Hash Function and Basics of Ciphers

Deepak Puthal

Email: Deepak.Puthal@uts.edu.au

41900- Fundamentals of Security

Overview

- Hash Functions
  - Context
  - One Way Functions

    Functions

    One Way Functions
  - Hash Functions

    Properties

    Applications

    Examples
- Cipher
  - Substitution Cypher
  - Symmetric Cipher

Context: Applied Cryptography

Cryptography is the study of mathematical techniques related to the design of ciphers.

Cryptanalysis is the study of breaking them.

Cryptology (or crypto) is the study of both.

Crypto building blocks are otherwise known as cryptographic primitives. For example:

- hash functions
- block ciphers
- stream ciphers
- digital signatures
- random number generators

Functions

A function $f : X \rightarrow Y$ is defined by:

- The domain, a set $X = \{x_1, x_2, \ldots, x_n\}$.
- The codomain, a set $Y = \{y_1, y_2, \ldots, y_m\}$.
- A rule f assigning each element of X to an element of Y.

When $f : X \rightarrow Y$ is a function:

- The image of $x \in X$ is called f(x), an element of Y.
- The range of f is the set of all images, and is a subset of Y.
- If f(x) = y, then x is called a preimage of y.
- The set of all preimages of y is written $f^{-1}(\{y\})$.

Functions: Example

Example: let $f : \{-1, 0, 1\} \rightarrow \{0, 1, 2\}$ be defined by $f(x) = x^2$.

- $f(-1) = 1$, $f(0) = 0$, and $f(1) = 1$.
- The preimage of 1 is $f^{-1}(\{1\}) = \{-1, 1\}$.
- The preimage of 2 is $f^{-1}(\{2\}) = \{\}$.
- The range of f is $\{0, 1\}$.

One Way Functions

Write $\{0, 1\}^n$ for the set of all binary strings of length n. For example:

- $\{0, 1\}^1 = \{0, 1\}$.
- $\{0, 1\}^2 = \{00, 01, 10, 11\}$.

A function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ is one way (a OWF) if:

- It is "*easy*" to compute f(x) for all $x \in X$
- It is "*computationally infeasible*" to find a preimage.

Intuitively:

- Given x, it is easy to compute f(x)
- Given f(x), it is hard to compute x

One Way Function Examples

- **Example**

1. Write a message on the side of a plate: x
2. Smash the plate: f(x)
3. Finding the inverse f^{-1} (x) is difficult (but not impossible)

- **Data Encryption Standard Cipher**

1. f(x) = DES(*m, k*) = *c*
2. Given *c* and *m*, it is difficult to find *k*

Hash Functions

A hash function, *h*, is an efficiently computable mapping of arbitrarily long strings to short fixed length strings.

Minimum properties:

- Compression

  Typically any number of bits to < 512 bits

  e.g. MD5, SHA256, SHA512

- Easy of computation

  Given *h* and *x*, *h(x)* is easy to compute.

Hash Functions

Keyed Hash Functions

Some hash functions take both a key (*k*) and a message (*m*)

$$MAC_k(m) = h(m, k)$$

They are also called message authentication codes (**MAC**) or hash-based message authentication codes (**HMAC**).

Properties of Secure Hash Functions

Not all hash functions are secure. In cryptography we consider secure hash functions.

Let $h : X \rightarrow Y$ be a hash function. In order to be secure, it must satisfy the following properties:

#1. Preimage Resistance

Given y it is “hard” to find a preimage x such that $h(x) = y$.

#2. Second Preimage Resistance

Given a particular x (and hence y), it is “hard” to find $x' \neq x$ such that: $h(x') = h(x) = y$.

#3. Collision Resistance

It is “hard” to find any pair $x \neq x'$ such that $h(x) = h(x')$.

Properties of Secure Hash Functions

A one way hash function satisfies #1 and #2

A collision resistant hash functions satisfies #3 (and hence #2)

Very Useful!
Hash functions are extremely useful for confirmation of knowledge without revealing what you know. Rather than sending Alice a secret across the Internet, just send the hash of it. If Alice knows the secret, she can hash it and verify that you know it too. This is much safer and efficient than sending the secret, which can of course be intercepted (provided the hash function is strong).

