

Chapter 2: Cryptographic Building Blocks

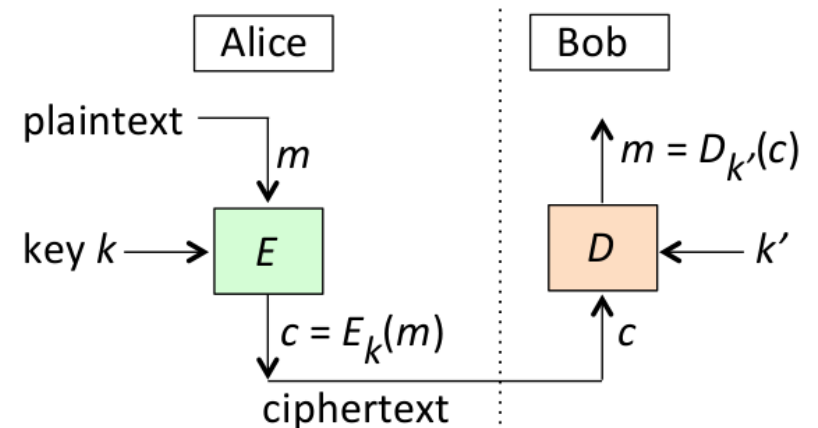
SYSC 4810: Introduction to Network and
Software Security

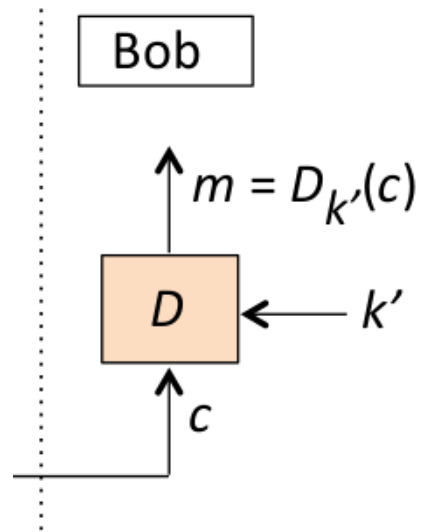
Prof. Hala Assal



Encryption and Decryption

- Encryption transforms *plaintext* into *ciphertext*.
- Encryption is reversible.
- Access to the *decryption key* controls access to the plaintext
 - only authorized parties are given the key.
- Encryption algorithms are typically non-secrets.
- Transmitted or stored secret data should be encrypted.



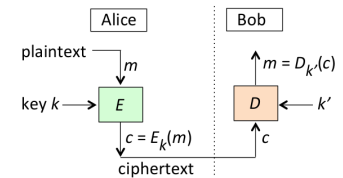


Fundamental Property

- It should be infeasible to recover **m** from **c** without knowledge of **k'**.
- A good encryption algorithm requires key brute force.
 - Thus, increasing the key space exhausts the adversary's trials.
 - How many expected trials for a key of size **n** bits?
 - And what does that entail?
 - How would the adversary know if the key was correct?

Attack Models

What is the adversary's objective?



Ciphertext-only attack	Attacker has ciphertext Tries to recover PT (or key)
Known-plaintext attack	Attacker has CT, PT pair Tries to recover unknown PT or key from another CT
Chosen-plaintext attack	Attacker choose some amount of plaintext and see the resulting ciphertext.
Chosen-ciphertext attack	Attackers can provide ciphertext of their choosing, and receive back the corresponding plaintext.

Attack Models

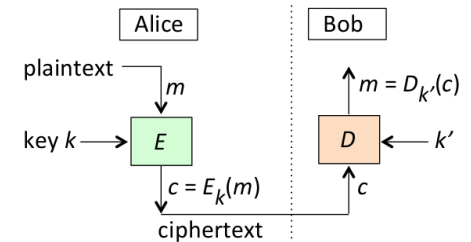
Ciphertext-only attack	Attacker has ciphertext Tries to recover PT (or key)	Passive adversary
Known-plaintext attack	Attacker has CT, PT pair Tries to recover unknown PT or key from another CT	
Chosen-plaintext attack	Attacker choose some amount of plaintext and see the resulting ciphertext.	Active adversary
Chosen-ciphertext attack	Attackers can provide ciphertext of their choosing, and receive back the corresponding plaintext.	



Symmetric Encryption



Symmetric Encryption



- AKA: symmetric-key, secret key
- Encryption and decryption keys are the same
 - ie, $k = k'$
- Examples:
 - One-time pads – or Vernam cipher; 1949
 - Data Encryption Standard (DES)
 - International Data Encryption Algorithm
 - Advanced Encryption Standard (AES); 1998

Vernam Cipher – One-time Pads

- Message length = key length = cipher length
- $c = m \oplus k$
- Unbreakable! More formally: “information theoretic secure”
 - Secure against an unbound adversary!
- Also: “Perfectly Secret.” Why?
 - All cipher space is possible.
 - Since any $k \in K$ is equally likely, any $c \in C$ is likewise.
 - Probability of c depends on k .
- Observing c gives the adversary no new information.
 - Only the message length
- One-time pads are impractical due to:
 - key distribution challenges
 - key length

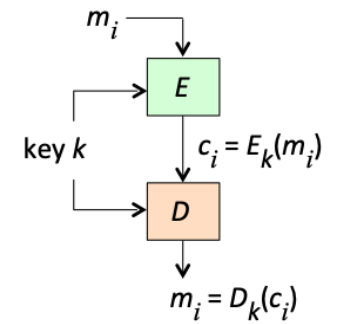
Computational Security

- Important term – leads to practicality
- Means protection against attackers modeled to have *fixed computational resources*
- ...and thus assumed unable to exhaustively try all keys in huge key spaces.
- In contrast, **information-theoretic security**:
 - The algorithm does not depend on unproven assumptions about computational hardness.

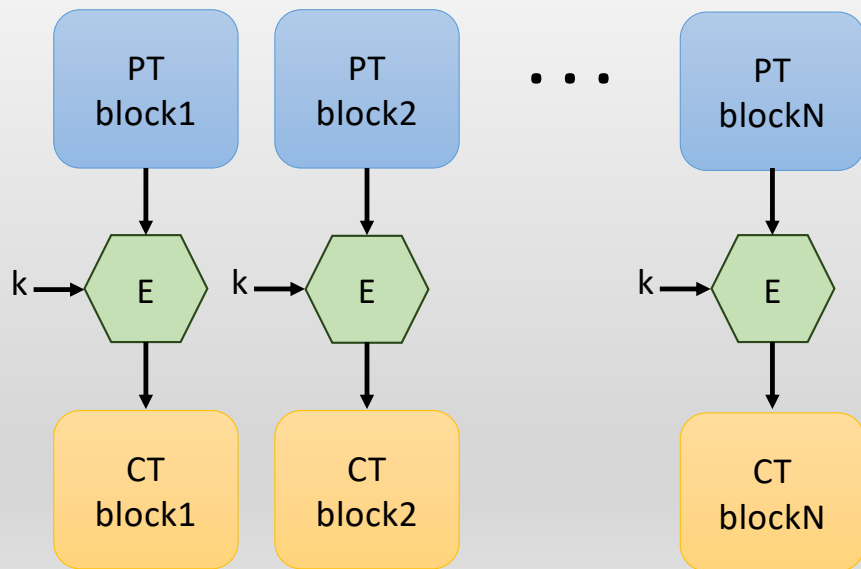
Cipher Modes

- Stream cipher
 - Encrypts bits in a stream
- Block ciphers
 - More common (and protective)
 - Parameters: block length, key distribution on blocks
 - Padding
 - Examples:
 - Advanced Encryption Standard (AES) 🏆
 - Electronic Code-book (ECB) – **insecure!**
 - Cipher-Block Chaining (CBC)
 - Cipher Feedback (CFB)
 - Counter (CTR)
 - Etc...

