STAM Center
SECURE, TRUSTED, AND ASSURED MICROELECTRONICS

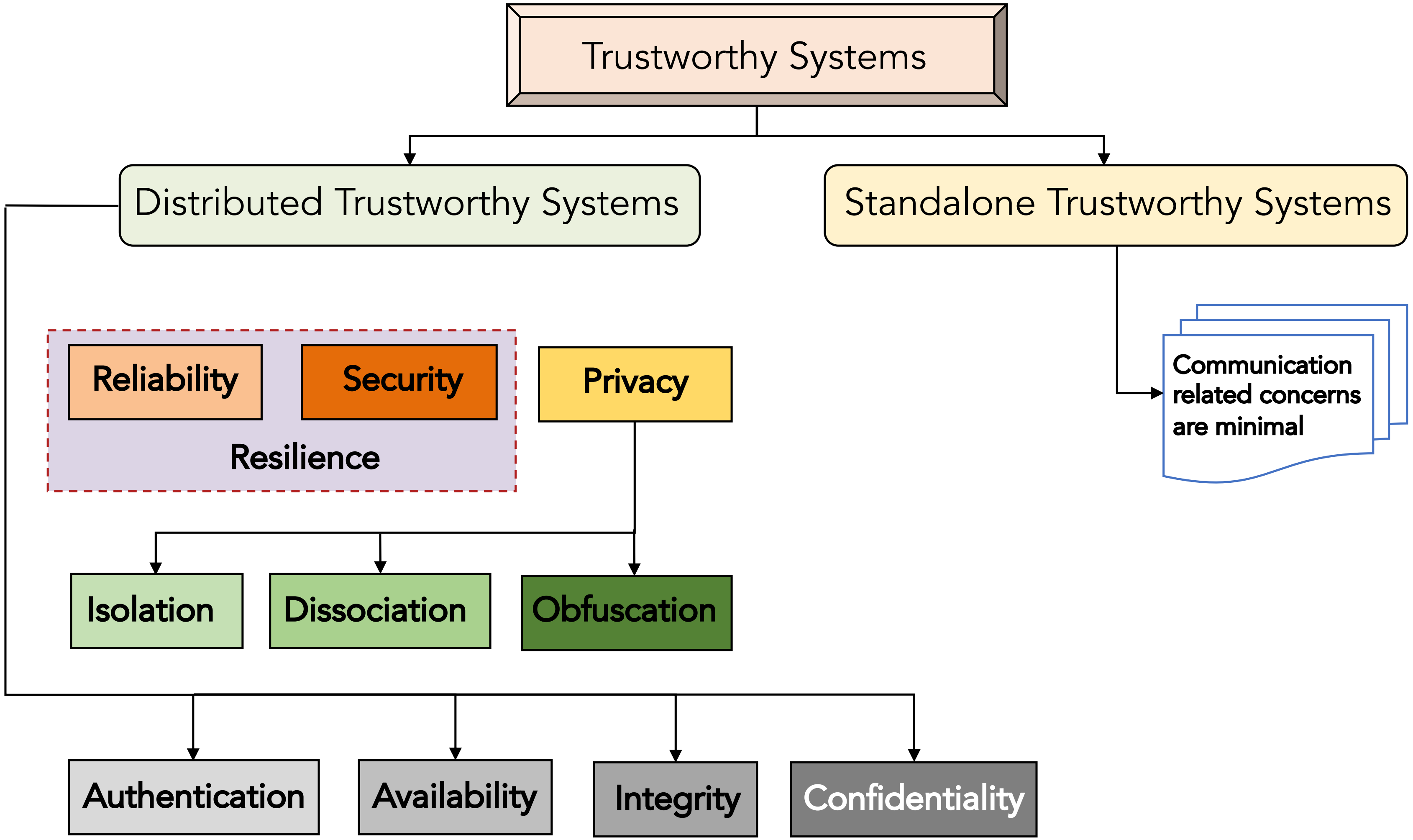
Ira A. Fulton Schools of Engineering
Arizona State University

CSE/CEN 598
Hardware Security & Trust

Classic and Modern Encryption Algorithms

Prof. Michel A. Kinsy

Trustworthy Hardware System Design



Foundations of Computer Security

- It is not necessary to be a cryptographer to properly use cryptography
  - Not everything is math – knowing your assumptions & inherent vulnerabilities, correctly modeling your threats, understanding information flows, and applying right solutions are all important
- Cryptography is a large and diverse field, ranging from very practical to very abstract concepts
  - Often taught as a potpourri of methods
  - Hard (at first) to separate abstraction layers
    - "Is e.g., zero-knowledge proofs a concept? An algorithm/method? A property?"
  - Can we do better?

Functional Security Properties of Systems

Confidentiality:

- Secure channels / symmetric cryptography:
  - One-time pads
  - Stream ciphers
  - Block ciphers
    - Modes of operation
- Key exchange / key distribution:
  - Public / private cryptography
  - Forward secrecy
- Obfuscation:
  - Indistinguishability obfuscation
  - Deniable encryption
  - Program obfuscation
    - Opaque predicates
  - Hardware obfuscation
    - Anti-tamper, split manufacturing, …
- Private lookups, private metadata:
  - Mix networks, oblivious RAM, onion routing
- Isolation
  - Virtualization, containerization, sandboxing…
  - Secure architectures,
    - Trusted execution engines, secure enclaves
  - Formal verification
- Zero-knowledge proofs

Functional Security Properties of Systems

Confidentiality:

- Secure channels / symmetric cryptography:
  - One-time pads
  - Stream ciphers
  - Block ciphers
    - Modes of operation
- Key exchange / key distribution:
  - Public / private cryptography
  - Forward secrecy
- Obfuscation:
  - Indistinguishability obfuscation
  - Deniable encryption
  - Program obfuscation
    - Opaque predicates
  - Hardware obfuscation
    - Anti-tamper, split manufacturing, …
- Private lookups, private metadata:
  - Mix networks, oblivious RAM, onion routing
- Isolation
  - Virtualization, containerization, sandboxing…
  - Secure architectures,
    - Trusted execution engines, secure enclaves
  - Formal verification
- Zero-knowledge proofs

Integrity:

- Message integrity:
  - Error correction codes
  - (Cryptographic) hash functions
- Privacy-preserving computation:
  - Multi-Party Computation (MPC):
    - Oblivious Transfer
    - Yao's Garbled circuits
    - Universal composability
  - Homomorphic Encryption (HE)
  - Hardware Root-of-Trust (HRoT):
    - Physical unclonable functions, e-fuses
  - Federated Learning
  - Distributed Consensus:
    - Digital currency, private voting
- Software security:
  - Virtual memory, file system permissions
  - App signing, sandboxing
  - Control flow integrity:
    - Shadow stacks
  - Buffer overflow protection:
    - ASLR, stack canaries
  - Malware detection:
    - Antiviruses, malware signatures
    - Hardware performance counters

Functional Security Properties of Systems

Confidentiality:

- Secure channels / symmetric cryptography:
  - One-time pads
  - Stream ciphers
  - Block ciphers
    - Modes of operation
- Key exchange / key distribution:
  - Public / private cryptography
  - Forward secrecy
- Obfuscation:
  - Indistinguishability obfuscation
  - Deniable encryption
  - Program obfuscation
    - Opaque predicates
  - Hardware obfuscation
    - Anti-tamper, split manufacturing, …
- Private lookups, private metadata:
  - Mix networks, oblivious RAM, onion routing
- Isolation
  - Virtualization, containerization, sandboxing…
  - Secure architectures,
    - Trusted execution engines, secure enclaves
  - Formal verification
- Zero-knowledge proofs

Integrity:

- Message integrity:
  - Error correction codes
  - (Cryptographic) hash functions
- Privacy-preserving computation:
  - Multi-Party Computation (MPC):
    - Oblivious Transfer
    - Yao's Garbled circuits
    - Universal composability
  - Homomorphic Encryption (HE)
  - Hardware Root-of-Trust (HRoT):
    - Physical unclonable functions, e-fuses
  - Federated Learning
  - Distributed Consensus:
    - Digital currency, private voting
- Software security:
  - Virtual memory, file system permissions
  - App signing, sandboxing
  - Control flow integrity:
    - Shadow stacks
  - Buffer overflow protection:
    - ASLR, stack canaries
  - Malware detection:
    - Antiviruses, malware signatures
    - Hardware performance counters

Authentication:

- Asymmetric cryptography
  - One-way functions, trapdoor functions
  - Key exchange / key distribution algorithms
  - Digital signatures
- Public key infrastructure:
  - Web of trust
  - Certificate authorities, root certificates, self-signed certificates
- Passwords, biometrics, …
- Password-based key derivation

Authorization:

- Access Control Lists
- Role-Based Access Control
- Capability-Based Security

Non-repudiation:

- Digital signatures
- Commitment schemes
- Message authentication codes
- Deniable encryption, undeniable signatures

