Cryptography and Steganography

CS-3113: PRINCIPLES OF CYBER SECURITY

BENJAMIN R. ANDERSON

Definitions – Lots of Definitions

Encryption: The cryptographic transformation of plaintext to produce ciphertext

Decryption: The cryptographic transformation of ciphertext to product plaintext

Plaintext: Data that ***can*** be viewed or used, without requiring a key, password, or method of decryption (the "original" message)

Ciphertext: Data that results from encryption, it ***cannot*** be viewed or used without requiring a key, password, or method of decryption (the "unreadable" message)

Cryptographic system: The specific method used for encryption and decryption

Transposition: Rearranging the order of elements (Ex: ABCD becomes BDAC)

Substitution: Replacing the elements with a different one (Ex: ABCD becomes EFGH)

Key: A piece of information that is required to encrypt and/or decrypt data

Steganography: The practice of hiding a message in such a way that its very existence is hidden

Encryption

The goal is to protect information by transforming it into an unreadable format

Encryption is used to prevent information from "falling into the wrong hands"

- With encryption, data is only available to the people that are supposed to see it
- Encryption is ***NOT*** the same thing as a code

Code: Words are substituted for other words

- As a famous example from history - Navajo code talkers from WW II
    - "dah-he-tih-hi (hummingbird) substituted for "fighter plane“
    - "besh-lo" (iron fish) meant "submarine"

Goal of Cryptography and Steganography

Looking at the CIA Triad

- Both “crypto” and “stego” are primarily interested in Confidentiality
- Both are aimed at keeping someone from reading the actual information
  - Stego also tries to hide the existence of the data by obscuring it in some way
- However, some crypto schemes also allow for integrity
  - If a small part of the message is changed, then the decryption results in unintelligible information
  - Therefore, it can’t be changed in a meaningful way unless an attacker can:
    - Decrypt the ciphertext
    - Modify the plaintext
    - Re-encrypt the modified data

History

Cryptography has been around for thousands of years

Read: *A Brief History of Cryptography* by Huzaifa Sidhpurwala

- https://www.redhat.com/en/blog/brief-history-cryptography
- You will not be responsible for knowing exact dates, but should understand the general progression of cryptographic methods

How Encryption Works

A cryptographic system has the following process:

- `Encrypt`$_{key}$`(Plaintext)  = Ciphertext`
- `Decrypt`$_{key}$`(Ciphertext) = Plaintext`

This can also be written as:

- `Encrypt(Plaintext, key)  = Ciphertext`
- `Decrypt(Ciphertext, key) = Plaintext`

For this to be successful, you must know the method used to encrypt the message and the key (if used)

Not all crypto methods use keys

Let's look at a simple Caesar cipher

- Plaintext = "hello"
- Encryption method = substitute letter with letter plus 3
- h → i → j → k
- `Encrypt("hello") = Ciphertext = "khoor"`

This is relatively easy to decipher since some letters appear more often than others

- For example, the most common ciphertext letter (for English) is probably an ‘e’
- In addition, there are doubled up letters (called bigrams) that happen regularly
  - For example, the ‘tt’ in “letters” is a bigram
  - Many bigrams are incredibly rare – such as yy

However, modifying the cipher – slightly – can make it more secure

For example, start with a shift of 3, then 4, then 5

- When you reach the end of the alphabet, you just start over
- Essentially you use “mod 26” after encryption

Plaintext = "hello“

- Encryption method = substitute letter with letter + i
- where i starts at 3 and increments 1 for each letter (mod 26)
- `Encrypt("hello") = Ciphertext = "kiqrv"`

More Definitions

Block Cipher: A cryptographic system that operates on a block of plaintext (or ciphertext) at a time

- A block could be any length – 64 bits, 256 bits, 1024 bits, etc.
- If there is not enough data to complete a block, it is padded with extra bits to make a complete block

Stream Cipher: A cryptographic system that operates on a single character, byte, or even bit at a time

- Often uses the XOR function for encryption and decryption
- A special version of the stream cipher is a one-time pad

Encryption Methods

There are two broad category of encryption methods used today

- Symmetric encryption
- Asymmetric encryption.

Symmetric: The same key is used for both encryption and decryption (also called private key encryption)

- Data Encryption Standard (DES)
  - Loosely based on Lucifer which used 128 bit keys
  - NSA asked that the key length be shortened to 56 bits
- Triple-DES
  - Variant of DES with various modes and keys
  - $\text{Ciphertext} = E_{K3}(D_{K2}(E_{K1}(\text{Plaintext})))$
  - $\text{Plaintext} = D_{K3}(E_{K2}(D_{K1}(\text{Ciphertext})))$
- Advanced Encryption Standard (AES)
- International Data Encryption Algorithm (IDEA)

Data Encryption Standard

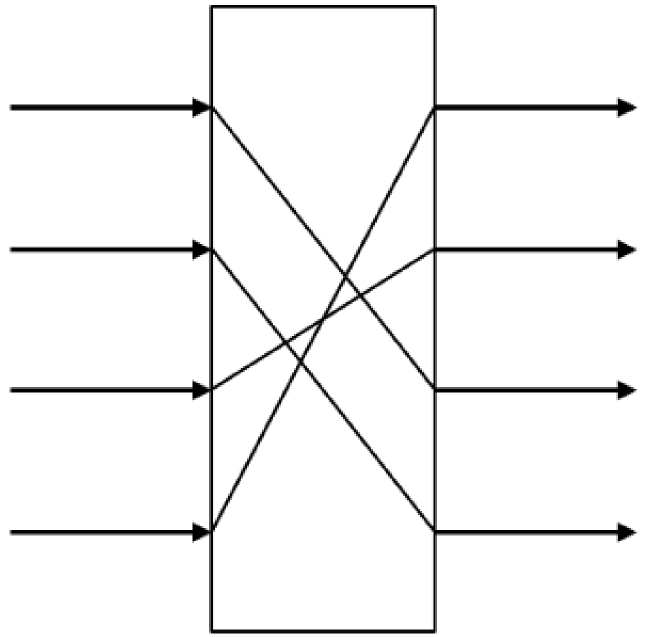
Properties:

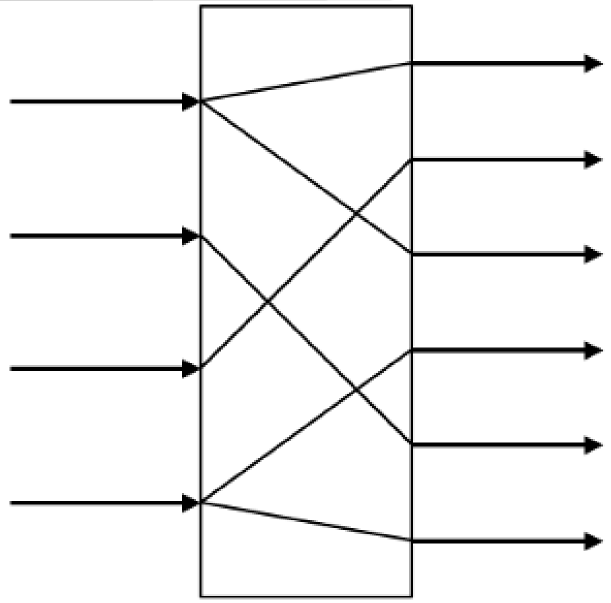
- Block size: 64 bits
- Key size: 56-bits
- Drop bits: 8, 16, 24, 32, 40, 48, 56, 64
    - These are used for parity bits to ensure the previous 7 bits were read correctly
    - Ex: 1-7 verified by bit 8; 9-15 verified by bit 16, etc.
    - This is because, at this time, keys were often loaded by pulling a paper tape through a reader and was prone to errors
- Utilizes: Substitution and Transposition

