

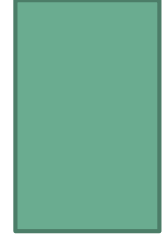
# Android Deep Dive With Chuck Easttom

Lesson 6 Cryptography for Android

# Why do you need to know this?

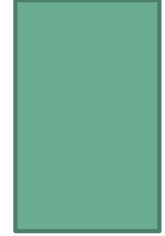
- 1 ECB or CBC mode?
- 2 What about hashes?
- 3 What key exchange algorithm?
- 4 Kerkhoff Principle

# Android Q Encryption



- ▶ In February 2019, Google unveiled Adiantum, an encryption cipher designed primarily for use on devices that do not have hardware-accelerated support for the Advanced Encryption Standard (AES), such as low-end devices.
- ▶ Adiantum is a cipher construction for disk encryption, which uses the ChaCha and Advanced Encryption Standard (AES) ciphers, and Poly1305 cryptographic message authentication code (MAC)
- ▶ ChaCha is a variant of the Salsa stream cipher.
- ▶ In 2013, Mouha and Preneel published a proof that 15 rounds of Salsa20 was 128-bit secure against differential cryptanalysis. (Specifically, it has no differential characteristic with higher probability than  $2^{-130}$ , so differential cryptanalysis would be more difficult than 128-bit key exhaustion)

# Programming Crypto

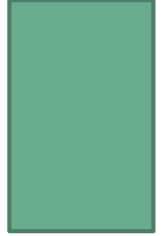


- ▶ Android has built in cryptography for programmers
- ```
byte[] plaintext = ...;
```
- ```
KeyGenerator keygen =  
KeyGenerator.getInstance("AES");
```
- ```
keygen.init(256);
```
- ```
SecretKey key = keygen.generateKey();
```
- ```
Cipher cipher =  
Cipher.getInstance("AES/CBC/PKCS5PADDING");
```
- ```
cipher.init(Cipher.ENCRYPT_MODE, key);
```
- ```
byte[] ciphertext = cipher.doFinal(plaintext);
```
- ```
byte[] iv = cipher.getIV();
```

# Java Cryptography Library

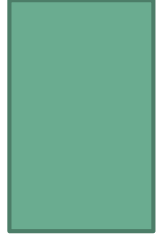
- ▶ <https://developer.android.com/reference/javax/crypto/package-summary>
- ▶ Complete Tutorial
- ▶ <https://proandroiddev.com/secure-data-in-android-encryption-in-android-part-1-e5fd150e316f>

# Android Security



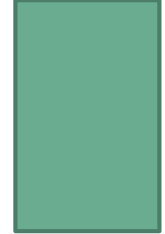
- ▶ It supports full disk encryption
- ▶ <https://source.android.com/security/encryption/full-disk>
- ▶ <https://www.howtogeek.com/141953/how-to-encrypt-your-android-phone-and-why-you-might-want-to/>

# How android encrypts



- ▶ Android does not offer anything that can approach iOS in terms of encryption. Instead, Android employs a straightforward block-level encryption of the data partition
- ▶ Android 7 introduced a new file-based encryption scheme.
- ▶ <https://source.android.com/security/encryption/full-disk>

# Android Cryptography



Class	Recommendation
Cipher	AES in either CBC or GCM mode with 256-bit keys (such as <code>AES/GCM/NoPadding</code> )
MessageDigest	SHA-2 family (eg, <code>SHA-256</code> )
Mac	SHA-2 family HMAC (eg, <code>HMACSHA256</code> )
Signature	SHA-2 family with ECDSA (eg, <code>SHA256withECDSA</code> )

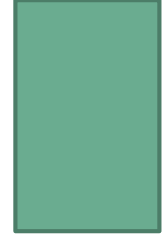


# Android Crypto Java Code

```
byte[] plaintext = ...;
KeyGenerator keygen = KeyGenerator.getInstance("AES");
keygen.init(256);
SecretKey key = keygen.generateKey();
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
cipher.init(Cipher.ENCRYPT_MODE, key);
byte[] ciphertext = cipher.doFinal(plaintext);
byte[] iv = cipher.getIV();
```

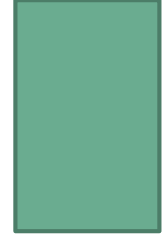
```
byte[] message = ...;
MessageDigest md = MessageDigest.getInstance("SHA-256");
byte[] digest = md.digest(message);
```

# Supported Algorithms



Algorithm	Supported (API Levels)	Notes
AES	23+	Supported sizes: 128, 192, 256
HmacSHA1	23+	<ul style="list-style-type: none"><li>Supported sizes: 8–1024 (inclusive), must be multiple of 8</li><li>Default size: 160</li></ul>
HmacSHA224	23+	<ul style="list-style-type: none"><li>Supported sizes: 8–1024 (inclusive), must be multiple of 8</li><li>Default size: 224</li></ul>
HmacSHA256	23+	<ul style="list-style-type: none"><li>Supported sizes: 8–1024 (inclusive), must be multiple of 8</li><li>Default size: 256</li></ul>
HmacSHA384	23+	<ul style="list-style-type: none"><li>Supported sizes: 8–1024 (inclusive), must be multiple of 8</li><li>Default size: 384</li></ul>
HmacSHA512	23+	<ul style="list-style-type: none"><li>Supported sizes: 8–1024 (inclusive), must be multiple of 8</li><li>Default size: 512</li></ul>