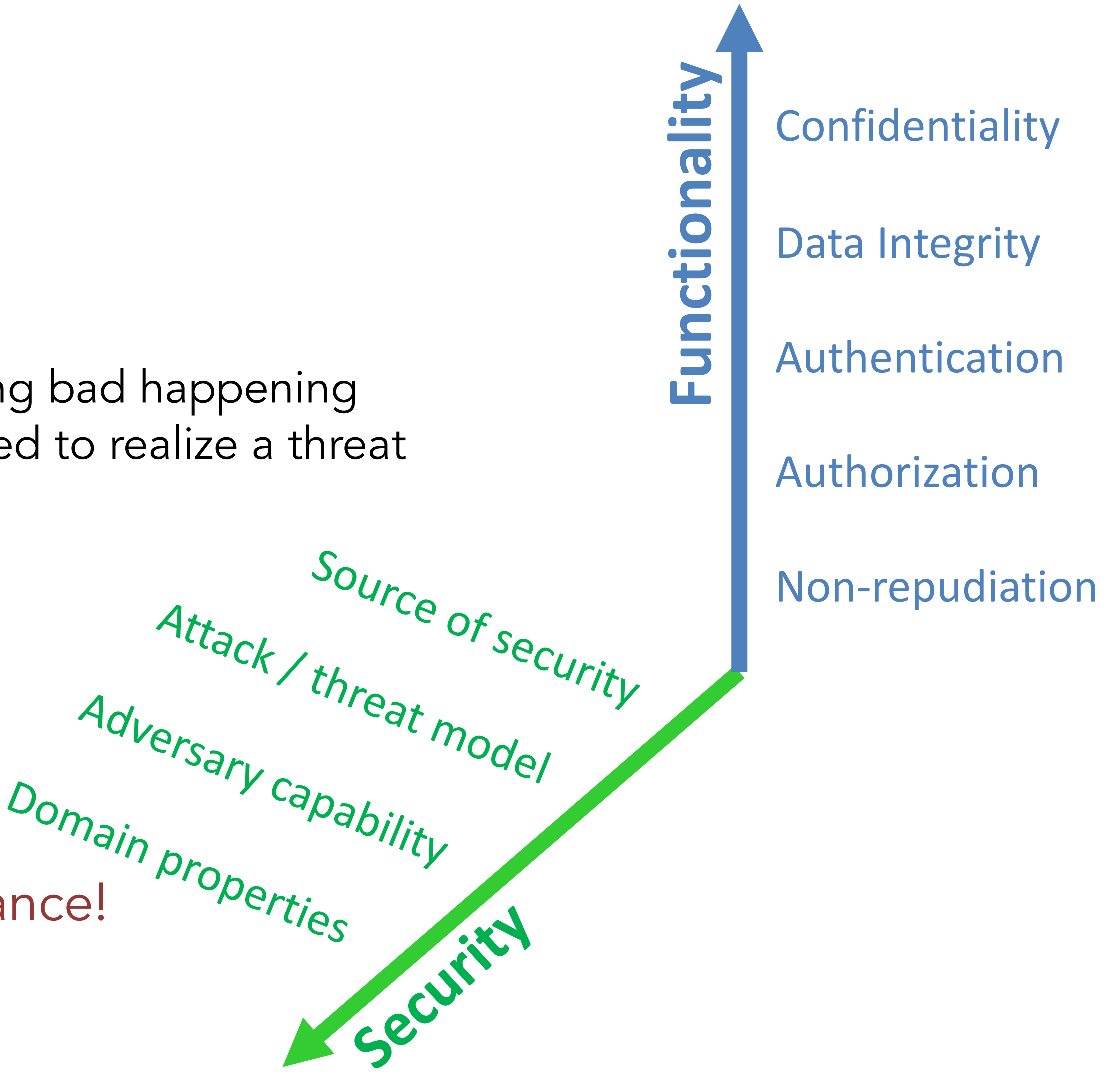
Functional Security Properties of Systems

- Functionality:
  - CIA triad:
    - Confidentiality
    - Integrity
    - Availability
  - Types of security services, according to NIST [1]:
    - Confidentiality
    - Data Integrity
    - Authentication
    - Authorization
    - Non-repudiation
  - Not ordered by importance!

[1] Elaine Barker, NIST Special Publication 800-57 Part 1 Revision 5, Recommendation for Key Management.

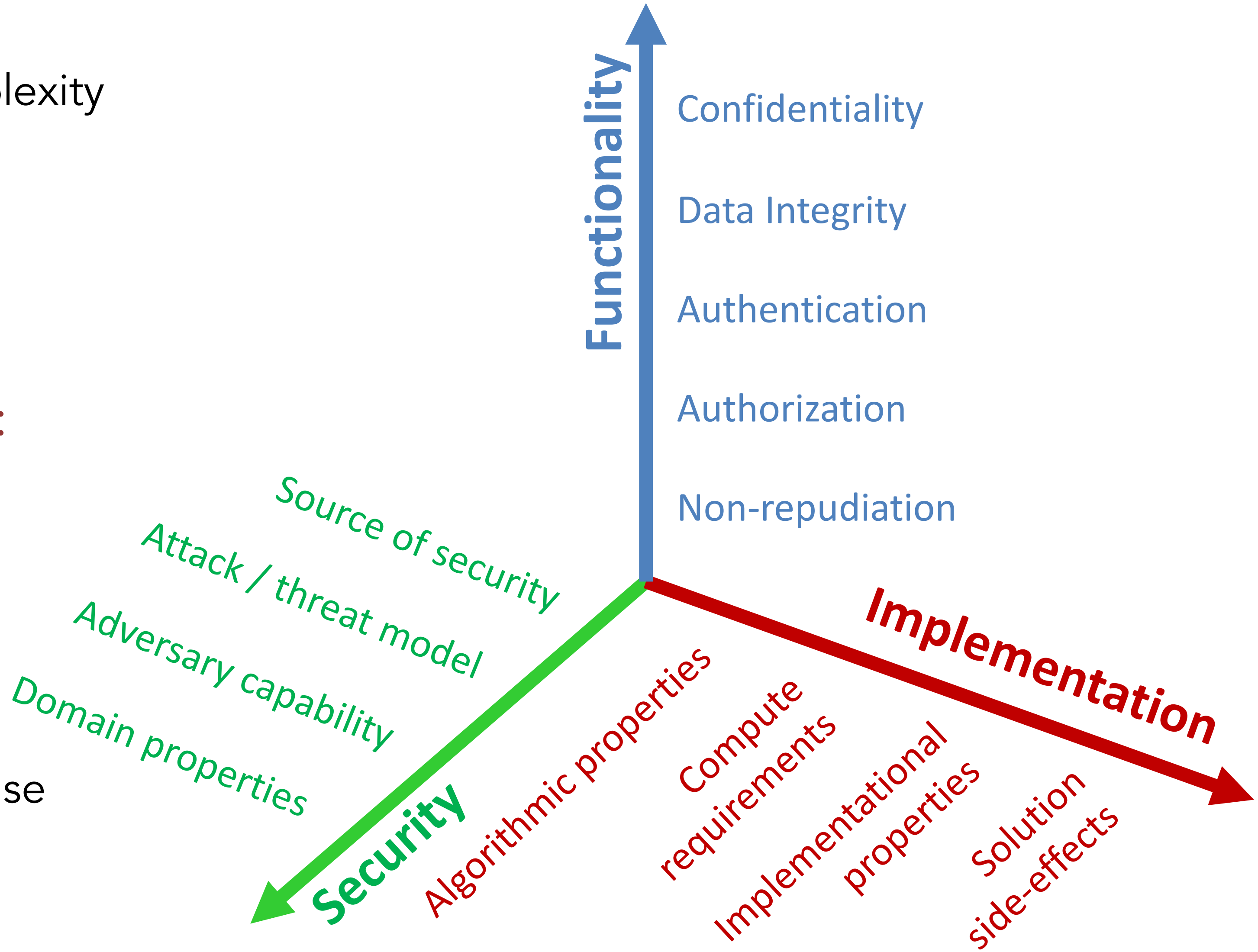
Functional Security Properties of Systems

- Security:
  - Source of security:
    - Information theoretic security
    - Computational security
      - Security nonuniformity
  - Attack & threat models:
    - threat = possibility of something bad happening
    - attack = a vulnerability exploited to realize a threat
  - Adversary capability:
    - Computational model
    - Computational resources
  - Domain properties:
    - Secure channels
    - Trusted parties / hardware
    - Domain assumptions
  - Again, not ordered by importance!



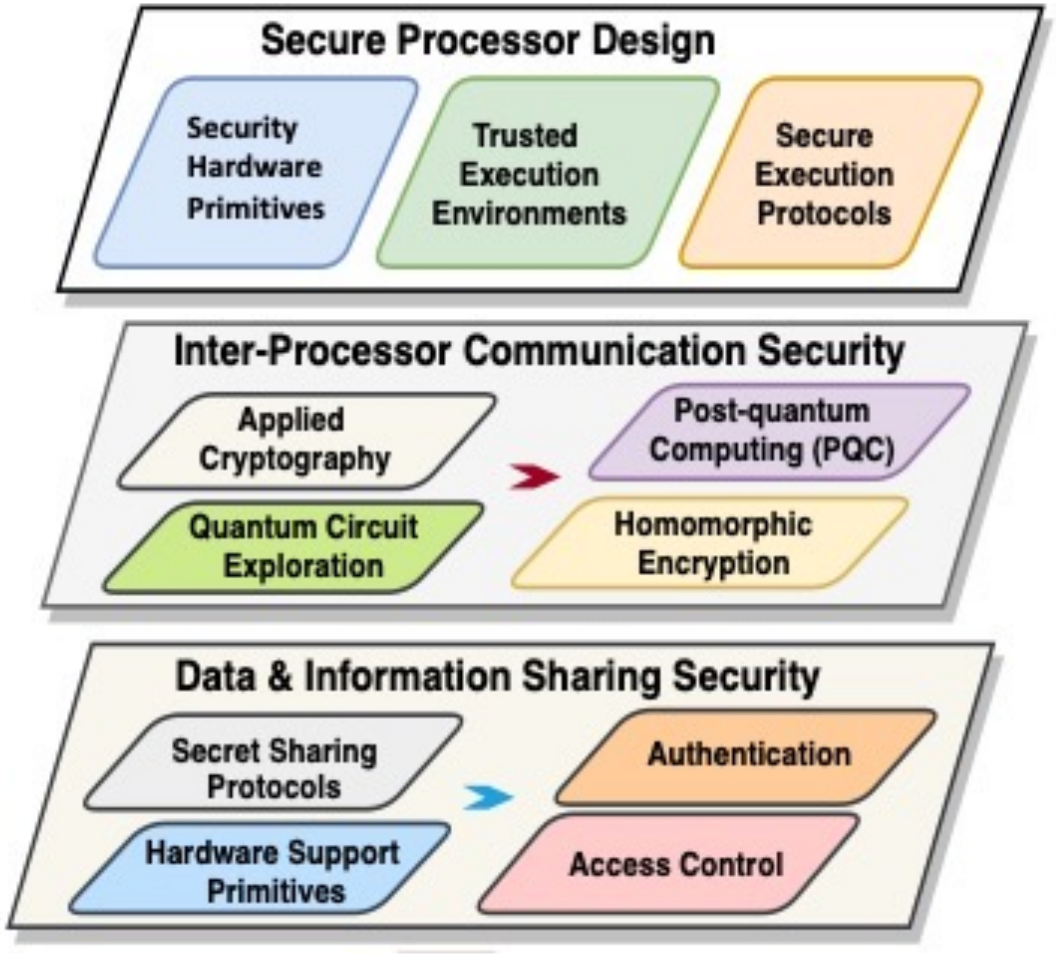
Security Concepts: Implementational Properties

