

Unit #2a

MIS5214

Cryptography

Agenda

- Cryptography terminology
- Symmetric Key Cryptography
 - Symmetric stream cryptography
 - Symmetric block cryptography
- Key sharing problem
- Public Key Cryptography
 - Diffie-Hellman algorithm: symmetric key generation through asynchronous cryptography
 - RSA algorithm
- Hybrid-Cryptography
 - Perfect Forward Secrecy
- Where do cryptographic controls go in the FedRAMP System Security Plan
- *If we have time: Brief review of Hashing & Digital Signatures*

Services of cryptosystems

- **Confidentiality** – Renders information unintelligible except by authorized entities
- **Authentication** – Verifies the identity of the user or system that created, requested or provided the information
- **Nonrepudiation** – Ensure the sender cannot deny sending the information
- **Integrity** – Data has not been altered in an unauthorized manner since it was created, transmitted, or stored

Key Cryptographic Terms and Their Relationships

- 1. **Plaintext** – The original, readable data or message before encryption.
- 2. **Algorithm** – A set of rules or mathematical functions used to encrypt and decrypt data.
- 3. **Cipher** – The specific method or algorithm used to transform plaintext into ciphertext.
- 4. **Encryption** – The process of converting plaintext into ciphertext using a key and a cipher.
- 5. **Decryption** – The process of converting ciphertext back into plaintext using a key and a cipher.
- 6. **Cryptanalysis** – The study and practice of analyzing ciphers and encrypted data to break the encryption or discover weaknesses.

Text-Based Relationship and Explanation

- 1. The sender has a **plaintext** message.
- 2. The plaintext is passed through an **encryption algorithm** with a **key**.
- 3. The encryption algorithm produces **ciphertext** (unreadable data).
- 4. The receiver uses a **decryption algorithm** with a **key** to convert ciphertext back into plaintext.
- 5. **Cryptanalysis** is performed by attackers trying to break the encryption.

Visualizing the Relationship

Process	Input	Operation	Output
Plaintext	"Hello World"	Input message	"Hello World"
Encryption	"Hello World" + Key	Apply encryption algorithm	"Xj2@&Lm#5!"
Ciphertext	"Xj2@&Lm#5!"	Data in transit (secure)	"Xj2@&Lm#5!"
Decryption	"Xj2@&Lm#5!" + Key	Apply decryption algorithm	"Hello World"
Cryptanalysis	"Xj2@&Lm#5!"	Attack to find plaintext or key	Possible key discovery

Attack Scenario (Cryptanalysis)

- If an attacker intercepts the ciphertext (Xj2@&Lm#5!), they will attempt to reverse the encryption process using **cryptanalysis** techniques such as brute force, frequency analysis, or differential cryptanalysis.

Visualizing the Relationship

Process	Input	Operation	Output
Plaintext	"Hello World"	Input message	"Hello World"
Encryption	"Hello World" + Key	Apply encryption algorithm	"Xj2@&Lm#5! "
Ciphertext	"Xj2@&Lm#5! "	Data in transit (secure)	"Xj2@&Lm#5! "
Decryption	"Xj2@&Lm#5! " + Key	Apply decryption algorithm	"Hello World"
Cryptanalysis	"Xj2@&Lm#5! "	Attack to find plaintext or key	Possible key discovery

This model represents how cryptography ensures secure communication and how cryptanalysis tries to break the encryption.

Cipher = encryption algorithm

2 main attributes combined in a cypher

Confusion:

- Makes the relationship between the plaintext, ciphertext, and key as complex as possible.
- Prevents attackers from deducing the key from ciphertext.
- **Typically implemented using substitution ciphers** (e.g., replacing one character with another).
- **Example:** AES uses a substitution-permutation network where confusion is introduced using the **SubBytes** step (S-Box transformation).

Diffusion:

- Spreads out the influence of each plaintext bit across multiple ciphertext bits.
- Ensures that altering a single bit of plaintext affects multiple ciphertext bits, making statistical attacks difficult.
- **Typically implemented using transposition ciphers** (e.g., rearranging characters).
- **Example:** AES uses the **MixColumns** step to distribute bits across the block.

