Cryptography Primer

CS-576 Systems Security

Instructor: Georgios Portokalidis

Fall 2018

Overview

Symmetric encryption

Public-key encryption

Hashing and message authentication codes

Public-Key Encryption

Publicly proposed by Diffie and Hellman in 1976

Based on mathematical functions

- …on the practical difficulty of factoring the product of two large prime numbers

Asymmetric

- Uses two separate keys a public and a private key
- Public key is made public for others to use

Multiple algorithms with different uses

- Establish a shared secret key
- Encrypt a message
- Digital signatures

Requirements for Public-Key Cryptosystems

Computationally easy …

- … to create key pairs
- … for sender knowing public key to encrypt messages
- … for receiver knowing private key to decrypt ciphertext

Computationally infeasible …

- … for opponent to determine private key from public key
- … for opponent to otherwise recover original message

Useful if either key can be used for each role

Symmetric vs Asymmetric

Which one is best?

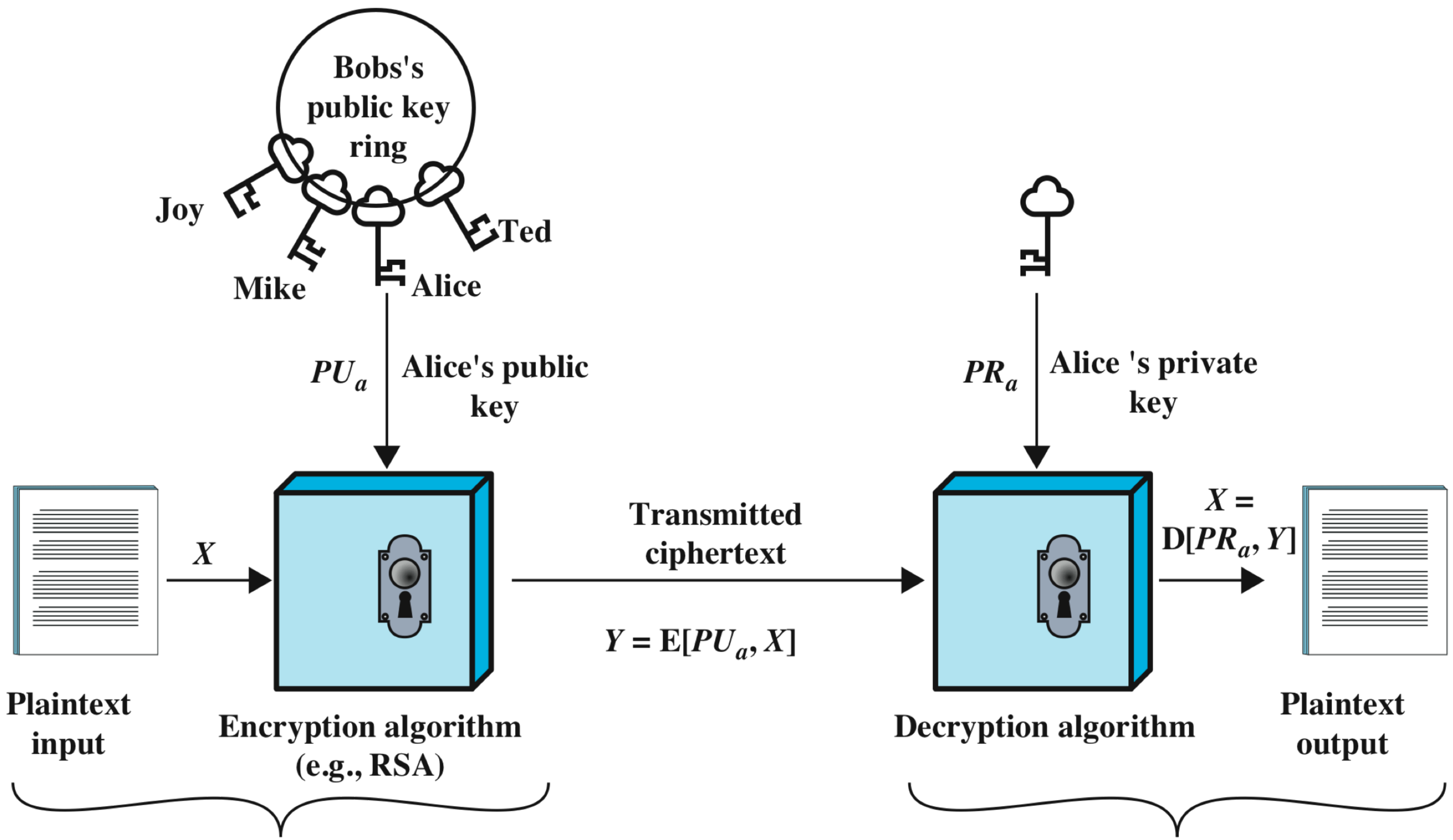
The strength of public-key cryptography depends more heavily on the length of the key

