SAC Summer School 2016

Implementation and analysis of cryptographic protocols

Dr. Douglas Stebila

McMaster University

https://www.douglas.stebila.ca/teaching/sac-2016

Implementation and analysis of cryptographic protocols

1. Cryptographic building blocks
2. The TLS protocol
3. Attacks
    1. Bleichenbacher’s attack
    2. BEAST
    3. CRIME & BREACH
    4. Cross-ciphersuite
    5. Renegotiation
    6. Logjam
4. Provable security of TLS
5. TLS 1.3

SAC Summer School 2016

Implementation and analysis of cryptographic protocols

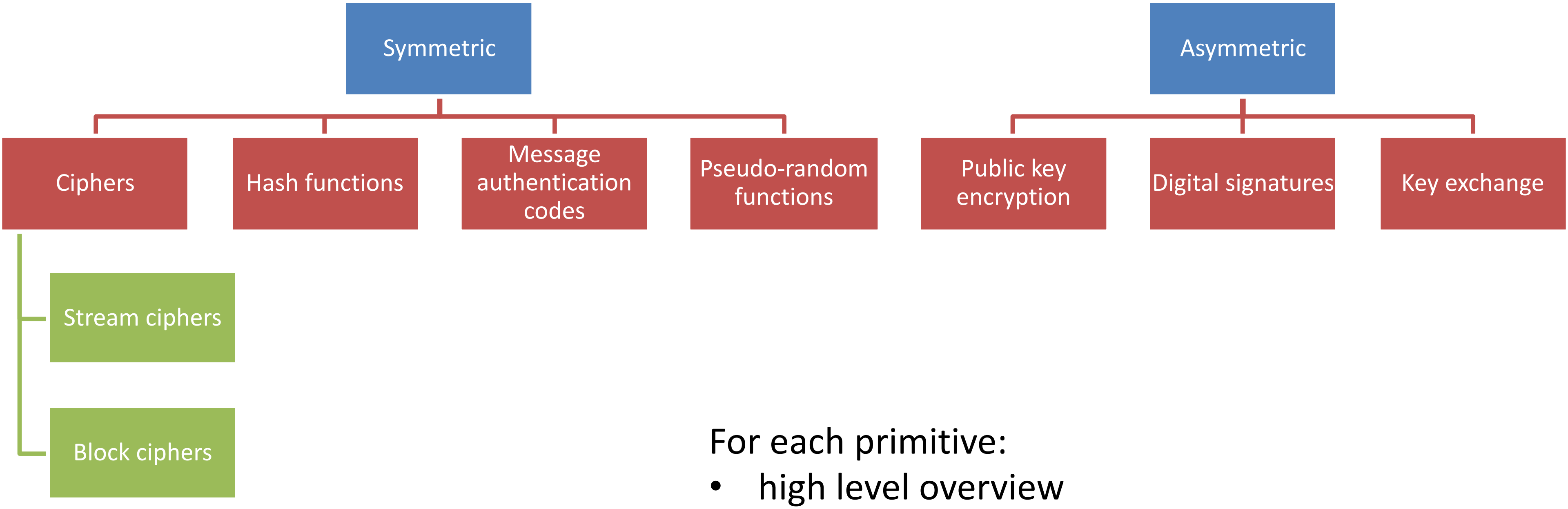
Part 1: Cryptographic Building Blocks

Dr. Douglas Stebila

McMaster University

https://www.douglas.stebila.ca/teaching/sac-2016

Cryptographic Building Blocks

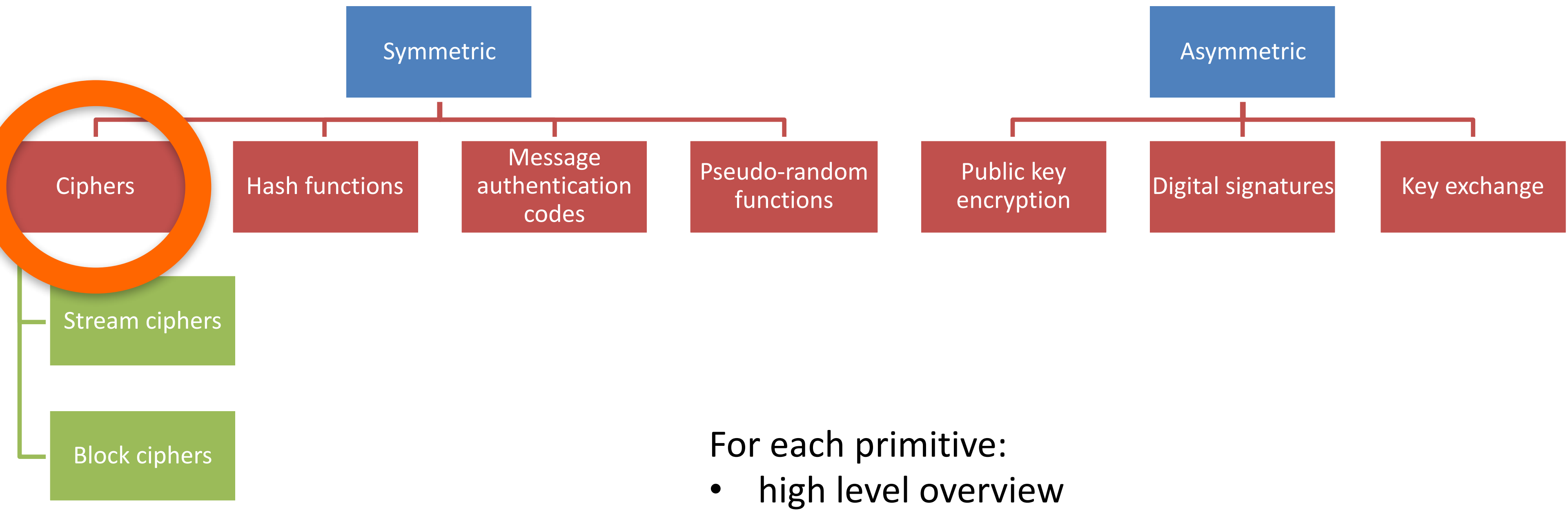


For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

SYMMETRIC CRYPTOGRAPHY

Cryptographic Building Blocks



For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Ciphers: Overview

- Encrypt an arbitrary length binary string using a shared secret key
- Provide confidentiality

Ciphers: Algorithms

KeyGen
(1^λ)
$\rightarrow$ k

Generates a secret key k.

Encrypt
(k, iv, m)
$\rightarrow$ c

Encrypt a message m using secret key k and initialization vector iv to obtain ciphertext c.

Decrypt
(k, iv, c)
$\rightarrow$ m

Decrypt a ciphertext c using secret key k and initialization vector iv to obtain message m.

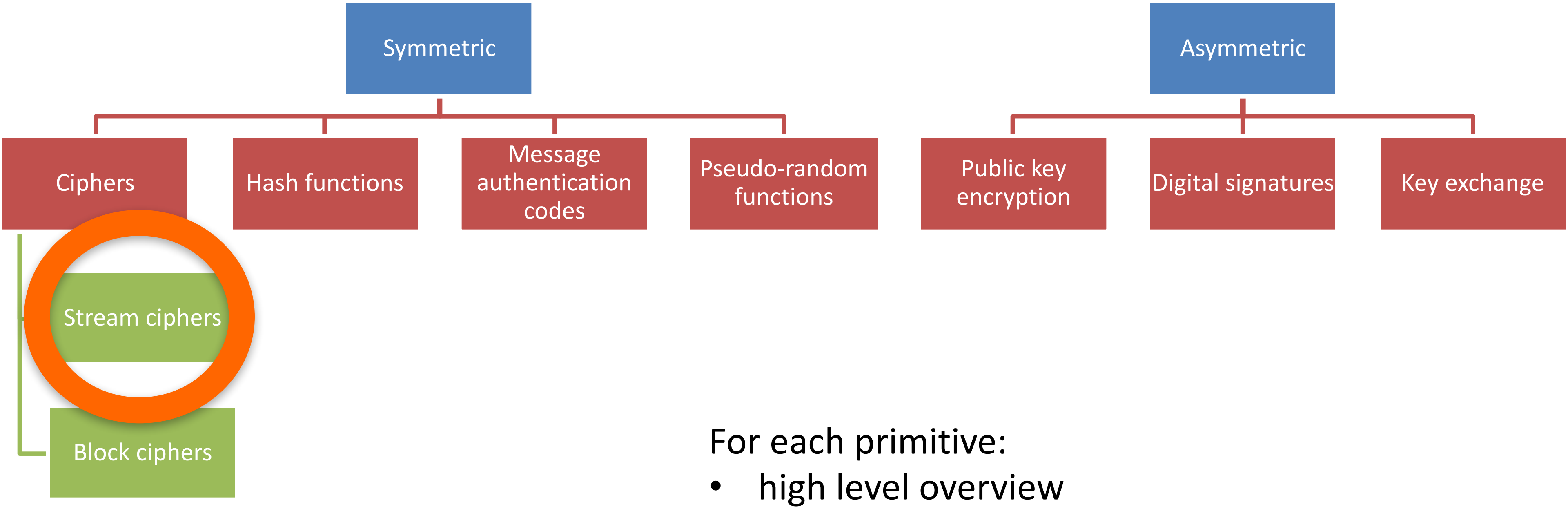
Need an IV so that we can encrypt different messages using the same key. (IV omitted in older cipher designs.)