- Implementational properties:
  - Algorithmic properties:
    - Computational / space complexity
    - Strong / weak scaling
  - Compute requirements:
    - FLOPS
    - Memory
    - Network bandwidth
  - Implementational properties:
    - Throughput
    - Latency
    - Power & area
    - Error correction, noise robustness
  - Solution side-effects:
    - Side-channel attacks & defense



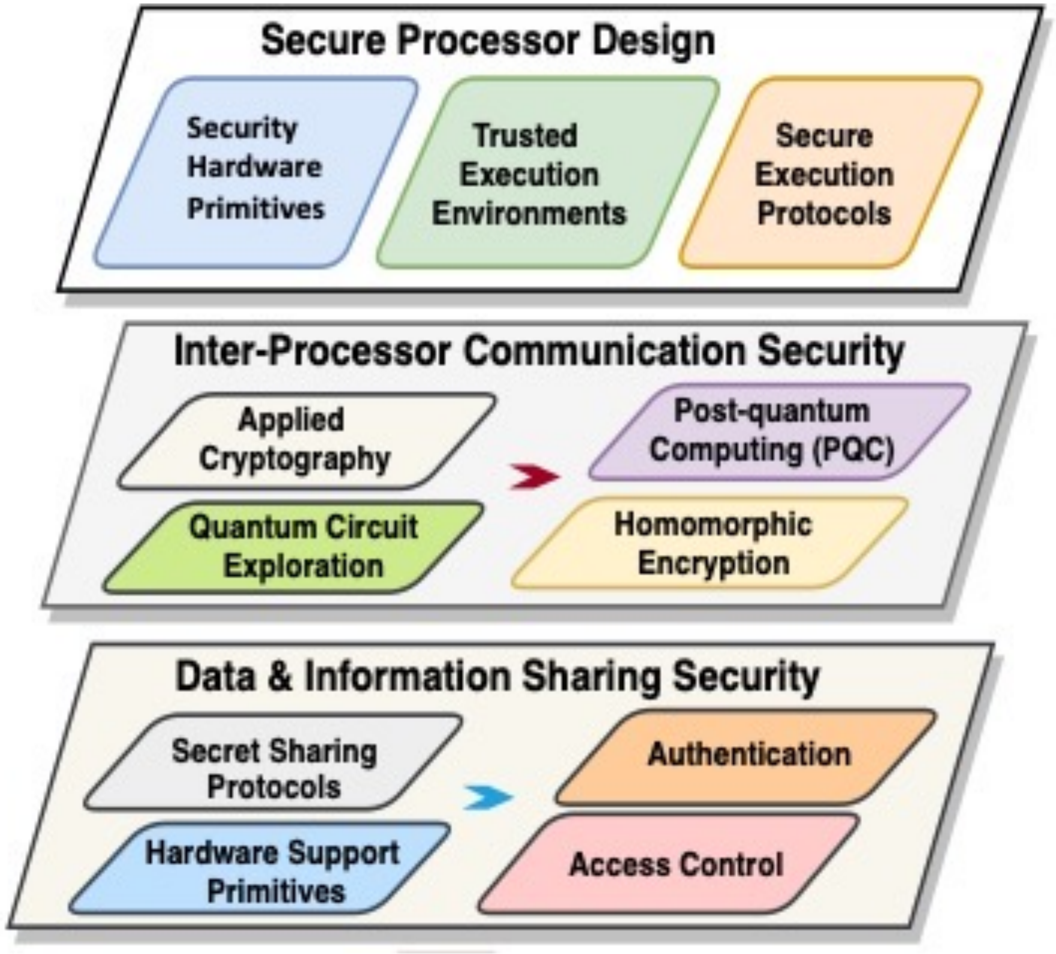
Computer Systems Security

- Processing - Data in manipulation
  - Program obfuscation
  - Opaque predicates
  - Virtualization, containerization, sandboxing
  - Secure architectures,
  - Trusted execution engines, secure enclaves
- Communication - Data in motion
  - Secure channels / cryptography
  - Key exchange / key distribution
  - Forward/backward secrecy
  - Oblivious Transfer
- Storage - Data at rest
  - Certificate authorities
  - Root certificates,
  - Self-signed certificates
  - Message authentication codes
- System-in-Use - Side-Channel
  - Control flow integrity
  - Shadow stacks
  - Buffer overflow protection:
  - ASLR, stack canaries
- Supply-Chain Trust Issues
  - Hardware obfuscation
  - Anti-tamper
  - split manufacturing



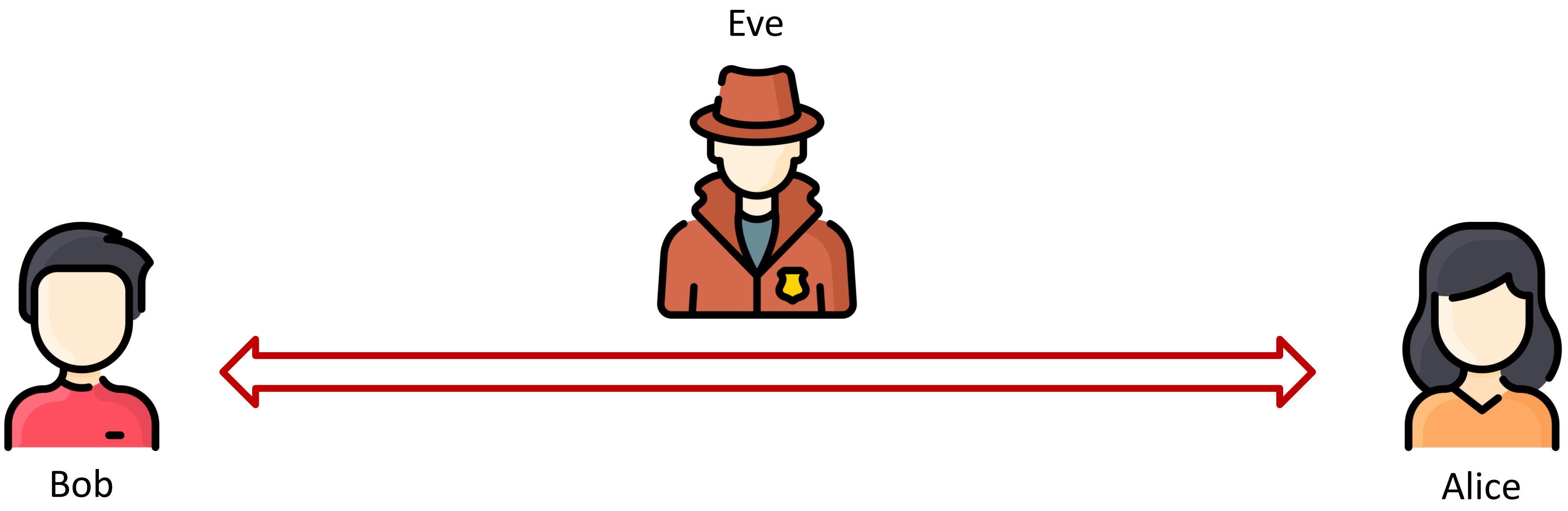
Computer Systems Security

- Processing - Data in manipulation
  - Program obfuscation
  - Opaque predicates
  - Virtualization, containerization, sandboxing
  - Secure architectures,
  - Trusted execution engines, secure enclaves
- Communication - Data in motion
  - Secure channels / cryptography
  - Key exchange / key distribution
  - Forward/backward secrecy
  - Oblivious Transfer
- Storage - Data at rest
  - Certificate authorities
  - Root certificates,
  - Self-signed certificates
  - Message authentication codes
- System-in-Use - Side-Channel
  - Control flow integrity
  - Shadow stacks
  - Buffer overflow protection:
  - ASLR, stack canaries
- Supply-Chain Trust Issues
  - Hardware obfuscation
  - Anti-tamper
  - split manufacturing



Secure Channels

- Scenario:
  - Alice and Bob need to privately communicate
  - The only channel between them is being eavesdropped on Eve
- Goal:
  - Need a method to privately transmit data over unsecure channel
- Assumptions:
  - Alice and Bob can securely and privately communicate ahead of time