Substitution Method Using Bit Shifting

Keyspace is the number of possible keys

Key Entry: Caesar / ROT-13

Description
Here you can enter the key for the Caesar cipher.
Caesar is a mono-alphabetic substitution, where the characters of the cleartext alphabet are mapped to the ciphertext alphabet by shifting. This shifting value is the key. You can enter the key as a number or as a single character of the alphabet.
Rot-13 is a special variant, where the key has the fixed value of half the length of the cleartext alphabet. This variant is only selectable if the length of the alphabet is an even number.

Select variant
☒ Caesar
☐ Rot-13

Options to interpret the alphabet characters
☒ Value of the first alphabet character = 0 (e.g. "A"=0)
☐ Value of the first alphabet character = 1 (e.g. "A"=1)

Key entry as
☐ Alphabet character
☒ Number value

Properties of the chosen encryption
Shift of 4
Mapping of the alphabet (26 characters)
from: ABCDEFGHIJKLMNOPQRSTUVWXYZ
to: EFGHIJKLMNOPQRSTUVWXYZABCD

Encrypt Decrypt Text options Cancel

Substitution Cipher Example (Caesar Cipher)

•The **Caesar Cipher** is a simple substitution cipher where each letter in the plaintext is shifted by a fixed number of places in the alphabet.

•Example:

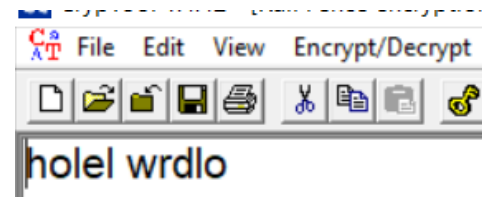
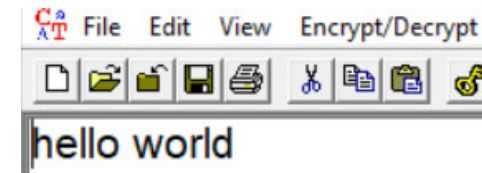
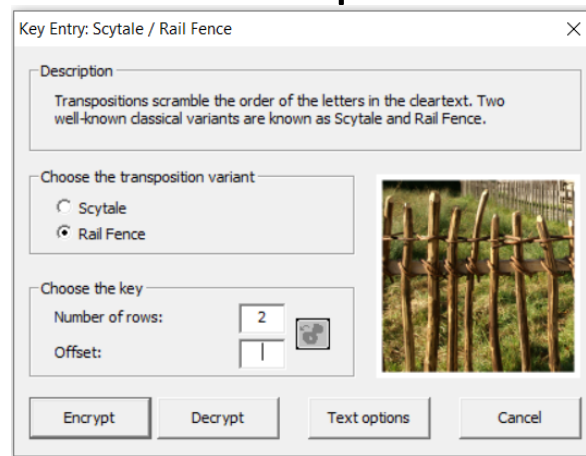
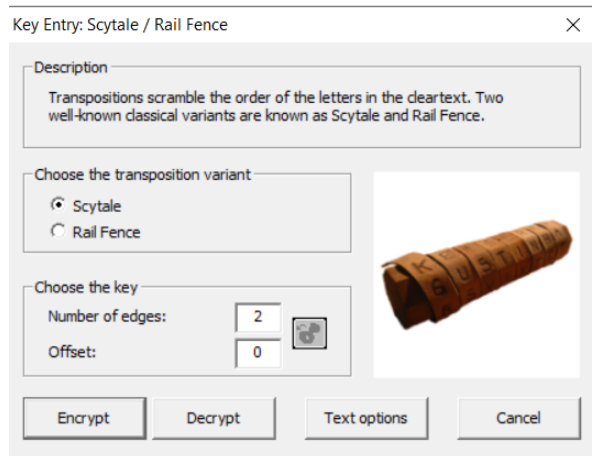
- Plaintext: **HELLO**
- Shift: **3**
- Ciphertext: **KHOOR**
(H → K, E → H, L → O, L → O, O → R)

Task 1: Encrypt a Message using Caesar Cipher

- 1.Choose a plaintext message: "SECURITY"
- 2.Use a shift of **3**.
- 3.Convert each letter by shifting **forward** in the alphabet.
- 4.Write down the ciphertext.