Intrinsically both offer similar guarantees against cryptanalysis

Public-key encryption is usually slower

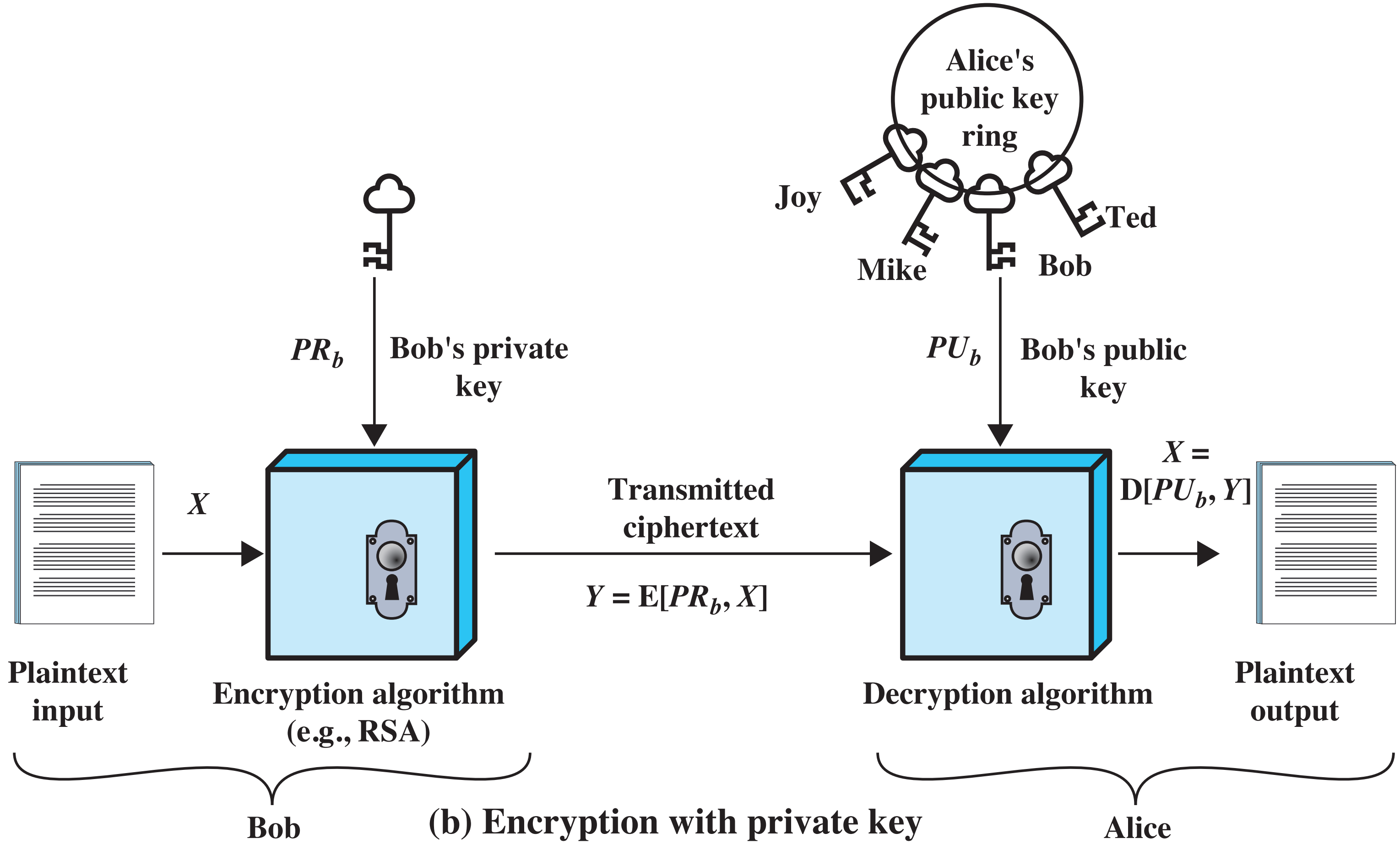
A shared key must be kept secret, similarly to the private key, but unlike the public key