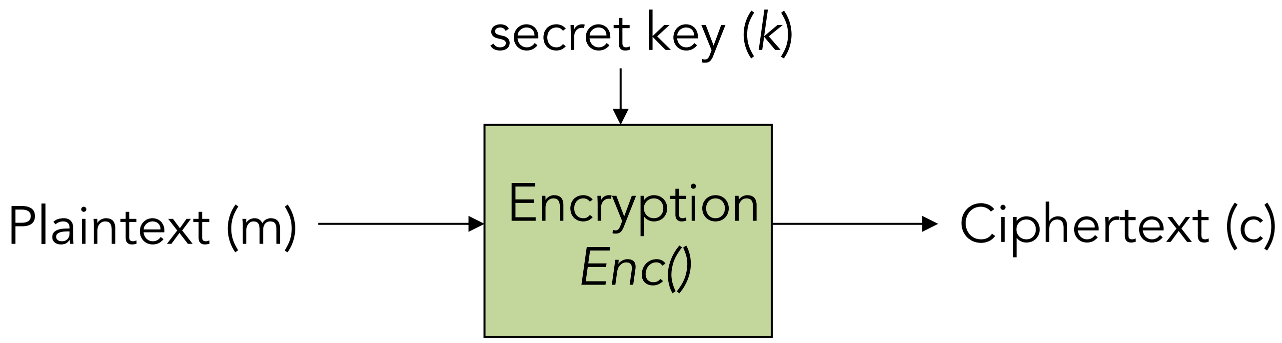


One-Time Pads

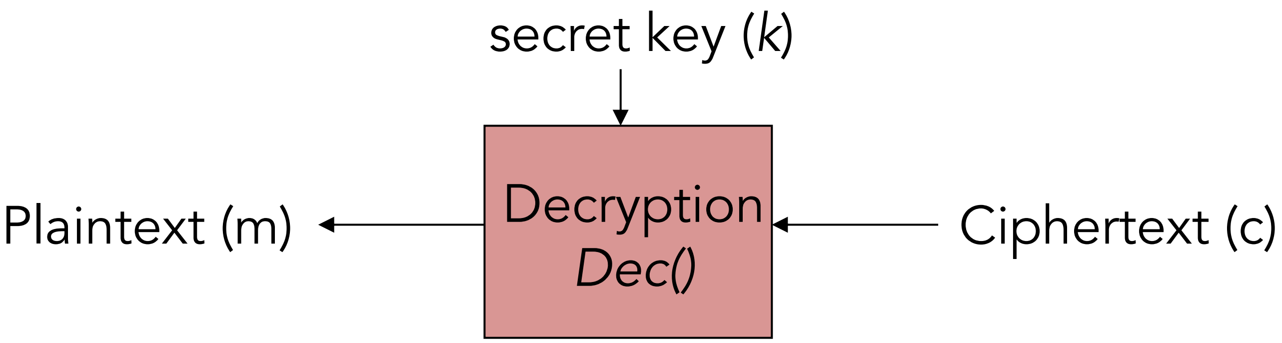
- Basic idea:
  - Alice and Bob exchange a codebook containing a stream of random numbers
  - When Alice wants to send Bob a message, she reads numbers from the codebook
  - Alice XORs the plaintext with them to create ciphertext
  - Alice sends ciphertext over an insecure channel
  - Bob performs the same steps to retrieve the plaintext
  - **Both cross out used numbers**
- Upsides:
  - Perfect provable secrecy
  - Plausible deniability
- Downsides:
  - "Key" is as large as the text
  - Must be communicated ahead of time
  - Cannot use encryption to send one-time pad if encryption is already using a one-time pad

Symmetric Cryptography

- Encryption
  - Input data, i.e., plaintext plus a secret key
  - Output is the ciphertext



- Decryption is the inverse function

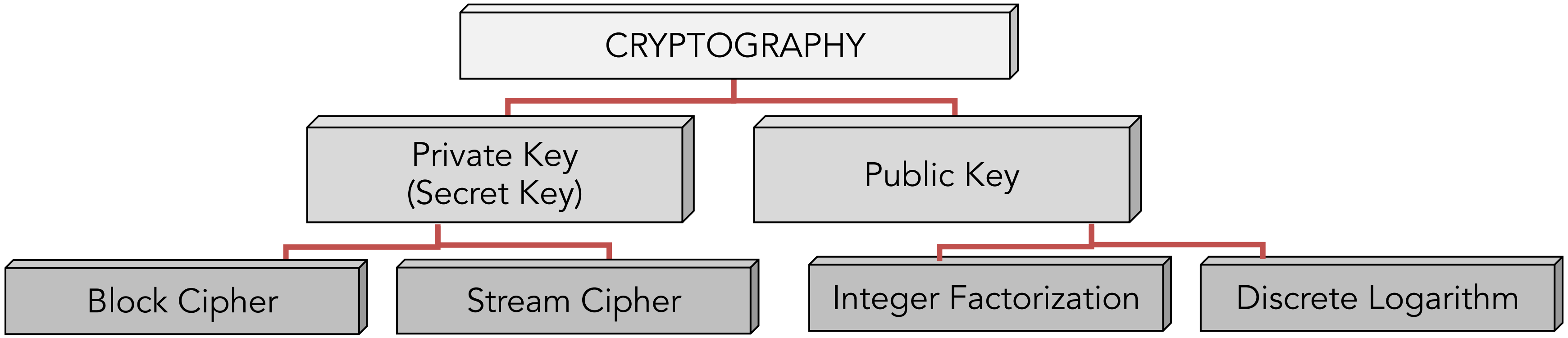


One-Way & Trapdoor Functions

- A one-way function is a function that is easy to compute but computationally hard to reverse
  - Easy to calculate f(x) from x
  - Hard to invert, i.e., calculate x from f(x)
- **There is no proof that one-way functions exist or that they can be constructed**
  - Generations of cryptographers have not made (public) progress
  - For example, the modular exponentiation function
    - Fairly easy to calculate (x^e mod n) from x
    - But hard to calculate x from (x^e mod n)
- A trapdoor one-way function has one additional property
  - With a certain knowledge, the function can be easily inverted
    - $x = (x^e \bmod n)^d \bmod n$
- We will see these more later when discussing asymmetric cryptography

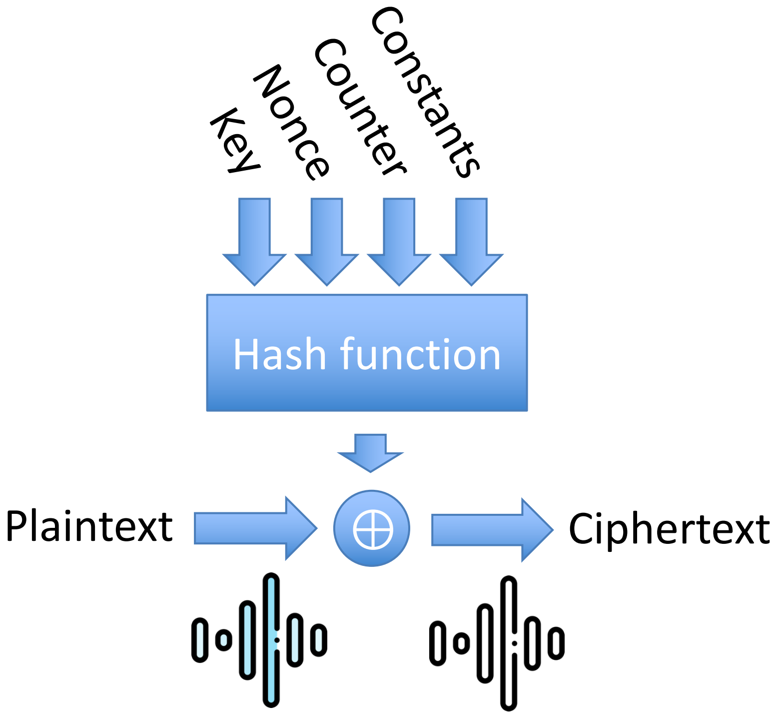
Aspects of Cryptography

- General categories of algorithms
  - Stream Ciphers
    - Operate on a variable-length stream
    - Generate a pseudo-random key stream and XOR with the plaintext
    - The algorithm key serves as the seed of the pseudo-random stream generator
  - Block Ciphers
    - Operate on blocks of predefined sizes