Algorithm:

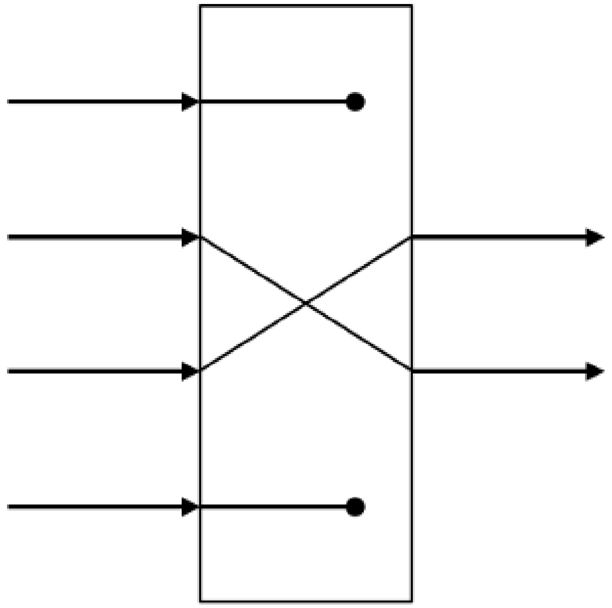
- 64-bit block is permutated
- 64-bit block split into two 32-bit blocks
- Each sub-block is combined with the key and processed 16 times
    - "cycles" of permutations, shifts, combinations (XOR), splits
    - Substitutions are made by “S-boxes” which was considered to be the true strength of the algorithm
- Sub-blocks are joined and sent through an inverse permutation process

Types of Permutations


Permutation


Expansion Permutation

Types of Permutations


Permuted Choice

Basically, permutations are moving the bits (or characters) around

- Expansion permutations uses some bits more than once to rearrange them
- Permuted choice moves some bits around, and drops other ones

Data Encryption Standard

You need to be familiar with the types of operations that happen in a cryptographic system

- This will illustrate how complex the encryption (and decryption) algorithms can be

You will not be required to memorize all the elements of DES

- However, you will be responsible for the things involved in the algorithm

Watch: *DES encryption by hand (simple low level example at a bit view)* by Ineapple:

- *https://www.youtube.com/watch?v=Sy0sXa73PZA*
- This video will walk you through a "round" of encrypting a block with DES

(***Optional***) Skim over the DES standard which starts on page number 8 (page 13 in the document)

- This will give you the exact implementation details of the algorithm – including the various lookup tables (Initial Permutation, etc.)
- https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf

Data Encryption Standard

Note: Some people felt that the NSA's involvement in the design of the S-Boxes meant they inserted a back door

- However, later on, it was discovered that they made them resistant to differential cryptanalysis – years before that technique was common knowledge

From Wikipedia:

- *Some of the suspicions about hidden weaknesses in the S-boxes were allayed in 1990, with the independent discovery and open publication by Eli Biham and Adi Shamir of differential cryptanalysis, a general method for breaking block ciphers. The S-boxes of DES were much more resistant to the attack than if they had been chosen at random, strongly suggesting that IBM knew about the technique in the 1970s.*
- *https://en.wikipedia.org/wiki/Data_Encryption_Standard#NSA's_involvement_in_the_design*

Data Encryption Standard

In addition to the S-boxes, there were other questions raised about DES:

- Number of Iterations
  - Why only 16 cycles?
  - Wouldn't more cycles be better?
- Key length
  - The longer the key, the harder it is to crack a message
  - This raised the question of why the DES key was shorted
    - The Lucifer algorithm was 128 bits
    - DES is only 64 bits – and only 56 are key bits

However, cryptography often involves trade-offs between speed and security

- Sometimes compromises must be made to make a system usable

Data Encryption Standard

Weaknesses of DES

- Weak keys
  - There are known keys that weaken the algorithm
  - e.g. keys where the shifted half of the key is all 1's or all 0's
- Semi-weak keys
  - Some keys exist for which another key can be found
  - This results in the identical decryption of a message encrypted with the first
- Potential issues with the S-Boxes
  - It is possible to produce the same encrypted values after one cycle with different inputs
  - This might indicate an inherent weakness
- Slow and only works on block sizes of 64 bits.
  - Key size is too small for power of today's computers

Exhaustive Key Search

An important aspect of any cryptographic system is the overall strength of it

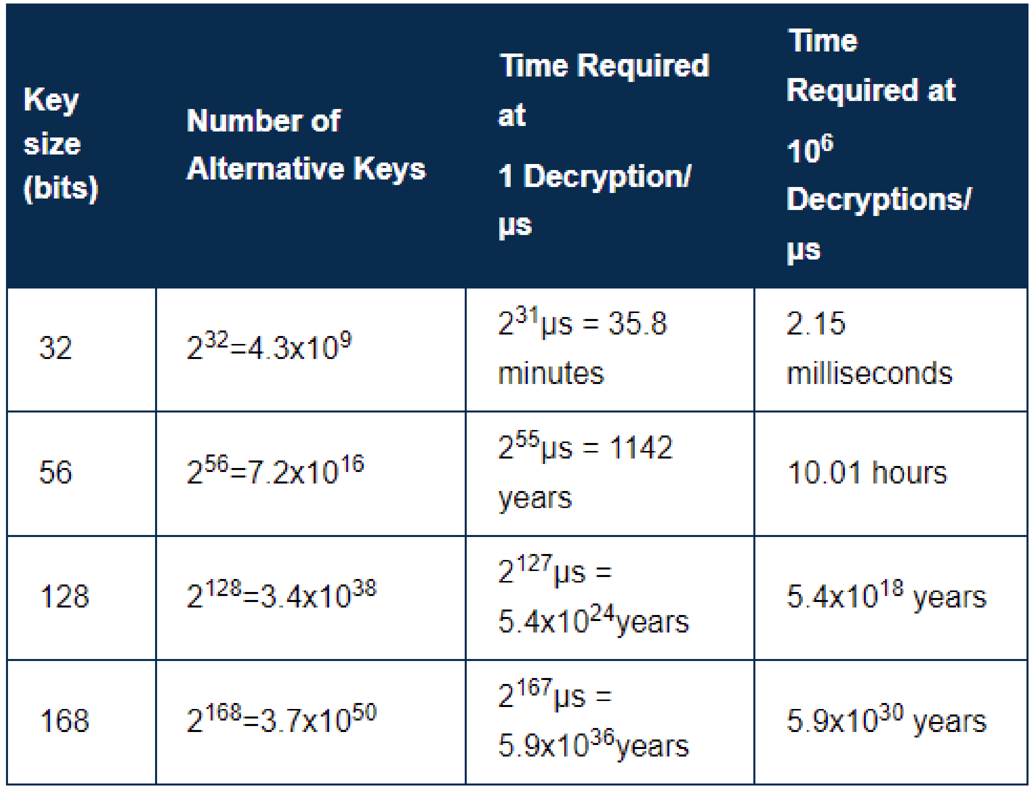
This can be measured by calculating how long it would take to do an exhaustive search of the key space

- This is directly impacted by the length of the key
- For DES that would require 2^{56} searches