Encryption with Public Key



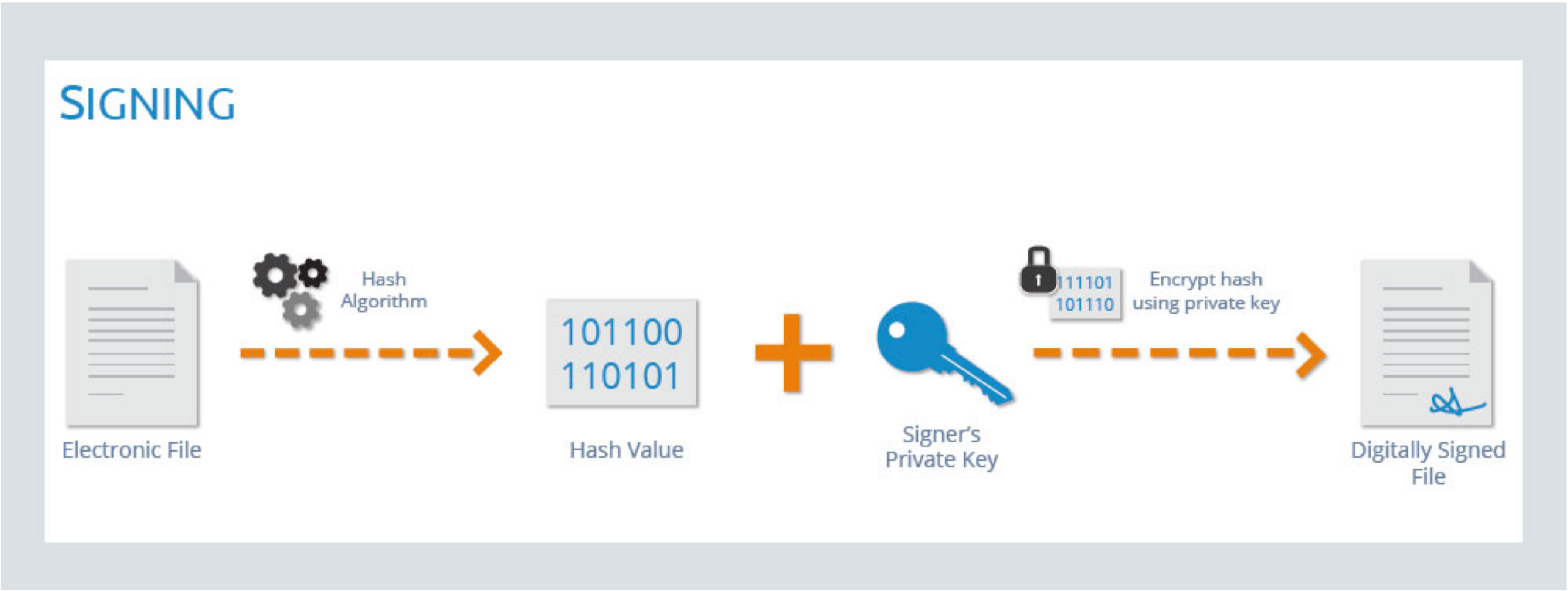
(a) Encryption with public key

Encryption with Private Key



(b) Encryption with private key

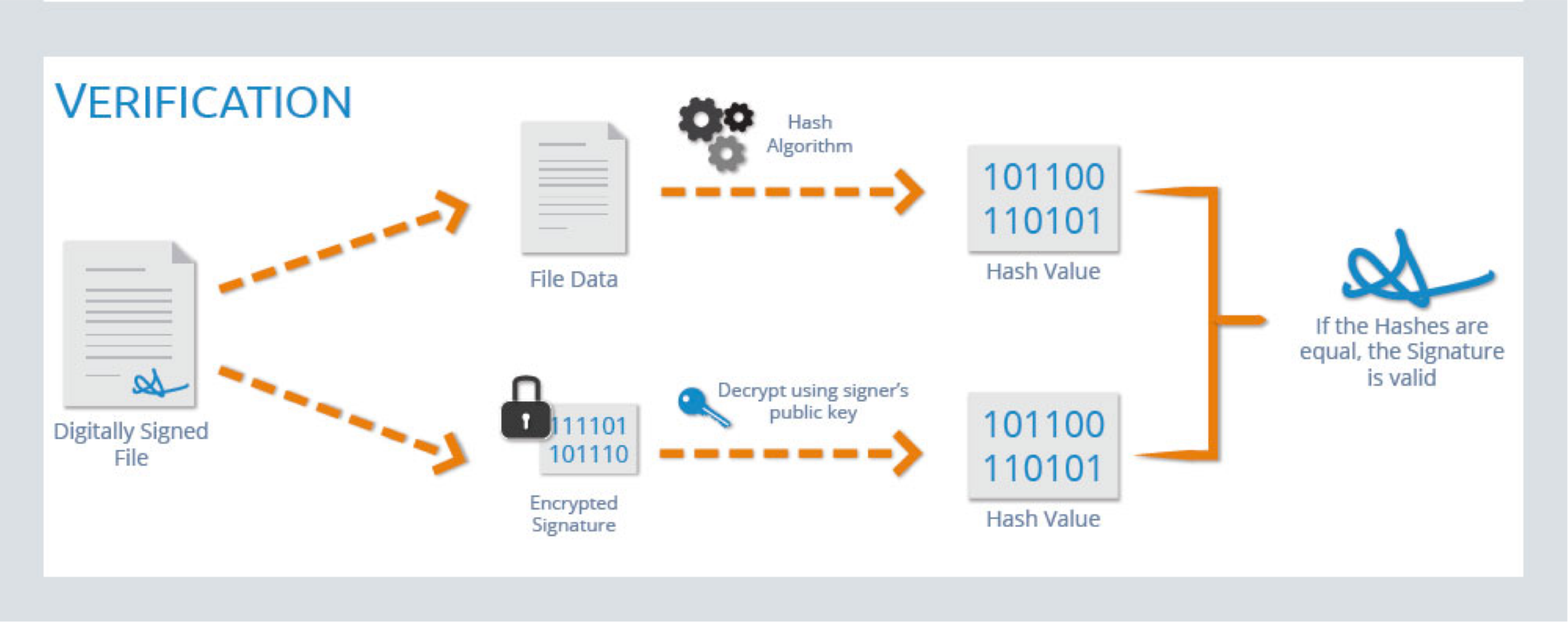
Digital Signing



Digital Signing

Verify ...

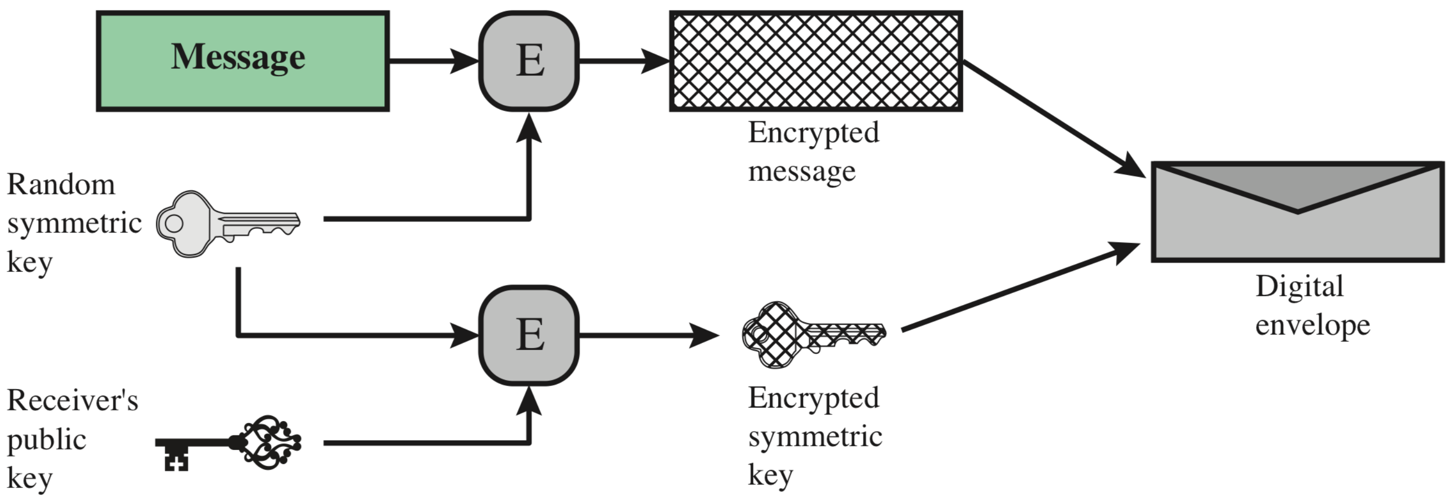
- ... the author of data
- ... the integrity of data