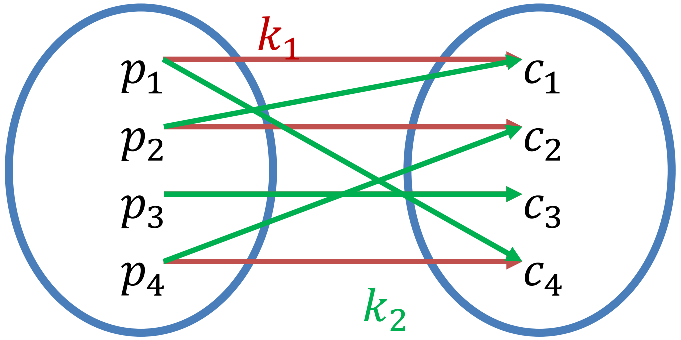
Stream Ciphers

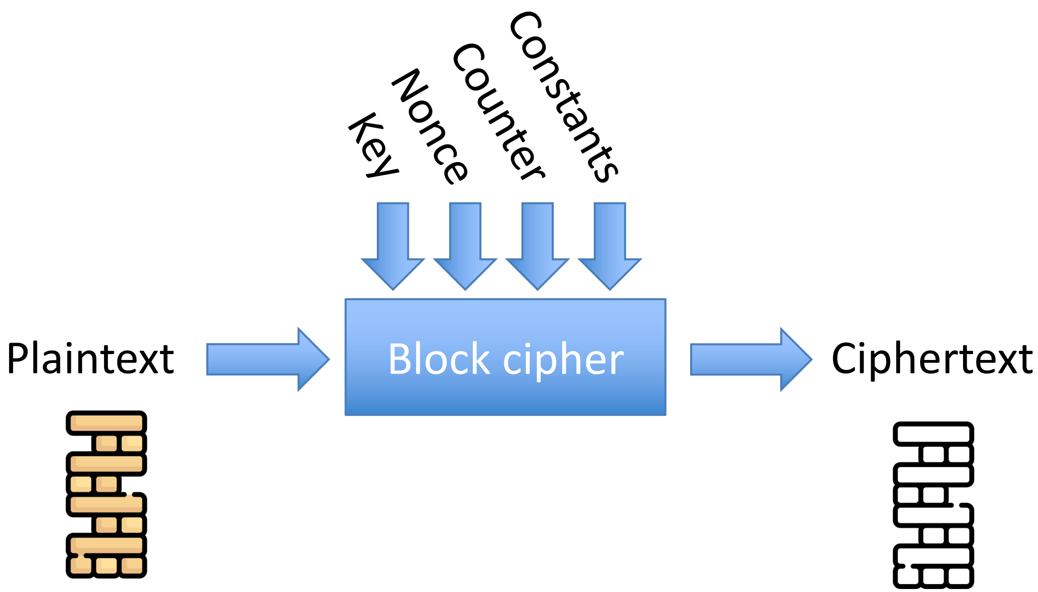
- Generate a stream of pseudo-random bits
- XOR bits with plaintext to create ciphertext
- To decrypt text, run RC4 in reverse
  - What does it mean to run "in reverse"?
  - Encryption and decryption must be perfectly aligned to work
- Upsides:
  - Very efficient implementation
  - No restrictions on plaintext size
  - Single-bit errors do not break rest of cipher
- Downsides:
  - By definition, stream ciphers operate on bits
  - A single ciphertext bit is a function of the key and **a single plaintext bit**
    - No avalanche effect w.r.t. the plaintext
      - Does this sound familiar?
    - Cipher feedback (CFB) can help (discussed in a couple of slides)
  - Malleable!



Block Ciphers

- A block cipher operating on b-bit inputs is a family of mappings on b bits specified by the key
  - k: q-bit key
  - p: b-bit string denoting a plaintext
  - c: b-bit string denoting a ciphertext
- Multiple *modes of operation* that can provide:
  - Confidentiality
  - Authentication
  - Error detection
- Upsides:
  - Better suited to modern networks
  - High diffusion:
    - Avalanche effect w.r.t. plaintext in some modes
- Downsides:
  - Slower than stream ciphers, may be impossible to preemptively execute the part of computation
  - One-bit errors threaten whole block





Example Block Cipher

- 2 bit block cipher, 2 bit key with encryption function defined by

Key 00

m	*c*
00	10
01	11
10	01
11	00

Key 01

m	*c*
00	11
01	00
10	01
11	10

Key 10

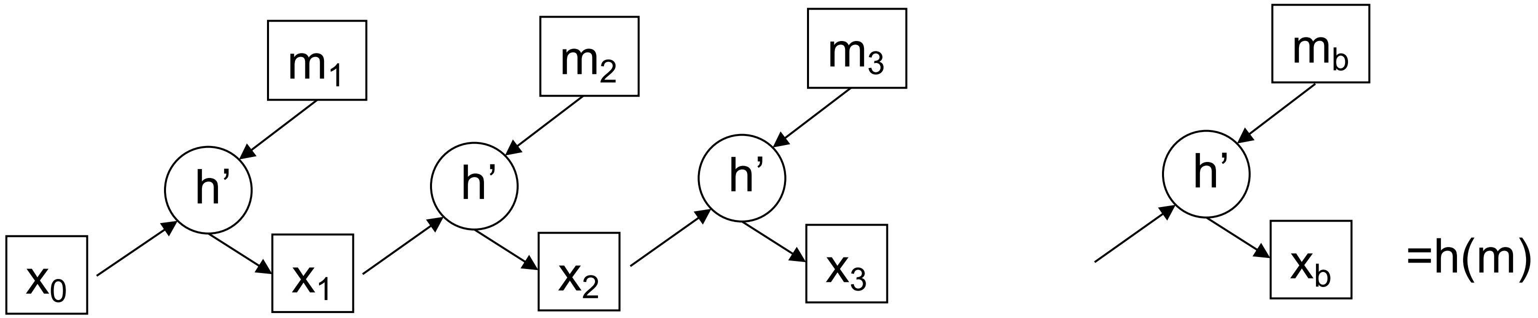
m	*c*
00	11
01	10
10	01
11	00

Key 11

m	*c*
00	01
01	00
10	11
11	10

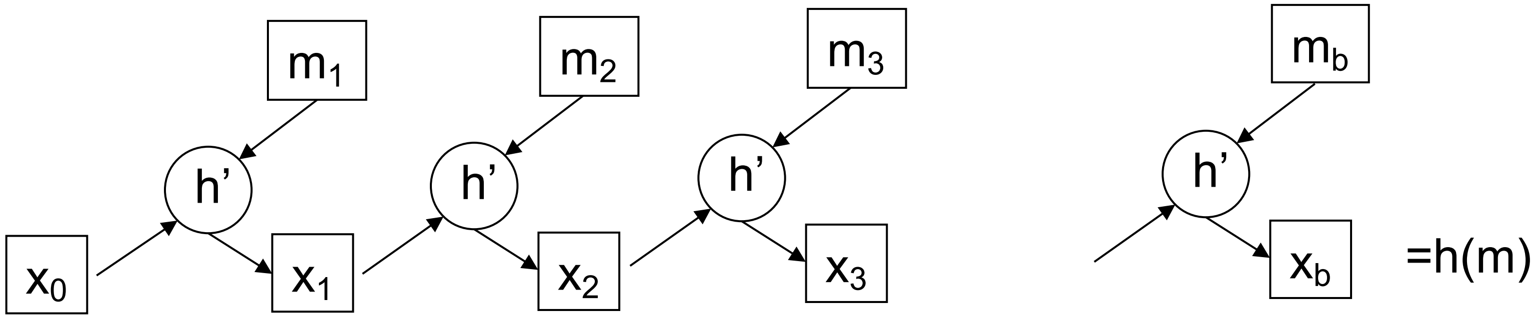
Block Ciphers as Hash Functions

- Iterate over all of the b blocks
- Use the output value from the previous block as input to the current block
  - x_0 is a constant



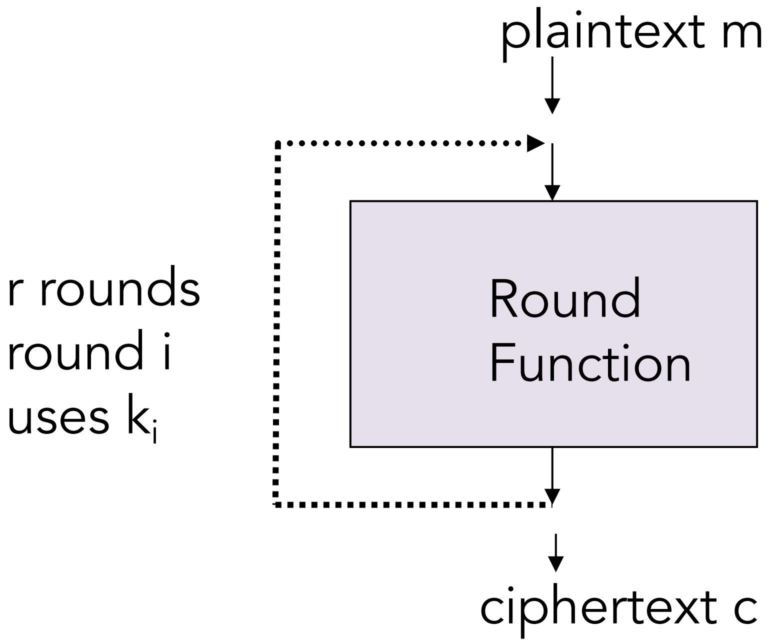
Block Ciphers as Hash Functions

- A commonly used cryptographic hash function is SHA-1
  - SHA-1 was originally designed by NIST and NSA in 1993/1995
  - It is used in the Digital Signature Standard (DSS)
  - SHA-256, SHA-384 and SHA-512