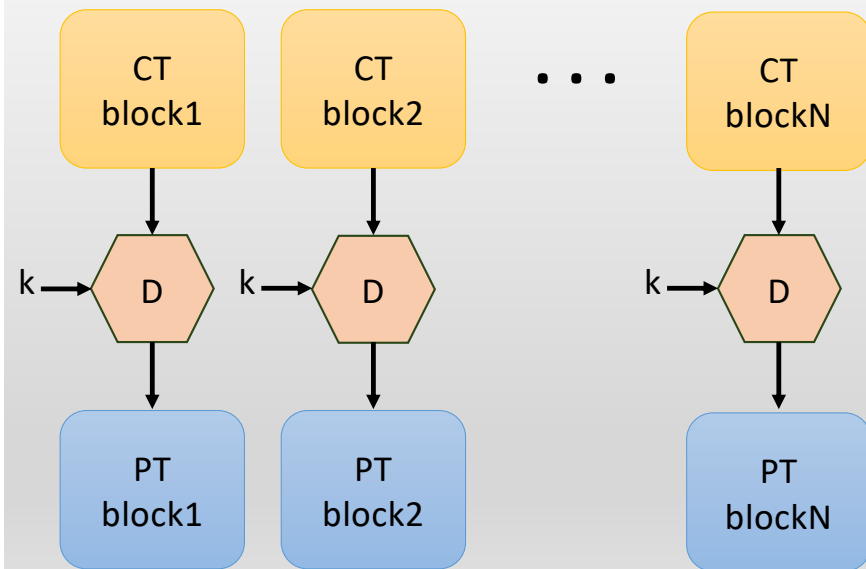
Electronic Code-book (ECB)



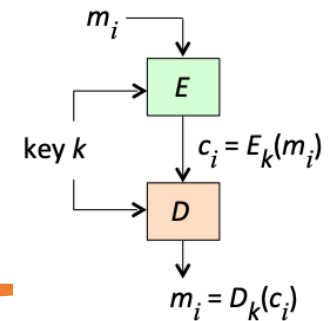
Encryption



Decryption



Electronic Code-book (ECB)



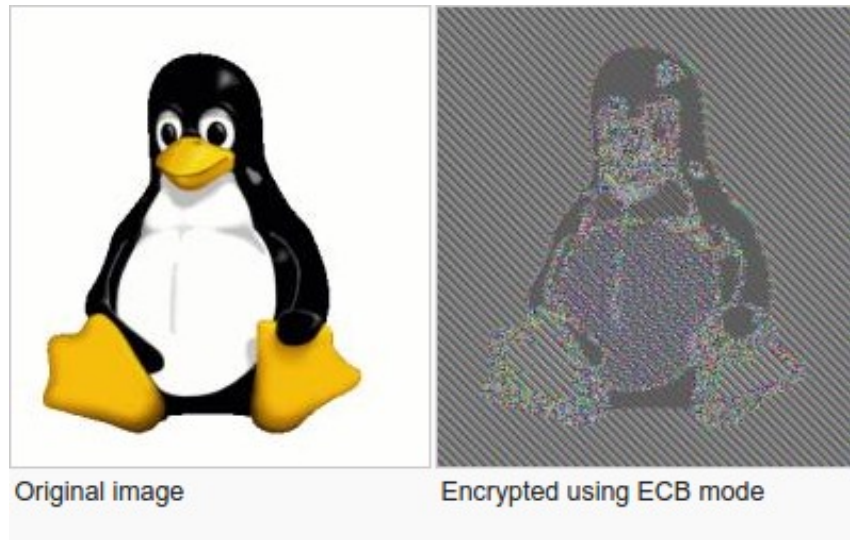
- Error propagation?
 - Error in block c_i affects only c_i
 - So errors don't propagate
- Encryption can be parallelized
- Decryption can be parallelized
- *What if one bit of plaintext is changed?*
 - Only one block needs to be recomputed
 - Good, e.g., for disc encryption, “random access” possible

Electronic Code-book (ECB)

- Is insecure
 - Also was shown to be vulnerable to chosen plain-text attack
- Was only used in the past

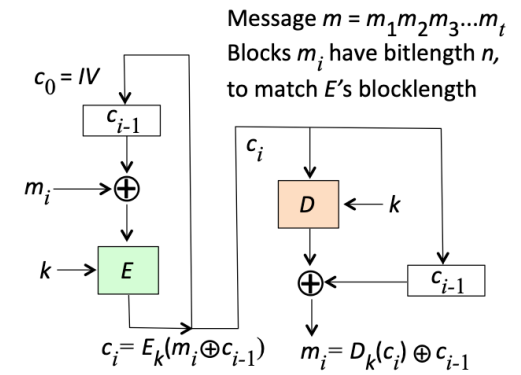
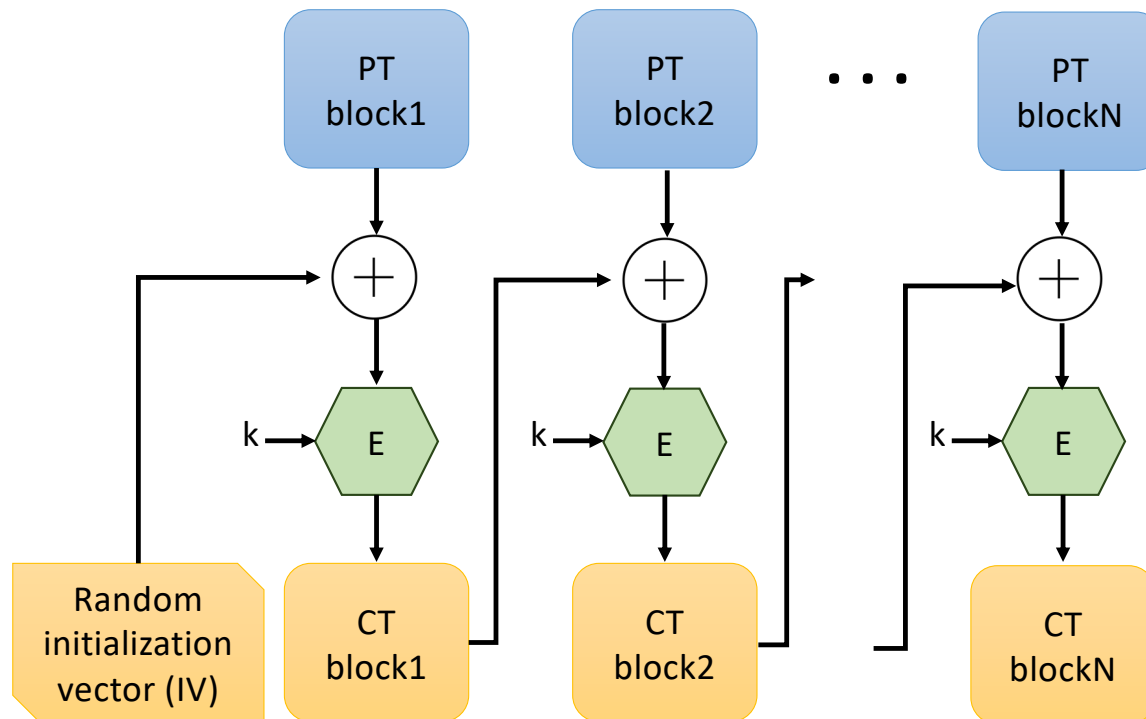
ECB Penguin Demonstration:

<https://github.com/robertdavidgraham/ecb-penguin>

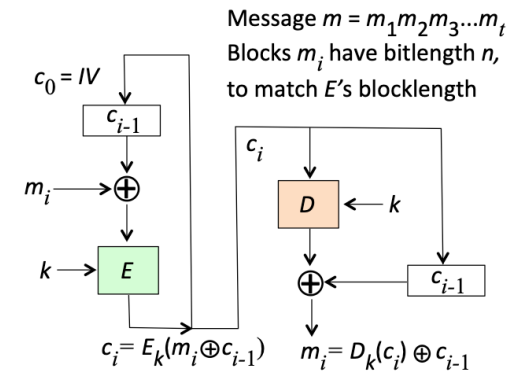
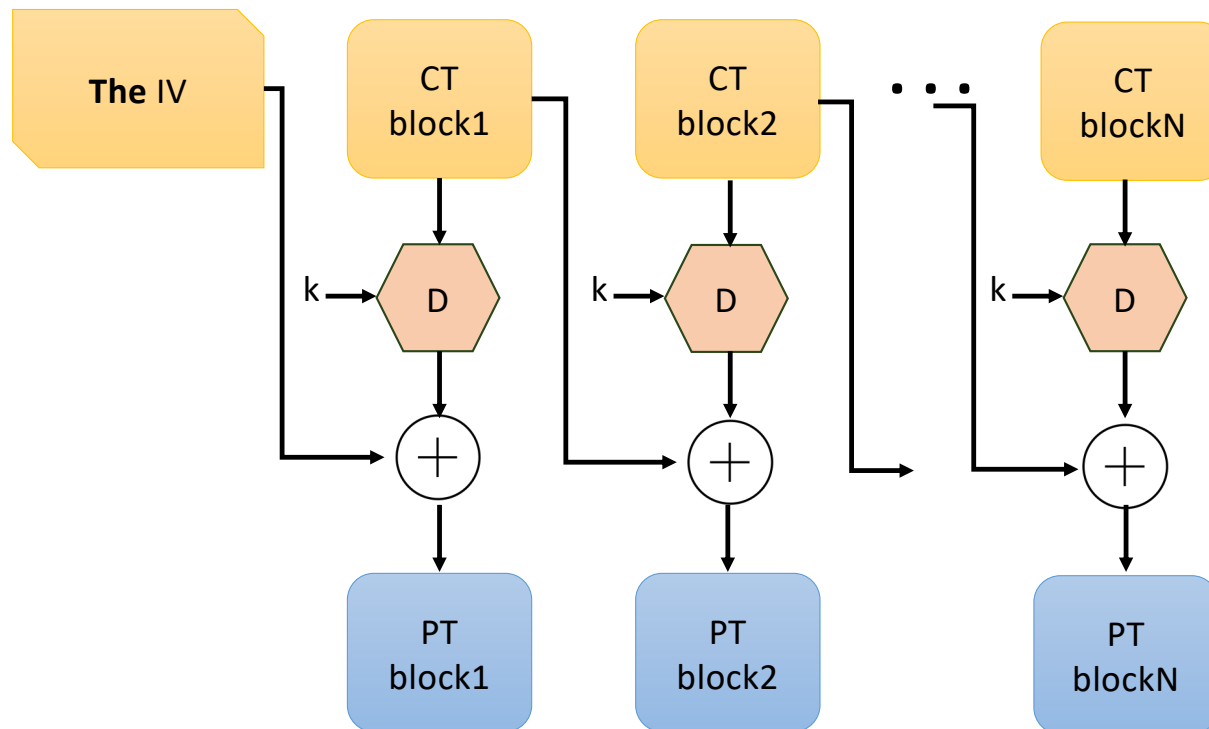


[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

Cipher-Block Chaining (CBC)

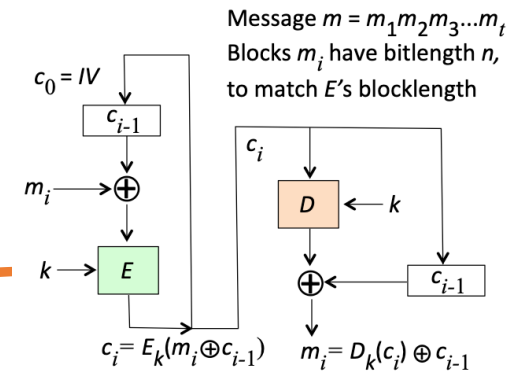


Cipher-Block Chaining (CBC)



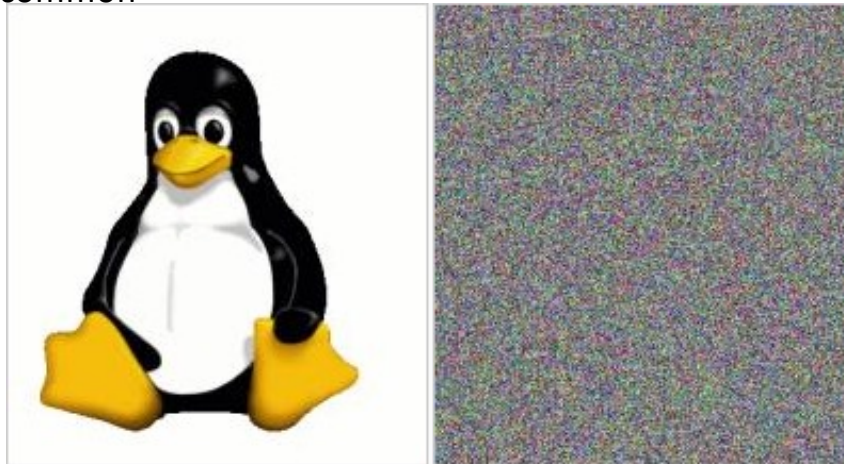
Cipher-Block Chaining (CBC)

- Error propagation?
 - Error in block c_i affects only c_i and c_{i+1} .
- Encryption **cannot** be parallelized
- Decryption **can** be parallelized
- What if one bit of plaintext is changed?
 - Everything needs to be recomputed!