Real World Hash Functions

Name	bits	h("lolcats")
MD2	128	4301aae7e3e791826b53b952859d0a14
MD4	128	52bb2839f24583f5af2fe74522db3e2e
MD5	128	c8ba0a4b74948d105bdb6f77b77a432e
RIPEMD	160	3dd2dec7cecec77219f644788e81ff26d328423c
SHA-1	160	ec9c175f8e3780cec9e93b66aea4f98b200764de
SHA224	224	c995ce647c889fdc3bbc7c8e4b43b3f5b5c3faf1525b640abc60ce54
SHA256	256	e06297effe5bcf6af177cead11f5c5d4a73777590a7a98a464287d5a6a7cdc2a
SHA384	384	14c41e98d0fb9b357922274adb9f70352f601d7b56aac8e4... ...39ed860b634b31a7c0f56e3d63284cfd4d04bde07ff3351d
SHA512	512	9a0552b9d165360fc08090a88f8c5274ca263c485417c73acb5c1a820b288549... ...12f8885bebbd49f9c229eac9be43441e061408f99e6e25dafaa5c4a946f50693

Some common hash functions and their bit sizes

Hash Function Applications

Password Files

e.g. /etc/shadow on UNIX

Instead of storing the password in cleartext, store the hash. Then just compute the hash whenever someone tries to log in and compare to what was stored.

If the password file gets stolen, the hash needs to be reversed before the attacker can use the passwords ("cracking passwords").

Virus Protection & Host Intrusion Detection

e.g. Tripwire

- For each file x, h(x) is stored off system.
- Periodically hash all files and check the hashes match
- Property #2 is critical as it should be hard to find x′ such that h(x) = h(x′) (otherwise viruses could hide)

Attacks on Hash Functions

To **brute force** in cryptanalysis is to search the entire space of possible alternatives.

A subset of this is a **dictionary attack** where we throw subsets of the keyspace (*dictionaries*) at the problem.

e.g. cracking UNIX passwords

We can use brute force to attack *pre-image resistance*:

- Say a hash produces an n-bit output: y = h(x)
- We must try 2^{n-1} hashes before Pr[h(a) = y] $\geqslant$ 0.5 (a $\in$ Z)
- Intuitively: if the secret key is one of (2^{10} = 1024) boxes, you have to open half of them (2^{9} = 512) on average before you find the secret key.

SHA-1 Break

Recently February 23, 2017!

Found by **Google** and **CWI**

https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

http://shattered.io/

Information	
Complexity:	9, 223, 372, 036, 854, 775, 808 SHA-1 compressions! Nine **QUINTILLION**
Comparisons:	Shattered: 110GPU = 1 year Brute Force: 12 million GPU = 1 year
What is affected?	Digital Certifcates, Email, Software Updates, **GIT**.

The Current State

Collision Resistance

Hash Function	Security Claim	Best Attack	Publish Date	Comment
MD5	2^{64}	2^{18} time	25-03-2013	This attack takes seconds on a regular PC. Two-block colliusions in 2^{18}, single block collisions in 2^{41}
SHA-1	2^{80}	$2^{60.3} \ldots 2^{65.3}$	19-06-2012 23-02-2017	SHAttered was the first public release of a collision. Attack is feasible with large amounts of computation.
SHA256	2^{128}	31/64 rounds ($2^{65.5}$)	28-05-2013	
SHA512	2^{256}	24/80 rounds ($2^{32.5}$)	25-11-2008	