Ciphers: Security

Security goal: <u>indistinguishability under adaptive chosen ciphertext attack (IND-CCA2)</u>.

Adaptive chosen ciphertext attack	Indistinguishability
• adversary can adaptively obtain encryptions of any messages and decryptions of any ciphertexts of his choosing	• the adversary cannot distinguish which of two messages m_0 or m_1 of its choosing was encrypted • equivalent to *semantic security*: attacker learns "nothing useful" from seeing ciphertext

Cryptographic Building Blocks



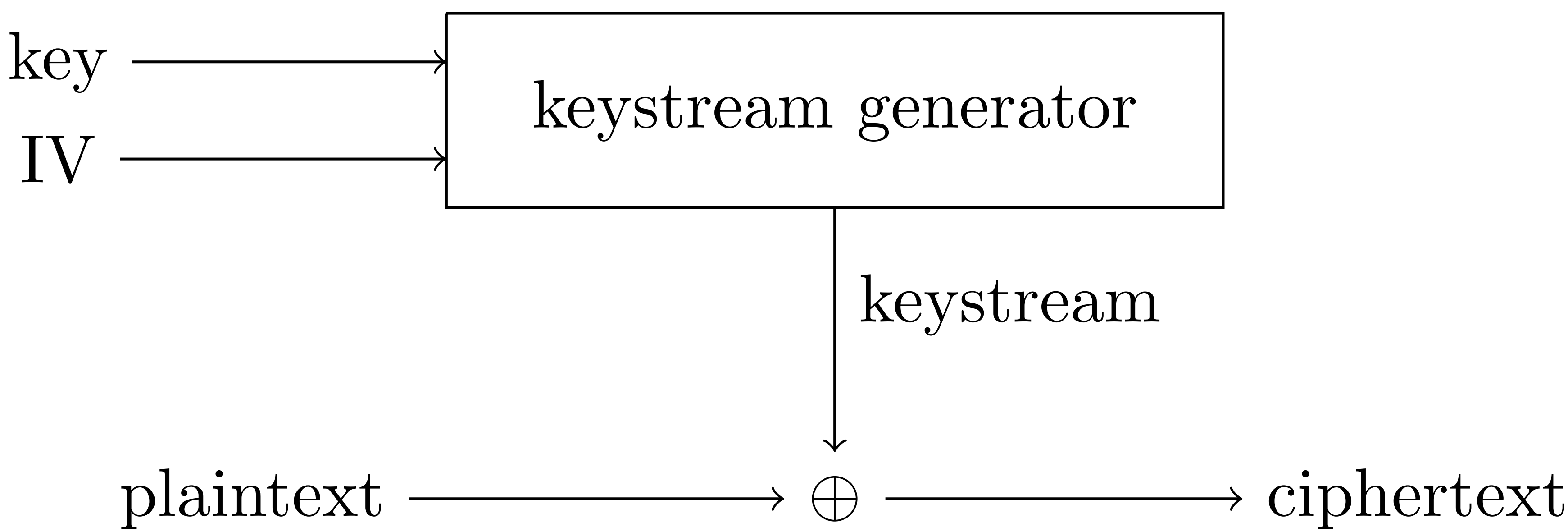
For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Stream ciphers: Overview

- Recall one-time pad: message is XORed with an encryption key of the same length
- Stream cipher encryption/decryption performed by having a keystream generator output a long encryption key from a short secret key, then XOR the long encryption key with the message

Stream ciphers: Overview

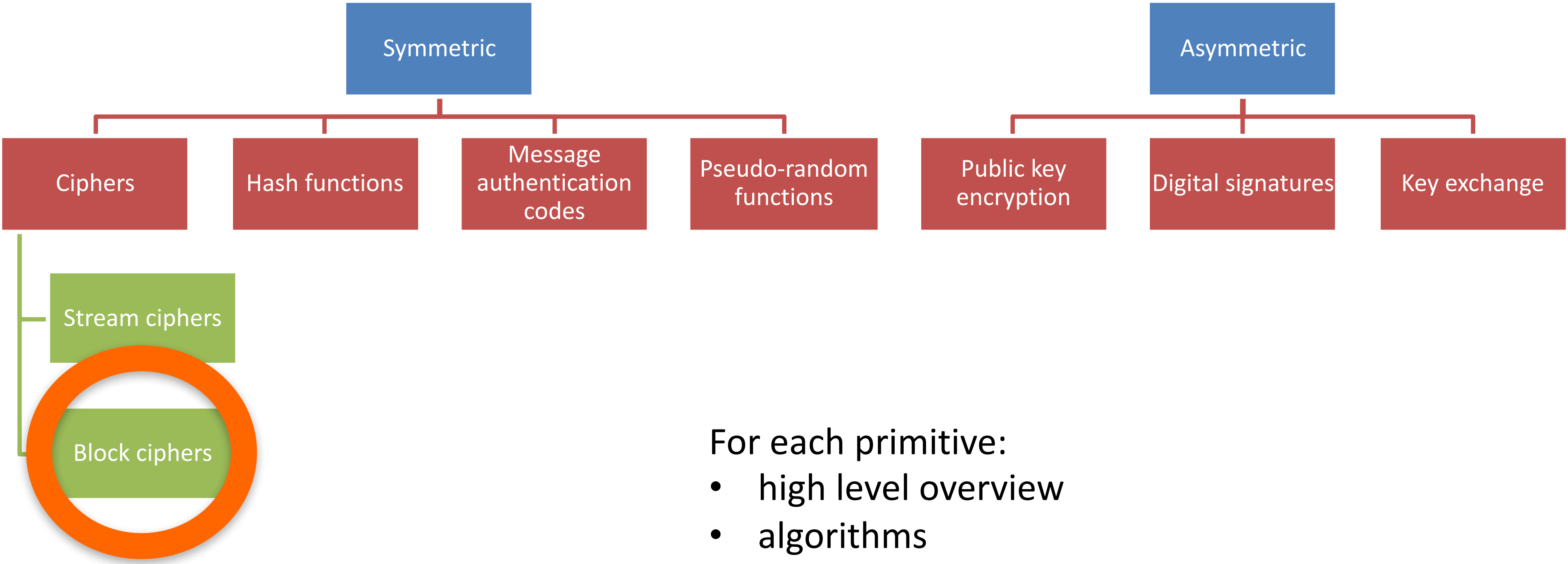


Stream ciphers: Schemes

- One common construction: linear feedback shift registers + non-linear filter or other non-linearity

Standardized schemes	
RC4	Weak; exploitable biases in keystream output.
A5/1 (A5/2)	Used in mobile phone communications; weak.
Salsa20 / ChaCha20	Family of extremely fast stream ciphers, ChaCha20 starting to be standardized.

Cryptographic Building Blocks



For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Block ciphers: Overview

- Message is divided into fixed-length blocks
- Each block is separately encrypted using:
  - a derived key
  - an initialization vector
  - the message block

Block ciphers: Data Encryption Standard (DES)

- Standardized by NIST in 1977 based on IBM design
- (effective) 56-bit key
- Uses a 16-round Feistel network

- Widely used in applications, some still active
- Small keyspace means can be readily brute force searched, in just a few hours on modern computers