Notice the extreme amounts of time it will take for us to search all possibilities for different key sizes

Consider: According to www.space.com our sun will burn out in about 5 billion years (5×10^9)

- Therefore, 128 bits (or more) will require more time to crack than there are years left for our solar system



Key size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/ µs	Time Required at 10^6 Decryptions/ µs
32	$2^{32}=4.3\text{x}10^9$	2^{31}µs = 35.8 minutes	2.15 milliseconds
56	$2^{56}=7.2\text{x}10^{16}$	2^{55}µs = 1142 years	10.01 hours
128	$2^{128}=3.4\text{x}10^{38}$	2^{127}µs = $5.4\text{x}10^{24}$years	$5.4\text{x}10^{18}$ years
168	$2^{168}=3.7\text{x}10^{50}$	2^{167}µs = $5.9\text{x}10^{36}$years	$5.9\text{x}10^{30}$ years

Advanced Encryption Standard

Given the short key size of DES, a more modern system can complete an exhaustive search in less than 10 hours

Over 25 years ago (1998), the Electronic Frontier Foundation (EFF), built a customized DES cracker for under $250,000

- It checked 88 billion keys a second for 56 hours
- “Worst case” would take 228 hours to try all 256 keys
- Done to prove the insecurity of DES
- https://en.wikipedia.org/wiki/EFF_DES_cracker

This resulted in the U.S. government putting out a call for a new encryption standard

- The ***Advanced Encryption Standard*** (AES)

Requirements for AES

Algorithm requirements:

- Handle block size of 128 bits
- Handle key lengths of 128, 192 and 256 bits

Algorithm evaluation criteria:

- Security
  - Resistance to known cryptanalysis techniques
  - Sound mathematical foundation
  - Randomness of output
- Cost
  - No licensing fees
  - No expensive hardware required
- Algorithm Characteristics
  - Efficiency in hardware and software
  - Memory requirements
  - Flexibility in key size and block size
  - Performs well in a variety of hardware such as smartcards

AES Development Timeline

September 20, 1997
- The call for algorithms is made

August 20, 1998
- 15 candidate algorithms named

August 1999
- NIST announced the 5 finalists: MARS, RC6, Rijndael, Serpent and Twofish

May 15, 2000
- The period for public comments ends

November 26, 2001
- Rijndael becomes the basis of the AES

June 2003
- 192 or 256 bit key AES authorized for encrypting top secret information

Properties of Rijndael

Supported key size: 128, 192, 256 bits

Support block sizes: 128, 192, 256 bits

AES standard only indicates block size of 128 bits

Number of "rounds" based on block size and key size

	128	**192**	**256**
128	10	12	14
192	12	12	14
256	14	14	14

Rijndael Algorithm

The easiest way to consider the block to be encrypted is as a 4x4 matrix of a single byte, (for a 128 bit block), call this A.

Before the first round, A is XORed with $K_{0.}$

Each round of the algorithm consists of 4 steps:

1. Byte substitution
2. Row shift
3. Column Mix
4. XOR with the appropriate round key

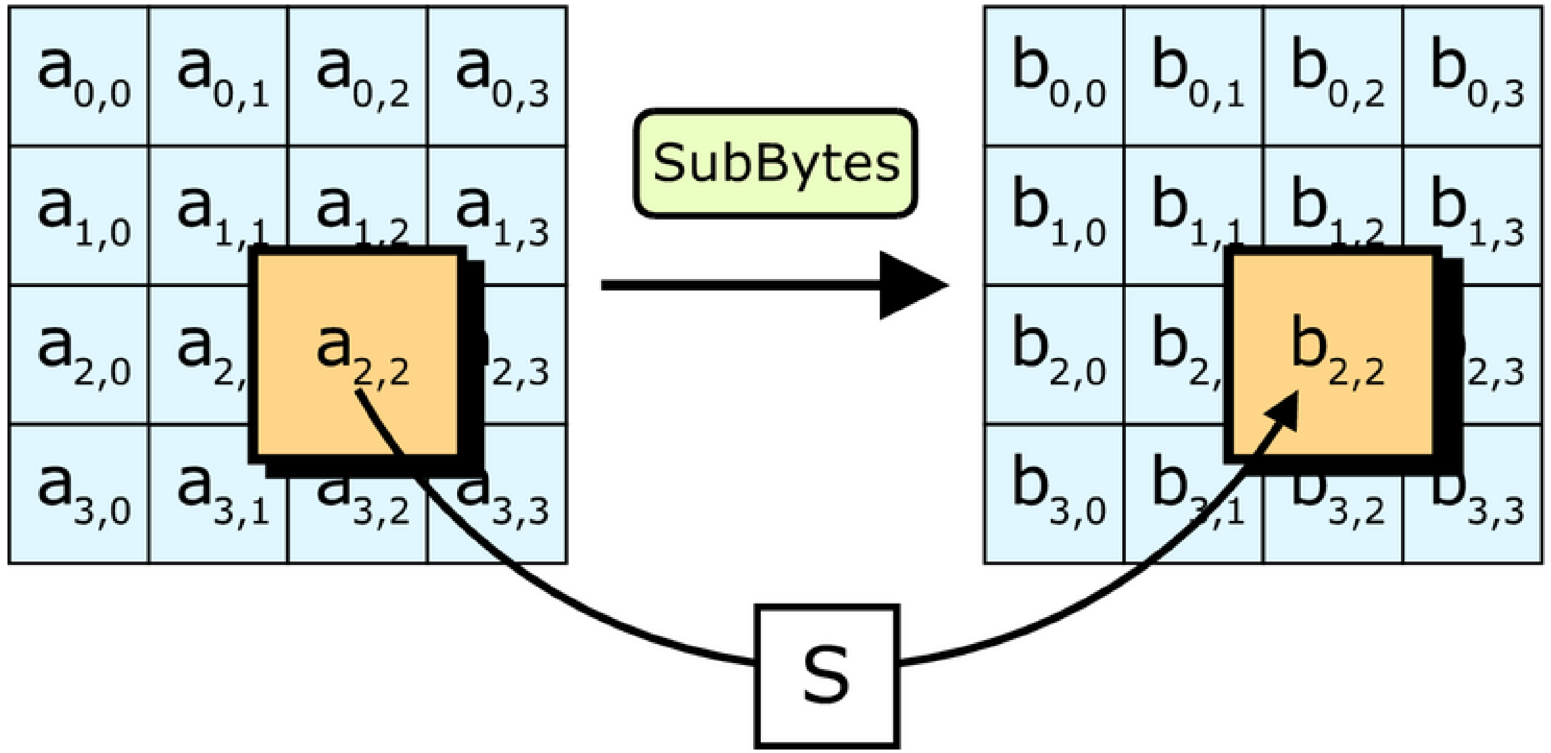
- The last round does not perform the Column Mix

Rijndael Algorithm - Byte Substitution

Take a single byte from the 4x4 matrix, and convert it to the “new” value using table lookup.

The “new byte” goes into the same position in the matrix as the original byte.