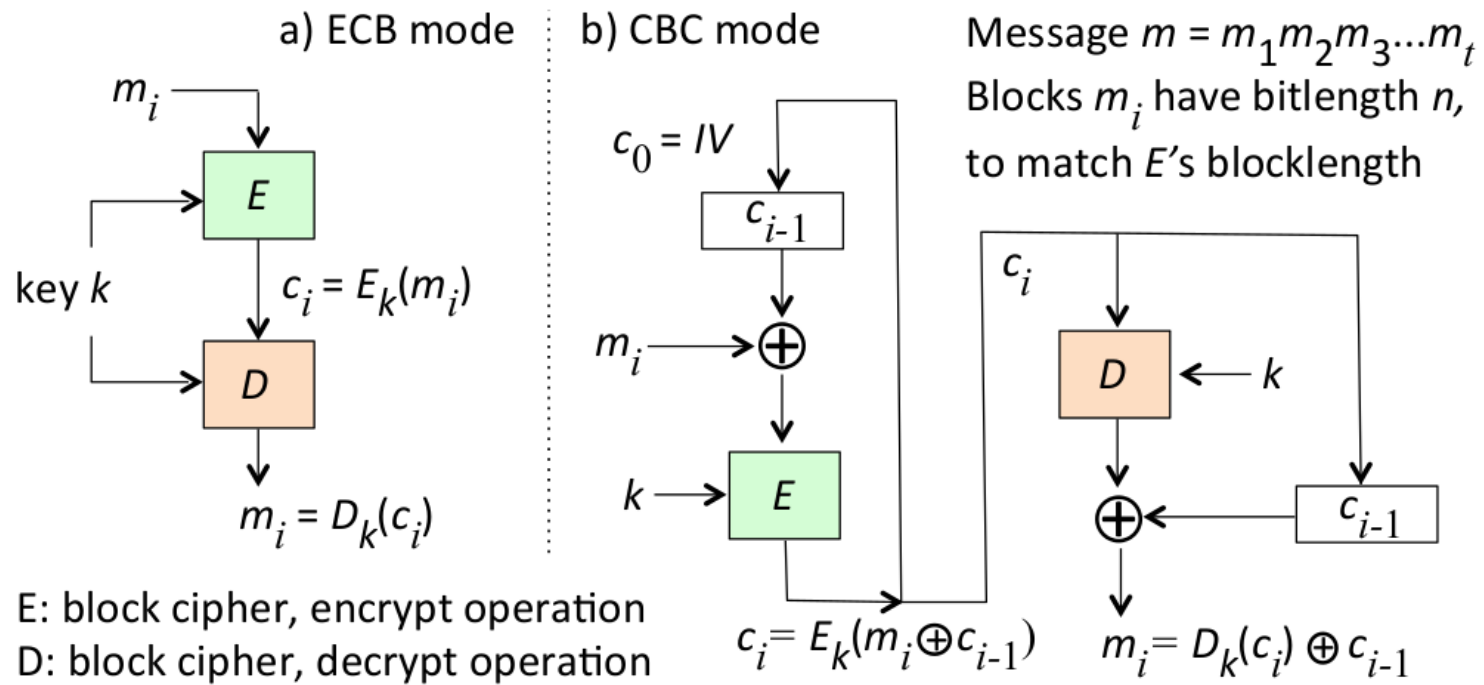
Cipher-Block Chaining

- Better than ECB
- Seldom used now
 - CTR is typically more common

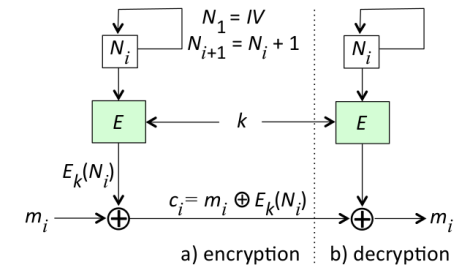


[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

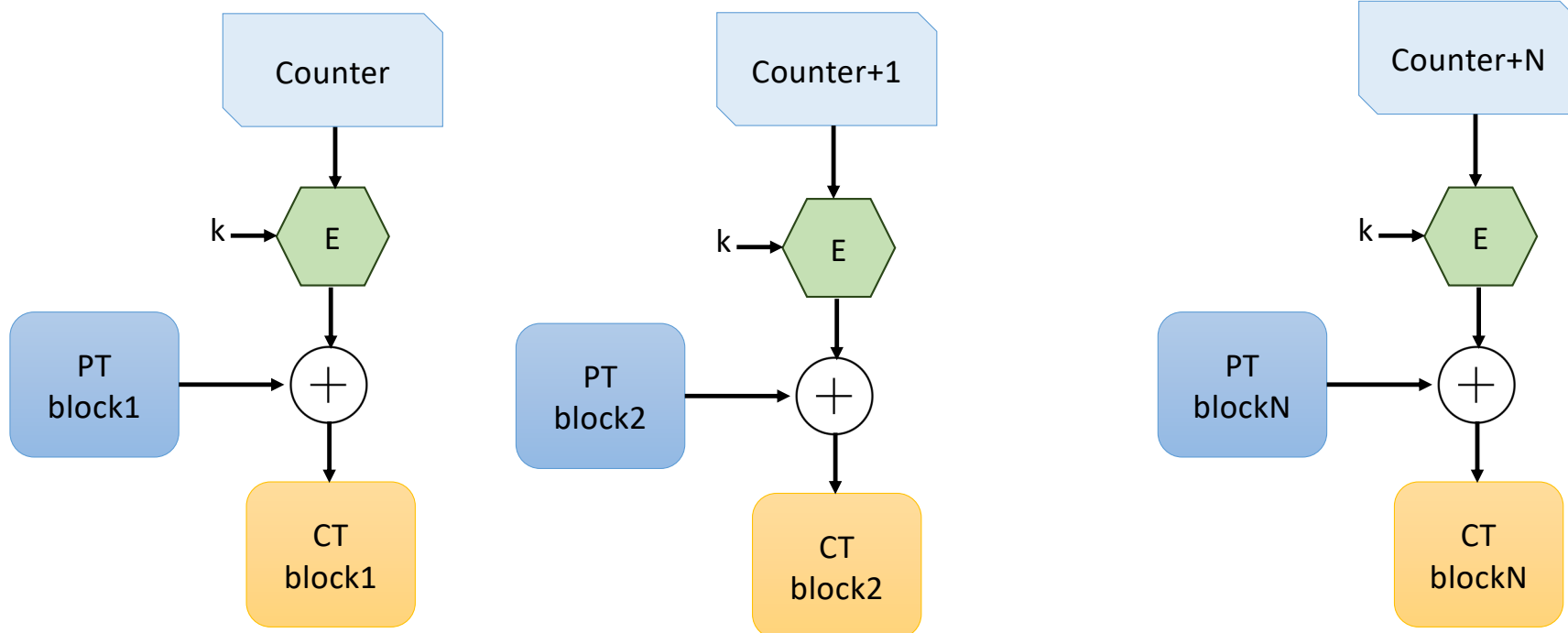
ECB vs CBC



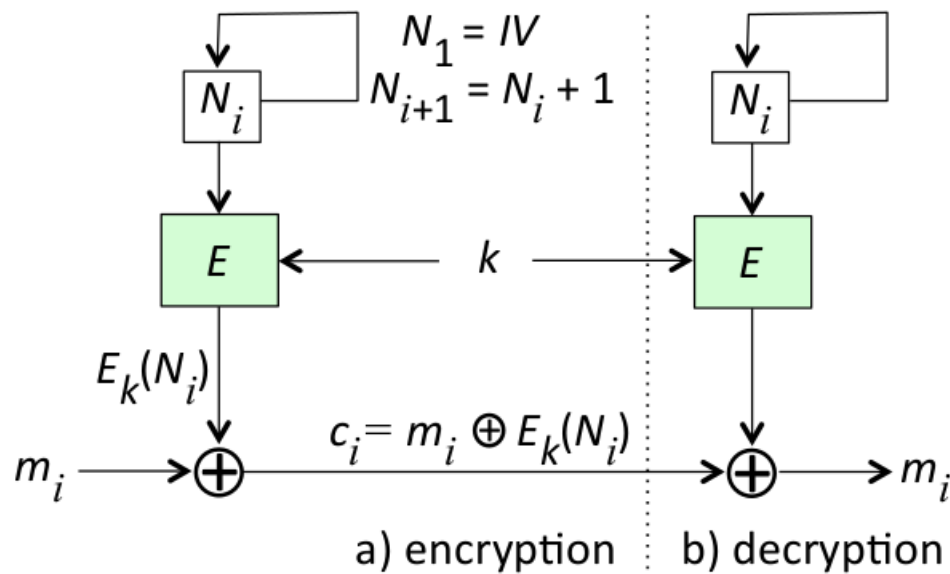
Counter (CTR)



...



Counter (CTR)



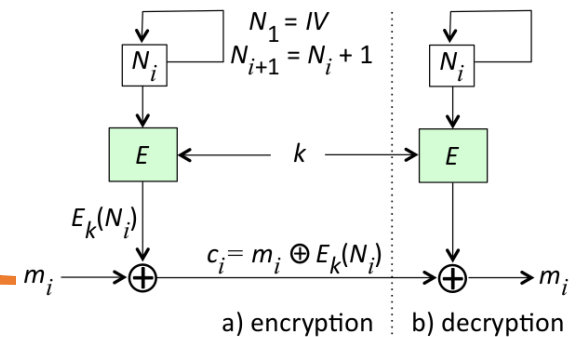
Counter (CTR) mode

Message $m = m_1 m_2 \dots m_t$
is encrypted to yield
ciphertext $c = c_1 c_2 \dots c_t$

Blocks m_i, c_i are n bits

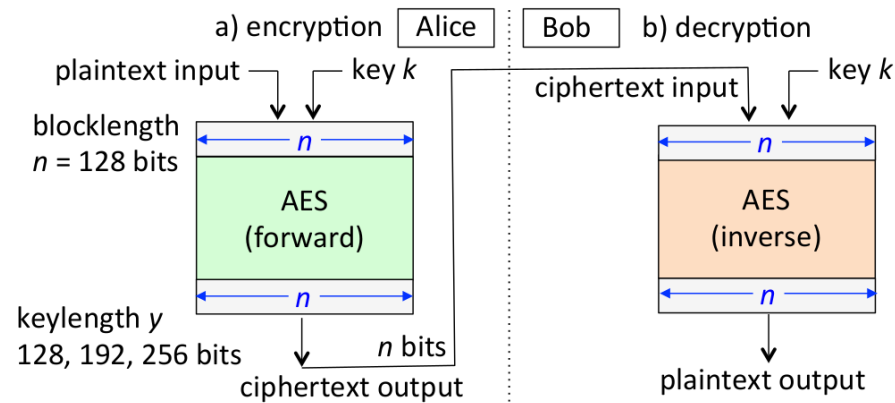
Counter (CTR)

- Must ensure that $IV + i$ never repeats
- Error propagation?
 - Error in block c_i affects only c_i
 - So like ECB, errors don't propagate
- Encryption can be parallelized
- Decryption can be parallelized
- What if one bit of plaintext is changed?
 - Only one block needs to be recomputed
 - Again like ECB, allows random access



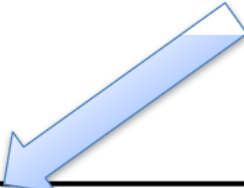
Advanced Encryption Standard (AES)

- Also called **Rijndael**.
- Resulted from a 1998 NIST competition calling for symmetric encryption standard.



Examples

A great design.
The only practical weakness: **short key**.
Can be broken by a **brute-force attack**.



	key length	block length
DES (1976) (Data Encryption Standard)	56	64
IDEA (1991) (International Data Encryption Algorithm)	128	64
AES (1998) (Advanced Encryption Standard)	128, 192 or 256	128



