Cryptography: Algorithms

Adapted by Joseph Lunderville
from slides by Dr. Brian Fraser
and course material by Dr. Steve Ko

Topics

- What is cryptography?
- What are the basics of cryptographic algorithms?
  - What are cryptographic hashes?
  - What is secret key encryption?
  - What is public key encryption?

The Absolute Basics

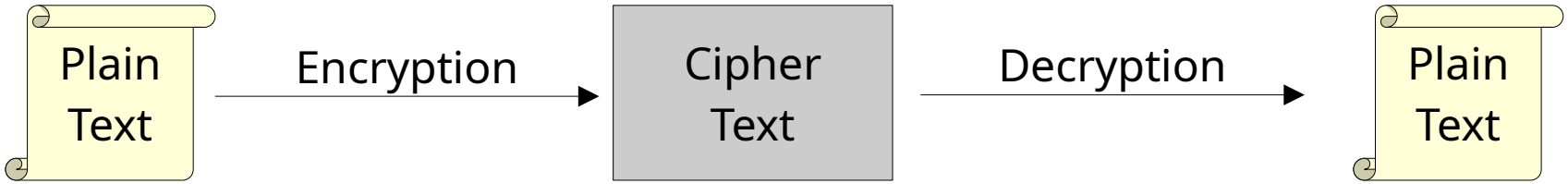
Context

- Cryptography
  - A *broad and deep* subject
  - We'll focus on how to use cryptography
  - We just touch on the most basic of basics!

The CIA Model

- *CIA model: the classic security model*
  - **Confidentiality:** information is only *disclosed* to those *authorized* to know it
  - **Integrity:** information is only *modified* in *allowed* ways, by *authorized* parties
  - **Availability:** those authorized for access *are not prevented from it*
- *Threat examples*
  - Against *confidentiality:* classified information leak
  - Against *integrity:* fake images, malicious updates
  - Against *availability:* Denial-of-Service (DoS) attacks

Message Encryption, Generally



- Cryptographers invented secret codes to hide messages from unauthorized observers
- *Challenges*
  - How can you hide a message from everyone but the intended recipient?
  - How can the recipient know the message is authentic?

Audience Participation - Historical Cryptography

- Historical cryptography
  - Secret codes, which are secret algorithms
  - **Caesar Cipher:** shift each letter a certain number of letters down the alphabet

    E.g., for +1, ‘A’ becomes ‘B’
- Which of the following is the cipher text from using a ***3****-letter shift* Caesar Cipher on the plain text “Hello world”?
- What is the problem with a secret algorithm?
  - When your algorithm (or code book) is compromised, your code is broken (useless)

a) EBIIL TLOIA b) KHOOR ZRUOG

c) IFMMP XPSME d) LOWOR LDHEL

Modern Encryption

- *Algorithms are public*
  - Keys are secret which provide the security
  - May be symmetric (secret key) or asymmetric (public key)
- *Why is this better?*
  - If the algorithm or code is secret, when it falls into the wrong hands it compromises the code
    - Good luck keeping your software out of the hands of an attacker forever!
  - If only key is private, then if it falls into the wrong hands, it is easy to replace with a new key

Cryptographic Algorithm Goals

- *Ideal properties of an encryption algorithm*
  - With the correct key, it should be relatively easy to encrypt or decrypt a message
  - Without the key, it should be hard to compute (invert) or decrypt a message
- *Key length sets a maximum bar for strength*
  - A shorter key is easier to guess, by brute force
  - Longer keys are stronger if the algorithm makes good use of the key

Window of Validity

- ***Window of Validity***
  - The minimum time to compromise a cryptographic algorithm
  - Must only use algorithm that have not been compromised!
- *May force updates*
  - Window of validity of your crypto function may be shorter than the lifetime of your system
  - Design systems so you can replace the cryptographic functions
    - This is hard to do well!
- *Example*
  - *1993:* SHA-0 was published
  - *1995:* Possible weakness was found in the SHA-0 algorithm; replaced with SHA-1
  - *2004:* Published way to compromise SHA-0
  - *2017:* Published way to compromise SHA-1
  - ????: Published way to compromise SHA-256?