Rijndael Algorithm - Byte Substitution



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S

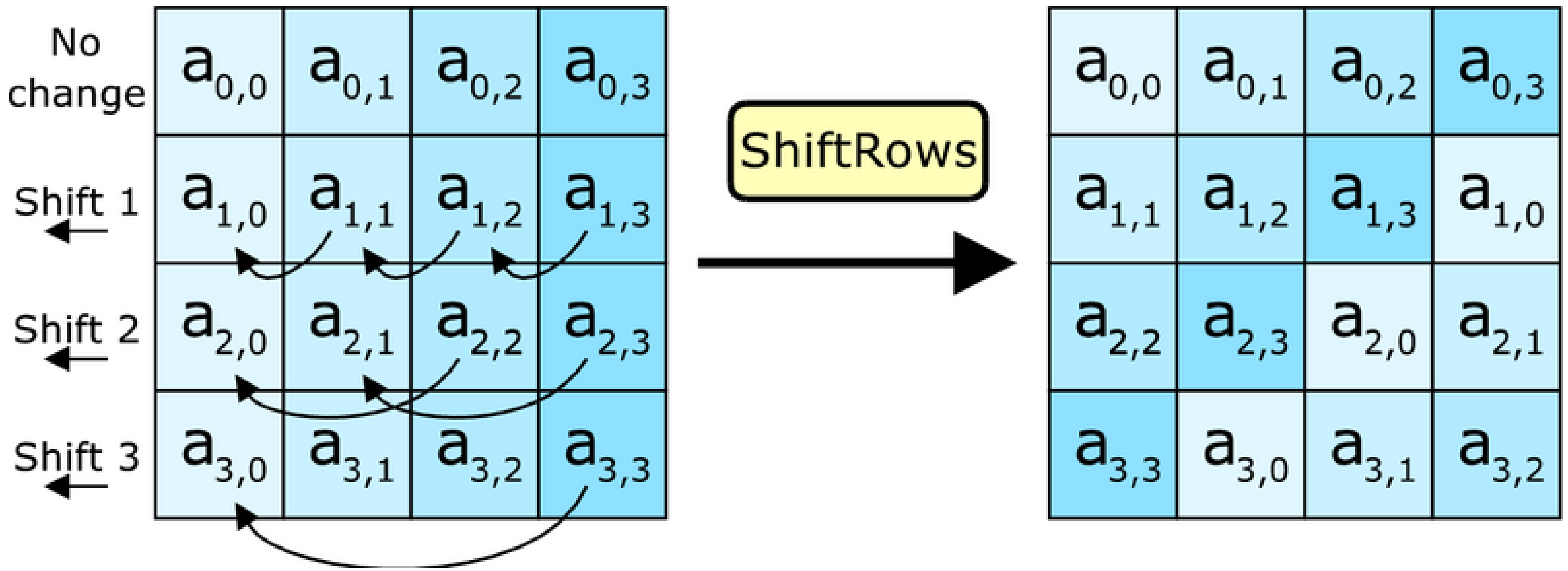
Image from Wikipedia.org: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Rijndael Algorithm – Row Shift

Each row of the 4x4 matrix is shifted “left” by a certain amount. (A permutation of the rows.)

- Row 0 is shifted left 0 times.
- Row 1 is shifted left 1 times.
- Row 2 is shifted left 2 times.
- Row 3 is shifted left 3 times.

Rijndael Algorithm – Row Shift



No change
Shift 1
Shift 2
Shift 3
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

Image from Wikipedia.org: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Column Mix

The mix is a linear transform and is done by treating the column as a polynomial over a Galois Field (F(2^8))

This is then multiplied by the polynomial:

- ($3x^3 + x^2 + x + 2$) modulo ($x^4 + 1$).

Can also be considered as:

$$t(A) = M \bullet A$$

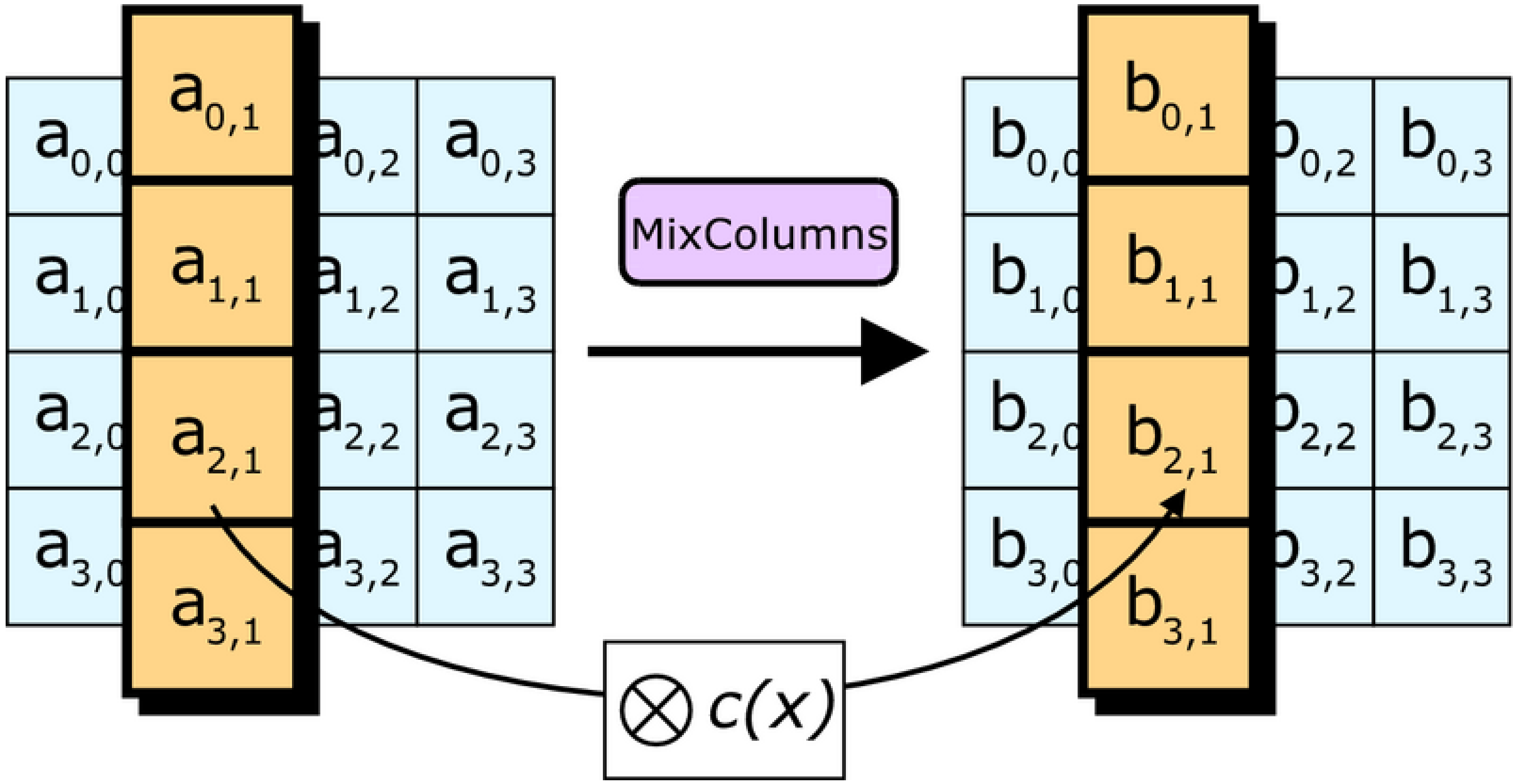
Where:

- M is a 32 x 32 bit matrix
- A is the 32 x 4 block being operated on

For more information, you can refer the NIST FIPS on AES, which can be found at:

- http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
- ***Note***: You are not required to know the underlying math for this operation