- Triple-DES uses three applications of DES to provide 112-bit security

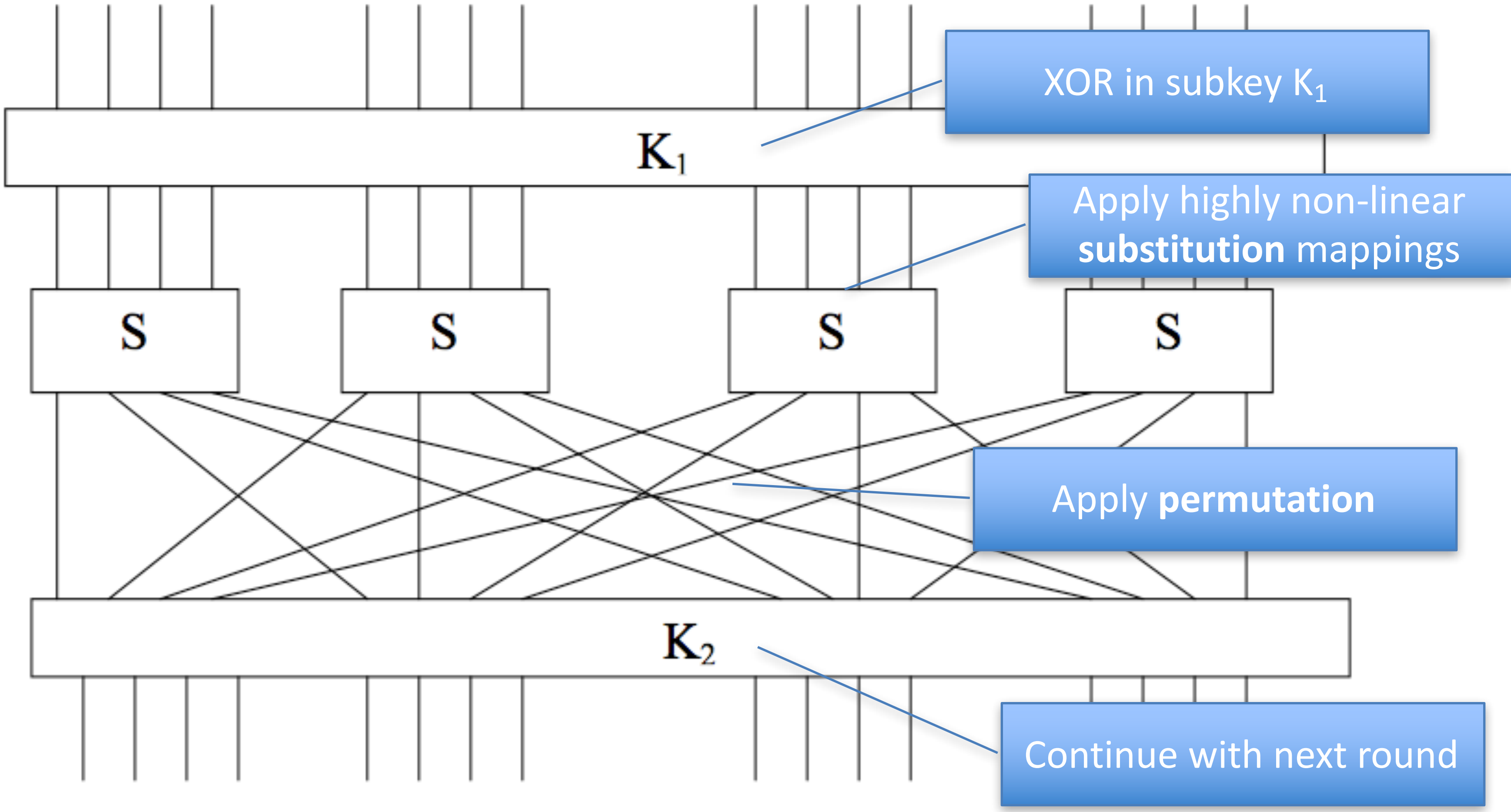
Block ciphers:
Advanced Encryption Standard (AES)

- Standardized by NIST in 2001 after an open competition, winner was Rijndael
- 128-, 192-, or 256-bit key
- Uses 10-14 rounds of a substitution-permutation network

- Widely used in applications
- Very fast on modern computers due to special processor instruction (AES-NI)

- No practical attacks, theoretical attacks barely better than brute force

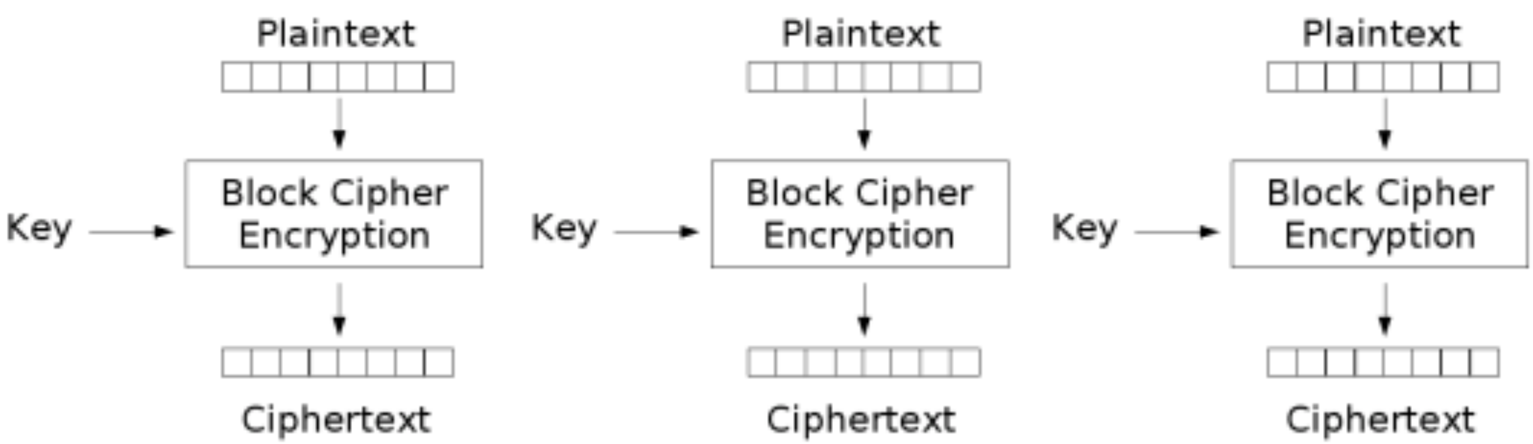
Block ciphers: Substitution-permutation network



Block ciphers: Modes of operation

- Since plaintext is divided into blocks when we use block ciphers, how should we process multi-block messages?

Block ciphers: Electronic Codebook (ECB) mode

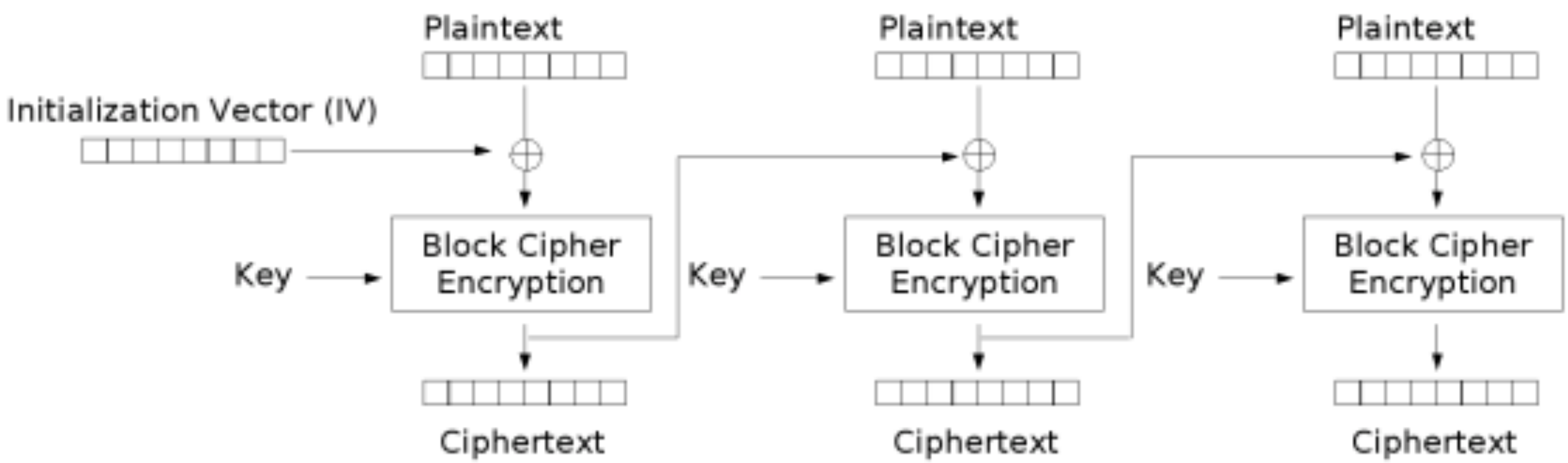


Electronic Codebook (ECB) mode encryption

If encryption is deterministic, then the same plaintext block is encrypted to the same ciphertext block every time.