The Current State of Affairs

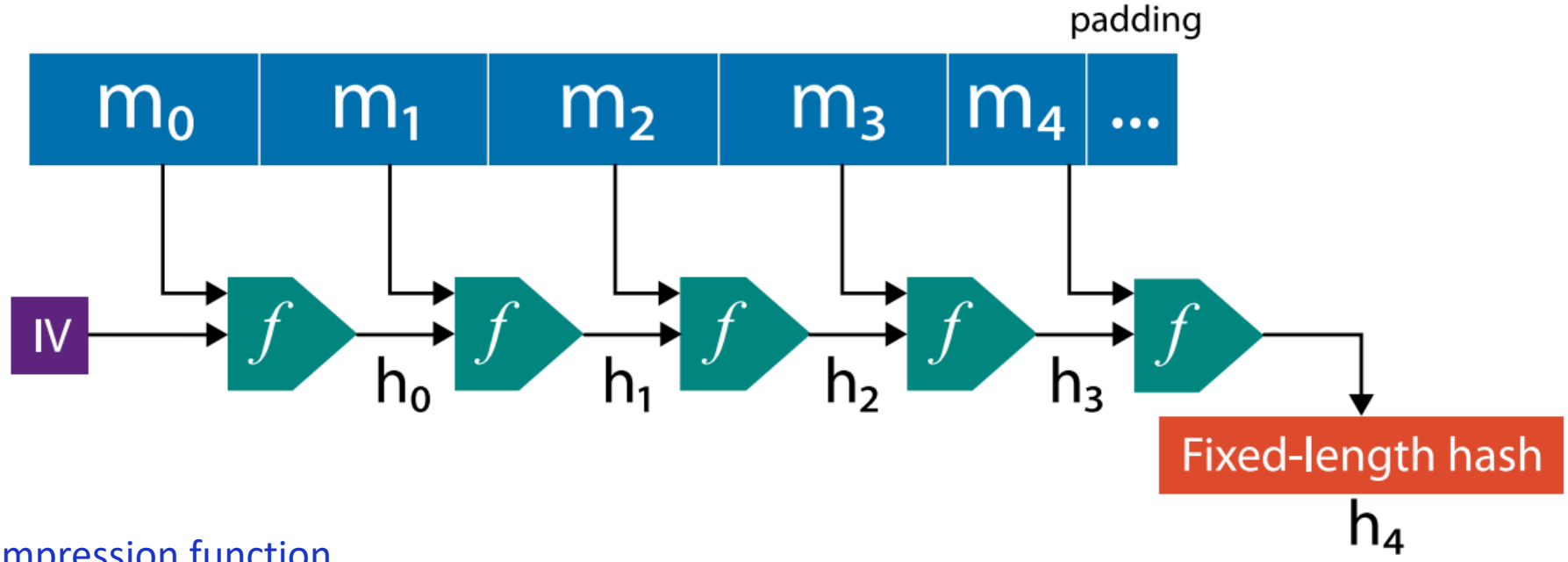
Preimage Resistance

Hash Function	Security Claim	Best Attack	Publish Date
MD5	2^{128}	$2^{123.4}$	27-04-2009
SHA-1	2^{160}	45 of 80 rounds	17-08-2008
SHA256	2^{256}	43 of 64 rounds ($2^{254.9}$ time, 2^{6} memory)	10-12-2009
SHA512	2^{512}	46 of 80 rounds ($2^{511.5}$ time, 2^{6} memory)	25-11-2008

htpp://en.wikipedia.org/wiki/Hash_function_security_summary

Iterated Hash Construction

Merkle-Damgård construction is a technique for building hash functions.



f is a one-way compression function.
Simply divide a message M into n r-bit blocks.
Length must be a multiple of a fixed number (e.g 512 bits), so the message must first pass through a padding function.

Why Use Merkle-Damgård?

Lemma

Suppose the compression function $f(m_n, h)$ is collision resistant.
Then the resulting hash function h(m) is also collision resistant.
To construct a CRHF, it is enough to construct CR compression functions:
$f : f\{0, 1\}^m \times \{0, 1\}^r \rightarrow \{0, 1\}^m$

Sponge Construction

Some recent CRHFs use sponge construction.
For example, SHA-3.

Sample Output

MD5

Input	Hash Value (as hex byte string)
“”	d41d8cd98f00b204e9800998ecf8427e
“a”	0cc175b9c0f1b6a831c399e269772661
“abc”	900150983cd24fb0d6963f7d28e17f72

SHA-1

Input	Hash Value (as hex byte string)
“”	da39a3ee5e6b4b0d3255bfef95601890afd80709
“a”	a9993e364706816aba3e25717850c26c9cd0d89d
“abc”	a9993e364706816aba3e25717850c26c9cd0d89d

Keyed Hash Functions (MACs)

Well known **Message Authentication Codes** (MACs).
A one-way hash function with the addition of a key:
$h_k : \{0, 1\}^* \rightarrow \{0, 1\}^n$
The key is secret and necessary to verify the hash $h_k(m)$ and can be thought of as a cryptographic checksum.

Goal
Provides message authentication where sender and receiver share a secret. An eavesdropper cannot fake a message with a valid MAC. Used for message integrity, *not* message **secrecy**.

Properties of MACs

Given m and k it is easy to construct $h_k(m)$.

Given pairs of messages and MAC_s $(m_i, h_k(m_i))$ it is hard to construct a valid new pair:

$(m_j, h_k(m_j))$ for $m_j \neq m_i$

Without knowledge of k.

Formally

A MAC is (ε, t, q, l).
It is secure if:
Given q pairs, each of length $\leqslant$ l, in time t an adversary can succeed in constructing new (message, MAC) pairs with probability < ε.