Rijndael Algorithm – Column Mixing



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes c(x)$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

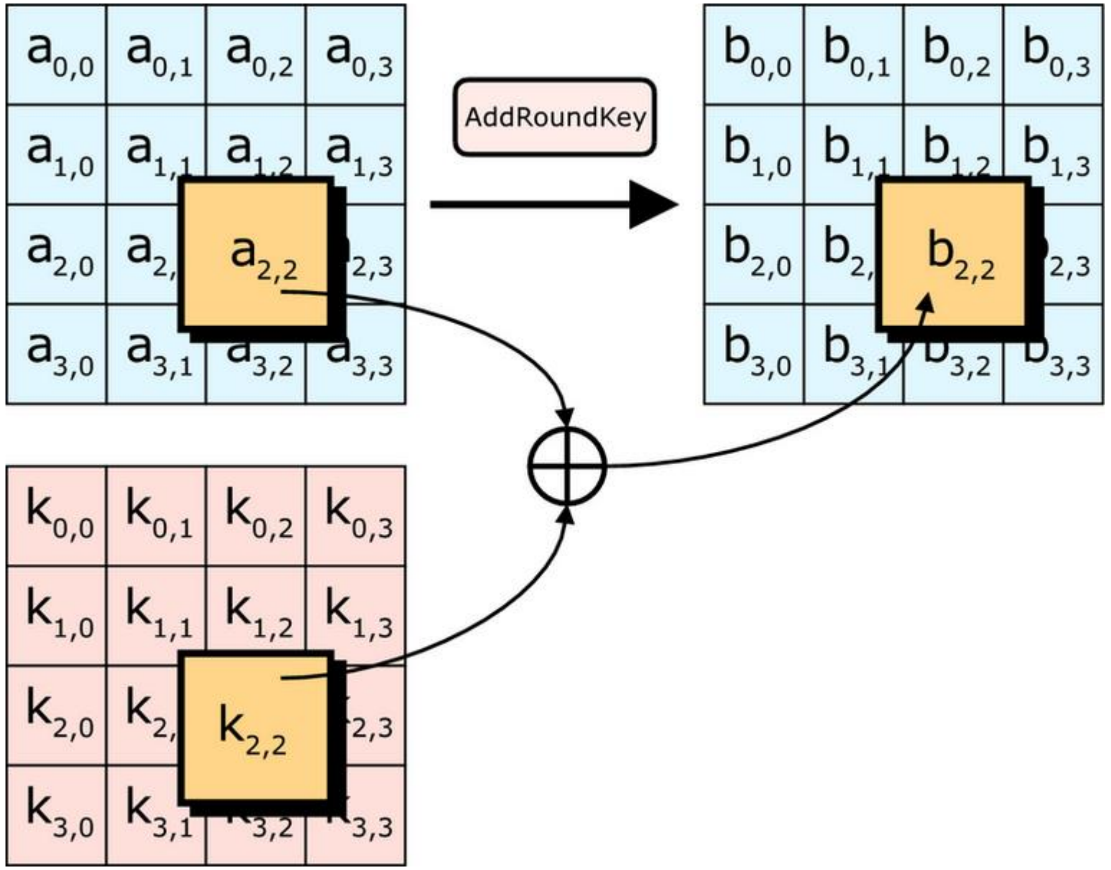
Image from Wikipedia.org: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Rijndael Algorithm – Round Key XOR

Each entry in the 4 x 4 matrix is XORed with the corresponding entry in the round key matrix.

Each entry in the key matrix is also 1 byte in size.

Rijndael Algorithm – Column Mixing



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$\oplus$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

Image from Wikipedia.org: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Rijndael Algorithm

Repeat this process 9 to 13 more times depending on key and block size.

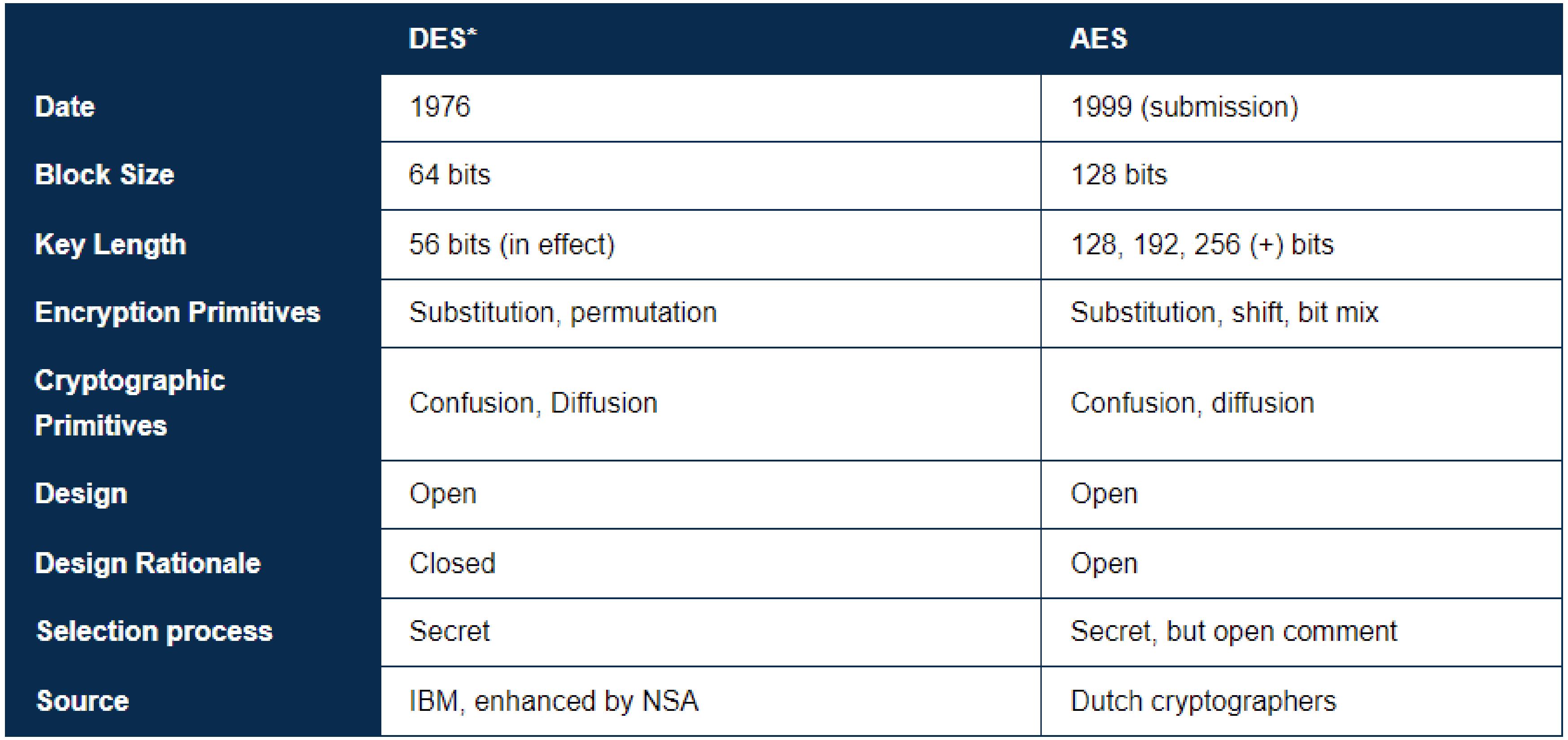
Note: The operation on each byte can be performed in parallel

Also, the various operations can be performed by using 4 table lookups, for a total table size of 1kB.