images courtesy of Wikipedia

Block ciphers: Cipher Block Chaining (CBC) mode

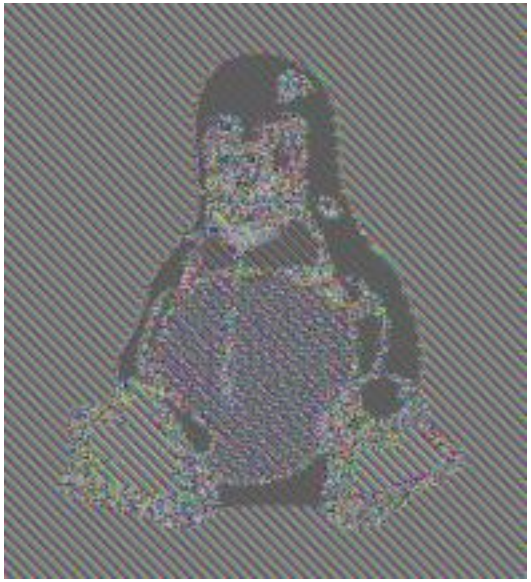


Cipher Block Chaining (CBC) mode encryption

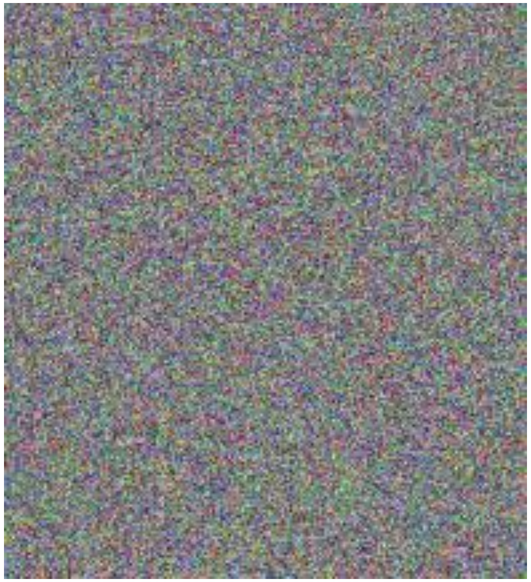
images courtesy of Wikipedia

Block ciphers: ECB vs CBC mode

Original image



ECB mode



CBC mode

images courtesy of Wikipedia

Block ciphers: Modes of operations

- Many different modes with many different properties
- Some more suitable for:
  - streaming media (lossy communication)
  - parallel processing
  - disk encryption
- Some provide integrity checking

Block ciphers vs. stream ciphers

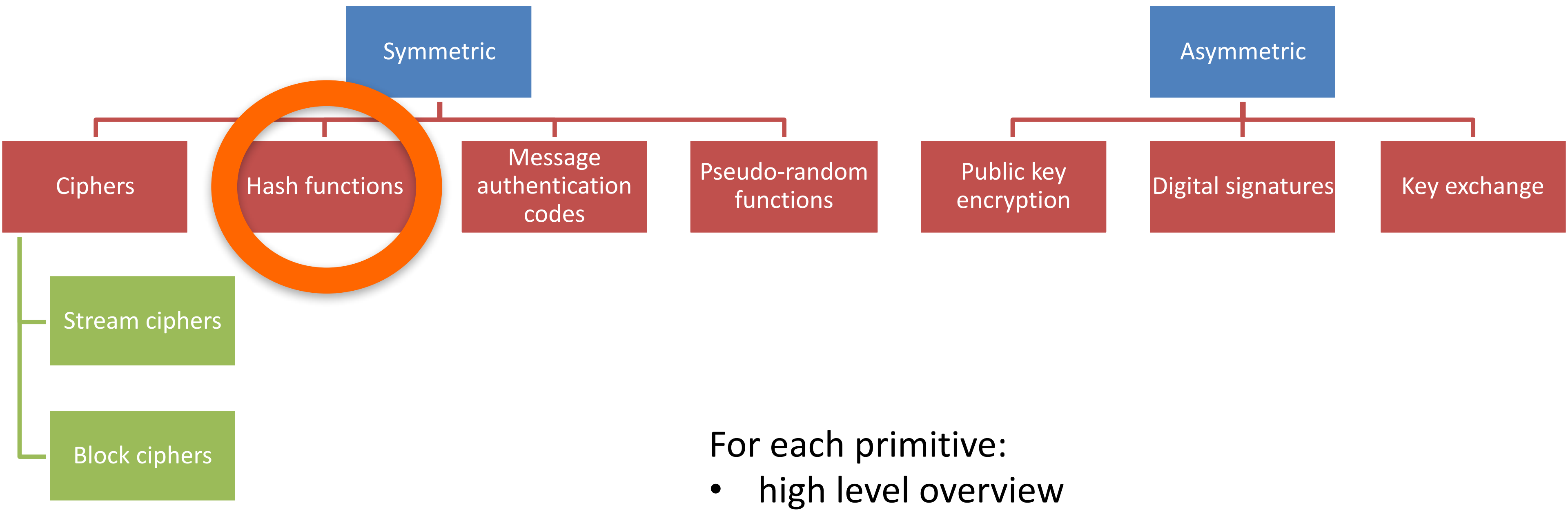
Block ciphers

- Often slower
- More complex implementation
- Better for storage
- Some modes good for streaming communication
- Viewed as being more secure

Stream ciphers

- Often faster
- Often easier to implement in software and hardware
- Better for streaming communication
- Viewed as being less secure

Cryptographic Building Blocks



For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Hash Functions: Overview

- Hashes an arbitrary length binary string into a fixed length binary string

- Useful for integrity and data origin authentication

Hash Functions: Algorithms

Keyed hash function (family)

KeyGen() → k

Generates a hash key k.

H(k, m) → h

Hashes a message m under a key k to obtain a hash h.

Unkeyed hash function

H(m) → h

Hashes a message m to obtain a hash h.

(Note k need not be secret, just random.)

Hash Functions: Security

Collision resistance

- It is hard to find two distinct values x_0 and x_1 such that $H(x_0)=H(x_1)$

Preimage resistance

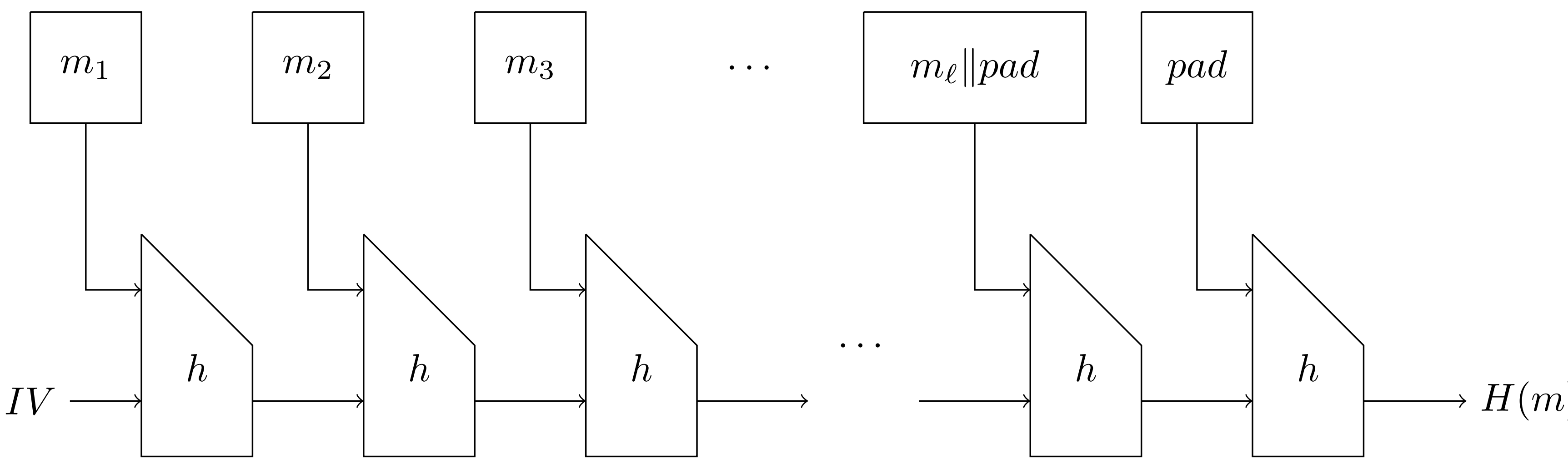
- Let x be chosen at random. Given y=H(x), it is hard to find x' such that H(x')=y.

Second preimage resistance

- Let x be chosen at random. Given x, it is hard to find a distinct x' such that H(x)=H(x').

Merkle–Damgård Construction

Common technique for constructing an arbitrary-length hash function *H* from a fixed-length compression function *h*.