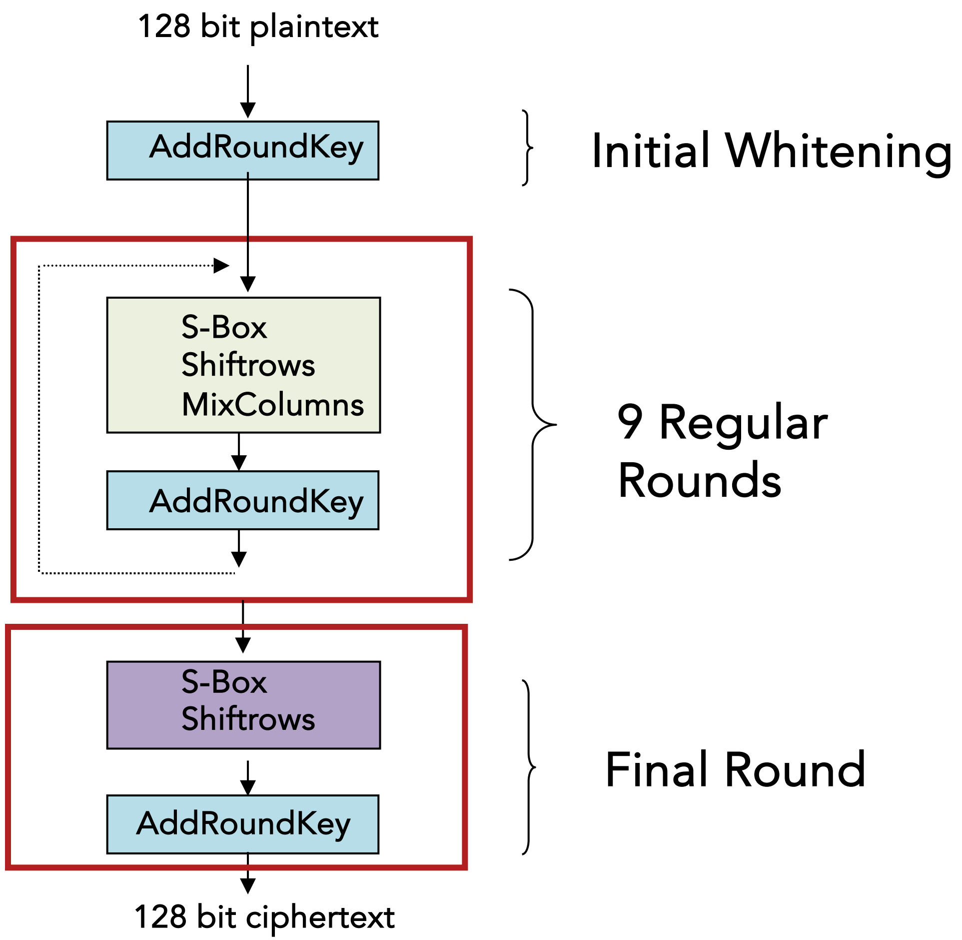
Modes of Operation

- *m* and *c* are fixed length (*e.g.*, 128 or 256 or 512 bits)
- Secret key, *k*, expanded via a function called a key schedule to create round keys $k_1, k_2, \ldots k_r$



Modes of Operation

- Advanced Encryption Standard (AES)
  - 10, 12, 14 rounds for 128, 192, 256 bit keys
    - Regular Rounds (9, 11, 13)
    - Final Round is different (10th, 12th, 14th)
  - Each regular round consists of 4 steps
    - Byte substitution (BSB)
    - Shift row (SR)
    - Mix column (MC)
    - Add Round key (ARK)



Modes of Operation

- *Diffusion*
  - *Byte Substitution*
    - Predefined substitution table
- *Confusion*
  - *Shift Row*
    - Left circular shift
- *Diffusion and Confusion*
  - *Mix Columns*
    - 4 elements in each column are multiplied by a polynomial
- *Confusion*
  - *Add Round Key*
    - Key is derived and added to each column

Modes of Operation

- 128-bit Shift Row

b0	b4	b8	b12
b1	b5	b9	b13
b2	b6	b10	b14
b3	b7	b11	b15

b0	b4	b8	b12
b5	b9	b13	b1
b10	b14	b2	b6
b15	b3	b7	b11

Modes of Operation

- Mix Column

$$\begin{bmatrix} S'_{0,I} \\ S'_{1,I} \\ S'_{2,I} \\ S'_{3,i} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} * \begin{bmatrix} S_{0,i} \\ S_{1,i} \\ S_{2,I} \\ S_{3,i} \end{bmatrix}$$

Modes of Operation

- Add Key

b0	b4	b8	b12
b1	b5	b9	b13
b2	b6	b10	b14
b3	b7	b11	b15

k0	k4	k8	k12
k1	k5	k9	k13
k2	k6	k10	k14
k3	k7	k11	k15

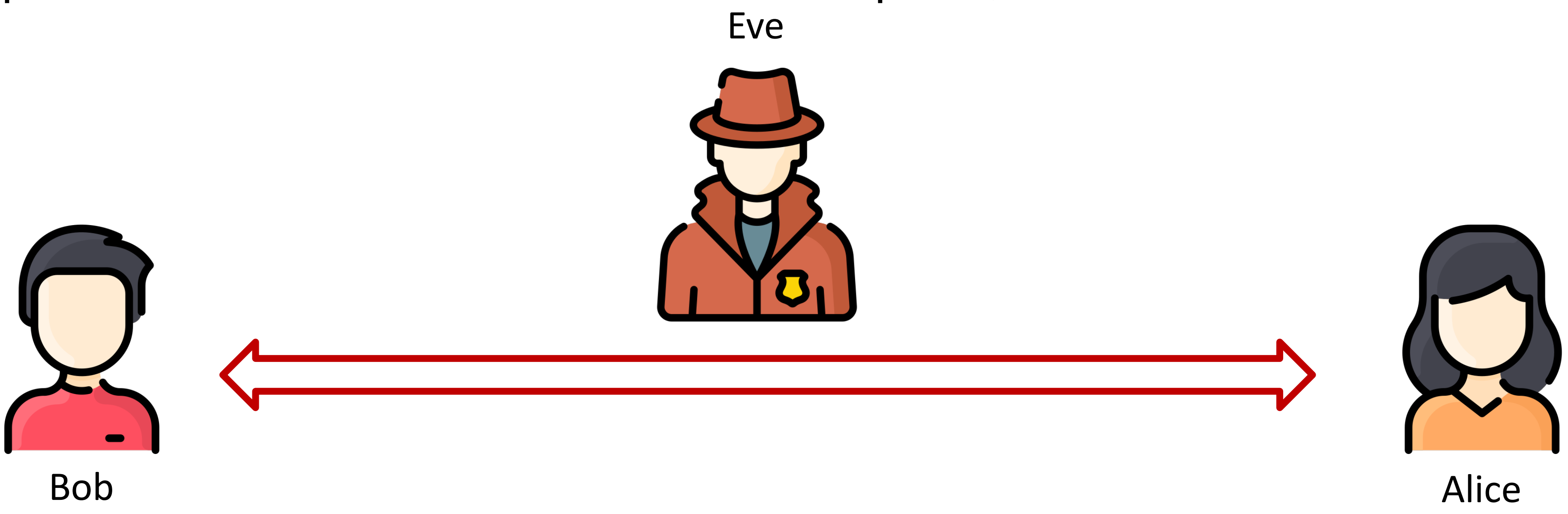
$$b'_x = b_x \text{ XOR } k_x$$

Data Encryption Standards

	DES (Data Encryption Standard)	AES
Date	1976	1999
Block size	64 bits	128 bits
Key length	56 bits	128, 192, 256, … bits
Encryption primitives	Substitution and permutation	Substitution, shift, bit mixing
Cryptographic primitives	Confusion and diffusion	Confusion and diffusion
Design	Open	Open
Design rationale	Closed	Open
Selection process	Secret	Secret (accepted public comment)
Source	IBM, enhanced by NSA	Belgian cryptographers

Problems with Symmetric Cryptography

- Alice and Bob require *prior communication* to privately communicate
  - Keys must be exchanged ahead of time
  - No secure method to exchange keys over the channel
- Not possible to authenticate other party



Diffie-Hellman Key Exchange Algorithm

- Allow two parties agree on a secret value
- Both parties compute the secret key $K=g^{xy}$
- Assuming the communication channel is authenticated
  - Which a very big assumption
- It cannot be used to exchange an arbitrary message
- It is a practical method for public exchange of a secret key
- It is based on exponentiation in a finite – Galois - field
  - Modulo a prime or a polynomial
    - This is easy
- The security relies on the difficulty of computing discrete logarithms
  - This is hard

Diffie-Hellman Key Exchange Algorithm

- Select two large numbers
  - One prime p and g a primitive root of p
  - p and g are both publicly available numbers
- Participant pick private values x and y
- Compute public values
  - $A = g^x \bmod p$
  - $B = g^y \bmod p$
- Public values A and B are exchanged
- Compute shared, private key
  - $k^x = B^x \bmod p$
  - $k^y = A^y \bmod p$
- $k^x = k^y$
- Participants now have a symmetric secret key to encrypt their messages

Diffie-Hellman

- This is just an introduction of the concept. There are number of issues to solve for its secure deployment
  - Man-In-The-Middle attack
  - Replay attack
  - Identity-misbinding attack
- Diffie-Hellman vs. RSA
  - Diffie-Hellman uses a symmetric key scheme, i.e., both participants agree on one key
  - RSA uses an asymmetric - public-private - key scheme such that a message encrypted by a public key, can only be decrypted by the corresponding private key

Asymmetric / Public Key Cryptosystem

- A public encryption method has
  - A public encryption algorithm
  - A public decryption algorithm
  - A public encryption key
- Using the public key and encryption algorithm anyone can encrypt a message
- The decryption key is known only to authorized parties
- RSA: Rivest, Shamir, Adleman

Necessary Math for RSA

- Modular arithmetic:
  - $a \cdot b = c \Rightarrow a \ (mod\ n) \cdot b \ (mod\ n) = c \ (mod\ n)$
  - $a \equiv b \ (mod\ n) \Rightarrow a^k \equiv b^k \ (mod\ n), k \in \mathbb{Z}$
  - $\left(a^x (mod\ n)\right)^y \ (mod\ n) = a^{xy} \ (mod\ n)$
  - $a \cdot \bar{a} \equiv 1 \ (mod\ n) \Rightarrow \bar{a}$ is the modular inverse of a
- Euler's totient function:
  - Euler's totient function $\phi(n)$ counts the positive integers up to a given integer n that are relatively prime to n
  - If n is prime, $\phi(n) = n - 1$
  - $\phi(pq) = \phi(p)\phi(q)$
- Euler's theorem:
  - If a and n are coprime integers, $a^{\phi(n)} \equiv 1 \ (mod\ n)$

Public Key Cryptosystem (RSA)

- Let p and q be two prime numbers
  - $n = pq$
  - $m = (p-1)(q-1)$
- x is such that $1 < x < m$ and $\gcd(m,x) = 1$
- y is such that $(xy) \bmod m = 1$
  - x is computed by generating random positive integers and testing $\gcd(m,x) = 1$ using the extended Euclid's gcd algorithm
  - The extended Euclid's gcd algorithm also computes y when $\gcd(m,x) = 1$