Using MACs - Example 1

Network Example

- Alice and Bob share a secret key **k**
- An adversary can't send a message with a valid MAC

$$MAC(m) = h_k(m)$$

Using MACs - Example 2

Say a hash function is used for virus protection and stores the signatures for each file in a database.

Couldn’t the virus also modify the database?

With a MAC, the virus can’t because it doesn’t know the key.

If it had write permissions, it could however corrupt the database or replace the verification program with a trojan/fake.

Constructing MACs

Cryptographic

- Non-Keyed hash functions (HMAC) – **fast**
- Block cyphers (CBC-MAC) – **slow**

Information Theoretic

- Based on universal hashing (outside of scope)

Hash Based MAC (HMAC)

MAC based on non-keyed hash function h

Attempt 1: MAC_k (m) = h(kjm)

Insecure: attacker can arbitrarily add to the end of the message. (Merkle-Damgård construction)

Attempt 2: MAC_k (m) = h(mjk)

Insecure: vulnerable to the birthday attack!

Attempt 3: $MAC_{k,k'}$ (m) = h(kjmjk′)

More secure: enveloped method

BEST: MACk k(m) = h((k $\oplus$ opad)jh((k $\oplus$ ipad)jm))

- *opad* is the **outer** padding

(0x5c5c5c5… one block of hex constant)

- *ipad* is the **inner** padding

(0x3636363… one block of hex constant)

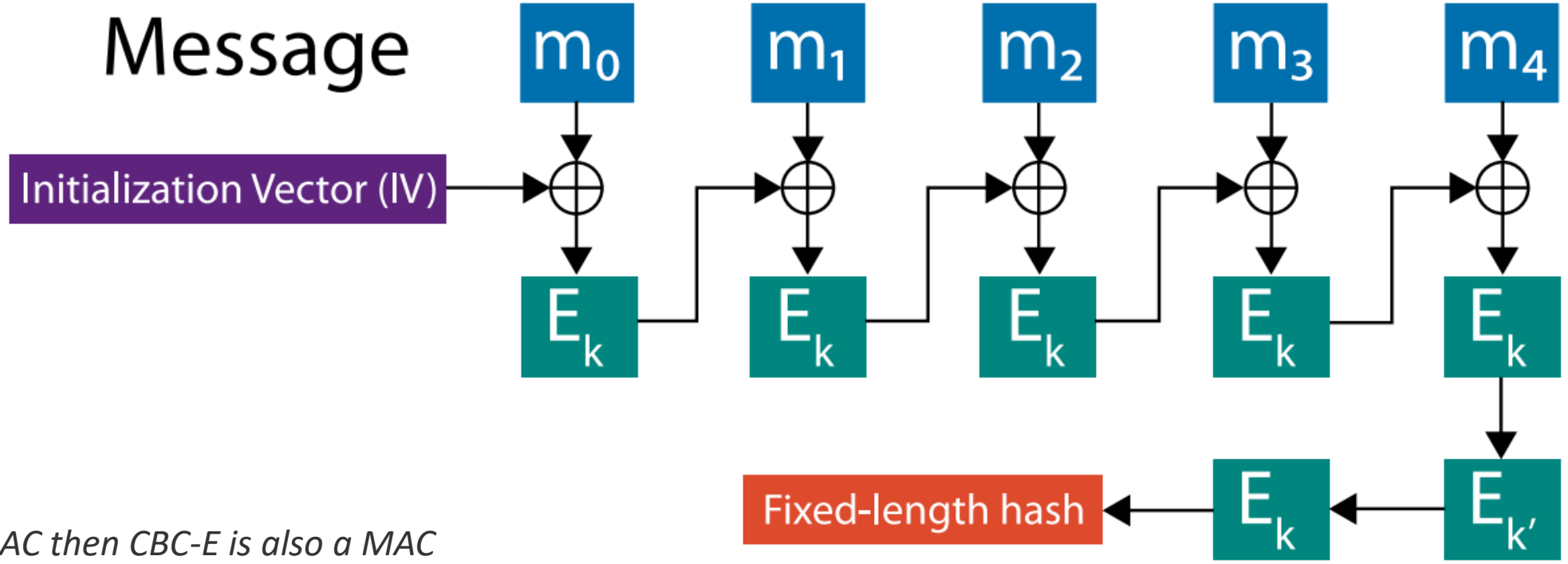
- from RFC2104

Cipher-Based MAC (CBC-MAC)

CBC-MAC uses a technique known as Cipher Block Chaining.