Hash Functions: Schemes

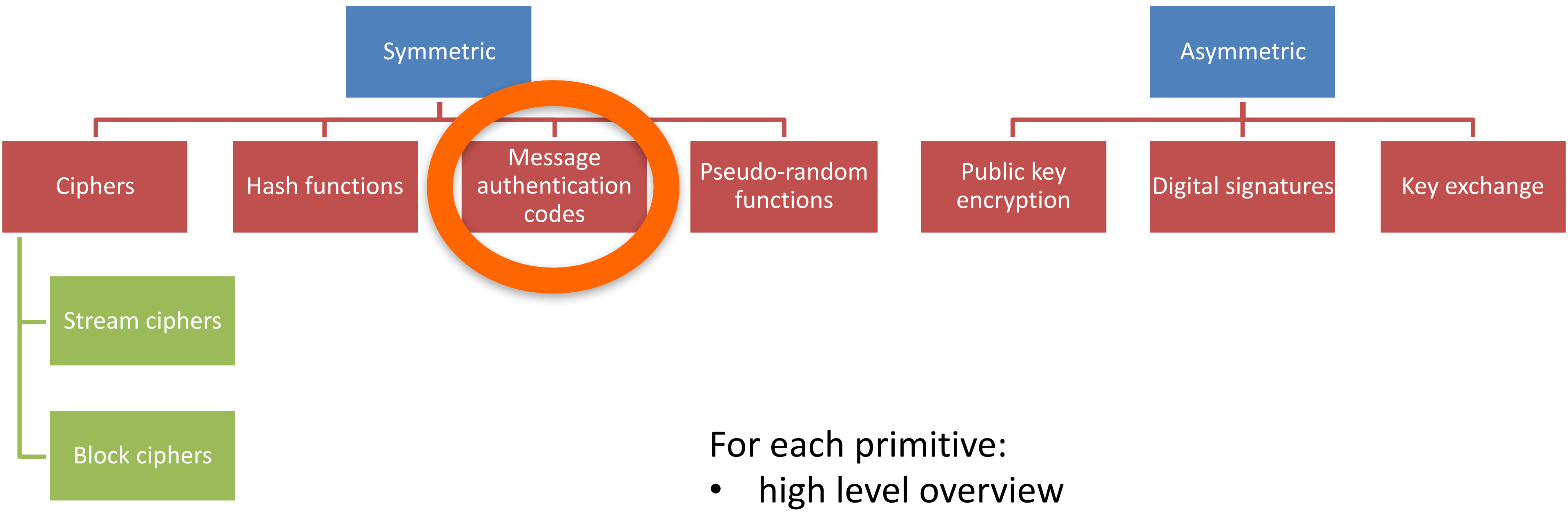
Standardized schemes	
MD5	Collision resistance broken.
SHA-1	Weak. Widely deployed.
SHA-2 (SHA-256, SHA-384, SHA-512)	Generally secure. Deployment in progress.
SHA-3 (a.k.a. Keccak)	Winner of NIST competition. NIST standarization August 2015; few deployments.

Quantum impact: For an n-bit hash function, Grover:

- pre-images in time $2^{n/2}$ (compared to 2^n classically)
- collisions in time $2^{n/3}$ (compared to $2^{n/2}$ classically)

Provably secure schemes (generally slower)	
Lattice-based	Based on learning with errors / shortest vector problem
RSA-based	Based on factoring / RSA problem.
Quantum fingerprinting	A quantum analogue of hashing

Cryptographic Building Blocks



For each primitive:

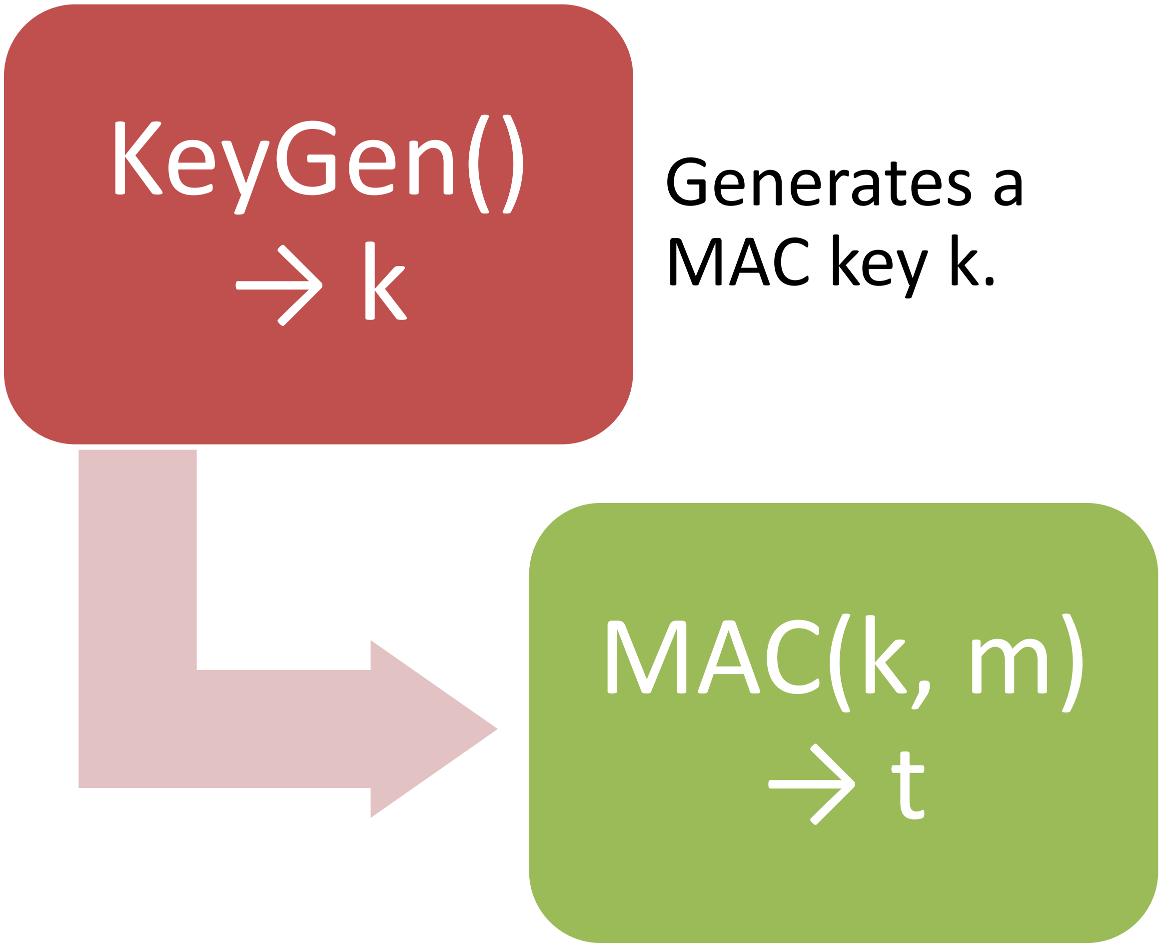
- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Message Authentication Codes: Overview

- Creates an authentication tag for a message.

- Provides integrity and data origin authentication

MACs: Algorithms



Generates a MAC key k.

Computes a tag t for a message m under key k.

Sender computes tag and sends tag and message; verifier recomputes tag and compares with received value.

MACs: Security

Security goal: existential unforgeability under chosen message attack (EUCMA).

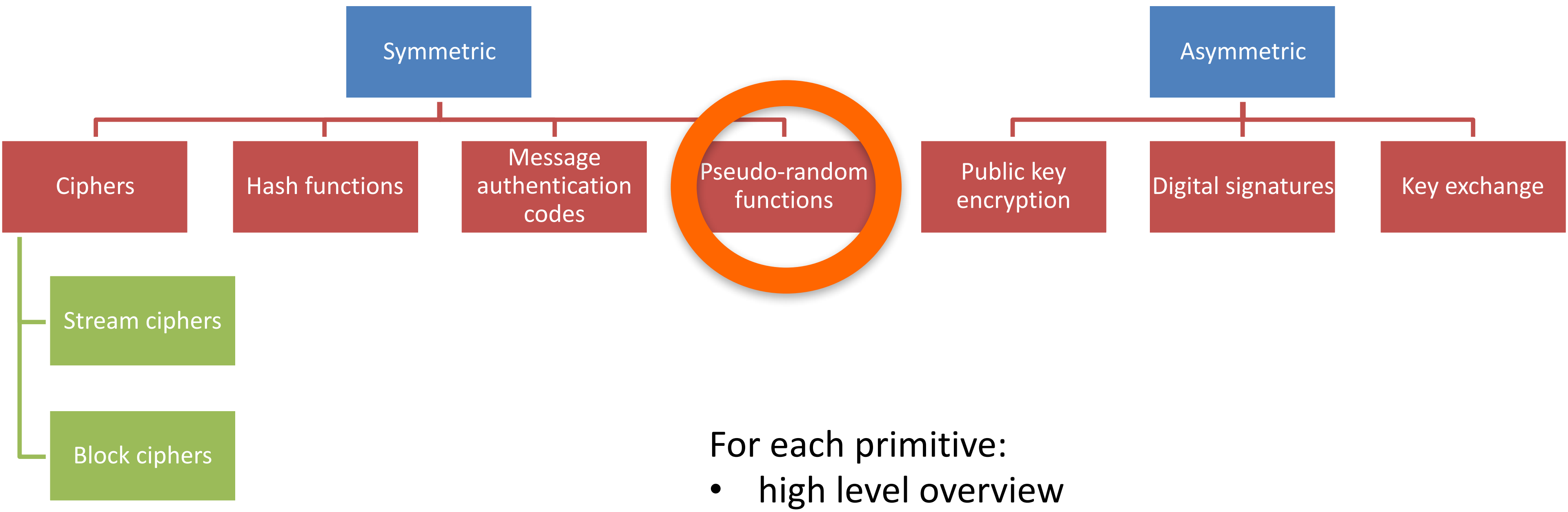
Chosen message attack	Existential unforgeability
• adversary can adaptively obtain tags for any messages of his choosing	• hard to construct a new valid message/tag pair (note: message doesn't have to be "meaningful")

MACs: Schemes

Standardized schemes	
HMAC-MD5 **HMAC-SHA1** **HMAC-SHA256** **…**	Almost universally used.
Quantum impact: For an n-bit key, Grover can break in time $2^{n/2}$	

Other schemes	
Wegman–Carter	Information-theoretically secure.
Poly1305-AES	High speed.