Digital Envelopes

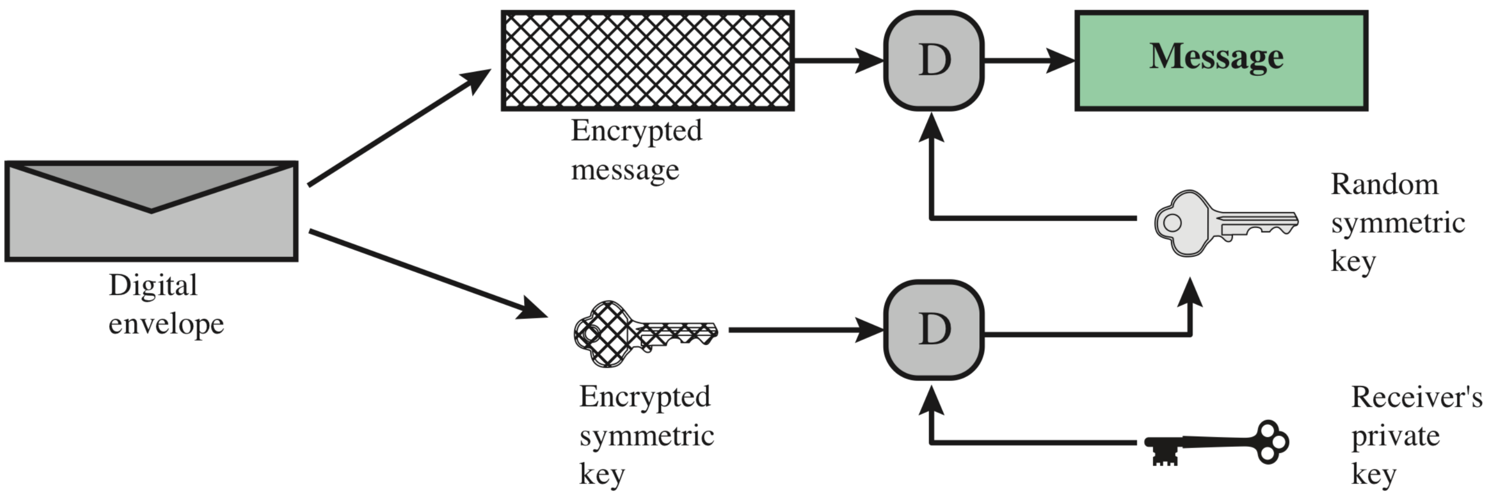
Use PK cryptography for encrypting a randomly generated symmetric key, which is used to encrypt a (large) message

- PK is only used to encrypt the key



Digital Envelopes

Opening an envelope



PK Encryption Algorithms

Diffie-Hellman key exchange algorithm

- Enables two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages
- Limited to the exchange of the keys

RSA (Rivest, Shamir, Adleman)

- Developed in 1977
- Most widely accepted and implemented approach to public-key encryption

Elliptic curve cryptography (ECC)

- Security like RSA, but with much smaller keys

Comparison

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

RSA Security

Based on the assumption that factoring numbers is hard

Variable key length

- Largest, publicly known, factored RSA number is 768 bits
- It is generally believed that 1024-bit keys may have already been broken or will soon be
- 2048-bit keys are recommended as the minimum

Part of the Public Key Cryptography Standards (PKCS)

In practice used with digital envelopes

RSA Security

Brute force

- Trying all possible private keys
- Defense is to use a large key space, however this slows speed of execution

Mathematical attacks

- Several approaches, all equivalent in effort to factoring the product of two primes

Timing attacks

- Depend on the running time of the decryption algorithm
- Comes from a completely unexpected direction and is a ciphertext-only attack
- Countermeasures: constant exponentiation time, random delay, blinding

Chosen ciphertext attacks

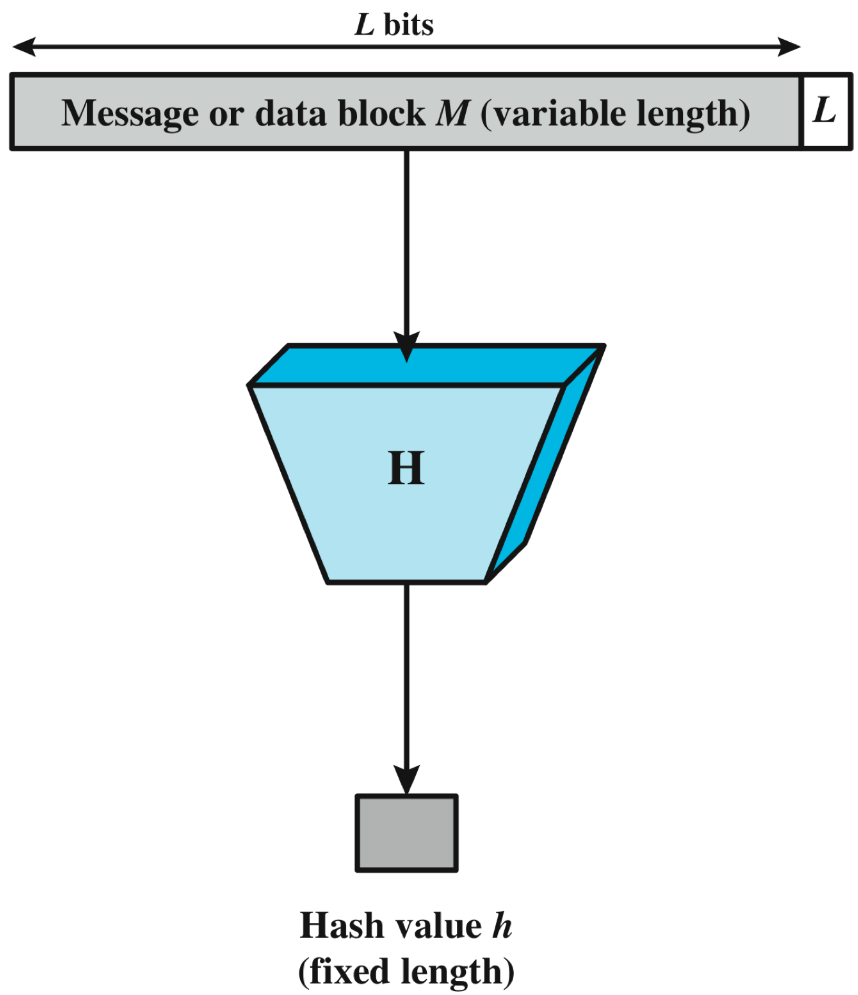
- Attack exploits properties of the RSA algorithm

Cryptographic Key Recommendation

Year	Symmetric	Factoring (modulus) (1)	(2)	Discrete Logarithm Key	Group	Elliptic Curve	Hash
2015	82	1613	1248	145	1613	154	163
2016	83	1664	1312	146	1664	155	165
2017	**83**	**1717**	**1344**	**147**	**1717**	**157**	**166**
2018	84	1771	1376	149	1771	158	168
2019	85	1825	1440	150	1825	160	169

https://www.keylength.com/en/1/

Hashing and Message Authentication Codes


L bits
Message or data block M (variable length)
L
H
Hash value h
(fixed length)