Decryption occurs by using inverse functions for the byte substitution, row shift and column mixing

These properties make Rijndael extremely fast for a cryptographic algorithm

Comparison of DES and AES



	DES*	AES
Date	1976	1999 (submission)
Block Size	64 bits	128 bits
Key Length	56 bits (in effect)	128, 192, 256 (+) bits
Encryption Primitives	Substitution, permutation	Substitution, shift, bit mix
Cryptographic Primitives	Confusion, Diffusion	Confusion, diffusion
Design	Open	Open
Design Rationale	Closed	Open
Selection process	Secret	Secret, but open comment
Source	IBM, enhanced by NSA	Dutch cryptographers

Table taken from *Security in Computing* by Pfleeger and Pfleeger

Skipjack and Clipper Chip

Skipjack was developed by the NSA

- Used an 80-bit key
- Ended up as one of the most controversial encryption algorithms developed
- It allowed for key escrow – a “backdoor” that could allow decryption
- This backdoor used a separate mechanism called the Law Enforcement Access Field (LEAF)

Skipjack was the algorithm integrated into the Clipper chip

Clipper chip

- Meant to help FBI combat criminals using encryption
- Built in back door for law enforcement
- Used key escrow: the session key was held by a third-party for later for release to law enforcement when needed
- Government tried to force this as a standard but it was not widely accepted

Skipjack and Clipper Chip

Controversy:

- The government wanted to basically "outlaw" the use of any public encryption algorithm except Skipjack as used by the Clipper chip
- This included a public relations campaign to gain acceptance of Clipper
  - It was necessary to fight against child pornography
  - It was necessary to combat terrorism
  - Would require a warrant to get the key held in escrow (similar to a warrant for a wiretap) so it wasn’t government intrusion
- Nothing would stop a criminal from using a different cryptographic system
  - However, that would have been a crime – *Illegal use of a cryptographic algorithm*
  - Essentially, this would allow the FBI to charge someone with a crime, even if they couldn’t read the (possibly) incriminating data
  - This is similar to the Prohibition Era when Al Capone was finally convicted of tax evasion instead of other crimes
- Eventually the idea met with such opposition, and was basically a very poor idea anyway, that the government dropped the idea

Issues With Symmetric Encryption

Problems with symmetric methods:

- Same key is used to encrypt and decrypt
- Shared key is more likely to be compromised
- It is possible to brute force short keys
- Certain keys are weak
- Different keys can produce identical ciphertext

Distribution of keys is also difficult

- Large groups make this difficult
- This problem is called “key management”

The method does not scale well

- The number of keys required for any two individuals to communicate securely is proportional to the square of the users

Issues With Symmetric Encryption

The number of keys required can be visualized as a 2-dimenstional matrix

- Number of rows and columns is the number of users
- Since the keys are symmetric, we can get rid of half the matrix (below the diagonal)
  - The key for User1 to User2 communication is the same as the User2 to User1 key
  - A user doesn't need a key for themselves, so we can also eliminate the diagonal
  - This gives us the formula for the number of keys as:
  - `(n * (n-1)) / 2`

Number of Users	Unique Keys Needed
2	1
3	3
4	6
5	10
10	45
100	4950
1000	499,500
100,000	4,999,950,000
1,000,000	499,999,500,000
34,734 UTSA Students Fall 2021	603,208,011

Asymmetric Algorithms

To address the key management problem, asymmetric encryption was developed

- Also called public key cryptography

Diffie-Hellman

- Developed the idea for public key cryptography in 1976

Private Key: Known only to the owner

Public Key: Can be known to others

Public Key Algorithms:

- RSA
  - Named for its inventors: Ron Rivest, Adi Shamir, and Leonard Adleman
- PGP
  - Pretty Good Privacy
  - Developed by Phil Zimmerman

Diffie-Hellman Method

Ciphertext = Encrypt(Plaintext, public_key)

Plaintext = Decrypt(Ciphertext, private_key)

Steps:

- Each party creates their own private key
- Each party computes a public key using a mathematical function of the private key
- Public keys are exchanged
- Message key is computed from other person's public key and your own private key
- Given the mathematical properties of key selection, the message key is the same on both sides

Public Key Algorithm Key Management

The management of public keys in this scheme is known as public key infrastructure

- These are the key management systems that allow users to lookup and retrieve public keys
- Assuming the other user maintains the security of their private key, only they can decrypt it
  - Note this assumption

This scheme greatly reduces the key management problem

- From: `(n * (n-1)) / 2`
- To: `(2 * n)`

For the UTSA students in Fall 2021 (34,734 students):

- ***Symmetric keys required***: 603,208,011
- ***Asymmetric keys required***: 69,468 (one public and one private for each student)
- Almost a 99.99% reduction – for even this small of an organization
- Imagine something like the Department of Defense with millions of users

Diffie-Hellman Key Exchange

Read: Diffie-Hellman key exchange

- https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
- For your general understanding – but make note of the "Secrecy Chart" section

Cryptographic Math

Developing algorithms for cryptographic systems is difficult

Requires algorithms that are easy to verify (i.e. – encrypt/decrypt) but are very difficult to brute-force

The general approach is to use an NP-Complete problem

- NP-Complete: Is a special class of non-deterministic polynomial (NP) problems that very strongly suggest that there is not a “crack” or “shortcut” that will find an answer in less than exponential time:
  - Ex: If there are n keys, it cannot be cracked in less than 2^n operations
- This is different from polynomial (P) problems which have solutions that can run in a bounded polynomial function of the problem size
  - Ex: If there are n keys, it could be cracked in n^2 or n^3 (it doesn’t have to be a power of 2 or 3 – but it is still a polynomial)
  - Even if n is 1,000,000 that is still in the set p

There is one unbreakable crypto method – the One-Time Pad (OTP) which will be discussed later

NP-Complete

From Wikipedia:

- *In computational complexity theory, a problem is NP-complete when:*
  1. *it is a problem for which the correctness of each solution can be verified quickly (namely, in polynomial time) and a brute-force search algorithm can find a solution by trying all possible solutions.*
  2. *the problem can be used to simulate every other problem for which we can verify quickly that a solution is correct. In this sense, NP-complete problems are the hardest of the problems to which solutions can be verified quickly. If we could find solutions of some NP-complete problem quickly, we could quickly find the solutions of every other problem to which a given solution can be easily verified.*
- *The name "NP-complete" is short for "nondeterministic polynomial-time complete". In this name, "nondeterministic" refers to nondeterministic Turing machines, a way of mathematically formalizing the idea of a brute-force search algorithm.*
- https://en.wikipedia.org/wiki/NP-completeness

Another problem that is an example of an NP-Complete problem is the Knapsack Problem

- ***Read***: Knapsack Problem: https://en.wikipedia.org/wiki/Knapsack_problem
- This is for your general understanding

Asymmetric Encryption

Remember:

- Each user has a public key and a private key
- They hold onto their private key – and keep it safe!
- However, the public key can be made available to anyone

Watch: *Asymmetric Encryption – Simply explained* by Simply Explained

- https://www.youtube.com/watch?v=AQDCe585Lnc
- You should understand how a message is protected using public and private keys

Watch: *What is digital signature?* by Sunny Classroom

- https://www.youtube.com/watch?v=TmA2QWSLSPg
- You should understand the parts of a digital signature and how the public and private keys are used to create a digital signature

Other Uses of Cryptography

Digital Certificates

- Used to encode and verify messages
- Requires a Certificate Authority that creates a digital certificate based on a private key and other authentication information
- Implements the "trusted third party" concept
- X.509 is a popular standard for defining digital certificates

VPN (Virtual Private Network)

- Connects geographically separate offices using public communication means
- Packets are usually "tunneled" – entire packet is encrypted and encapsulated in a new packet before sending
- Hardware or software based
- Sometimes integrated into firewalls
- Very good for mobile employees that need access to the company network

Transport Layer Security (TLS)

- Replacement for SSL (Secure Sockets Layer) used in web transactions
- Uses Asymmetric encryption to exchange symmetric keys

Encryption is also increasingly being used in the offensive tools of cyberwarfare – such as malware

- The actual payload is encrypted to defeat signature analysis
- The malware decrypts the payload before delivering it

Public Key Infrastructure

Problem:

- How does Bob know that the public key he is using belongs to Alice?