Turn a message into blocks and repeated encryption using a block cipher is *XORed* (⊕)

Secret key = (k, k″, IV); **IV: Initialisation Vector** (random bits)



If E is a MAC then CBC-E is also a MAC
Often used in the banking industry!

CBC-MAC Length

Typical key length

Very small, e.g. 40 bits

Security ~240 (*easily guessed*)

No Birthday Attack on MACs

Implies MACs are shorter than message digests!

CBC-MAC Length

Name	Key Size (bits)	Hash length (bits)	Relative Speed	Class	Notes
Blowfsh	Up to 448	64	23	Block Cipher	Bruce Schneier
DES	56	64	10.6	Block Cipher	Lucifer/NSA
3DES	112	64	3.7	Block Cipher	Banking
IDEA	128	64	11.8	Block Cipher	Massey and Lai
RC5 (r=12)	Up to 2048	32, 64, 128	19.6	Block Cipher	Ron Rivest (RSA)
AES (r=10, 128 bits)	128,192,256	128,192,256	21.1	Block Cipher	Rijndael
CRC32	-	32	173	Checksum	Very weak - linear
MD4	-	128	176	Hash Function	Ron Rivest (RSA)
MD5	-	128	127	Hash Function	Ron Rivest (RSA) Block collisions
SHA-1	-	160	81.5	Hash Function	NSA Hash Collisions

Keep up-to-date

There has been a steady stream of breaks against popular hashing functions like MD5 and SHA-1.

Be sure to pay attention when new hash functions are recommended and when others have been depricated.

In 2012, NIST ran a competition to choose the latest generation of hash functions, now known as the SHA-3 family.

Cipher

Substitution Ciphers

Substitution ciphers are the oldest form of cipher.

The secret key consists of a table which maps letter substitutions between plaintext and ciphertext.

The most famous is the Caesar cipher where each letter is shifted by 3 (modulo 26):

abcdefghijklmnopqrstuvwxyz

DEFGHIJKLMNOPQRSTUVWXYZABC

Similar to ROT13 which shifts plaintext 13 places – largest advantage is that encrypting twice results in the plaintext:

$$ROT13(ROT13(m)) = m$$

Substitution Ciphers

There are 26! (factorial) different possible keys (≈ 2^{88} or 88-bits).
Monoalphabetic (single character) substitution cipher:

Src = abcdefghijklmnopqrstuvwxyz
Key = XNYAHPOGZQWBTSFLRCVMUEKJDI
m = thiscourserockstheblock
C = MGZVYFUCVHCFYWVMGHNBFYW

Substitution ciphers are easy to break using frequency analysis of the letters:
Single letters

- Digraphs (pairs of letters)
- Trigraphs (three letters)

This is a ciphertext only attack.

Homophonic Ciphers (Improved Substitution Ciphers)

Homophonic ciphers are substitution ciphers that replace a common letter with multiple symbols (i.e. E can go to [C, ε, O])

Peaks or troughs in the letter frequency are hidden as they're broken down into multiple smaller spikes.

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
D	X	S	F	Z	E	H	C	V	I	T	P	G	A	Q	L	K	J	R	U	O	W	M	Y	B	N
9				7				3					5	0				4	6						
				2																					
				1																					

As a result, the high frequency of the letter "E" (the most common letter in English) is spread amongst several characters, making frequency analysis much more difficult.

Homophonic Ciphers (Improved Substitution Ciphers)

Still difficult to decipher even using modern computing:

success rate is measured in terms of alphabet size and ciphertext length.

Deciphered using nested hill climbing (heuristic algorithm / educated guessing):

- Outer layer determines the number of symbols each letter maps to
- Inner layer determines the exact mapping

Used by the Zodiac killer:

- "Zodiac 408" was solved within days of publication
- "Zodiac 340" still remains unsolved

http://www.cs.sjsu.edu/faculty/stamp/RUA/homophonic.pdf

Vigenere Cipher

Originated in Rome in the sixteenth century, a Vigenere cipher is a polyalphabetic substitution cipher (made of multiple monoalphabetic substitution ciphers). The secret key is a repeated word with encryption performed by adding the key modulo 26.