Hash Function Properties

Can be applied to a block of data of any size

Produces a fixed-length output

Security properties:

- One-way or pre-image resistant: computationally infeasible to find x such that H(x) = h
- Given x and H(X), it is computationally infeasible to find y ≠ x such that H(y) = H(x)
- Collision resistant or strong collision resistance: computationally infeasible to find any pair (x,y) such that H(x) = H(y)

Simple Hash Function

Split input in blocks of n bits

$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$

	Bit 1	Bit 2	• • •	Bit n
Block 1	b_{11}	b_{21}		b_{n1}
Block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
Block m	b_{1m}	b_{2m}		b_{nm}
Hash code	C_1	C_2		C_n

Secure Hash Algorithm (SHA)

SHA was originally developed by NIST

- Published as FIPS 180 in 1993
- Revised in 1995 as SHA-1
- Produces 160-bit hash values

SHA-2 adds 3 additional versions of SHA

- SHA-256, SHA-384, SHA-512 with 256/384/512-bit hash values
- Same basic structure as SHA-1 but greater security

SHA Comparison

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

Notes: 1. All sizes are measured in bits.

2. Security refers to the fact that a birthday attack on a message digest of size n produces a collision with a work factor of approximately $2^{n/2}$.

SHA-3

SHA-1 considered insecure and has been phased out for SHA-2

SHA-2 shares same structure and mathematical operations as its predecessors and causes concern

Due to the time required to replace SHA-2 should it become vulnerable, NIST announced in 2007 a competition to produce SHA-3

SHA-3, a subset of the cryptographic primitive family **Keccak**

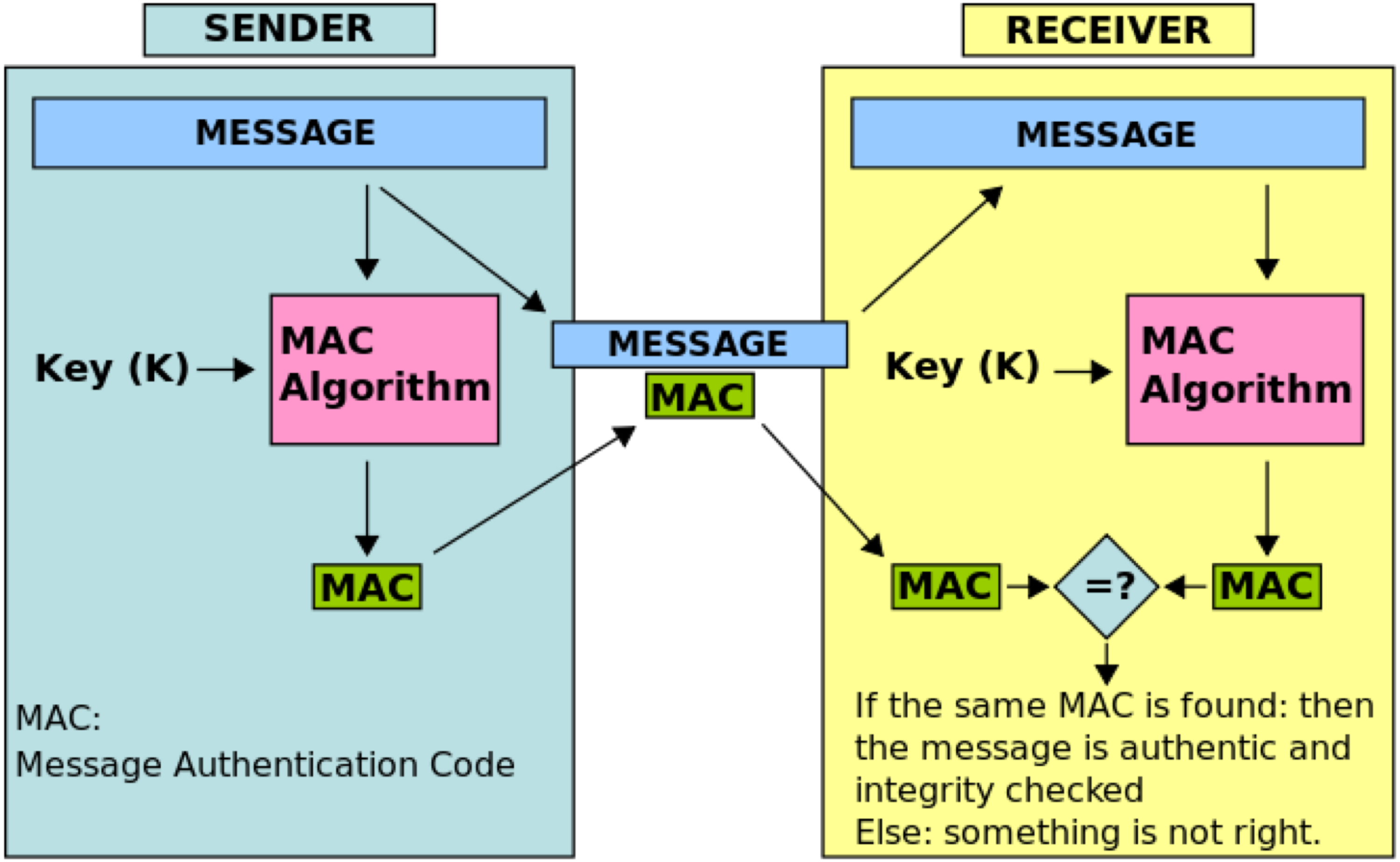
- Better security (resist attacks against SHA-2)
- Appropriate for fast implementation in hardware