Solution:

- Use a trusted third-party to verify Alice's identity and digitally sign a certificate that binds her identity to that public key
- This is called a certificate authority (CA)

The infrastructure to support the hardware and software elements to make this solution workable is called ***public key infrastructure*** or ***PKI***

Public Key Infrastructure

How it works:

- Assume the CA has issued a self-signed certificate for itself (this creates the “root of trust”) with its public key
- Alice and Bob agree to use the CA to verify their identities
- Alice requests a public key certificate from the CA
- The CA:
  - Verifies her identity
  - Computes a hash of the content that will make up her certificate
  - Signs the hash by using the private key that corresponds to the public key in the published CA certificate
  - Creates a new certificate by concatenating the certificate content and the signed hash
  - Makes the new certificate publicly available.
- Bob :
  - Retrieves the certificate
  - Decrypts the signed hash by using the public key of the CA
  - Computes a new hash of the certificate content, and compares the two hashes
  - If the hashes match, the signature is verified
  - Bob can assume the public key in the certificate belongs to Alice
- Bob uses Alice's verified public key to encrypt a message to her
- Alice uses her private key to decrypt the message from Bob

The certificate signing process enables Bob to verify that the public key was not tampered with or corrupted during transit

Therefore, he has some level of assurance/trust that the public key does belong to Alice

Note: Attacks against CA’s have been used to get fraudulently signed certificates

- These are revoked by the CA as soon as they are detected
- Make sure you check the revocation lists!

Operational Realities

In general

- Symmetric algorithms are fast – but have the key distribution problem
- Asymmetric algorithms are slow – but have a much easier key distribution problem

Operational Solution:

- Use the slow, asymmetric algorithm to distribute symmetric keys for a single connection or transfer
- Once done, use the faster symmetric encryption for transferring the data

From the Wikipedia article on Transport Layer Security (which is used in HTTPS):

- *The protocols use a handshake with an asymmetric cipher to establish not only cipher settings but also a session-specific shared key with which further communication is encrypted using a symmetric cipher.*
- https://en.wikipedia.org/wiki/Transport_Layer_Security#Websites

One-Time Pad

The only known encryption method that is completely unbreakable is the one-time pad (OTP)

However, there are 4 requirements:

- The key must be at least as long as the plaintext
- The key must be random
- The key must never be reused in whole or in part
- The key must be kept completely secret by the communicating parties

The randomness ***must be true randomness*** – not pseudo-random or statistically random

No reuse! It is a ONE-time pad, and not a "Use-a-couple-times pad!"

It is unbreakable because any message could be any other message of the same length encrypted with a different key

However, this is an even worse key management problem than symmetric key encryption

- Not only do you need a key for every pair of users – ***the key has to be the length of every message you intend to send***
- So, if you want to send 10 MB worth of information, the OTP must be at least 10 MB in length

One-time Pad Example

Given the Cipher text:

- `gpprvynawosypzccup`

If you use a key pad of:

- `zpdqbhhwfwshlgckbrasdfghjk`

You will obtain the possible plain text:

- "Hamburgers are tasty"

If you use a key pad of:

- `abcdefghijklmnopqrstuvwxyz`

You will obtain the possible plain text:

- "Go north to find money"

If the key is truly random, there is no way to know what the actual message is compared to any of the other possible messages

Other Concepts

COMSEC: Communications Security

Link Encryption: Encrypting information at the data link level as it is transmitted between two points within a network

- Link encryption takes place in the lowest protocol layers (layers 1 and 2 in the OSI model)
- Plaintext in the host is encrypted when it leaves the host
- It is decrypted at the next link (which may be a relay point)
- The information is then re-encrypted before it continues to the next link
- It continues in this manner until it reaches its destination.

End-to-end Encryption: Encryption takes place at the ***origin*** and decryption at the ***destination*** without intermediate (link) decryption

Onion Routing

A hybrid link encryption approach that will become more important in the next lesson is Onion Routing

Read: Wikipedia: *Onion routing*

- https://en.wikipedia.org/wiki/Onion_routing
- You will be responsible for the purpose of onion routing, how it is used, and the general steps to create and read the "onion"

Breaking Cryptography

There are a number of ways to compromise a cryptographic scheme

For example, there might be inherent weaknesses:

- The human factor
  - Somebody reveals the secret key
- Security of key and message
  - Maintaining the secret key and decrypted message in a secure manner
  - Don’t use a file named “passwords” or “keys” – and don’t post keys (including API keys) to online code repositories
- Key length
  - Short keys can be broken even with a good algorithm
- Algorithm (some algorithms are just not very secure -- ex: rot13)
  - It is very difficult to develop a secure algorithm
  - A weak algorithm can be insecure even with a long key

Breaking Cryptography

There are also a variety of technical attacks that can be used:

- ***Ciphertext only***: The only thing available are copies of encrypted messages
- ***Known plaintext***: The attacker has both the encrypted and unencrypted version of a message
- ***Chosen plaintext***: The attacker has the ability to choose the text that will be encrypted and will then have both the plaintext and ciphertext versions
- ***Adaptive chosen plaintext***: The attacker can choose multiple inputs to encrypt and can make small changes in successive messages to examine the changes in the resulting ciphertext
- ***Related Key Attacks***: Analyzing ciphertext that had been encrypted with different but related keys

In the attack on Enigma, the German cryptographic system from World War II, the daily setting was the same but different operators had different settings

- A related key attack

Breaking the Enigma

Read: From Wikipedia: Cryptanalysis of the Enigma

- https://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma
- You will be responsible to know the general techniques they used and the major details
  - Ex. The machines used to crack Enigma were known as ***bombes***