Asymmetric Encryption



Asymmetric Encryption

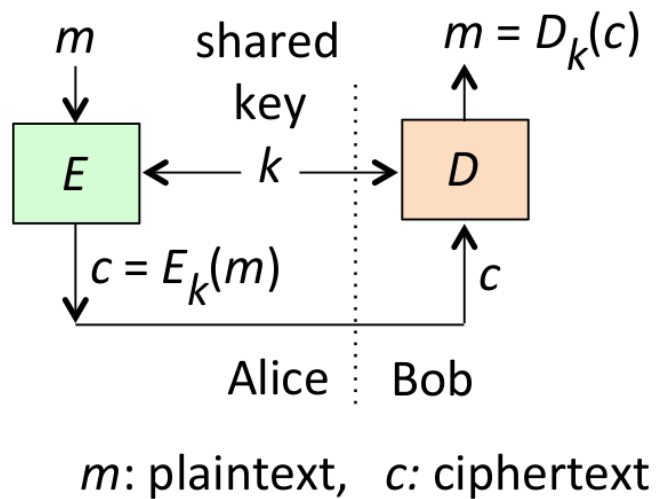
- AKA: *Public Key Encryption*

$$c = E_{e_B}(m) ; m = E_{d_B}(c)$$

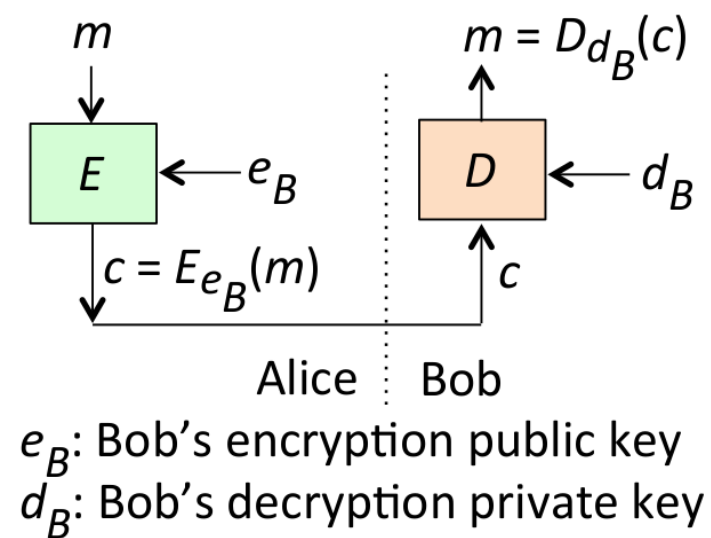
- How to communicate e_B ?
- How to protect d_B ?
- How many keys to distribute?
- How many keys are needed among n parties?
 - For symmetric: ?
 - For asymmetric: ?

Symmetric vs Asymmetric

a) Symmetric-key encryption

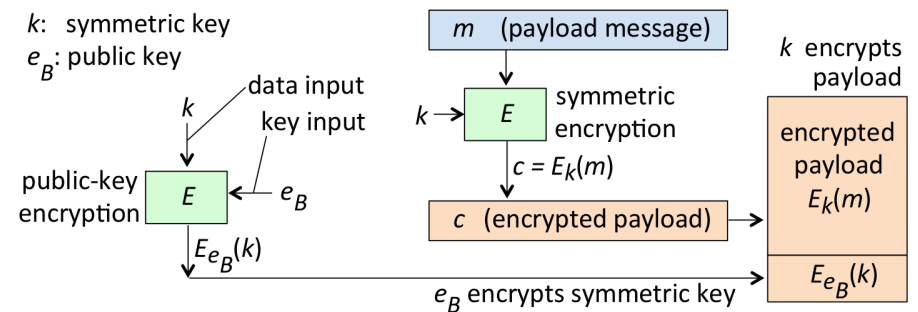


b) Public-key encryption



Hybrid Encryption

- Asymmetric Encryption is slow.
- Hybrid is most commonly used in practice.
- Combines the benefits of both encryption tactics.



RSA

- The first popular public-key (i.e., asymmetric) encryption.
- Ronald Rivest, Adi Shamir and Leonard Adleman. MIT, 1977.

Key generation

Choose numbers p, q	p, q are prime and $p \neq q$
Compute $n = pq$	
Compute $\phi(n) = (p-1)(q-1)$	totient (a number of numbers $< n$ & coprime to n)
Choose e	e is relative prime to $\phi(n)$; $1 < e < \phi(n)$
Compute d	$ed \bmod \phi(n) = 1$, d is a multiplicative inverse of e
public key	$PU = \{e, n\}$
private key	$PR = \{d, n\}$

Encryption

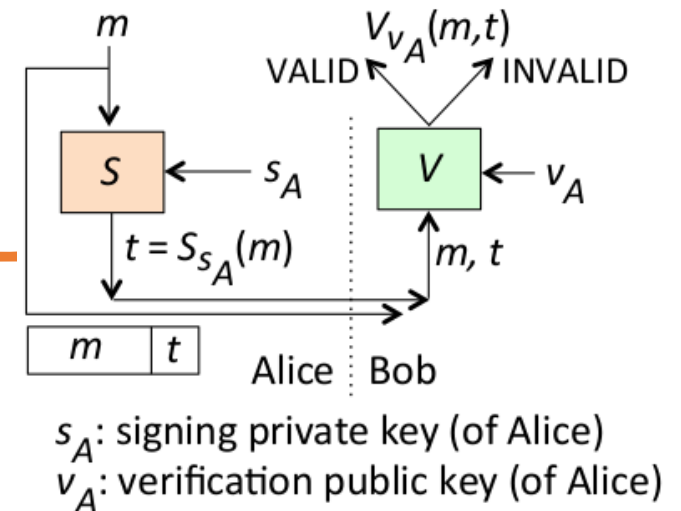
Plaintext:	$m < n$
Ciphertext:	$C = m^e \bmod n$

Decryption

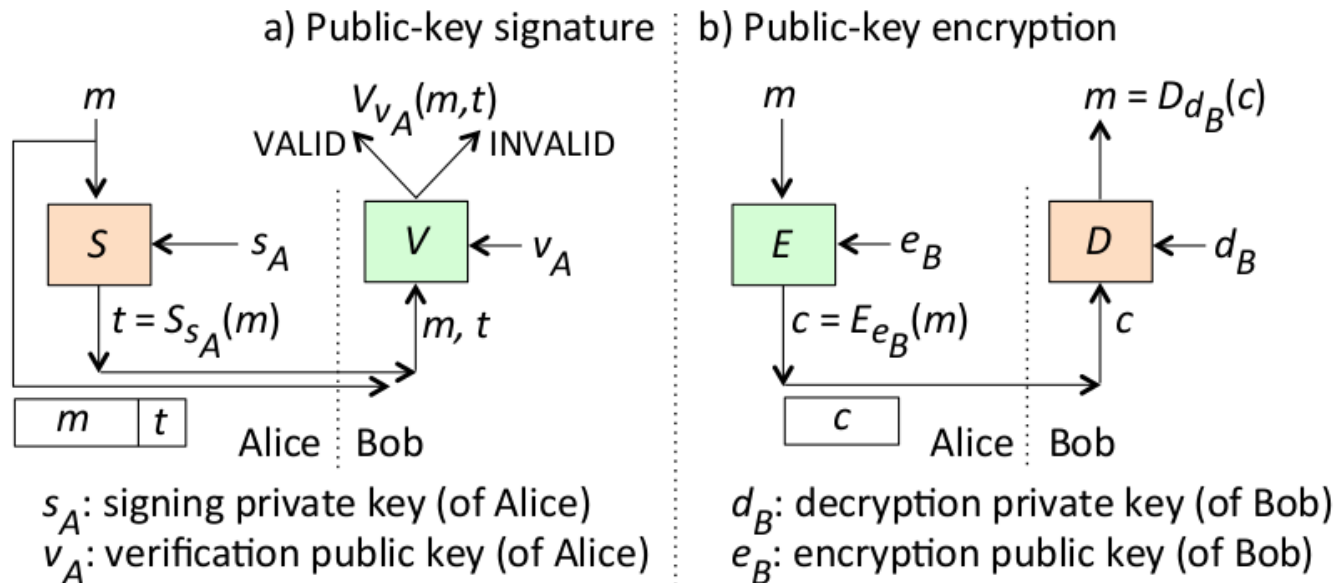
Ciphertext:	C
Plaintext:	$m = C^d \bmod n$