Public Key Cryptosystem (RSA)

- Security relies on the fact that prime factorization is computationally very hard
  - If k is the number of bits in the binary representation of n
  - There is no known algorithm, polynomial in k, to find the prime factors of n
- RSA Encryption And Decryption
  - Message $M < n$
  - Encryption key = (a, n)
  - Decryption key = (b, n)
  - Encrypt
    - $Enc(M) = M^a \bmod n$
  - Decrypt
    - $Dec(M) = E^b \bmod n$

RSA Asymmetric Cryptosystem

Setup:

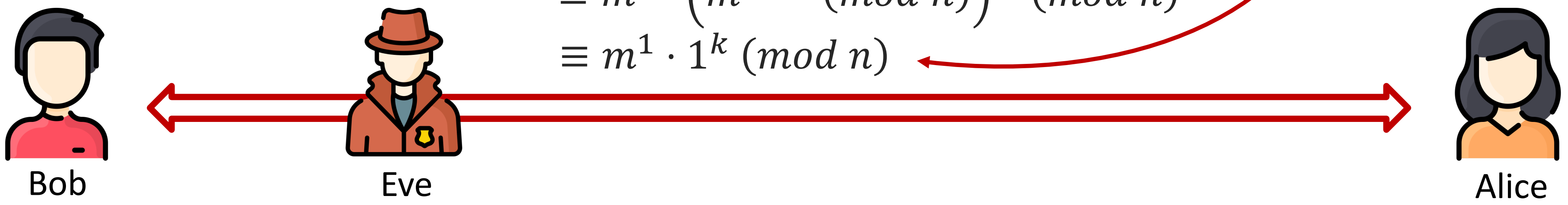
- Alice publishes a public key (n, e):
  - $n = pq$, where p and q are large prime numbers
  - e is some large number, e.g., $e = 2^{16} + 1 = 65537$
    - Chosen to be relatively prime to $\phi(n) = (p-1)(q-1)$
- Alice calculates her private key d:
  - $ed \equiv 1 \left(mod\ \phi(n)\right)$
    - $ed = 1 + k\phi(n)$

Transmitting messages:

- Bob wants to send message m to Alice
  - $m < n$, so Bob potentially splits message into smaller pieces
- Bob sends $c \equiv m^e\ (mod\ n)$
- Alice calculates $c^d \equiv m\ (mod\ n)$

$$
\begin{aligned}
m &\equiv c^d (mod\ n) \\
&\equiv \left(m^e (mod\ n)\right)^d (mod\ n) \\
&\equiv m^{ed}\ (mod\ n) \\
&\equiv m^{1+k\phi(n)}\ (mod\ n) \\
&\equiv m^1 \cdot m^{k\phi(n)}\ (mod\ n) \\
&\equiv m^1 \cdot \left(m^{\phi(n)}\ (mod\ n)\right)^k\ (mod\ n) \\
&\equiv m^1 \cdot 1^k\ (mod\ n)
\end{aligned}
$$

Euler's theorem:
$a^{\phi(n)} \equiv 1\ (mod\ n)$



Security Foundation of RSA

- A large value of n prevents finding prime factors p and q
  - Factorizing large numbers is very hard
- e chosen to be a very large integer relatively prime to $(p-1)(q-1)$:
  - $m^e \ (mod\ n)$ "wraps" many times
  - Finding discrete logarithms is very hard

Brief Review of Number Theory

Brief Review of Number Theory

- Divisibility
  - Given integers x and y, with x > 0, x divides y (denoted x|y) if there exists an integer a, such that y = ax
    - x is then called a divisor of y, and y a multiple of x
  - Given integers x, y such that x>0, x<y then there exist two unique integers q and r, 0 <= r < x such that y = xq + r
  - r = y mod x
- An integer p > 1 is a prime number if only positive divisors of p are 1 and p
- Any integer number p > 1 that is not prime, is a composite number

Brief Review of Number Theory

- Fundamental Theorem of Arithmetic
- Any integer number x > 1 can be written as a product of prime numbers that are greater than 1
- The product is unique if the numbers are written in increasing order

$$X = d_1^{e1} . d_2^{e2} . d_3^{e3} \ldots d_k^{ek}$$

- Given integers x > 0 and x > 0, we define gcd(x, y) = z, the greatest common divisor (GCD), as the greatest number that divides both x and y
- The integers x and y are relatively prime (rp) if gcd(x, y) =1

Brief Review of Number Theory

- Given integers x, y > 0 and m > n, then z = gcd(x,y) is the least positive integer that can be represented as z = mx + ny
- Given integers x, y, z >1
  - If gcd(x, z) = gcd(y, z) = 1, then gcd(xy, z) = 1
- The least common multiple (lcm) of the positive integers x and y is the smallest positive integer that is divisible by both x and y
- What is the least common multiple of $2^3 3^5 7^2$ and $2^4 3^3$?

Brief Review of Number Theory

- What is the least common multiple of $2^3 3^5 7^2$ and $2^4 3^3$?
  - $lcm(2^3 3^5 7^2, 2^4 3^3) = 2^{max(3,4)} \cdot 3^{max(5,3)} \cdot 7^{max(2,0)} = 2^4 3^5 7^2$
- Let x and y be positive integers, then $xy = gcd(x,y).lcm(x, y)$
- All of these transformations and definitions have formal proofs

Brief Review of Number Theory

- Euclidean Algorithm
  - Given integers x and y great or equal to 1, on can use the division algorithm repeatedly

$$y = q_1 x + r_1 \qquad 0 \leq r_1 < x$$

$$x = q_2 r_1 + r_2 \qquad 0 \leq r_2 < r_1$$

$$\ldots$$

$$r_{k-2} = q_k r_{k-1} + r_k \qquad 0 \leq r_k < r_{k-1}$$

$$r_{k-1} = q_{k+1} r_k$$

  - The remainders r_i get smaller
    - $r_1 > r_2 > \cdots \geq 0$

Brief Review of Number Theory

- Let (x, y) be in Z^2, and n in Z^+, then x is a congruent to y modulo n if n divides a – b
  - $x \equiv y \pmod{n}$
- Similarly, given n> 0, x, y, we say that y is a multiplicative inverse of x modulo n if $xy \equiv 1 \pmod{n}$
  - $(x \bmod n) = (y \bmod n) \rightarrow x \equiv y \pmod{n}$

Modular Arithmetic

- Cummutative Laws
  - (y + x) mod n = (x + y) mod n
  - (y * x) mod n = (x * y) mod n
- Associative Laws
  - [(z + x) + y] mod n = [z + (x + y)] mod n
  - [(z * x) * y] mod n = [z * (x * y)] mod n
- Distributive Law
  - [z * (x + y)] mod n = [(z * x) + (z * y)] mod n
- Identities
  - (0 + x) mod n = x mod n
  - (1 * x) mod n = x mod n
- Additive Inverse (-w)
  - For each x in Z_n, there exists a r such that $x + r \equiv 0 \bmod n$

Brief Review of Number Theory

- Quadratic residues
  - If there is an integer s, with 0< x <p, such that $x^2 = q \pmod p$
  - If the congruence $x^2 = q \pmod p$ has a solution, then q is a quadratic residue of p
  - If the congruence $x^2 = q \pmod p$ has no solution, then q is a quadratic nonresidue of p
- Quadratic reciprocity
  - It relates the solvability of the congruence
    - $x^2 = q \pmod p$
  - To the solvability of the congruence
    - $x^2 = p \pmod q$
    - Where p and q are distinct odd primes

Brief Review of Number Theory

- Our goal in this class is to quickly run through some these concepts as they form the foundation of modern cryptography and by default computer security
  - This allows us to better understand the gap between the theoretical aspects of these problems and the impurities introduced by their software and/or hardware implementation or even their susceptibility to side-channel attacks
- For example, understanding of prime factorization
  - Prime Factorization Theorem
    - Every integer $n > 2$ can be written as a product of one or more primes
- There is an infinite number of primes

Review of Groups

- Definition of a Group
  - A Group G is a collection of elements together with a binary operation* which satisfies the following properties
    - Closure
    - Associativity
    - Identity
    - Inverses
- * A binary operation is a function on G which assigns an element of G to each ordered pair of elements in G.
  - For example, multiplication and addition are binary operations

Review of Groups

- Groups may be finite or infinite
  - They are finite when they have a finite number of elements
- Groups may be commutative or non-commutative
- A set G with a binary operation + (addition) is called a commutative group if
- The commutative property may or may not apply to all elements of the group
  - Commutative groups are also called Abelian groups

Review of Groups

- Groups may be finite or infinite
  - They are finite when they have a finite number of elements
- Groups may be commutative or non-commutative
- A set G with a binary operation + (addition) is called a commutative group if

1. $\forall\, x,y \in G,\ x+y \in G$
2. $\forall\, x,y,z \in G,\ (x+y)+z=x+(y+z)$
3. $\forall\, x,y \in G,\ x+y=y+x$
4. $\exists\, 0 \in G,\ \forall\, x \in G,\ x+0=x$
5. $\forall\, x \in G,\ \exists\, -x \in G,\ x+(-x)=0$

Review of Groups

- The commutative property may or may not apply to all elements of the group
  - Commutative groups are also called Abelian groups
- Infinite and Abelian:
  - For example, the integers under the addition operation (Z +)
  - The rational numbers without 0 under multiplication (Q*, x)
- Infinite and non-Abelian
- Finite and Abelian
  - The integers mod n under modular addition operation (Z_n, +)
- Finite and non-Abelian