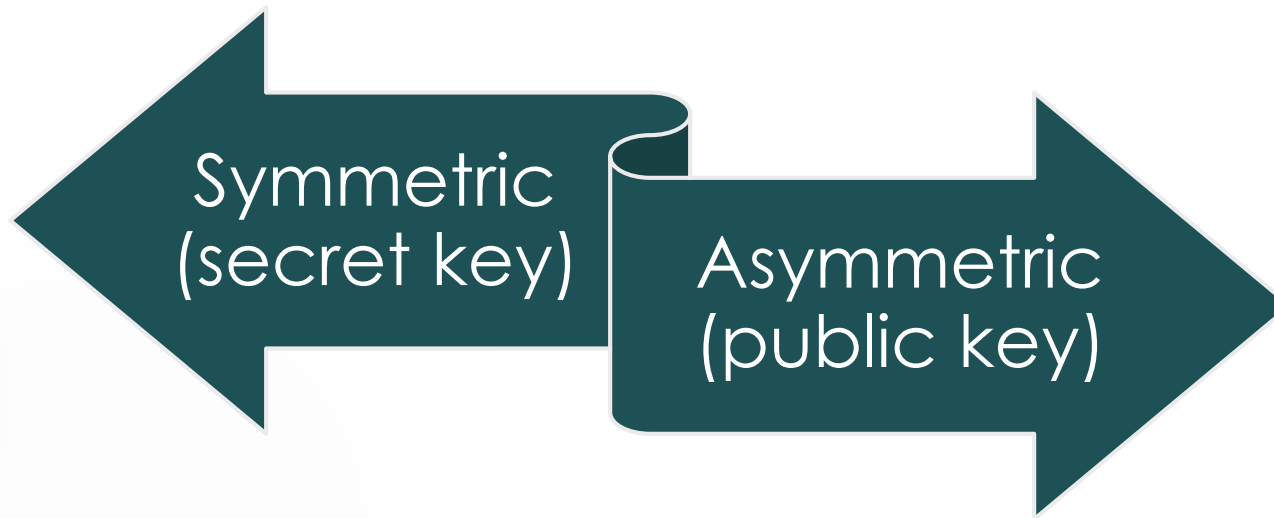
# Supported Algorithms



Algorithm	Supported (API Levels)	Notes
DSA	19–22	
EC	23+	<ul style="list-style-type: none"><li>Supported sizes: 224, 256, 384, 521</li><li>Supported named curves: P-224 (secp224r1), P-256 (aka secp256r1 and prime256v1), P-384 (aka secp384r1), P-521 (aka secp521r1)</li></ul> <p>Prior to API Level 23, EC keys can be generated using <code>KeyPairGenerator</code> of algorithm "RSA" initialized <code>KeyPairGeneratorSpec</code> whose key type is set to "EC" using <code>setKeyType(String)</code>. EC curve name cannot be specified using this method – a NIST P-curve is automatically chosen based on the requested key size.</p>
RSA	18+	<ul style="list-style-type: none"><li>Supported sizes: 512, 768, 1024, 2048, 3072, 4096</li><li>Supported public exponents: 3, 65537</li><li>Default public exponent: 65537</li></ul>

# Intro to Encryption

12



# History of Encryption

13

- ▶ Caesar Cipher
- ▶ ROT 13
- ▶ Multi alphabet encryption
- ▶ Scytale
- ▶ Atbash
- ▶ Enigma machine – used by Nazi Germany

# Caesar Cipher Example

14

- ▶ Shifting One to the Left



File Edit Format View Help

**I Like Computers**  
**H Khjd Bnlotsdqr**

# Atbash Cipher

15

Hebrew scribes copying the book of Jeremiah used this cipher. It is very simple, just reverse the alphabet. This is, by modern standards, a very primitive and easy to break cipher. But it will help you get a feel for how cryptography works.

The ATBASH cipher is a Hebrew code which substitutes the first letter of the alphabet for the last and the second letter for the second to the last, etc. It simply reverses the alphabet.

A becomes Z, B becomes Y, C becomes X, etc.

K	i	F	CH	Z	V	HE	D	G	B	A
Kaf	yod	Tet	het	zain	waw	he	dalut	gimel	bet	alef
כ	י	ט	ח	ז	ו	ה	ד	ג	ב	א
ל	מ	נ	ס	ע	פ	צ	ק	ר	ש	ת
lamed	mem	nun	samech	ayin	pe	tsade	qof	res	shin	taw
L	M	N	S	H	P	Z	Q	R	S	T

# Atbash Cipher Example

16

A cat sleeps

Becomes

Z xzg hovvkh

This is obviously a rather simple cipher and not used in modern times. However it illustrates the basic concept of cryptography. That is to perform some permutation on the plain text to render it difficult to read by those who don't have the key to 'un scramble' the cipher text.



# ROT 13

17

This is another single alphabet substitution cipher. All characters are rotated 13 characters through the alphabet.

The phrase

A CAT

Becomes

N PNG

# Scytale

18

- ▶ A cylinder tool used by Spartans for substitution encryption.



# Rail Fence cipher

- ▶ write message letters out diagonally over a number of rows
- ▶ then read off cipher row by row
- ▶ eg. write message out as:

L	N	A	D	W	
	A	D	T	A	N

- ▶ giving cipher text

LNADWADTAN

# Vigenère Cipher

20

Perhaps the most widely known poly-alphabet cipher is the Vigenère cipher. This cipher was invented in 1553 by Giovan Battista Bellaso. It is a method of encrypting alphabetic text by using a series of different mono-alphabet ciphers selected based on the letters of a keyword. This algorithm was later misattributed to Blaise de Vigenère, and so it is now known as the "Vigenère cipher", even though Vigenère did not really invent it.

# Vigenère Cipher Example

Match the letter of your keyword on the top with the letter of your Plain text on the left to find the cipher text.

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

# Vigenère Cipher Example

Using the previous chart if you are encrypting the word 'cat' and your key word is 'horse' then: the cipher text is *jok*

# The Enigma Machine

23

It is an rotor based cipher system used in WWI and WWII. There were multiple variations on this machine. The machine was designed so that when the operator pressed a key the encrypted cipher text for that plain text was altered each time. So if the operator pressed the A key, he or she might generate an F in the cipher text, and the next time it might be a D. Essentially this was a multi-alphabet cipher consisting of 26 possible alphabets.