Algorithm Types

- *Types of cryptography algorithms based on their keys*
  - Zero keys: hash functions (message authentication)
  - One key: secret-key functions (symmetric)
  - Two keys: public-key functions (asymmetric encryption)

Cryptographic Hash Functions (Zero Keys)

Cryptographic Hash Functions

- *Suppose we have a cryptographic* ***hash function*** $h()$
  - It takes a message m of arbitrary length as input and produces a smaller (short) number $h(m)$
  - This can be used as a stand-in for the message, when we don't want to process, transmit, or store the whole thing
- *Toy example*
  - $h(m) = m^2 \bmod 4321$

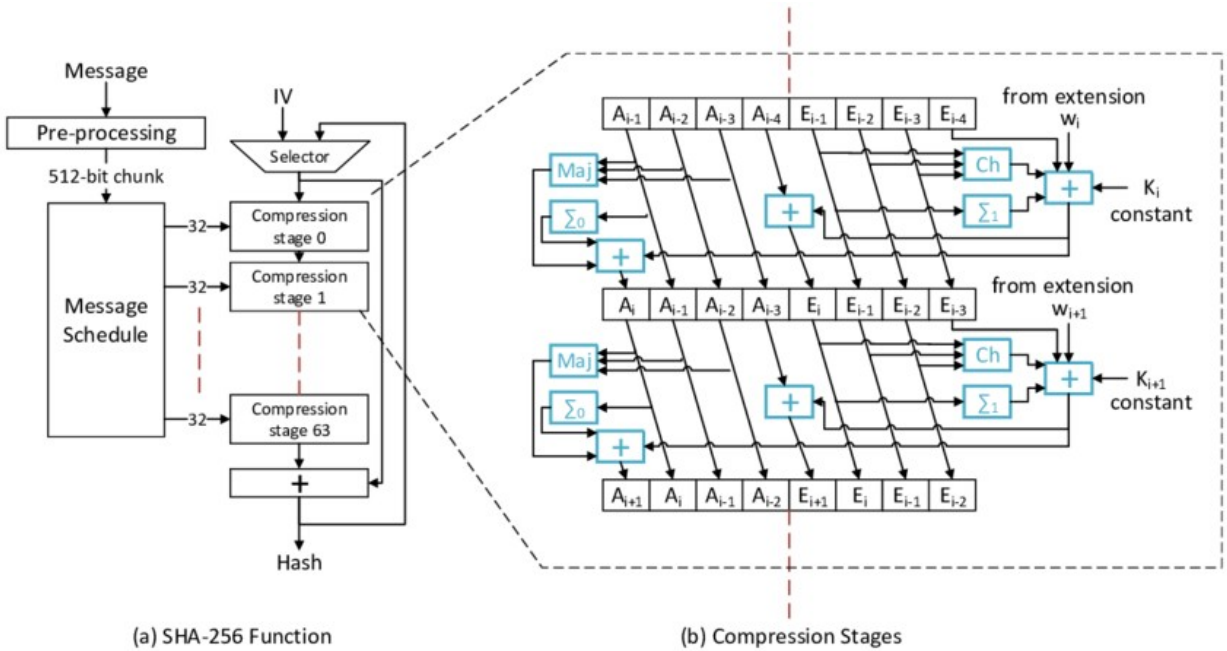
m	m in hex		h(m)
AAAA	0x41414141	→	2242
BBBB	0x42424242	→	893
CCCC	0x43434343	→	2558
DDDD	0x44444444	→	2916
EEEE	0x45454545	→	1967
FFFF	0x46464646	→	4032

Hash Function Properties

- *Easy to compute*
  - It should be easy to compute $h(m)$
- *One-way function*
  - Given $h(m)$, it should be difficult to find m
  - i.e., the reverse of $h()$ should be difficult to compute
- *Weak collision resistance*
  - Given m, it should be difficult to find $m' \neq m$ where $h(m') = h(m)$
  - I.e., given a value and a hash function, it should be difficult to find another value that produces the same hash
- *Strong collision resistance*
  - It should be difficult to find *any* two messages $m \neq m'$ where $h(m) = h(m')$
  - I.e., given a hash function, it should be difficult to find two different values that produce the same hash