Cryptographic Building Blocks



For each primitive:

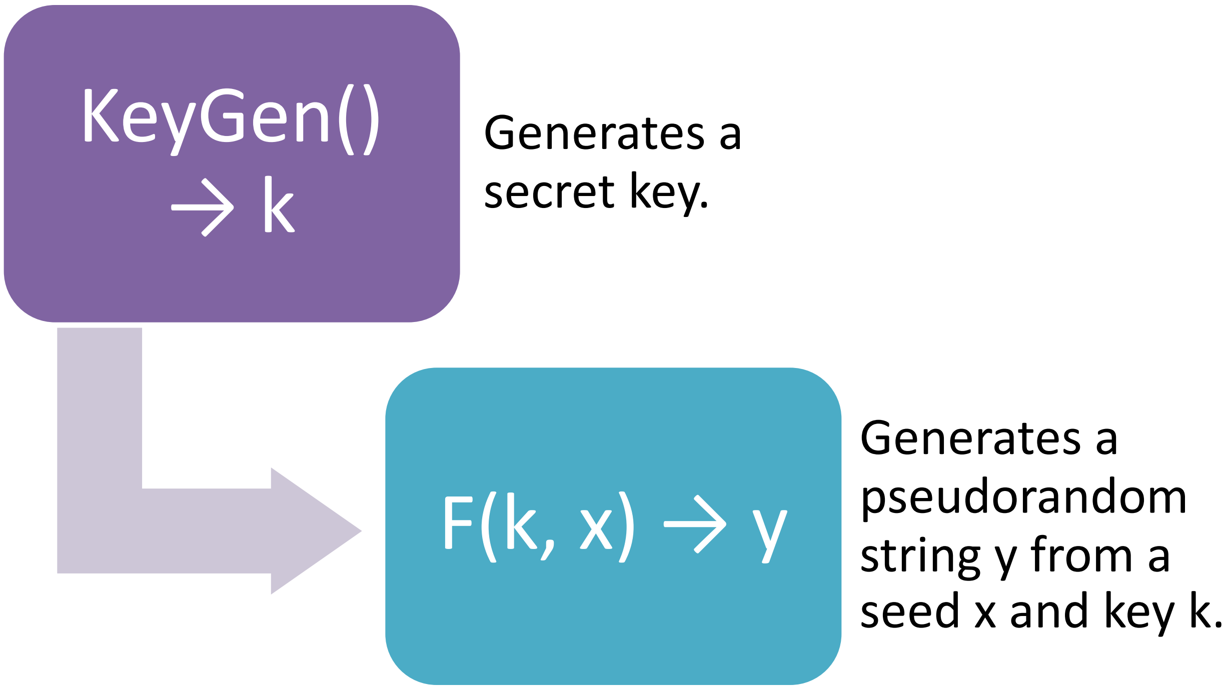
- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Pseudorandom Functions: Overview

- Generates a binary string that is indistinguishable from random

- Useful for confidentiality and key generation

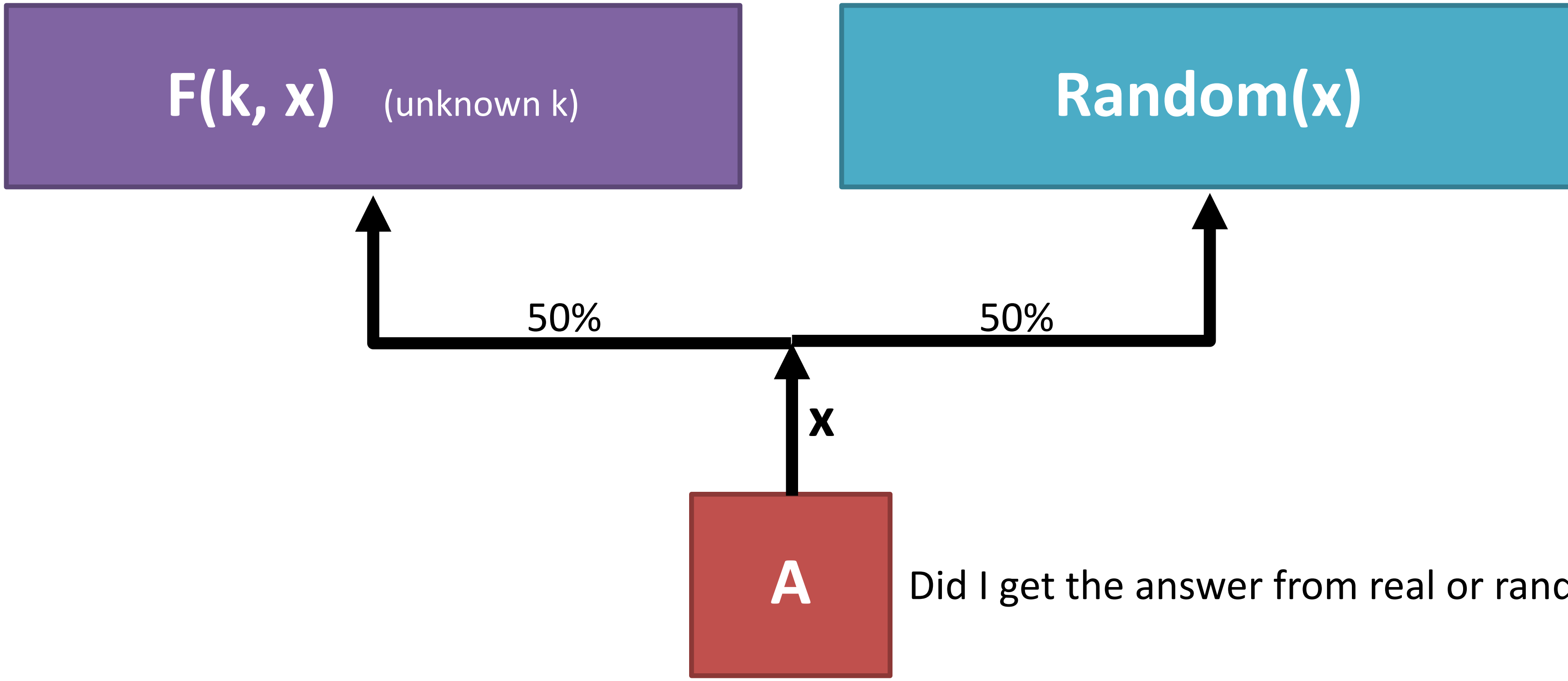
Pseudorandom Functions: Algorithms



Pseudorandom functions: Security

Security goal: pseudorandomness:

– Hard to distinguish the output of F(k, x) from the output of a truly random function Random(x).



Did I get the answer from real or random?