# Binary Numbers

File Edit Format View Help	
<b>First Number</b>	<b>1101</b>
<b>Second Number</b>	<b>0110</b>
<b>Resulting Number (OR)</b>	<b>1111</b>

## ► Binary AND

File Edit Format View Help	
<b>First Number</b>	<b>1101</b>
<b>Second Number</b>	<b>0110</b>
<b>Resulting Number (AND)</b>	<b>0100</b>



# Binary XOR

- ▶ Like Caesar Cipher, this is given only for historical/educational purposes. This is not true encryption

## Binary XOR

File Edit Format View Help

<b>First Number</b>	<b>1101</b>
<b>Second Number</b>	<b>0110</b>
<b>Resulting Number (XOR)</b>	<b>1011</b>

## Binary Decrypt

File Edit Format View Help

<b>First Number</b>	<b>1101</b>
<b>Second Number</b>	<b>0110</b>
<b>Resulting Number (XOR)</b>	<b>1011</b>

<b>now take the result</b>	<b>1011</b>
<b>Re-XOR it with the second number</b>	<b>0110</b>
<b>And result is the original number</b>	<b>1101</b>

# Block vs Stream Cipher

- ▶ Block ciphers encrypt plaintext in blocks or chunks of many bytes at a time.
- ▶ Stream Ciphers encrypt each byte (or sometimes each bit) at a time.

# Symmetric Block Cipher Algorithms

Here are some of the most widely known.

- The Feistel Network
- DES
- 3DES
- AES
- Blowfish
- Twofish
- Skipjack
- IDEA
- Serpent

# DES & AES

## DES

- ▶ The Data Encryption Standard
- ▶ Now considered out dated
- ▶ 3 DES
- ▶ 56-bit key
- ▶ Symmetric block cipher

## AES

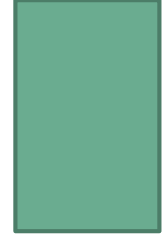
- ▶ Advanced Encryption Standard
- ▶ Rijndael block cipher
- ▶ key size of 128, 192, or 256 bits
- ▶ Block size of 128 bits

# Blowfish

## Blowfish

- ▶ Symmetric Block Cipher
- ▶ designed in 1993 by Bruce Schneier
- ▶ intended as a replacement for the aging DES
- ▶ 64 bit block size
- ▶ Key length varies from 32 up to 448 bits

# AES



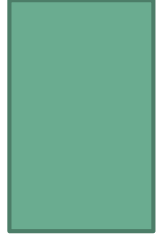
AES can have three different key sizes: 128, 192, or 256 bits.

The three different implementations of AES are referred to as AES 128, AES 192, and AES 256.

The block size is always 128 bits.

The original Rijndael cipher allowed for variable block and key sizes in 32-bit increments. However, the U.S. government uses these three key sizes with a 128-bit block as the standard for AES.

# AES



**Key Expansion:** Round keys are derived from the cipher key using Rijndael's key schedule

**Initial Round:** AddRoundKey—each byte of the state is combined with the round key using a bitwise XOR

**Rounds:**

1. SubBytes—a nonlinear substitution step in which each byte is replaced with another according to a lookup table.
2. ShiftRows—a transposition step in which each row of the state is shifted cyclically a certain number of steps.
3. MixColumns—a mixing operation that operates on the columns of the state, combining the four bytes in each column.
4. AddRoundKey

**Final Round (no MixColumns):**

1. SubBytes
2. ShiftRows
3. AddRoundKey

# Serpent

32

- ▶ Serpent has a block size of 128 bits and can have a key size of 128, 192 or 256 bits, much like AES. The algorithm is also a substitution-permutation network like AES. It uses 32 rounds working with a block of four 32-bit words. Each round applies one of eight 4-bit to 4-bit S-boxes 32 times in parallel. Serpent was designed so that all operations can be executed in parallel.