Comparison from Wikipedia

Algorithm and variant		Output size (bits)	Block size (bits)	Max message size (bits)	Security (bits)	Example Performance (MiB/s)[12]
MD5 (as reference)		128	512	$2^{64} - 1$	<64 (collisions found)	335
SHA-0		160	512	$2^{64} - 1$	<80 (collisions found)	-
SHA-1		160	512	$2^{64} - 1$	<80 (theoretical attack[13] in 2^{61})	192
SHA-2	SHA-224 SHA-256	224 256	512	$2^{64} - 1$	112 128	139
	SHA-384 SHA-512 SHA-512/224 SHA-512/256	384 512 224 256	1024	$2^{128} - 1$	192 256 112 128	154
SHA-3	SHA3-224 SHA3-256 SHA3-384 SHA3-512 SHAKE128 SHAKE256	224 256 384 512 d (arbitrary) d (arbitrary)	1152 1088 832 576 1344 1088	∞	112 128 192 256 min(d/2, 128) min(d/2, 256)	

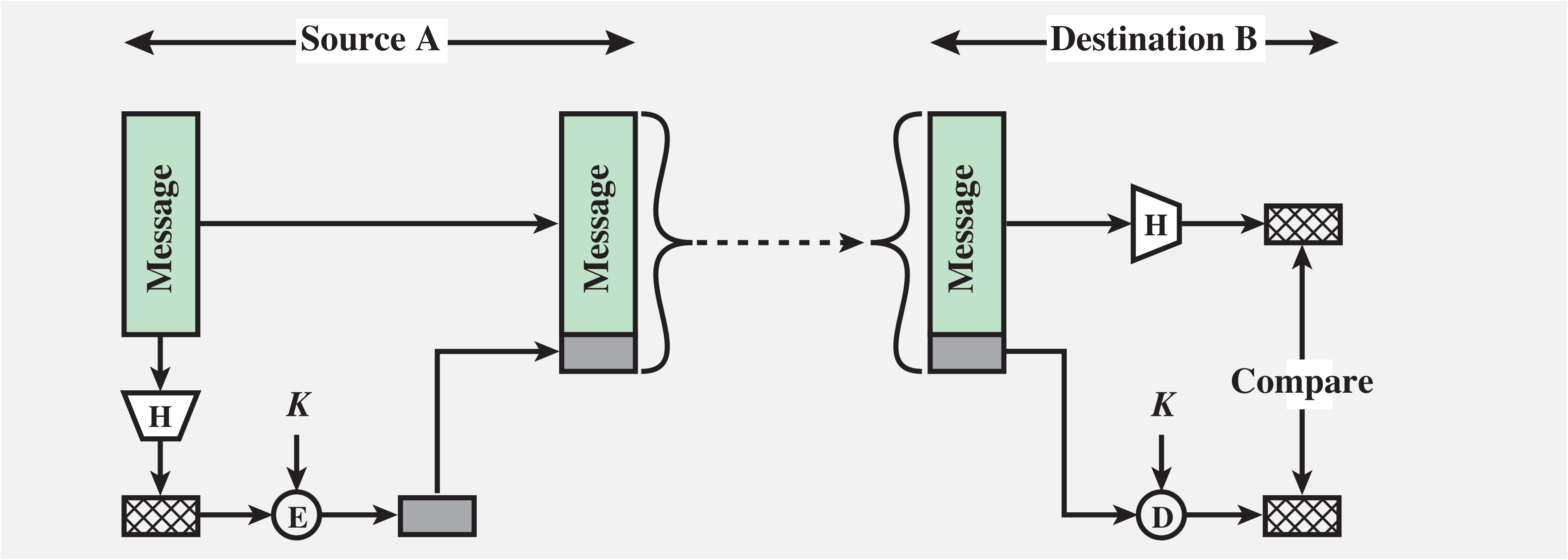
Message Authentication Code

Verify message integrity and authenticity



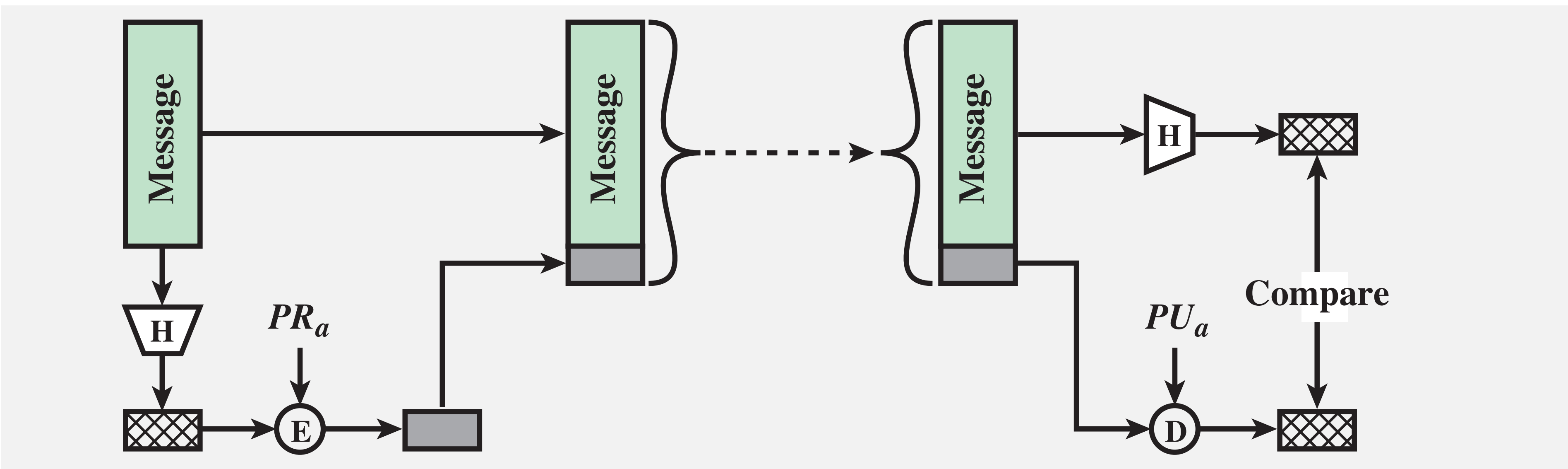
MAC with Symmetric Encryption

Encrypt hash of message using shared secret key, verify by decrypting with the same key