PRFs versus PRNGs versus KDFs

PRF

- Pseudorandom function
- Input: (short) uniform random key
- Output: (longer) computationally uniform random string

PRNG

- Pseudorandom number generator
- Input: (short) random seed
- Output: (longer) computationally uniform random string
- Update mechanism

KDF

- Key derivation function
- Input: (medium) (non-uniform) random key
- Output: (short) computationally uniform random key

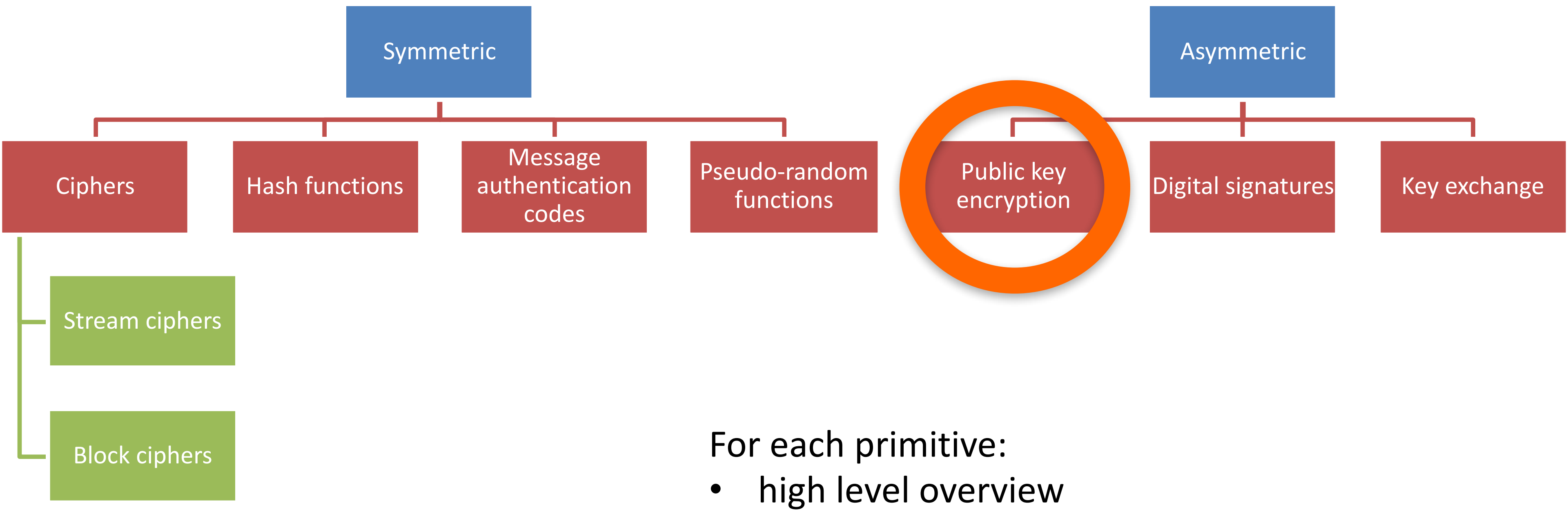
PRFs, PRNGs, KDFs: Schemes

Standardized Schemes	
Ad hoc constructions based on hash functions, HMAC, stream ciphers	
HMAC	Often used as a PRF or KDF.
Dual_EC_DRBG	NIST provably secure scheme based on elliptic curves, has a backdoor.
PBKDF2, Argon2	Used for deriving pseudorandom keys from passwords.
HKDF	Provably secure.

- PRNGs on computers also need to set and update seeds from a source of entropy

ASYMMETRIC CRYPTOGRAPHY

Cryptographic Building Blocks



For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Public Key Encryption: Overview

- Alice creates a private key / public key pair
- Anyone can encrypt messages for Alice based on her public key, but only Alice can decrypt those messages

- Provide confidentiality

- Versus ciphers: Anyone can encrypt using public key, whereas you need the shared secret for encrypting with ciphers.

Public Key Encryption: Algorithms

KeyGen() → (sk, pk)

Generates a private key sk and a public key pk.

Encrypt(pk, m) → c

Encrypt a message m using public key pk to obtain ciphertext c.

Decrypt(sk, c) → m

Decrypt a ciphertext c using private key sk to obtain message m.

Public Key Encryption: Security

Security goal: indistinguishability under adaptive chosen ciphertext attack (IND-CCA2).

Adaptive chosen ciphertext attack

- adversary can adaptively obtain decryptions of any ciphertexts of his choosing

Indistinguishability

- the adversary cannot distinguish which of two messages m_0 or m_1 of its choosing was encrypted

Public Key Encryption: Schemes

Standardized schemes	
RSA PKCS#1	Based on factoring
DHIES	Based on finite-field discrete logarithms
ECIES	Based on elliptic curve discrete logarithms
Quantum impact: Shor's algorithm can break all of these in polynomial time.	

Post-quantum schemes	
Lattice-based	Based on (ring) learning-with-errors problem
	Based on NTRU problem
Code-based	Based on bounded distance decoding problem
Multi-variate quadratic	

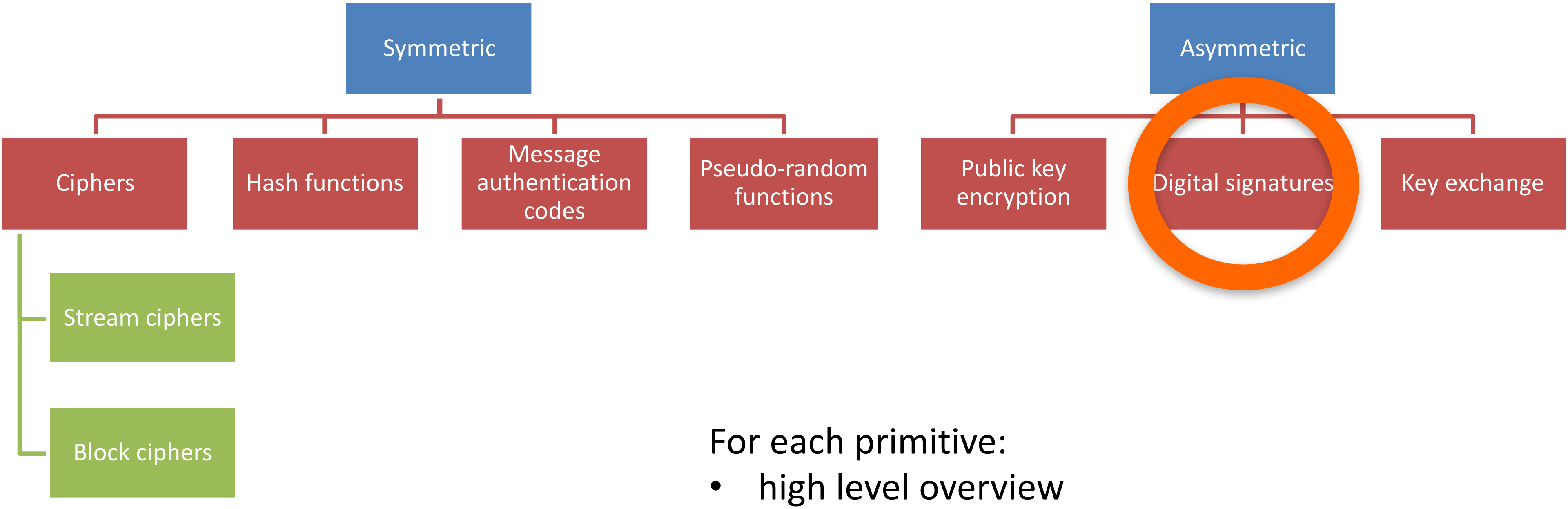
Hybrid encryption

To encrypt a long message m, typically use hybrid public key encryption:

1. Pick a random secret key k for a symmetric cipher like AES.
2. $c_1 \leftarrow$ AES.Encrypt(k, m)
3. $c_2 \leftarrow$ RSA.Encrypt(pk, k)
4. ciphertext = (c_1, c_2)

Faster than encrypting the whole message using public key encryption.

Cryptographic Building Blocks



For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Digital Signatures: Overview

- Alice creates a private key / public key pair
- Only the person with the private key (Alice) can create valid signatures, but anyone with the public key can verify

- Provide data origin authentication, integrity, non-repudiation
- Useful for entity authentication

- Versus MACs: Anyone can verify using public key.

Digital Signatures: Algorithms

KeyGen() → (sk, vk)

Generates a signing key sk and a verification key vk.

Sign(sk, m) → σ

Sign a message m using signing key sk to obtain a signature σ.

Verify (vk, m, σ) → {0,1}

Check validity of signature σ of a message m under verification key vk and output 0 or 1.

Digital Signatures: Security

Security goal: existential unforgeability under chosen message attack (EUCMA).