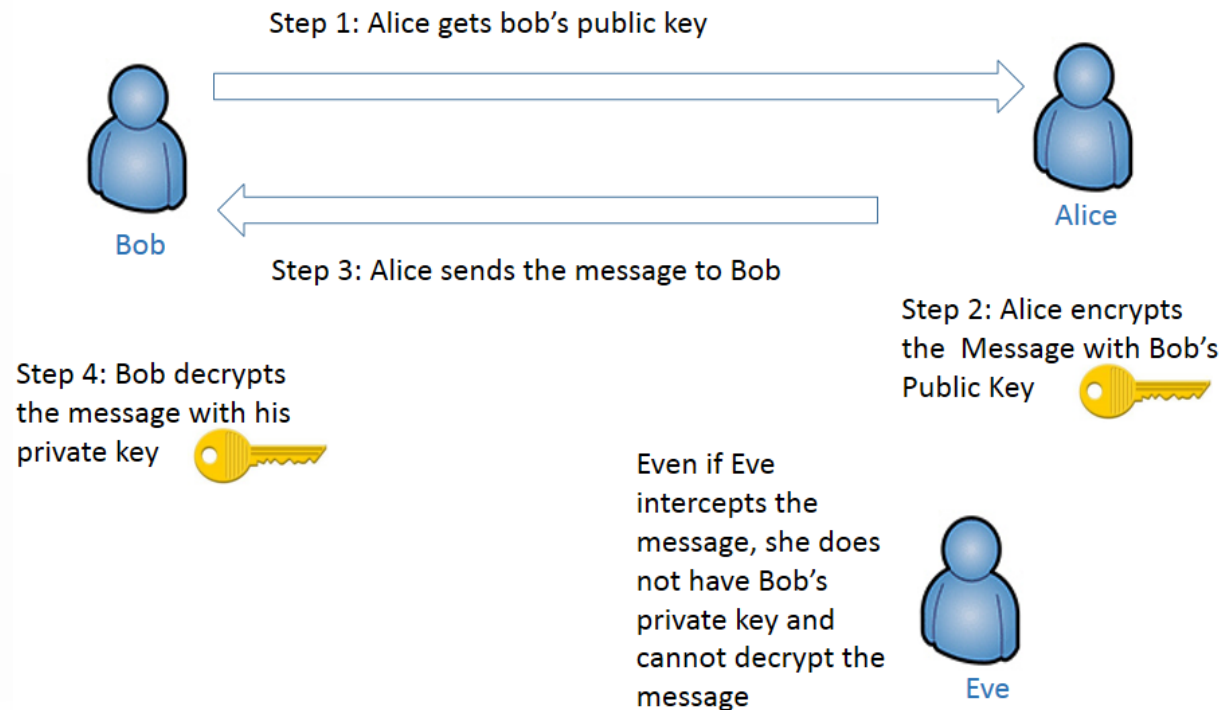


# Asymmetric Encryption

33

- ▶ Asymmetric systems and schemes use key pairs which consist of a public key and private key. The former is made public (for example, by publishing it in a directory) and the latter is kept secret. So the asymmetric cryptography does not involve shared secrets.
- ▶ Advantages: Provides a secure way to communicate; provides method of validation; non-repudiation Disadvantages: Slower than Symmetric
- ▶ RSA is the first full-fledged and most widely used public-key cryptographic algorithm designed by R. Rivest, A. Shamir, and L. Adleman. Its security is based on the difficulty of factoring large numbers.
- ▶ Diffie-Hellman key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets. The Diffie-Hellman key exchange is vulnerable to a man in the middle attack.

# How public/private key encryption works



# RSA & Diffie-Helman

## RSA

- ▶ Public key encryption
- ▶ The algorithm was publicly described in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT; the letters **RSA** are the initials of their surnames.

## Diffie-Helman

- ▶ a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel.

# CONCEPTS for RSA

36

Except for basic operations like multiplication, you really only need four concepts from number theory/discrete math to understand RSA. Those concepts are:

- ▶ Prime
- ▶ Co-Prime
- ▶ Euler's Totient (or Euler's Phi Function)
- ▶ Modulus operation

# How Does RSA Work?

## Key generation

Generate two large random primes,  $p$  and  $q$ , of approximately equal size such that their product  $n = pq$  is of the required bit length (such as 2048 bits, 4096 bits, and so on):

Let  $n = pq$

Let  $m = (p - 1)(q - 1)$

Choose a small number  $e$ , co-prime to  $m$ . (Note: Two numbers are co-prime if they have no common factors.)

Find  $d$ , such that  $de \bmod m \equiv 1$

Publish  $e$  and  $n$  as the public key.

Keep  $d$  as the secret key.

# RSA Continued

- ▶ Encrypt

- ▶ Ciphertext = Plaintext<sup>e</sup> mod n
- ▶ Put another way
  - ▶ Computes the cipher text  $c = m^e \bmod n$

- ▶ Decrypt

- ▶ Plaintext = Ciphertext<sup>d</sup> mod n
- ▶ Put another way
  - ▶ Uses his private key (d,n) to compute  $m = c^d \bmod n$ .

# RSA Example

1. Select primes:  $p=17$  &  $q=11$
2. Compute  $n = pq = 17 \times 11 = 187$
3. Compute  $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select  $e=7$
1. Find  $d$ , such that  $de \bmod m = 1$ ,  $d = 23$
5. since  $23 \times 7 = 161 \bmod M(160) = 1$
6. Publish public key  $7, 187$
7. Keep secret private key  $23, 187$

# RSA Continued

40

- ▶ Now use the number 3 as the plain text. Remember  $e = 7$ ,  $d = 23$ , and  $n = 187$ 
  - ▶  $\text{Ciphertext} = \text{Plaintext}^e \bmod n$  or
  - ▶  $\text{Ciphertext} = 3^7 \bmod 187$
  - ▶  $\text{Ciphertext} = 2187 \bmod 187$
  - ▶  $\text{Ciphertext} = 130$
- ▶ Decrypt
  - ▶  $\text{Plaintext} = \text{Ciphertext}^d \bmod n$
  - ▶  $\text{Plaintext} = 130^{23} \bmod 187$
  - ▶  $\text{Plaintext} = 4.1753905413413116367045797\text{e}+48 \bmod 187$
  - ▶  $\text{Plaintext} = 3$



# RSA why the public key won't decrypt

41

- ▶ Remember from the last slide the ciphertext is 130. Remember  $e = 7$ ,  $d = 23$ , and  $n = 187$ . What if you tried the public key  $(e, n)$  again, instead of the private key  $(d, n)$ ?
- ▶ Decrypt
  - ▶  $\text{Plaintext} = \text{Ciphertext}^e \bmod n$
  - ▶  $\text{Plaintext} = 130^7 \bmod 187$
  - ▶  $\text{Plaintext} = 627485170000000 \bmod 187$
  - ▶  $\text{Plaintext} = 37$

# RSA why the public key won't decrypt

42

- ▶ Remember from the last slide the ciphertext is 130. Remember  $e=7$ ,  $d=23$ , and  $n=187$ . What if you tried the public key  $(e,n)$  again, instead of the private key  $(d,n)$ ?
- ▶ Decrypt
  - ▶  $\text{Plaintext} = \text{Ciphertext}^e \bmod n$
  - ▶  $\text{Plaintext} = 130^7 \bmod 187$
  - ▶  $\text{Plaintext} = 627485170000000 \bmod 187$
  - ▶  $\text{Plaintext} = 37$

# RSA Continued

43

- ▶ Now take 3 as the plain text. Remember  $e = 7$ ,  $d = 23$ , and  $n = 187$ . But what if even one of these numbers is wrong? What if we did not pick an  $e$  that is coprime to  $m$  (160)? So let's pick  $e = 6$  which is not coprime to 160
  - ▶  $\text{Ciphertext} = \text{Plaintext}^e \bmod n$  or
  - ▶  $\text{Ciphertext} = 3^6 \bmod 187$
  - ▶  $\text{Ciphertext} = 729 \bmod 187$
  - ▶  $\text{Ciphertext} = 168$
- ▶ Decrypt
  - ▶  $\text{Plaintext} = \text{Ciphertext}^d \bmod n$
  - ▶  $\text{Plaintext} = 168^{23} \bmod 187$
  - ▶  $\text{Plaintext} = 1.5209448956267486762854590239666e+51 \bmod 187$
  - ▶  $\text{Plaintext} = 93$
- ▶ See it does not work! Only by using numbers that have the coprime relationships can this work

# RSA Continued

44

- ▶ Also note RSA cannot encrypt any plaintext larger than the modulus. If you do, then the public key will decrypt as well as encrypt!