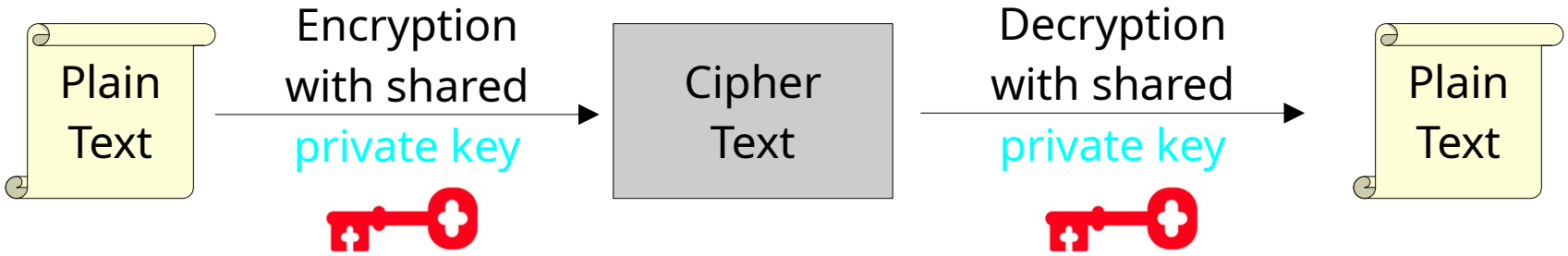
Ideal Hash

- *Ideally, we want all these properties*
  - For a strong cryptographic hash function
  - Not all hash functions provide all these properties
  - In cryptography, a weak hash often ruins the whole system!
- *Example good crypto hash function: SHA-256*