MAC with Public-Key Encryption

Encrypt hash of message using private key, verify using public key of sender



Digital Signatures

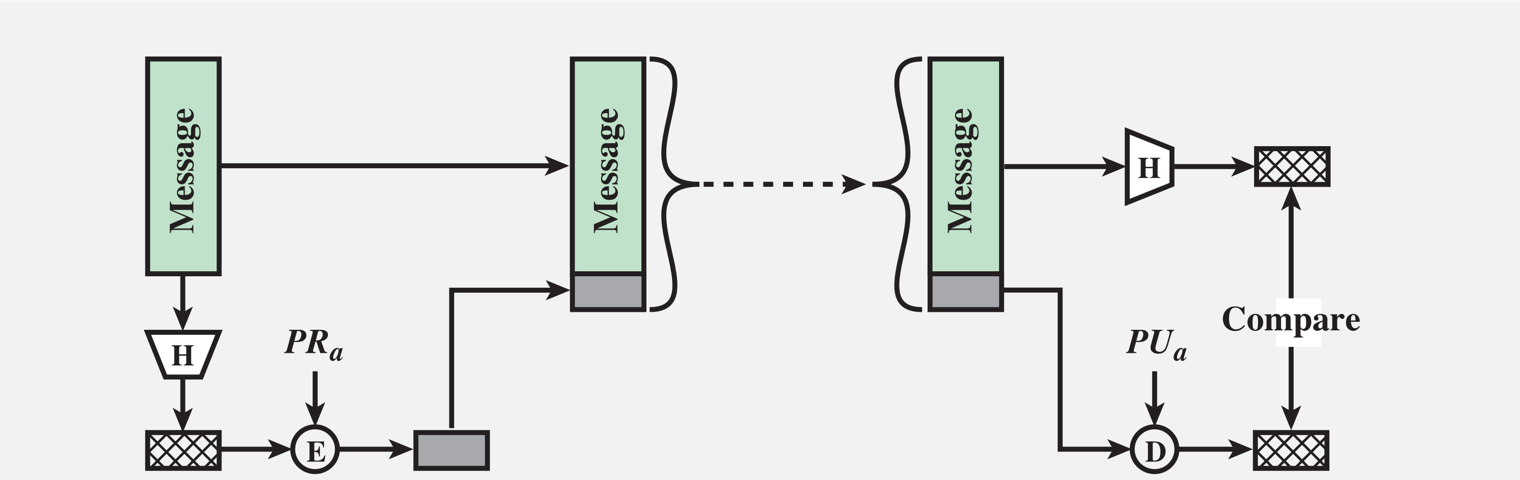
Similar to MAC using public-key cryptography

Used for authenticating both source and data integrity

Created by encrypting hash code with private key

Does not provide confidentiality

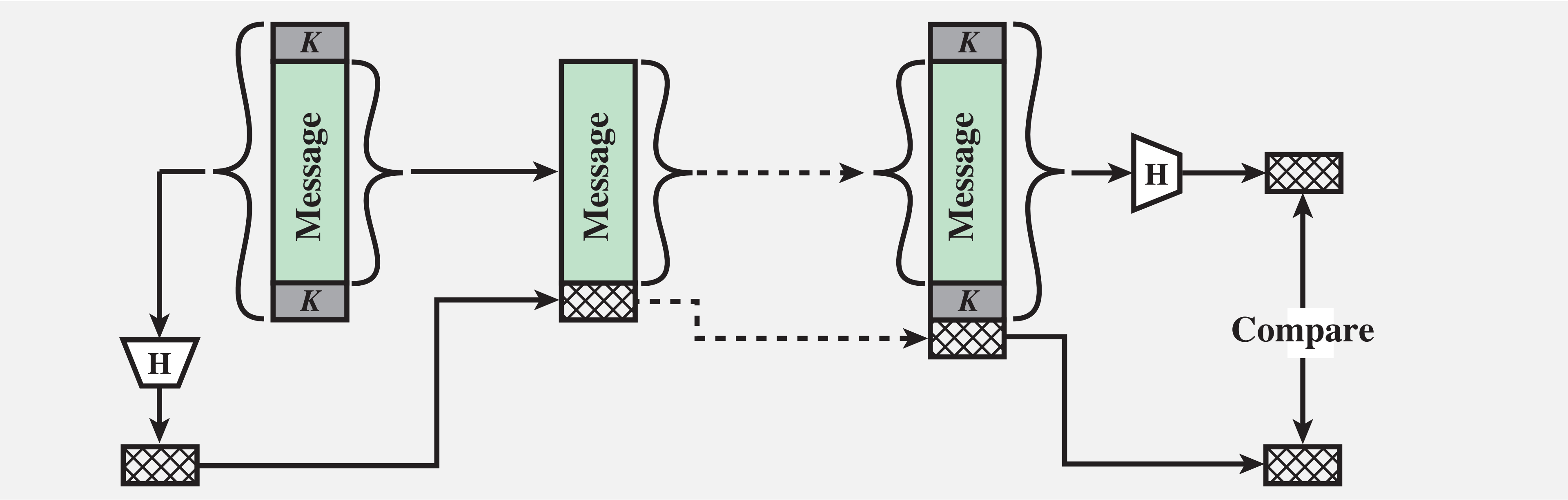
- Message is safe from alteration but not eavesdropping



(b) Using public-key encryption

MAC with Secret Value

Prefix and suffix message using nonce and hash the result, verify using the reverse