# Video

- ▶ RSA Explained

[https://www.youtube.com/watch?v=wXB-V\\_Keiu8&t=834s](https://www.youtube.com/watch?v=wXB-V_Keiu8&t=834s)



# Diffie-Hellman

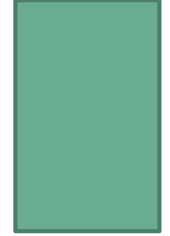
Diffie-Hellman is a cryptographic protocol that allows two parties that have no prior knowledge of each other to establish a shared secret key jointly over an insecure communication channel.



# Video

- ▶ Diffie Hellman Explained

[https://www.youtube.com/watch?v=YEBfamv-\\_do](https://www.youtube.com/watch?v=YEBfamv-_do)



# Elliptic Curve

This algorithm was first described in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) .

The security of Elliptic Curve cryptography is based on the fact that finding the discrete logarithm of a random elliptic curve element with respect to a publicly-known base point is difficult to the point of being impractical to do.

The size of the elliptic curve determines the difficulty of the finding the algorithm, and thus the security of the implementation. The level of security afforded by an RSA-based system with a large modulus can be achieved with a much smaller elliptic curve group.



# ElGamal

49

- ▶ It is based on the Diffie–Hellman key exchange. It was first described by Taher Elgamal in 1984.
- ▶ It has three components: the key generator, the encryption algorithm, and the decryption algorithm.

# DSA

50

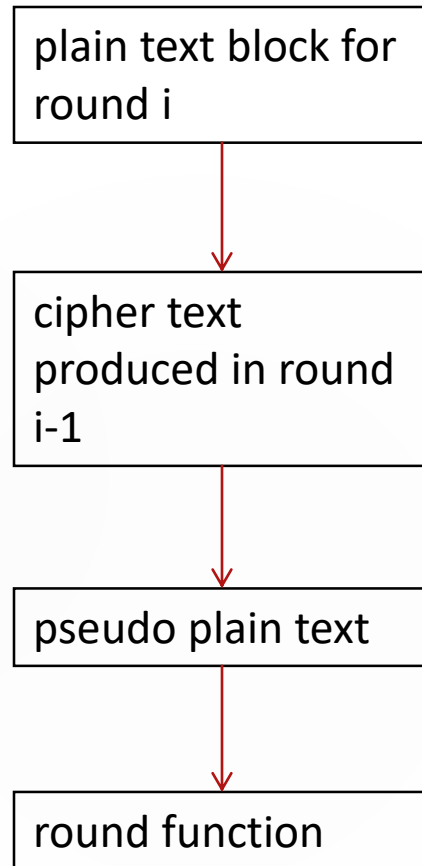
- ▶ Digital Signature Algorithm
- ▶ U.S. Patent 5,231,668, filed July 26, 1991, and attributed to David W. Kravitz. It was adopted by the US government in 1993 with FIPS 186.

# Electronic codebook (ECB)

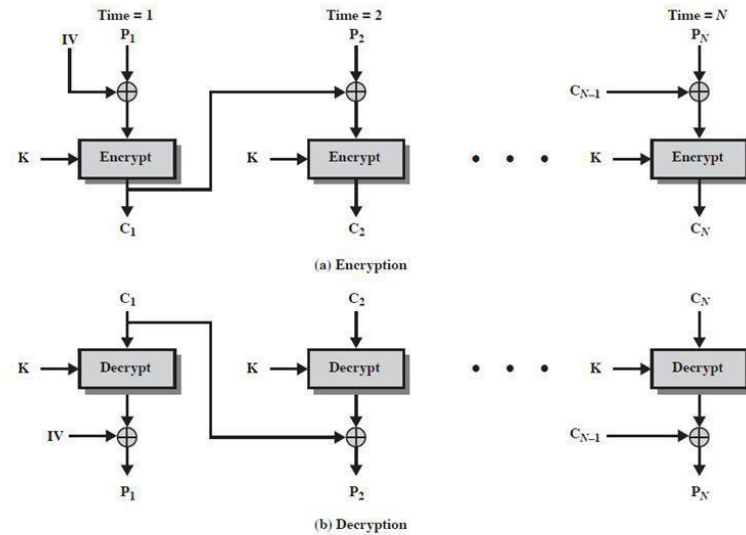
The most basic encryption mode is the electronic codebook (ECB) mode. The message is divided into blocks and each block is encrypted separately. The problem is that if you submit the same plain text more than once, you always get the same cipher text. This gives attackers a place to begin analyzing the cipher to attempt to derive the key.

# CBC

52



## Cipher Block Chaining (CBC)



10

## Initialization vector (IV)

An IV is a fixed-size input to a cryptographic primitive that is random or pseudorandom. Some cryptographic methods require the IV only to be non-repeating, not truly random. In this case, the IV is commonly called a nonce (*number used once*), and the methods are described as *stateful* as opposed to *randomized*.

In a block ciphers using Electronic Code Book (ECB) mode, encryption of the same plain text with the same key results in the same cipher text. Use of an initialization vector that is xor'd with the first block of plaintext or included in front of the plaintext prior to encryption solves this problem.

# Hardware Encryption

- ▶ Full disk encryption (FDE) or whole disk encryption often signify that everything on disk is encrypted
- ▶ Disk encryption does not replace file encryption
- ▶ Trusted Platform Module (TPM) is a secure cryptoprocessor embedded in the motherboard that can be used to authenticate a hardware device
- ▶ TPM can be used to tie the hard disk drive (HDD) encryption to a particular device. If the HDD is removed from that particular device and placed in another, the decryption process will fail
- ▶ *Hardware Security Modules (HSMs)* are devices that handle digital keys. They can be used to facilitate encryption as well as authentication via digital signatures. Most HSMs support tamper resistant mechanisms to prevent the tampering of the keys.