Private Key AKA Symmetric Encryption

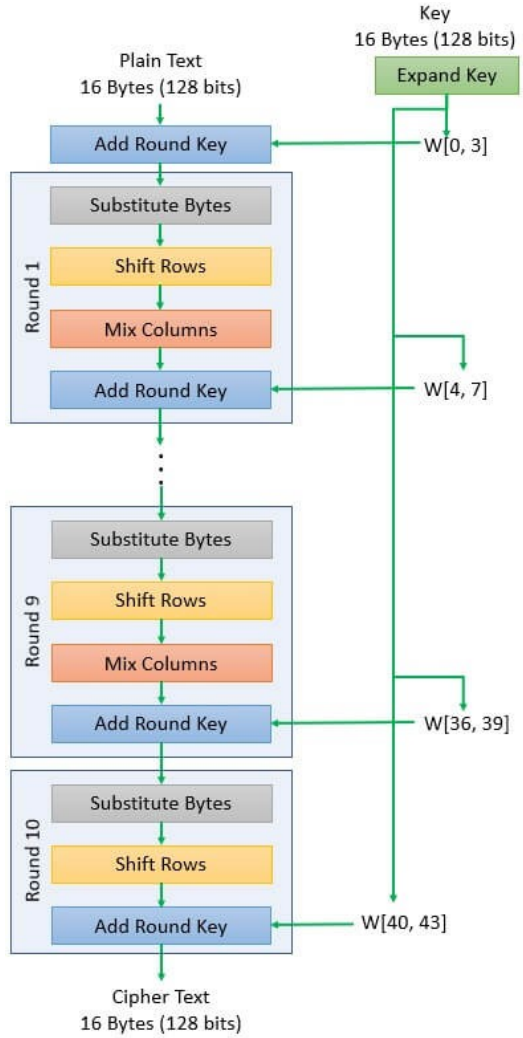
Private Key (Symmetric) Encryption



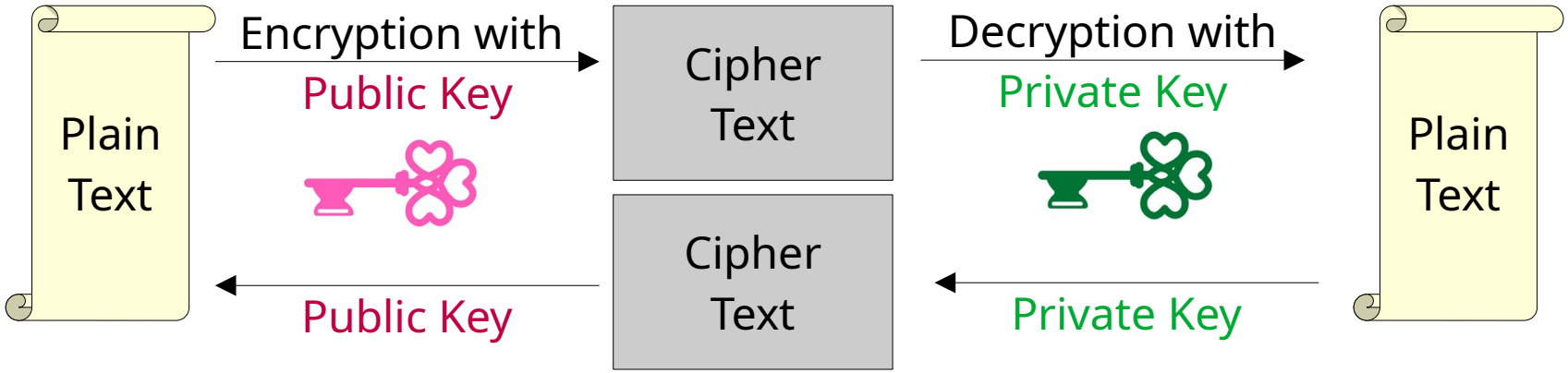
- *One key used for both encryption and decryption*
  - Requires a secure way to share the secret key!
  - This was the only type of encryption prior to invention of public-key in 1970's

Private Key Encryption: AES

- ***AES*** *is an example private key algorithm*
  - Need the same key to encrypt and decrypt



Public Key AKA Asymmetric Encryption



- *There are two keys*
  - **Public key:** can be known to anybody
    - Used to *en*crypt and *verify* signatures
  - **Private key:** should be known *only to the owner* of the key
    - Used to *de*crypt and *sign* signatures
- *Fundamental property of public key encryption*
  - When encrypted with one key, only the other key can decrypt it

Toy Asymmetric Algorithm

- *How does asymmetric encryption actually work?*
- *Key 1 and Key 2 are effectively an "inverse" of each other*
  - Applying the same algorithm using each key gets you back to the original message
- *Analogy to Caesar's Cipher*
  - Key 1: Encrypt by shifting letter to right by 7
  - Key 2: Decrypt by shifting letter to right by 19
  - Example: Message is "Hi"

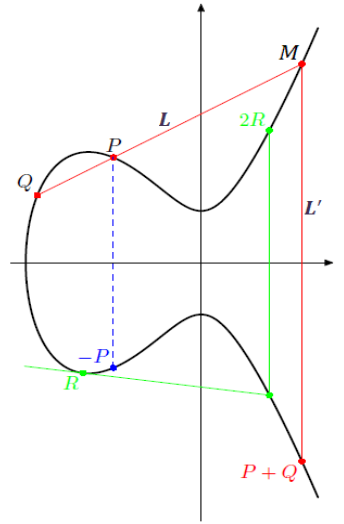
    A B C D E F G **H** I J K L M N **O** P Q R S T U V W X Y Z

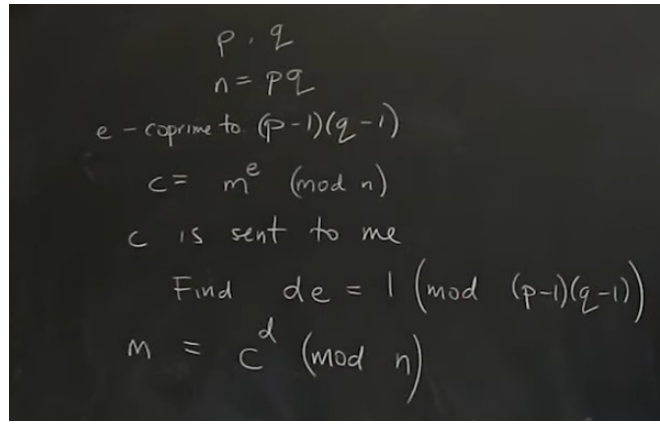
    

    - H shifted right by 7 = O
    - O shifted right by 19 = H
- We are always doing the same algorithm (toy is shifting right).

