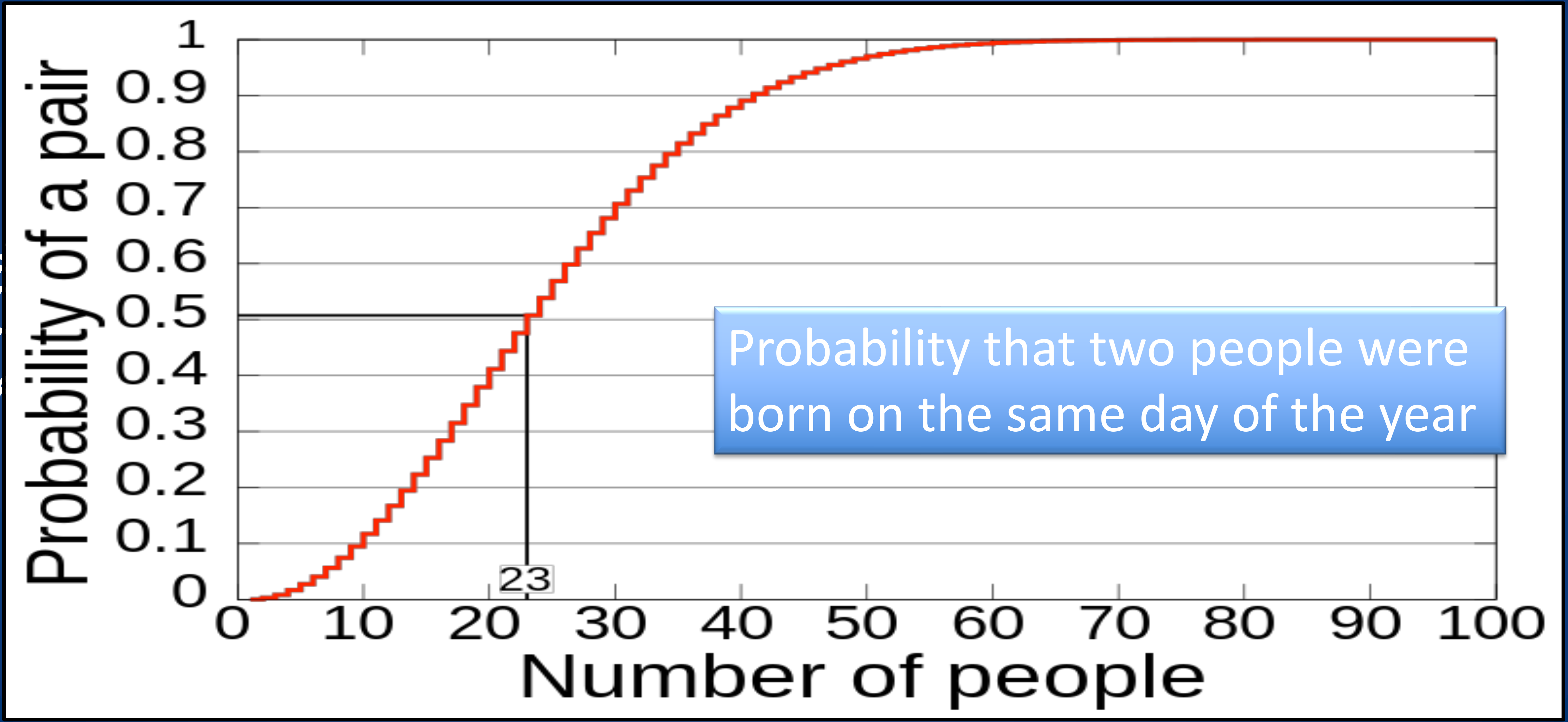
- Enter your birthday here:

https://forms.gle/rDwsU1wjdncDA1856

(Link also on lectures page of website)

- If you don't want to enter your birthday, just pick any random date

Birthday Paradox



Probability that two people were born on the same day of the year

Hash Functions vs. Hash table

- Hash function
  - Function h mapping plaintext P to fixed-length value x = h(P), called hash value or digest of P
  - Should take time proportional to length of plaintext
- Collision
  - Pair of plaintexts P and Q that map to the same hash value, h(P) = h(Q)
  - Collisions are unavoidable

- Hash table
  - Widely used data structure
  - Stores items into locations associated with hash values
  - Chaining or open addressing deal with collisions
  - Constant expected search time if hash function spreads items uniformly

Applications

- File integrity
  - Alice stores her files on a cloud server managed by Bob
  - She later retrieves the files
  - How can she make sure the files were not corrupted?
  - She wants something more efficient than keeping a copy of all her files
- Solution
  - Alice computes and keeps a crypto hash for each file
  - Security ensured by weak collision resistance
  - Efficient scheme since Alice stores short hashes (e.g., 32 bytes) instead of files

Applications

- Password authentication
  - How to authenticate users without storing passwords?
  - We want to avoid server breach to leak passwords
  - We want to defend against password-guessing attacks

- Solution
  - Store crypto hash of password but not the password
  - One-way makes it difficult to recover password from hash
  - Weak collision resistance makes it hard to guess other password with same hash

In Practice

- Practical hash functions
  - Functions widely believed to perform in practice like a cryptographic hash function
  - No mathematical proof that they satisfy the three properties
  - No significant attacks
  - Standardized by NIST

- MD5 (128 bits)
  - Developed by Ron Rivest (1991)
  - Considered insecure, do NOT use
- SHA-1 or RIPEMD160 (160 bits)
  - SHA-1 NIST 1995
  - RIPEMD Developed by Hans Dobbertin, Antoon Bosselaers and Bart Preneel (1996)
- SHA-2: different lengths (224, 256, 384, 512 bits)
  - Developed by the NSA (2002)
- SHA-3: Keccack (different number of bits)
  - Developed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche (2011)
  - Won SHA-3 Competition **(11/2/2007 – 10/2/2012)**
  - Not widely used

Let’s try together

- Practicing with different hashes
  - www.tools4noobs.com/online_tools/hash/
- Two different images same hash:
  - https://www.hacksandsecurity.org/posts/two-images-have-same-md5-hash-md5-collision-example
  - https://github.com/sunjw/fhash

Clicker Question (Th:821033)

Bob.com authenticates users by storing a cryptographic hash of each user's password in a server-side database. Which property of hash functions is most important when protecting against an attacker who has direct access to the password database?

A. One-way
B. Weak collision resistance
C. Strong collision resistance
D. All of the above

Clicker Question - Answer

Bob.com authenticates users by storing a cryptographic hash of each user's password in a server-side database. Which property of hash functions is most important when protecting against an attacker who has direct access to the password database?

A. **One-way**
B. **Weak collision resistance**
C. Strong collision resistance
D. All of the above

Summary

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