# File and Drive encryption

55

## ▶ File Encryption

- ▶ Prevents alteration of data
- ▶ Prevents file from being replaced
- ▶ Usually uses public key
- ▶ Windows uses EFS
  - ▶ Enhances NTFS permissions security
  - ▶ Might need cipher.exe to decrypt files

## ▶ Drive Encryption

- ▶ Microsoft Bit Locker
- ▶ Key stored on separate device
- ▶ Requires TPM or USB flash drive

# Digital Signatures/Certificates

- ▶ Digital Signature is usually the encryption of a message or message digest with the sender's private key. To verify the digital signature, the recipient uses the sender's public key. Good digital signature scheme provides:
  - ▶ authentication
  - ▶ integrity
  - ▶ non-repudiation
- ▶ RSA algorithm can be used to produce and verify digital signatures; another public-key signature algorithm is DSA.



# Digital Signature Basics

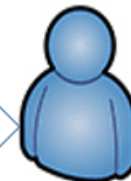
57

Step 1: Bob signs the message with his private key



Bob

Step 2: Bob Sends the message with signature



Alice

Step 3: Alice verifies the signature using Bob's public key.

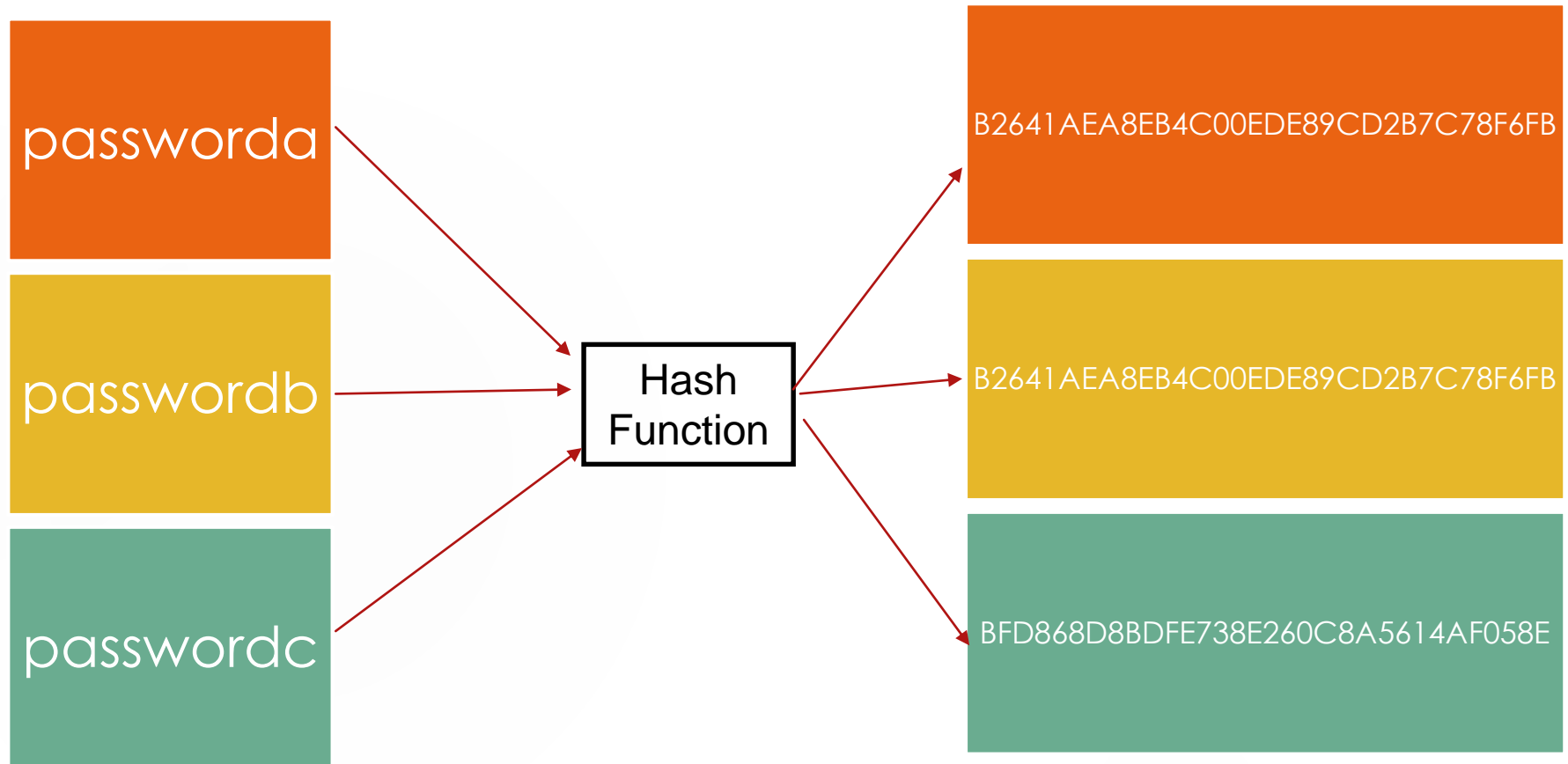


# Hashes

- Hash: Cryptographic hash is a one-way function that takes an input of a variable size and produces a fixed-size output which is commonly referenced to as "hash" or "digest". It is "one-way", which means that when given: ,an input, it is easy to compute its hash; a hash, it is hard to compute the corresponding input; a block of data as an input, it is hard to find another block of data with the same hash Another important requirement to hash functions in cryptography is the collision-resistance: it is hard to find two random inputs with the same hash.
- MD5 Produces 128 bit digest
- SHA1 produces 160 bit digest

# What is a collision

Two different inputs produce the same output



# Hashes

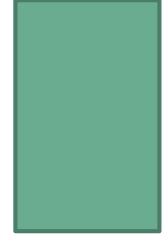
- ▶ SHA-2: This is actually two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. They differ in the word size; SHA-256 uses 32-byte (256 bits) words where SHA-512 uses 64-byte (512 bits) words. There are also truncated versions of each standardized, known as SHA-224 and SHA-384. These were also designed by the NSA
- ▶ SHA-3: On October 2, 2012, the Keccak algorithm was selected as the winner of the NIST hash function competition. SHA-3 uses the sponge construction [5][6] in which message blocks are XORed into the initial bits of the state. The state consists of a  $5 \times 5$  array of 64-bit words/1600 bits.