Expected Answer:

Transposition



Writing the Text in the Zigzag Order

We move down the rows, then up in a repeating sequence:

Start at the top rail (Row 1) → Place "H".
Move down to Row 2 → Place "E".
Move down to Row 3 → Place "L".
Reverse direction (start moving up) to Row 2 → Place "L".
Move up to Row 1 → Place "O".
Move down to Row 2 → Place "W".
Move down to Row 3 → Place "O".
Move up to Row 2 → Place "R".
Move up to Row 1 → Place "I".
Move down to Row 2 → Place "D".

Task 3

Encrypt "HACKINGTOOLS" using **Rail Fence (Depth 2)**.

Linguistic cryptanalysis uses knowledge of the English language, for example: Remember the prior scenario

Attack Scenario (Cryptanalysis)

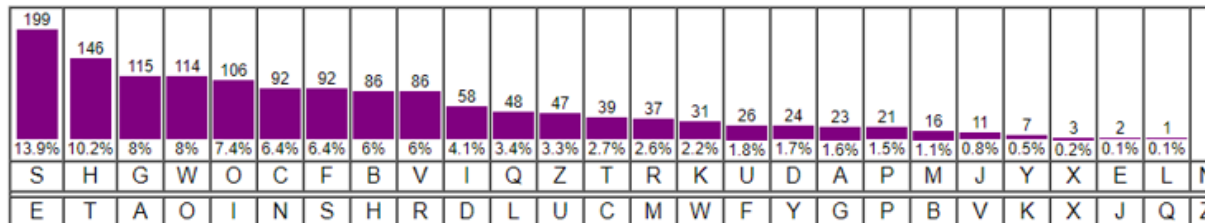
- If an attacker intercepts the ciphertext (Xj2@&Lm#5!), they will attempt to reverse the encryption process using **cryptanalysis techniques** such as brute force, frequency analysis, or differential cryptanalysis.

Common English Letter Frequencies:

- The most common letter in English is E, followed by T, A, O, I, N, S, R.

Frequency analysis recognizes that different letters have different probabilities of frequencies of use in words

What is N-grams



The second row of letters represent the frequency of letters in plaintext form.

Order by: Replace with

HVS QOSGOF QWDVSF WG BOASR OTHSF XIZWIG
QOSGOF, KVC, OQOCFRWBU HC GISHCBWIG, IGSR WH
KWHV O GWNTH CT HVFSS HC DFCHSQH ASGGOUSG CT
AWZWHOFM GWUBHTWQOBQS. KVVZS QOSGOF'G KOG
HVS TWFGH FSQCFSR IGSR CT HVMG GOVSAS, CHVSF
GIPGHWHIHWCB QWDVSFG OFS YBCKB HC VOJS PSSB
IGSR SOFZWSF.

WT VS VOR OBHVMWBU QCBTWSBHMZOZ HC GOM, VS
KFCHS WH WB QWDVSF, HVOH WG, PM GC QVOBUNBU
HVS CFRSF CT HVS ZSHHSFG CT HVS OZDVOPSH,
HVOH BCH O KCFR QCIZR PS AORS CIH. WT OBHMCBS
KWGVSG HC RSQWDVSF HVS GS, OBR USH OH HVS NF

HVS QOSGOF QWDVSF WG BOASR OTHSF XIZWIG
QOSGOF, KVC, OQOCFRWBU HC GISHCBWIG, IGSR WH
KWHV O GWNTH CT HVFSS HC DFCHSQH ASGGOUSG CT
AWZWHOFM GWUBHTWQOBQS. KVVZS QOSGOF'G KOG
HVS TWFGH FSQCFSR IGSR CT HVMG GOVSAS, CHVSF
GIPGHWHIHWCB QWDVSFG OFS YBCKB HC VOJS PSSB
IGSR SOFZWSF.

WT VS VOR OBHVMWBU QCBTWSBHMZOZ HC GOM, VS
KFCHS WH WB QWDVSF, HVOH WG, PM GC QVOBUNBU
HVS CFRSF CT HVS ZSHHSFG CT HVS OZDVOPSH,
HVOH BCH O KCFR QCIZR PS AORS CIH. WT OBHMCBS
KWGVSG HC RSQWDVSF HVS GS, OBR USH OH HVS NF

Polyalphabetic Cipher

Vigenère Cipher (Polyalphabetic Substitution)

Description: Uses multiple shifting alphabets based on a keyword.