Review of Groups

- Let (G, +) be a group, (H,+) is a sub-group of (G,+) if it is a group, and $H \subseteq G$
  - If (G, +) be a finite group, $H \subseteq G$, and H is closed under +, then (H,+) is a sub-group of (G,+)
  - Lagrange theorem
    - If G is finite and (H,+) is a sub-group of (G,+) then |H| divides |G|
- Let x^n denote $\underbrace{x+\ldots+x}_{\text{(n times)}}$
- The x is of order n if $x^n = 0$, and for any m<n, $x^m \neq 0$
- Euler theorem
  - In the multiplicative group of Z_n, every element is of order at most $\varphi(n)$

Review of Groups

- If G be a group and x be an element of order n, then the set $<x>=\{1, x,\ldots,x^{n-1}\}$ is a sub-group of G
  - x is then the generator of the set $<x>$
- If G is generated by x, then G is called cyclic, and x is a primitive element of G
- For any prime p, the multiplicative group of Z_p is cyclic
- If G is a group with $x \in G$, then $H=\{x^n | n \in Z\}$ is a sub-group of G
  - It is the cyclic sub-group $<x>$ of G generated by x
- Every cyclic group is abelian cyclic

Review of Groups

- Rings
  - A set G with two binary operations + and * is called a commutative ring with identity if

1. $\forall\ x,y \in G,\ x+y \in G$
2. $\forall\ x,y,z \in G,\ (x+y)+z=x+(y+z)$
3. $\forall\ x,y \in G,\ x+y=y+x$
4. $\exists\ 0 \in G,\ \forall\ x \in G,\ x+0=x$
5. $\forall\ x \in G,\ \exists\ {-x} \in G,\ x+(-x)=0$
6. $\forall\ x,y \in G,\ x*y \in G$ 6.
7. $\forall\ x,y,z \in G,\ (x*y)*z=x*(y*z)$
8. $\forall\ x,y \in G,\ x*y=y*x$
9. $\exists\ 1 \in G,\ \forall\ x \in G,\ x*1=x$
10. $\forall\ x,y,z \in G,\ x*(y+z)=x*y + x*z$
11. $\forall\ x \neq 0 \in G,\ x*x^{-1} =1$

Review of Groups

- Fields
  - A field is a commutative ring with identity where each non-zero element has a multiplicative inverse
    - $\forall x \neq 0 \in G, \exists x^{-1} \in G, x * x^{-1} = 1$
- Given a polynomial function f of degree n in one variable x over a field G, i.e., $a_n, a_{n-1}, \ldots, a_1, a_0 \in G$
  - $f(x) = a_n * x^n + a_{n-1} * x^{n-1} + a_{n-2} * x^{n-2} + \ldots + a_1 * x + a_0$
  - f(x)=0 has at most n solutions in G
- Polynomial remainders
  - $f(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \ldots + a_1 \cdot x + a_0$
  - $g(x) = b_m \cdot x^m + b_{m-1} \cdot x^{m-1} + b_{m-2} \cdot x^{m-2} + \ldots + b_1 \cdot x + b_0$
    - Two polynomials over G such that $m \leq n$
    - There is a unique polynomial r(x) of degree less than m over G such that $f(x) = h(x) * g(x) + r(x)$
    - r(x) is called the remainder of f(x) modulo g(x)

Review of Groups

- Finite field
  - A field (G,+,*) is called a finite field if the set G is finite
- Galois Fields GF(p^k)
  - For every prime power p^k (k=1,2,…) there is a unique finite field containing p^k elements.
  - These fields are denoted by GF(p^k)
  - There are no finite fields with other cardinalities

Discrete Logarithm

- Let G be a group, q ∈ G, and $y=q^x$ where x the minimal non negative integer satisfying $y=q^x$
  - x is the discrete log of y to base q
- Let $y=q^x$ mod p be in the multiplicative group of Z_p
  - The exponentiation steps are $O(\log^3 p)$
  - Standard discrete log is computationally hard
    - q^x given x is easy
    - Finding x given q^x is hard - computationally infeasible
- $X \vdash q^x$ is a one way function
- Finally we have arrived to the essence of modern cryptography

Birthday Paradox

- Let G be a finite set of elements of size n
- If we select k elements of G uniformly and independently, what is the probability of getting at least one collision?
- Consider the event E_k with no collision after k elements
- Prob(E_k) = $1(1-\frac{1}{r})(1-\frac{2}{r})\ldots(1-\frac{k-1}{r})$

$$< \exp(-\frac{1}{r})\exp(-\frac{2}{r})\ldots\exp(-\frac{k-1}{r})$$

$$= \exp(-(1+2+\ldots+\frac{k-1}{r})$$

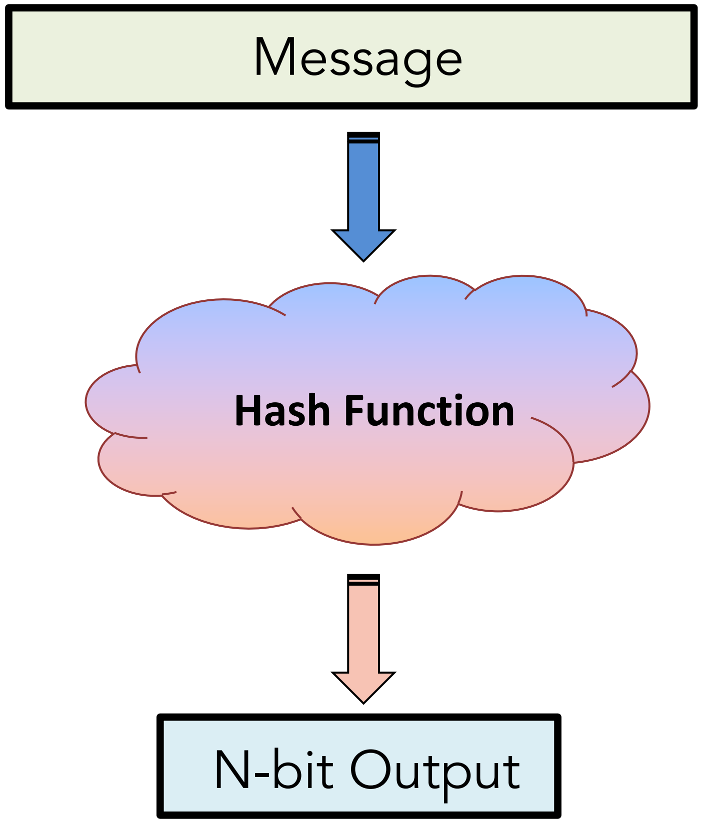
$$= \exp(-\frac{k(k-1)}{2r})$$

$$\sim \exp(-\frac{k^2}{2r})$$

- If $k=r^{1/2}$, then Prob(E_k)<0.607

Review of Hash Functions

- A hash function that maps a message of an arbitrary length to an n-bit output (digest)
- For a function f: $X \rightarrow Y$
  - It is injective if $f(x) = f(y)$ implies $x = y$ for all $x, y \in X$,
  - Surjective if for any $y \in Y$ there is $x \in X$ with $f(x) = y$,
  - Bijective if it is both injective and surjective
  - If there is a bijection between two finite sets, then the sets have the same number of elements



Review of Hash Functions

- A hash function that maps a message of an arbitrary length to an n-bit output
- Hash functions can be implemented using compression functions
- A hash function is a many-to-one function, so collisions can happen
- A cryptographic hash function has additional properties
  - One-wayness
    - It is computationally infeasible/expensive to find messages mapping to specific hash outputs
  - Collision freedom
    - It is computationally infeasible/very unlikely to find two messages that hash to the same output

Review of Hash Functions

- Message Integrity Check (MIC)
  - Send hash of message, i.e., digest
  - The digest is sent always encrypted
- Message Authentication Code (MAC)
  - Send keyed hash of message
  - MAC, message optionally encrypted
- Digital Signature for non-repudiation
  - Encrypt hash with private signing key
  - Verify with public verification key

Review of Hash Functions

- Pseudorandom function (PRF)
  - Generate session keys, nonces
  - Produce key from password
  - Derive keys from master key cooperatively
- Pseudorandom number generator (PRNG)
  - Vernam Cipher
  - S/Key, proof of "knowledge" via messages

Review of Hash Functions

- Lamport One-time Passwords
  - Provide password safety in distributed systems
    - Server compromise does not compromise the password
    - Interception of authentication exchange also does not compromise password
- Illustration
  - Alice picks a password p_A
  - She hashes the password n times, $h^n(p_A)$
  - Server stores (Alice, n, $h^n(p_A)$)
  - Attacker is not able to get p_A from $h^n(p_A)$

Review of Hash Functions

- Lamport One-time Passwords
  - Provide password safety in distributed systems
    - Server compromise does not compromise the password
    - Interception of authentication exchange also does not compromise password
- Illustration
  - Protocol
    - Alice sends "Alice"
    - Server sends "n-1"
    - Alice sends "x" where $x= h^{n-1}(p_A)$
    - Server verifies $h(h) = h^n(p_A)$
    - Server updates to (Alice, n-1, x)
    - Attacker still cannot extract p_A or impersonate Alice

In Summary

- Our goal in this class is to quickly run through some these concepts as they form the foundation of modern cryptography and by default computer security
  - This allows us to better understand the gap between the theoretical aspects of these problems and the impurities introduced by their software and/or hardware implementation or even their susceptibility to side-channel attacks
- You must understand to a certain degree some the mathematical underpinnings of these systems, their general design goals, approaches and strengths to be able to:
  - Select the appropriate and best fitting one for a given design situation or platform
  - Understand their potential (a) inherent vulnerabilities, (b) additional software implementation vulnerabilities, or (c) additional hardware implementation vulnerabilities

Next Topic

- Message Authentication: Secrecy vs. Integrity