# Hashes

61

- ▶ **MD5**
- ▶ **SHA 1**
- ▶ **SHA 2**
- ▶ **SHA 3**
- ▶ **NTLM**
- ▶ **NTLMv2**
- ▶ RipeMD
- ▶ Haval
- ▶ Tiger

# Encryption/Hash/Signature



Bob



Alice



Send message

Encrypt the entire package

I would like to order 10  
widgets

Hash: B24&456s\$a0982\$X

Signature: 54@aso&&

1. decrypt the entire package
2. Verify the signature
3. Hash the message received
4. Compare that to the hash received

# Message Authentication Code (MAC)

A MAC function is used to add a secret key to protect integrity

► Two types of MACs

1. Hash Message Authentication Code (HMAC)
2. Cipher Block Chaining Message Authentication Code (CBC-MAC)

# Digital certificates

- ▶ PKI (public key infrastructure) uses asymmetric key pairs and combines software, encryption and services to provide a means of protecting security of business communication and transactions.
- ▶ PKCS ( Public Key Cryptography Standards ) Put in place by RSA to ensure uniform Certificate management throughout the internet.
- ▶ A Certificate is a digital representation of information that identifies you as a relevant entity by a trusted third party ( TTP )
- ▶ A CA ( Certification Authority ) is an entity trusted by one or more users to manage certificates.
- ▶ RA ( Registration Authority ) Used to take the burden off of a CA by handling verification prior to certificates being issued. RA acts as a proxy between user and CA. RA receives request, authenticates it and forwards it to the CA.
- ▶ CPA ( Certificate Practice Statement ) describes how the CA plans to manage the certificates it issues.
- ▶ CP ( Certificate Policy ) is a set of rules that defines how a certificate may be used.



# Digital certificates

## Continued

- ▶ X.509 This is an international standard for the format and information contained in a digital certificate.
- ▶ CRL ( Certificate Revocation List ) is a list of certificates issued by a CA that are no longer valid. CRLs are distributed in two main ways: PUSH model: CA automatically sends the CRL out a regular intervals. Pull model: The CRL is downloaded from the CA by those who want to see it to verify a certificate. End user is responsible.
- ▶ Status Checking: The newer method is the “Online Certificate Status Checking Protocol” [OCSP].

# X.509 certificate content

- ▶ Version
- ▶ Certificate holder's public key info
  - ▶ Public Key Algorithm
  - ▶ Certificate holder's Public Key
- ▶ Serial number
- ▶ Certificate holder's distinguished name
- ▶ Certificate's validity period
- ▶ Unique name of certificate issuer
- ▶ Digital signature of issuer
- ▶ Signature algorithm identifier

RFC 5280



# Rainbow Table

- ▶ In 1980 Martin Hellman described a cryptanalytic technique which reduces the time of cryptanalysis by using precalculated data stored in memory. This technique was improved by Rivest before 1982. Basically these types of password crackers are working with pre-calculated hashes of all passwords available within a certain character space, be that a-z or a-zA-z or a-zA-Z0-9 etc, This is called a Rainbow table. If you search a Rainbow table for a given hash, whatever plaintext you find, must be the text that was input into the hashing algorithm to produce that specific hash.

# Rainbow Table

- ▶ Clearly such a Rainbow table would get very large very fast. Assume that the passwords must be limited to keyboard characters. That leaves 52 letters (26 upper case and 26 lower case), 10 digits, and roughly 10 symbols, or about 72 characters. As you can imagine even a 6 character password has a very large number of possible combinations. This means there is a limit to how large a Rainbow table can be, and this is why longer passwords are more secure than shorter passwords.

# VERY Simple Illustration of Rainbow Tables

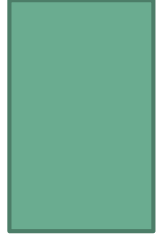
Password	MD5 Hash (in Hex)
aaaa	74b87337454200d4d33f80c4663dc5e5
aaab	4c189b020ceb022e0ecc42482802e2b8
aaac	3963a2ba65ac8eb1c6e2140460031925
aaa1	39dc4f1ee693e5adabddd872247e451f
aaa2	0ad346c93c16e85e2cb117ff1fcfada3
aaa4	ee93fca7c150d9c548aff721c87d0986

Password	MD5 Hash (in Hex)
aaaaa	594f803b380a41396ed63dca39503542
aaabb	120858a7016efcfab66967b834e9153c
aaacc	ee43671d755ac457cf e6e32d1894788e
aaa1a	5bbac29650eb36b4de16885c190a9fa3
aaa2a	597f0ce6d11567cc691b3f5df35594cb
aaa4a	4305dc076b3ba2bf8d55524cddf5a72d

# Hash - Salt

- ▶ Random bits added to further secure encryption or hashing. Most often encountered with hashing, to prevent Rainbow Table attacks.
- ▶ Essentially the salt is intermixed with the message that is to be hashed. Consider this example. You have a password that is
- ▶ pass001
- ▶ in binary that is
- ▶ 01110000 01100001 01110011 01110011 00110000 00110000  
00110001
- ▶ A salt algorithm would insert bits periodically, lets assume for our example that we insert bits every 4<sup>th</sup> bit giving us
- ▶ 0111100001 0110100011 0111100111 0111100111 0011100001  
0011100001 0011100011
- ▶ If you convert that to text you would get
- ▶ xZ7?? #

# The Birthday Paradox



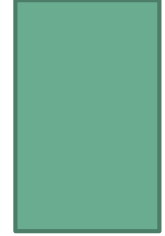
The basic idea is this: How many people would you need to have in a room to have a strong likelihood that two would have the same birthday (month and day, but not year).

Obviously, if you put 367 people in a room, at least 2 of them must have the same birthday, since there are only 365 days in a year, plus one more in a leap year.