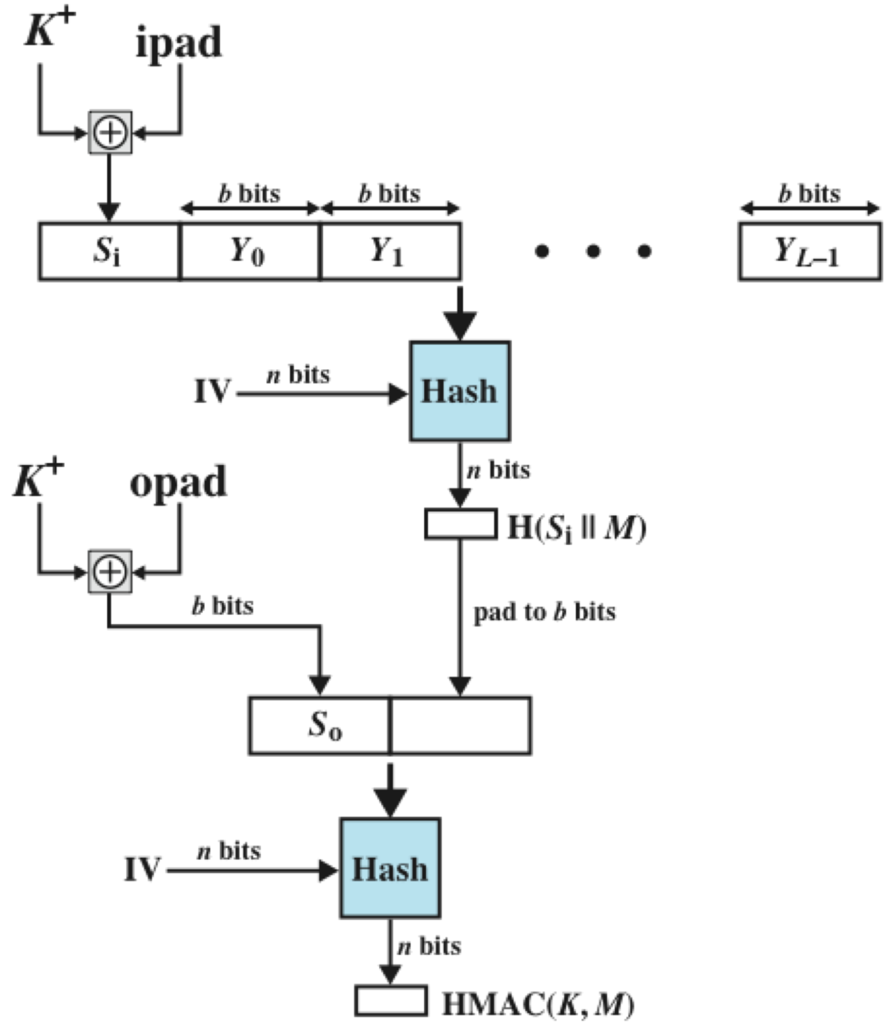
Hashed MAC (HMAC) Standard

A MAC using a secret key that enables the use of available hash functions without modifications

To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required

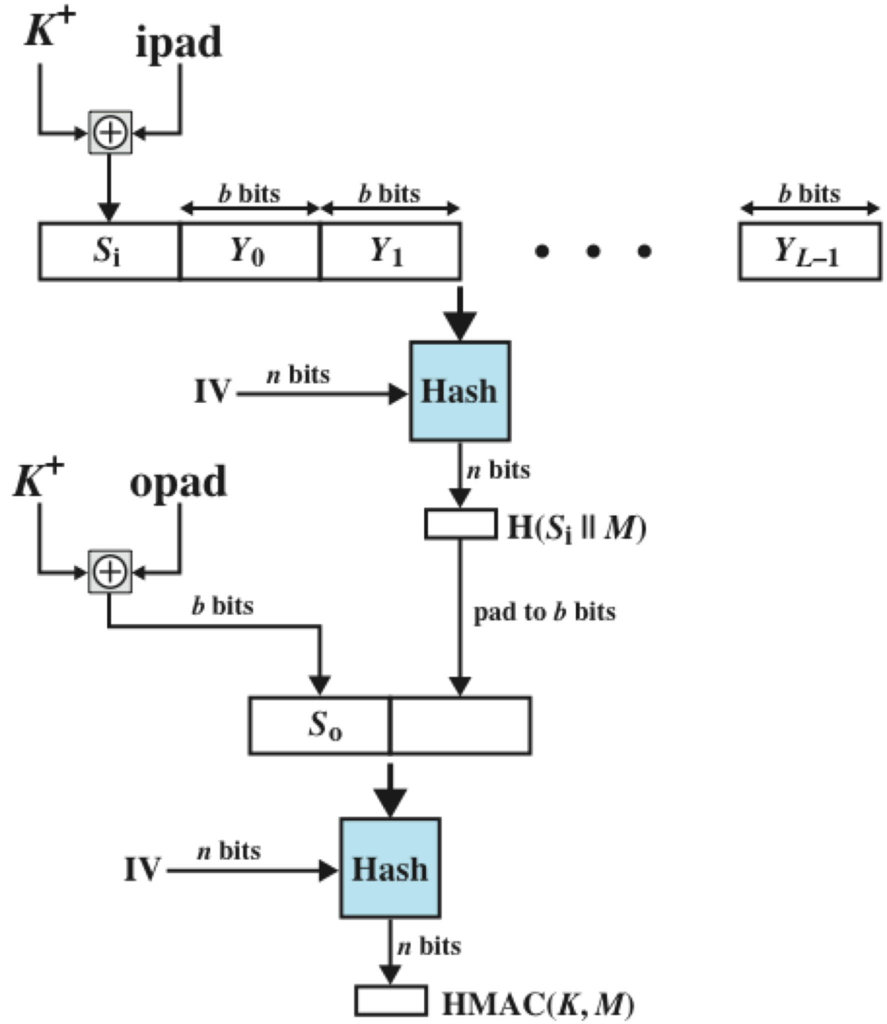
To use and handle keys in a simple way

- **K+** is K padded with zeros on the left so that the result is b bits in length
- **ipad** is a pad value of 36 hex repeated to fill block
- **opad** is a pad value of 5C hex repeated to fill block
- **M** is the message input to HMAC (including any padding)
- **IV** Initialization vector (if hash function requires one)



HMAC(K,M) = Hash[(K⁺ XOR opad) || Hash[(K⁺ XOR ipad) || M)]]

- Note that the XOR with ipad results in flipping one-half of the bits of *K* .
- Similarly, the XOR with opad results in flipping one-half of the bits of *K* , *but a* different set of bits. In effect, by passing S_i *and* S_o *through the hash algorithm, we* have pseudorandomly generated two keys from *K* .




```
HMAC(K,M) = Hash[(K⁺ XOR opad) || Hash[(K⁺ XOR ipad) || M)]]
```

Hashes vs MACs vs Signatures

	Hash	MAC	Signature
Integrity			
Authentication			
Non-repudiation			
Keys	None	Symmetric	Asymmetric