Chosen message attack	Existential unforgeability
• adversary can adaptively obtain signatures for any messages of his choosing	• hard to construct a new valid signature/message pair (note: message doesn't have to be "meaningful")

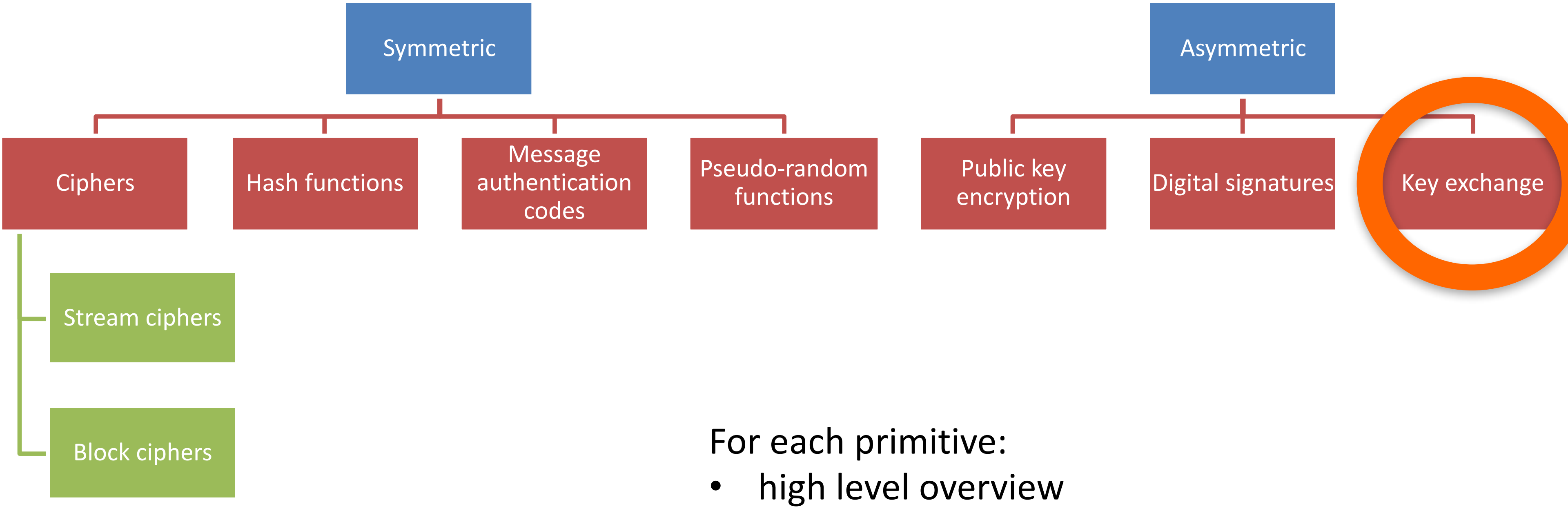
Digital Signatures: Schemes

Typically hash long message to short string then sign short string

Standardized schemes	
RSA PKCS#1	Based on factoring
DSA	Based on finite-field discrete logarithms
ECDSA	Based on elliptic curve discrete logarithms
Quantum impact: Shor's algorithm can break all of these in polynomial time.	

Post-quantum schemes	
Merkle-Lamport	Based on secure hash functions
Lattice-based	Based on short integer solution problem
	Based on (ring) learning-with-errors problem
Code-based	Based on bounded distance decoding problem
Multi-variate quadratic	

Cryptographic Building Blocks



For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Key Exchange: Overview

- Two parties establish an authenticated secret session key that they can use to exchange encrypted data

- Useful for entity authentication, confidentiality, data origin authentication, integrity

Key Exchange: Protocol
Example: Unauthenticated Diffie–Hellman

Let g be a generator of a cyclic group of prime order q.

Alice		**Bob**
$x \xleftarrow{\$} \{1, \ldots, q-1\}$		$y \xleftarrow{\$} \{1, \ldots, q-1\}$
$X \leftarrow g^x$		$Y \leftarrow g^y$
	$\xrightarrow{X}$	
	$\xleftarrow{Y}$	
$k \leftarrow Y^x$		$k \leftarrow X^y$

Key Exchange: Protocol Example: Signed Diffie–Hellman

Let g be a generator of a cyclic group of prime order q.

Alice		**Bob**
$(sk_A, pk_A) \leftarrow \texttt{SIG}.\mathsf{KeyGen}(1^\lambda)$		$(sk_B, pk_B) \leftarrow \texttt{SIG}.\mathsf{KeyGen}(1^\lambda)$
obtain pk_B		obtain pk_A
$x \xleftarrow{\$} \{1, \ldots, q-1\}$		$y \xleftarrow{\$} \{1, \ldots, q-1\}$
$X \leftarrow g^x$		$Y \leftarrow g^y$
$\sigma_A \leftarrow \mathsf{Sign}(sk_A, X)$		$\sigma_B \leftarrow \mathsf{Sign}(sk_B, Y)$
	$\xrightarrow{X, \sigma_A}$	
	$\xleftarrow{Y, \sigma_B}$	
abort if $\mathsf{Verify}(pk_B, Y, \sigma_B) = 0$		abort if $\mathsf{Verify}(pk_A, X, \sigma_A) = 0$
$k \leftarrow Y^x$		$k \leftarrow X^y$

Key Exchange: Security

Security goal: indistinguishability of session keys under various attack scenarios.

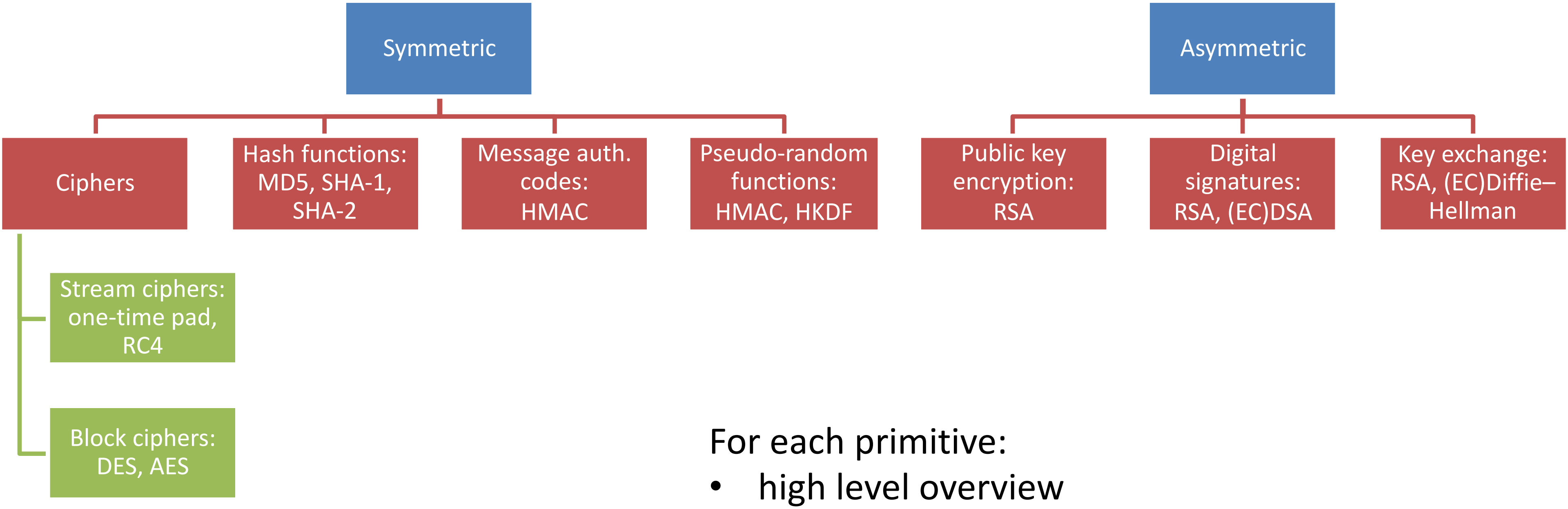
Attack scenarios	**Indistinguishability** of session key
• adversary can control communications, • learn session keys of other sessions, • learn parties’ long-term keys (“forward secrecy”) • learn parties’ random coins	• hard to distinguish the real session key from random string of the same length

Key Exchange: Schemes

Commonly used schemes	
RSA key transport	Based on factoring
Signed-Diffie–Hellman	Based on finite-field discrete logarithms
Signed elliptic curve Diffie–Hellman	Based on elliptic curve discrete logarithms
MQV / ECMQV	Based on discrete logarithms
Quantum impact: Shor's algorithm can break all of these in polynomial time.	

Post-quantum schemes	
Lattice-based key exchange	Based on (ring) learning-with-errors problem
	Based on NTRU problem
Code-based key exchange	Based on bounded distance decoding problem
Isogenies-based key exchange	Based on isogenies on super-singular elliptic curves
Quantum key distribution	Information-theoretically secure based laws of quantum mechanics

Cryptographic Building Blocks



For each primitive:

- high level overview
- algorithms
- security goal
- standardized schemes
- effect of quantum computers

Matching key sizes

- Applications often use multiple cryptographic primitives together
- Only as secure as strength of weakest scheme / key
- Lots of recommendations based on forecast computational power (but not cryptographic breakthroughs!)
  - http://www.keylength.com/

Security	Cipher	Hash size	Finite field (RSA/DSA)	Elliptic curve
Short-term protection	80	160	approx. 1024	160
Medium (e.g. until 2030)	128	256	2048-3072	256
Long-term (e.g. past 2030)	256	512	approx. 15360	512

Lots more cryptographic primitives

- minicrypt: oblivious transfer, bit commitment
- identity-based encryption, attribute-based encryption, functional encryption
- group signatures
- fully homomorphic encryption
- secure multi-party computation
- password-authenticated key exchange
- client puzzles / proofs of work -> Bitcoin, …
- …