The paradox is not asking how many people you need to guarantee a match, just how many you need to have a strong probability.

Even 23 people in the room, you have a 50 percent chance that 2 will have the same birthday.

# The Birthday Paradox



The probability that the first person does not share a birthday with any previous person is 100 percent, because there are no previous people in the set. That can be written as  $365/365$ .

The second person has only one preceding person, and the odds that the second person has a birthday different from the first are  $364/365$ .

The third person might share a birthday with two preceding people, so the odds of having a birthday from either of the two preceding people are  $363/365$ .

Because each of these are independent, we can compute the probability as follows:

$365/365 * 364/365 * 363/365 * 362/365 \dots * 342/365$

(342 is the probability of the 23rd person shares a birthday with a preceding person.)

When we convert these to decimal values, it yields (truncating at the third decimal point):

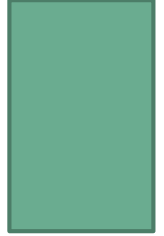
$1 * 0.997 * 0.994 * 0.991 * 0.989 * 0.986 * \dots 0.936 = 0.49$ , or 49 percent.

This 49 percent is the probability that 23 people will not have any birthdays in common; thus there is a 51 percent (better than even odds) chance that 2 of the 23 will have a birthday in common.

This relates to the odds of two inputs to a hash having the same output



# Other Crypto Attacks



**Downgrade:** A downgrade attack is often used against secure communications such as TLS in an attempt to get the user to shift to less secure modes.

**Brute Force:** Brute force tries every possible random combination.

▶ How effective is brute force?

- ▶ AES 128 bit, if you tried 1 trillion keys a second could take 112,527,237,738,405,576,542 years




This technique was invented by Mitsuru Matsui.

It is a known plain-text attack and uses a linear approximation to describe the behavior of the block cipher. Given enough pairs of plain text and corresponding cipher text, bits of information about the key can be obtained.


Obviously the more pairs of plain text and cipher text you have, the greater the chance of success.

Linear cryptanalysis is based on finding affine approximations to the action of a cipher. It is commonly used on block ciphers.




Cryptanalysis is an attempt to crack cryptography. For example, with the 56-bit DES key, a brute-force attack could take up to  $2^{56}$  attempts.

Linear cryptanalysis will take  $2^{47}$  known plain texts. This is better than brute force but still impractical for most situations. Matsui first applied this to the FEAL cipher, and then later to DES. However, the DES application required  $2^{47}$  known plain text samples, making it impractical.



With this method, a linear equation expresses the equality of two expressions that consist of binary variables that are XOR'd. For example, the following equation, XORs the sum of the first and third plain-text bits and the first cipher-text bit is equal to the second bit of the key.

$$P_{i1} \oplus P_{i2} \dots \oplus C_{j1} \oplus C_{j2} \dots = K_{k1} \oplus K_{k2} \dots$$



Differential cryptanalysis is a form of cryptanalysis applicable to symmetric key algorithms. Invented by Eli Biham and Adi Shamir, it involves the examination of differences in an input and how that affects the resultant difference in the output. In other words, differential cryptanalysis focuses on finding a relationship between the changes that occur in the output bits as a result of changing some of the input bits.

It originally worked only with chosen plain text. However, it could also work with known plain text and cipher text.

The attack is based on seeing pairs of plain-text inputs that are related by some constant difference. The usual way to define the differences is via an XOR operation, but other methods can be used. The attacker computes the differences in the resulting cipher texts and looks for some statistical pattern. The resulting differences are called the *differential*.



Differential cryptanalysis measures the XOR difference between two values.


Differentials are often denoted with the symbol  $\Omega$ . So you might have a differential  $\Omega_a$  and another differential  $\Omega_b$ .

A characteristic is composed of two differentials. For example, differential  $\Omega_a$  in the input produces differential  $\Omega_b$  in the output. These matching differentials are a *characteristic*. The characteristic demonstrates that the specified differential in the input leads to a particular differential in the output.



Regardless of the technique used, there are three resources for cryptanalysis:

- **Time:** The number of “primitive operations” that must be performed. This is quite loose—primitive operations could be basic computer instructions, such as addition, XOR, or shift, or entire encryption methods.
- **Memory:** The amount of storage required to perform the attack.
- **Data:** The quantity of plain texts and cipher texts required.



Particularly with e-mail and hard drive encryption, there is usually some password the user must know to decrypt and access the information.

Many people are not even aware when their e-mail password has been cracked. Several web sites keep lists of e-mail accounts that have been breached, such as [haveibeenpwned.com](http://haveibeenpwned.com) and [PwnedList.com](http://PwnedList.com)

If, for example, you are a law enforcement officer attempting to breach an encrypted drive belonging to a suspect, you can check these sites to see if that suspect's e-mail account has been breached. You can then use the e-mail password, and close permutations, to attempt to decrypt the hard drive.





People often choose passwords that have meaning for them to make memorization much easier.

I frequently advise forensic analysts to learn all they can about a suspect. Photographing or videotaping the area where the computer is seized can be quite valuable. If, for example, the suspect is an enthusiastic fan of a particular sports team, with memorabilia extensively displayed in his home or office, this might aid in breaking encrypted drives, phones, and e-mails. It is at least reasonably likely that this individual's cryptography password is related to a beloved sports team.



**Downgrade:** A downgrade attack is often used against secure communications such as TLS in an attempt to get the user to shift to less secure modes.

**Coppersmith's attack** describes a class of cryptographic attacks on the public-key cryptosystem RSA based on the Coppersmith method. It works best when implementations of RSA use a low exponent such as the common  $e$  values of 3, 17, and 65537 ( $2^{16}+1$ )

**Man in the middle:** intercepting the key exchange handshake.

**Drown**(Decrypting RSA with Obsolete and Weakened eNcryption) attack is a cross-protocol security bug that attacks servers supporting modern TLS protocol suites by using their support for the obsolete, insecure, SSL v2 protocol to leverage an attack on connections using up-to-date protocols that would otherwise be secure.