Generating Keys

- *Generating keys*
  - The public and private keys are generated together as part of a solution to some very hard to reverse computation
- *Example approaches to generating keys*
  - Factoring very large prime numbers,
  - Solving "Twisted Edwards Curves" (ed25519)
- *Description of RSA algorithm*
  - https://www.youtube.com/watch?v=qph77bTKJTM
  - Uses simple numbers to show how it works





Keeping Secrets

- *Example: Keeping Secrets*
  - Alice wants to send a secret message to Bob
  - Alice encrypts the *plain text* message using Bob's *public key*
  - Bob decrypts the *cipher text* using his *private key*
- *Analysis*
  - Since only Bob knows Bob's private key, only Bob can decrypt the cipher text
  - Hence Alice and Bob can securely share the message

Verifying Sender

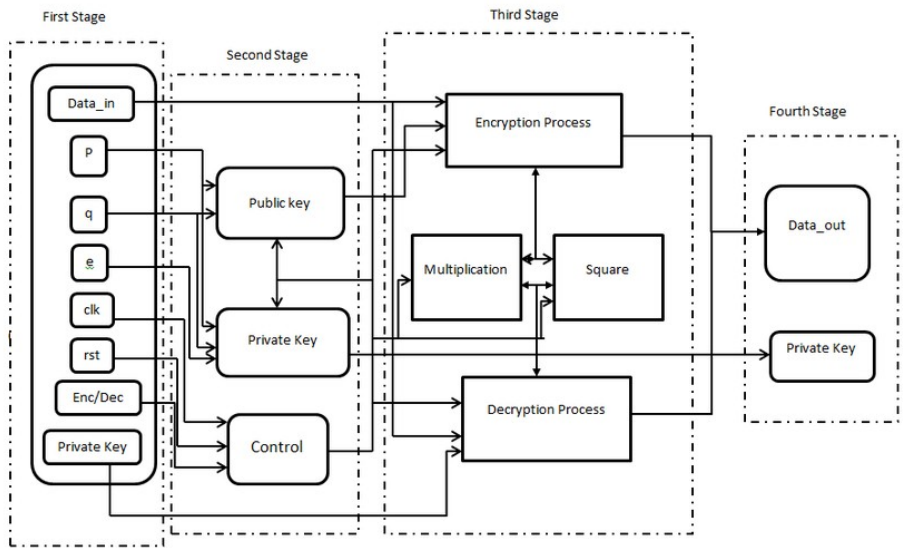
- *Example: Verifying Sender*
  - Bob wants Alice to know that he sent a messages and it has not been altered
  - Bob encrypts the plain text with his *private key*
  - Alice decrypts the cipher text using Bob's *public key*
- *Analysis*
  - Since only Bob knows Bob's private key, Bob is the only one who can encrypt a message that can be decrypted with Bob's public key
  - Alice knows it was Bob who created the message

Secret and Verified

- *Example: Secret and Verified*
  - Combine previous two examples
  - Alice wants to send a verified, secret message
  - Alice encodes a message with her private key
    - Anyone can decrypt it with her public key
    - But only she can encrypt with it; so we know she sent it!
  - Alice encodes the result with Bob's public key
  - Only Bob can decrypt it with his private key
- *Analysis*
  - Only Bob can decrypt the message (using his private key), and he'll know that only Alice can create it (using her private key)

Public Key

- Benefit
  - This does not require a secure key distribution mechanism
  - Lots of other use cases beyond encryption / decryption
- Example algorithm: RSA



Summary

- *Cryptography*
  - From plain text, create cipher text that others cannot read or change
- *Types of algorithms*
  - 0 Keys: **Hash function**
    - Turns a message into a token that is unique and hard to forge
  - 1 Key: **Symmetric encryption (private key)**
    - Both sides know the same secret key
  - 2 Keys: **Asymmetric encryption (public key)**
    - You share a public key with the world
    - *Anyone can encrypt* messages for you using this public key, but *only you can decrypt* messages using your secret private key (which matches the public key)
    - Alternatively, *only you can sign messages* using your secret private key, but *anyone can verify* messages using your public key