Digital Signatures

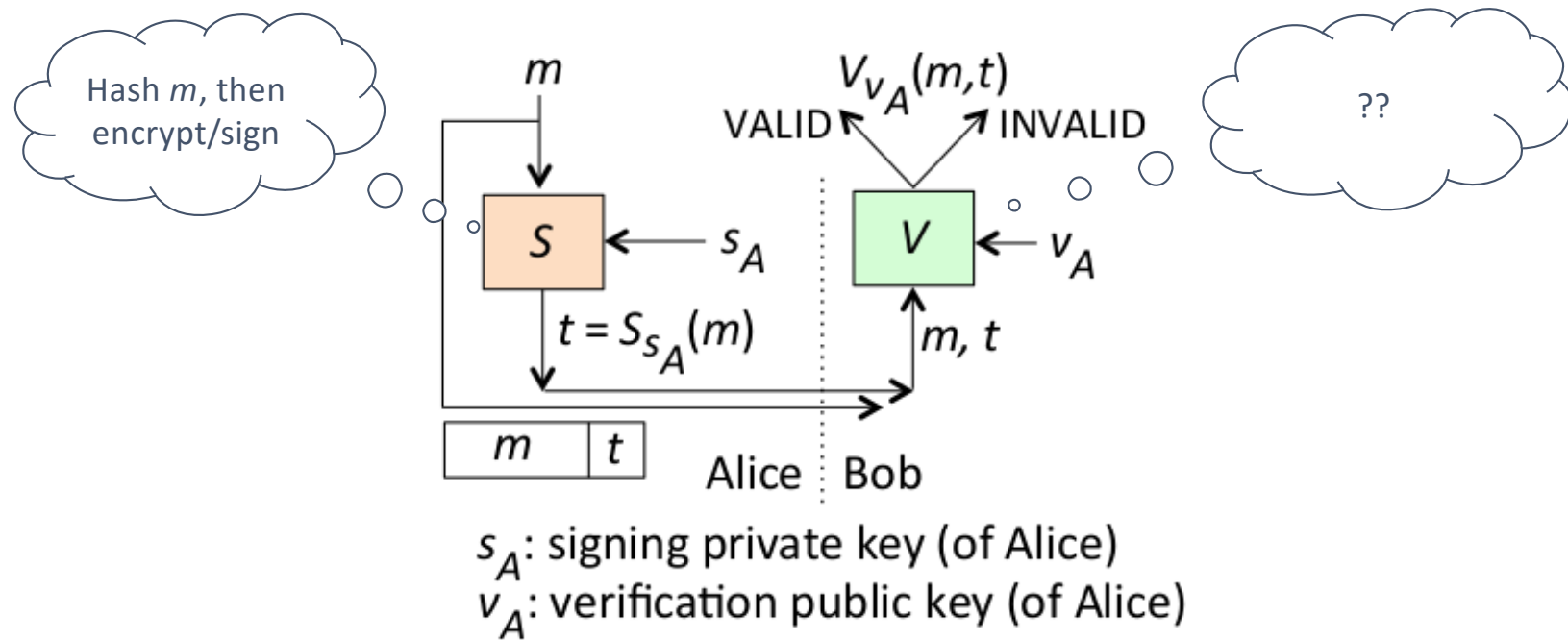
- Tags that accompany messages: based on **m** and **PR**
- When you encrypt with a *private key*, you generate a digital signature.
 - Now called “**signing key**”
- Decryption with the corresponding *public key* verifies the signature.
 - Now called “**verification key**”
- Additional mechanisms should be used to prevent message replay
 - e.g., buying a box of printer paper through a signed message? Receiving 100 boxes.
- Currently used mostly for authentication.
- Properties provided:
 - Data origin authentication
 - Data integrity
 - Non-repudiation
 - But in practice, can a digital signature hold in court for that purpose?
- Instead of saying “encrypt with the private key”, say “**sign**”



Digital Signatures



Digital Signatures

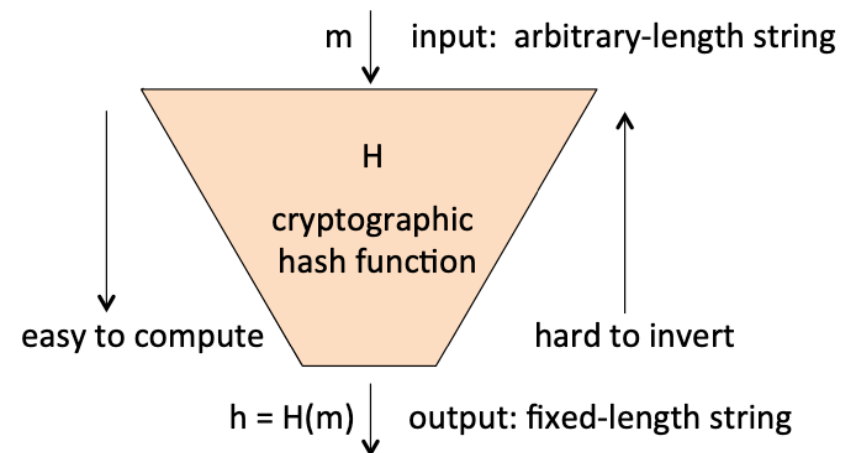


Cryptographic Hash Functions

- They take as input any binary string
 - (e.g., message or file)
- ... and produce a fixed-length output called:
 - a **hash value, hash, message digest or digital fingerprint**.
- They typically map longer into shorter strings.
- They have special properties.
- Not to confuse with regular, non-crypto, hashing

Cryptographic Hash Functions

- A hash value is a compact representation intended to be associated with a unique input.
- For a good hash function, changing a single bit results in entirely unpredictable output changes.
 - 50% of output bits change on average.
- Hashes are often used as a type of secure checksum whose mappings are too complex to predict or manipulate -- and thus hard to exploit.



Cryptographic Hash Functions: Properties

H1

One-way property (preimage resistance):

for essentially all possible hash values h , given h it should be infeasible to find any m such that $H(m) = h$.

H2

Second-preimage resistance:

given any first input m_1 , it should be infeasible to find any distinct second input m_2 such that $H(m_1) = H(m_2)$.

Note: there is free choice of m_2 but m_1 is fixed. $H(m_1)$ is the target image to match; m_1 is its preimage.

H3

Collision resistance:

it should be infeasible to find any pair of distinct inputs m_1, m_2 such that $H(m_1) = H(m_2)$.

Note: here there is free choice of both m_1 and m_2 .

When two distinct inputs hash to the same output value, we call it a collision.