VIGENERE ENCRYPTION

```
Plaintext:   launchmissilesatlosangeles
Keystream:   cryptocryptocryptocryptocr
=====================================
Ciphertext:  nrscvvozqhbzgjyiecurlvwzgj
```

$$L + C = 11 + 2 \quad = 13 \bmod 26 = 13^{th}\text{char} \quad \Rightarrow N$$

$$N + Y = 13 + 24 \quad = 37 \bmod 26 = 11^{th}\text{char} \quad \Rightarrow L$$

Cryptography

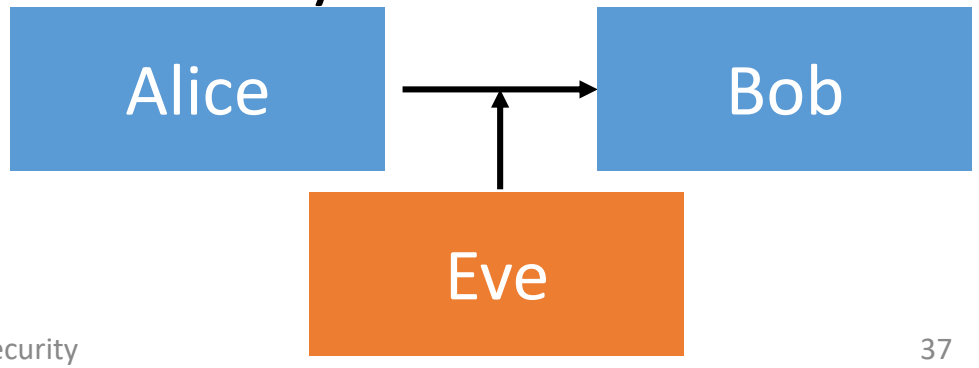
The fundamental application of cryptography is enabling **secure communications** over an **insecure channel**.

How can Alice send a secure message to Bob over an insecure channel when Eve is listening in?

Eve is an active attacker and may tap, insert or modify messages in transit.

How does one use cryptography to provide security such as:

Authentication
Integrity
Confdentiality
Non-Repudiatio

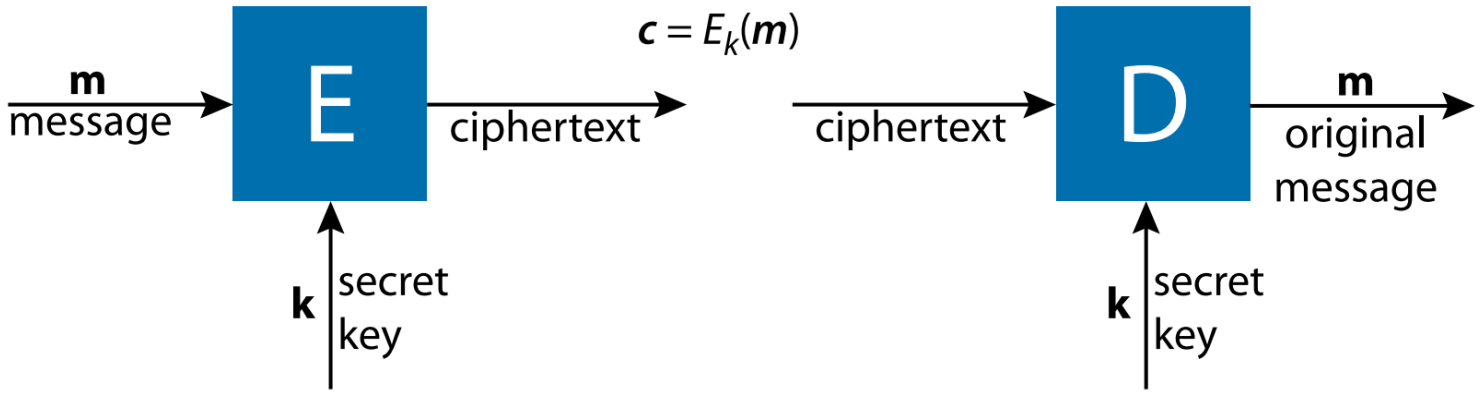


Symmetric Ciphers

The traditional way of achieving secrecy is through a **shared secret key**. This is also known as **symmetric encryption** since the key used to both encrypt and decrypt messages is the same.

A symmetric cipher is an encryption algorithm E_k and a decryption algorithm D_k which inverts E_k. In other words, for all keys k and messages m,

$$Dk(Ek(m)) = m$$



Types of Symmetric Ciphers

Stream Ciphers

Operate on a single bit or byte at a time.

Block Ciphers

Operate on blocks (numbers of bits) of plaintext at a time.