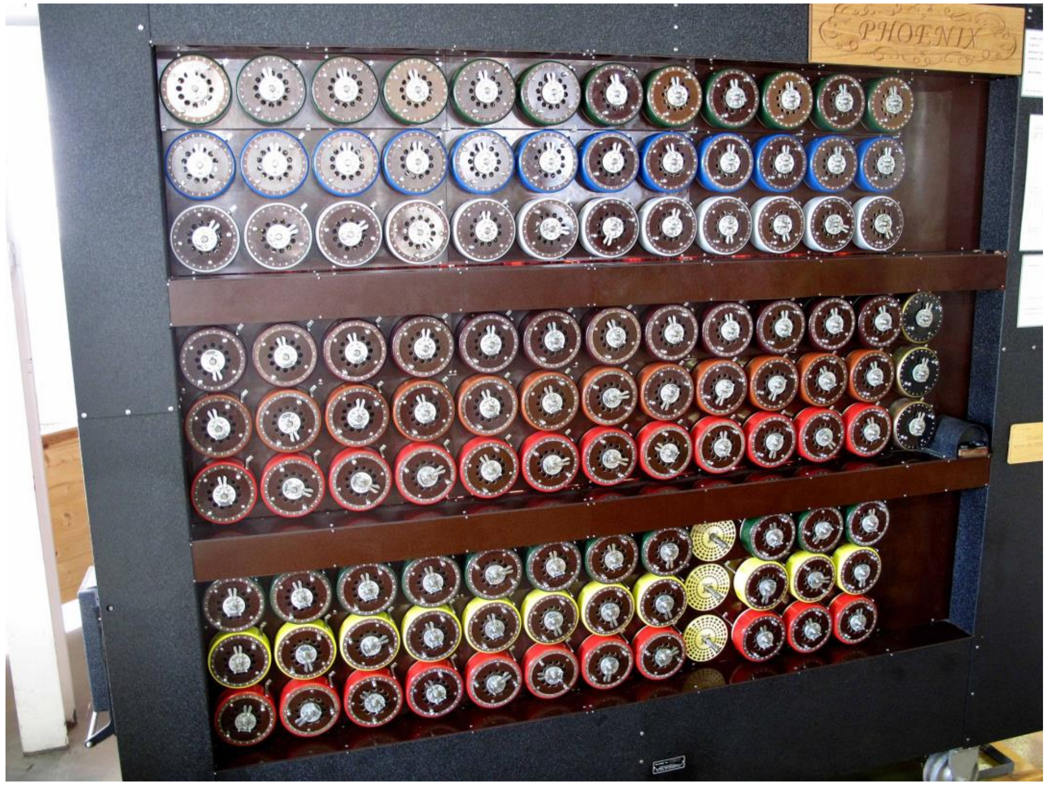


Image from Wikipedia: https://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma

Breaking Cryptography

Other methods:

- ***Brute Force***: Simply attempt to use every possible key until you find the correct one
  - Also known as exhaustive key search
- ***Acquire the Key***: If you can obtain the key, there is no need to worry about trying to crack the algorithm
- Ways to acquire the key include:
  - Social engineer the key from somebody who knows it
  - Attack the system where the key resides in order to grab it
    - This was done with the Heartbleed attack
  - Attack the key generation system in order to acquire them as they are produced
  - Exploit a flaw in the software
- ***Attacking the algorithm***: Attack the algorithm to find flaws, or portions of the algorithm such as the random number generator

Cryptographic Hash Function

From Wikipedia:

- *A cryptographic hash function (CHF) is a mathematical algorithm that maps data of an arbitrary size (often called the "message") to a bit array of a fixed size (the "hash value", "hash", or "message digest"). It is a one-way function, that is, a function for which it is practically infeasible to invert or reverse the computation.[1] Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes. Cryptographic hash functions are a basic tool of modern cryptography.*
- *A cryptographic hash function must be deterministic, meaning that the same message always results in the same hash. Ideally it should also have the following properties:*
  - *it is quick to compute the hash value for any given message*
  - *it is infeasible to generate a message that yields a given hash value (i.e. to reverse the process that generated the given hash value)*
  - *it is infeasible to find two different messages with the same hash value*
  - *a small change to a message should change the hash value so extensively that a new hash value appears uncorrelated with the old hash value (avalanche effect)*
- https://en.wikipedia.org/wiki/Cryptographic_hash_function

Random Number Generators

Useful in cryptography – especially for stream ciphers

Truly random is hard to do, so we use pseudorandom number generators (PNGs)

- If the PNG "seed" is compromised, an attacker can generate all the keys

One method is a Linear Congruential Generator

- `Z`$_{i+1}$` = (aZ`$_i$` + b) % M`
- If a is 237 and b is 23, and M is 2^{32}, we would generate the table on the right
- However, if an attacker gets any of the values, they can calculate all the others

Key	LCG Output
Z_0	23,485
Z_1	5,565,968
Z_2	1,319,134,439
Z_3	3,397,216,754
Z_4	1,981,486,369
Z_5	1,460,834,212

Steganography

Steganography literally means “covered writing”

The practice of hiding a message in such a manner that its very existence is concealed

- Done by embedding the message in some medium such as a document, image, sound recording, or video
- Those who know the medium contains a message can extract it
- For those who don't know about it, the message will be completely invisible

Related concept is digital "watermarking"

Terrorist groups, including Osama bin Laden's network, have used popular websites to post messages to their agents

- They upload a pornographic image (one was named “blond bombshell”) to the website
- Thousands of users download it – unaware of the message it contained
- Even if law enforcement discovered the message, they would have no idea who the recipient was intended to be

Since the message wasn’t apparent, this was not encryption, but steganography

Steganography

There is a very long history of steganography

In the Histories of Herodotus (430 B.C.):

- Demaratus wanted to notify the Spartans that Xerxes planned to invade Greece. He had the wax scraped off of writing tablets, the message carved into the wood, then recovered with the wax. The message was thus hidden.
- Another very early method was to shave the head of a messenger, tattoo the message on his head, then let his hair grow back. The message would then be hidden
  - While not timely, travel was much slower 2500 years ago

Codes, invisible ink, microdots are other examples

Steganography

Two ways of hiding the message "Pershing sails from N.Y. June 1"

Example 1:

- President's embargo ruling should have immediate notice. Grave situation affecting international law, statement foreshadows ruin of many neutrals. Yellow journals unifying national excitement immensely.
- This example uses the first letter of each word to hide each letter of the message.
- It is pretty easy to spot.

Example 2:

- Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on byproducts, ejecting suets and vegetable oils.
- This example is a bit harder to spot initially, but is just about as easy to implement
- It simply uses the 2nd letter of each word.

Either message could be used in something like a paragraph in a newspaper or online forum

Steganography

The most common method used in steganography is to hide a message in a picture

Any data file can potentially be used if there is any "slack space“ or hidden fields

- For example, Microsoft Office documents can have author information, revision information, etc.
- You could set the author to be, “Starbucks at noon”

Steganography in images takes advantage of the coding schemes used to encode a picture

- For pictures, each pixel (picture element) is represented by 1 or more bytes
- If the least significant bit is used to encode the message, small variations in the picture may occur but the message will be hidden inside

A very small image can still have a lot of pixels to manipulate

- A 400 x 300 image (tiny) will have 120,000 pixels
  - In an 8-bit coding scheme (256 colors) 120,000 bits of coded message can be encrypted or ~15kb
  - An RGB scheme used with 3 bytes/pixel (one for each color RGB) provides even more space
- Modern images have far, far more pixels than that

Steganography

Read: Wikipedia: Steganography

- https://en.wikipedia.org/wiki/Steganography
- You will be responsible for understanding the general techniques
- Pay close attention to the “Yellow Dot Code” (printer steganography)
- The Yellow Dot Code ***may*** have been used to identify Reality Winner, an NSA contractor, who stole and shared classified documents
  - https://arstechnica.com/information-technology/2017/06/how-a-few-yellow-dots-burned-the-intercepts-nsa-leaker/

There are many online tools that allow you to hide a message in an image

- One tool: https://manytools.org/hacker-tools/steganography-encode-text-into-image/
- You could try using a solid color image to see if you can detect the encoded message