Cryptographic Hash Functions: Properties

H1

One-way property (preimage resistance):

for essentially all possible hash values h , given h it should be infeasible to find any m such that $H(m) = h$.

H2

Second-preimage resistance:

given any first input m_1 , it should be infeasible to find any distinct second input m_2 such that $H(m_1) = H(m_2)$.

H3

Collision resistance:

it should be infeasible to find any pair of distinct inputs m_1, m_2 such that $H(m_1) = H(m_2)$.

- **H2** fails to guarantee **H3**
 - **H3** is harder to achieve
 - Birthday paradox
- **H3** implies **H2**
- **H3** almost always, but not always, implies **H1**
- **H2** almost always, but not always, implies **H1**

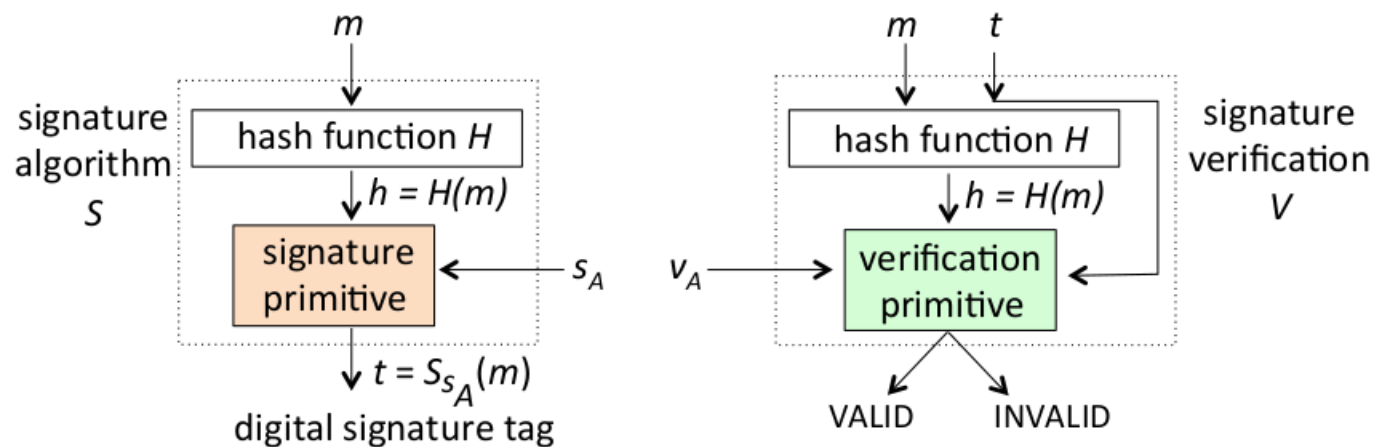
Cryptographic Hash Functions: Computational Security

- Consider a hash function that outputs 128 bits
- For an input of 512 bits, each 2^{512} should map to one of 2^{128} possible outputs.
- So, about 2^{384} will result in the same hash.
- Pigeonhole theory.
- Hence, we say “infeasible”, because it **is** possible.
- If it is infeasible but possible, it is computationally secure
 - But not “information-theoretically secure”

Cryptographic Hash Functions

Family name	Year	Output size		Alternate names and notes
		bitlength	bytes	
SHA-3	2015	224, 256	28, 32	SHA3-224, SHA3-256
		384, 512	48, 64	SHA3-384, SHA3-512 (NOTE 1)
SHA-2	2001	256, 512	32, 64	SHA-256, SHA-512
SHA-1	1995	160	20	Deprecated (2017) for browser certificates
MD5	1992	128	16	Widely deprecated, for many applications

Revisiting Digital Signatures

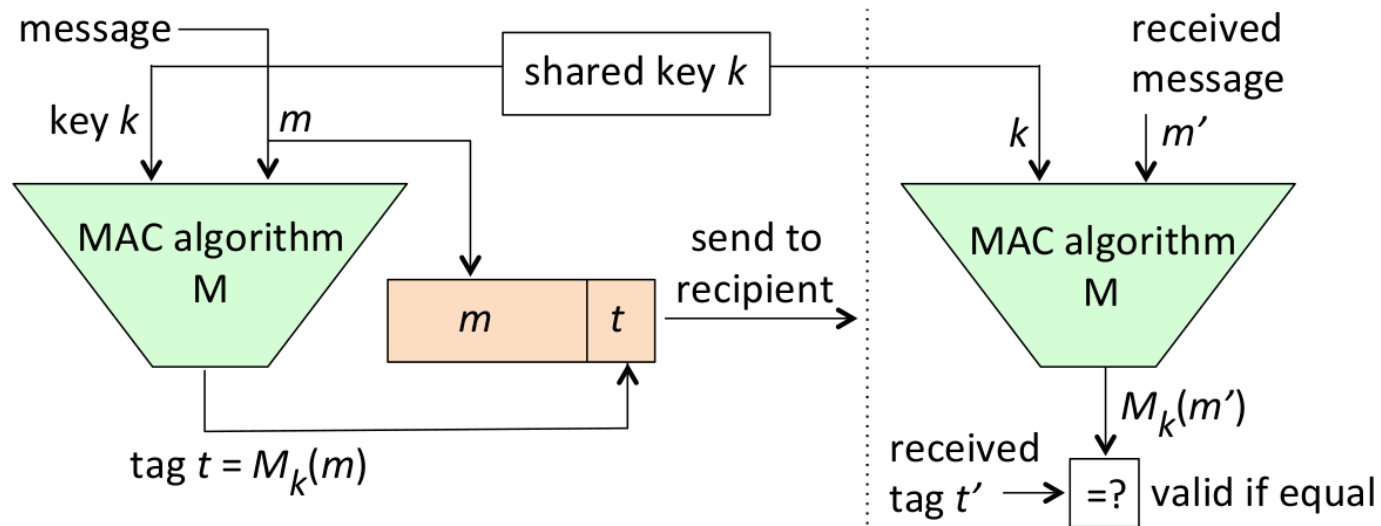


Message Authentication Codes (MAC)

*A tag sent with the message, that is computed using a MAC function which input is the **message** and a **secret key***

- Can be used for integrity protection
 - Like cryptographic hash functions
- Is also used for *data-origin authentication*
 - Unlike hash functions, because anyone can create a hash – no keys are needed to hash.
- MAC vs Digital Signature?
 - Highly comparable achievements
 - But different uses in practice.
 - MAC is symmetric encryption
 - Much faster than digital signatures
 - But requires shared key establishment
 - Therefore, typical for continuous streams of data, unlike digital signatures which are typically used to sign a document.

Message Authentication Codes (MAC)



Message Authentication Codes (MAC)

- Do MACs provide non-repudiation?
 - ??
- Do they, on their own, ensure freshness?
 - ??

Message Authentication Codes (MAC)

- Do MACs provide non-repudiation?
 - No, because the key is naturally shared among more than one party.
- Do they, on their own, ensure freshness?
 - No, because an attacker could replay the same message, with its MAC