Example (Keyword = KEY):

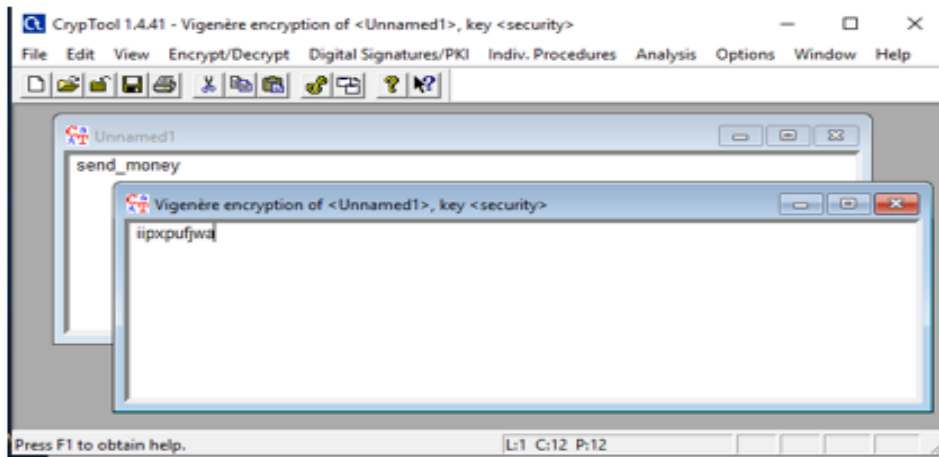
- **Plaintext:** HELLO
- **Key Repeated:** KEYKE
- **Ciphertext:** RIJVS

Ciphers can be made stronger, and frequency analysis made more difficult when more than one cipher alphabet is used

- For example, encrypt the plaintext message “SEND MONEY”
 - Using the word “SECURITY” as the key, but repeat its use in the key to make it have as many letters as the plaintext:

Plaintext: SEND MONEY (10 characters including the space “_”)

Key: SECURITYSE (10 characters)



Use the following to understand how to crack the code:

<https://www.youtube.com/watch?v=E352JJ8xv48>

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	-	.
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

Key = Security

s	e	c	u	r	i	t	Y
18	4	2	20	17	8	19	24

Encrypt

Increment Value	18	4	2	20	17	8	19	24	18	4
Plain Text	s	e	n	d	_	m	o	n	e	Y
Cipher Value	i	i	p	x	p	u	f	j	w	a

Random Polyalphabetic Cipher

What if we use a random polyalphabetic key that is as long as the message?

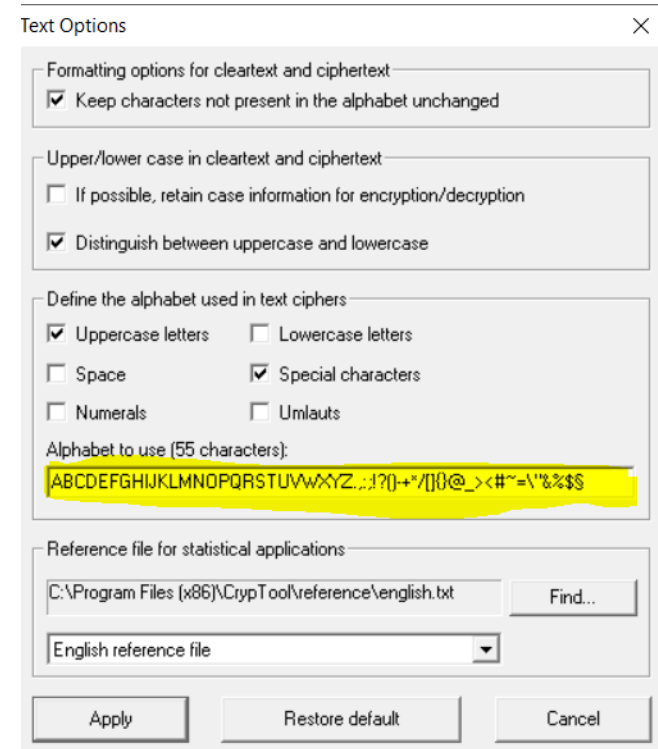
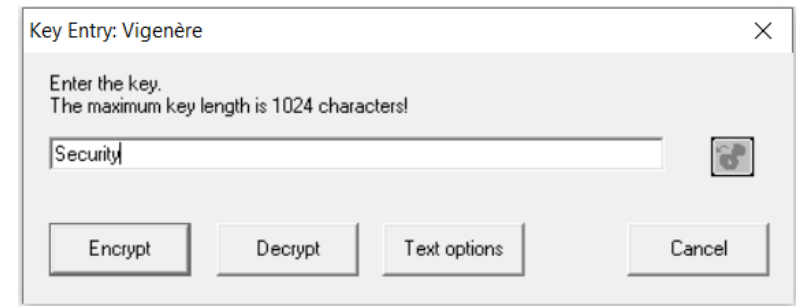
For example, let's say our plaintext is:

***We intend to begin on the first of February
unrestricted submarine warfare.***

And the polyalphabetic key is a string of random characters as long as the message:

**ackwulsjwkblogbzcukn.kqubpnnefjvcebuy
maclzvzmzwfbxpmmzqwmm.tejzfutjcqrsf_
hq**

How could an attacker attempt to crack this message?
Is an attack possible?



Translating what we type, into ASCII, and then into binary... which is what is sent as data packets across the network to other computers...

Binary – Decimal

0 0 0 0 0 0 0 0 = 0
 1 1 1 1 1 1 1 1 = 255

8 bits supports 256 numbers

2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
128	64	32	16	8	4	2	1



ASCII - Decimal

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	\$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e

ASCII Character Table

Name	Hex	Dec
. (period)	2E	046
0	30	048
1	31	049
2	32	050
3	33	051
4	34	052
5	35	053
6	36	054
7	37	055
8	38	056
9	39	057

Name	Hex	Dec
A	41	065
B	42	066
C	43	067
D	44	068
E	45	069
F	46	070
G	47	071
H	48	072
I	49	073
J	4A	074
K	4B	075

Name	Hex	Dec
L	4C	076
M	4D	077
N	4E	078
O	4F	079
P	50	080
Q	51	081
R	52	082
S	53	083
T	54	084
U	55	085
V	56	086

Name	Hex	Dec
W	57	087
X	58	088
Y	59	089
Z	5A	090

XOR – Exclusive OR

Creating “confusion” (i.e. substitution) through a binary mathematical function called “exclusive OR”, abbreviated as XOR

Step 1: Convert "HELLO" into Binary

To encrypt "HELLO", we first convert each letter into its **ASCII binary representation**.

Character	ASCII (Decimal)	Binary Equivalent
H	72	01001000
E	69	01000101
L	76	01001100
L	76	01001100
O	79	01001111

Step 2: Choose a Key and Convert It to Binary

- Let's use a **1-byte key**: "K"
- "K" has an **ASCII decimal value of 75**, which in binary is:

K = 75 → Binary: `01001011`

Step 3: Apply XOR Operation

We apply XOR (\oplus) between each character's binary and the key's binary (01001011).

Character	Binary (ASCII)	XOR Key κ (01001011)	XOR Result (Cipher)
H (72)	01001000	01001011	00000011 (Decimal 3 → ETX)
E (69)	01000101	01001011	00001110 (Decimal 14 → SO)
L (76)	01001100	01001011	00000111 (Decimal 7 → BEL)
L (76)	01001100	01001011	00000111 (Decimal 7 → BEL)
O (79)	01001111	01001011	00000100 (Decimal 4 → EOT)

Ciphertext (Encrypted Data) in Binary:

00000011 00001110 00000111 00000111 00000100

Agenda

- ✓ Cryptography terminology
- Symmetric Key Cryptography
 - Symmetric stream cryptography
 - Symmetric block cryptography
- Key sharing problem
- Public Key Cryptography
 - Diffie-Hellman algorithm: symmetric key generation through asynchronous cryptography
 - RSA algorithm
- Hybrid-Cryptography
 - Perfect Forward Secrecy
- Where do cryptographic controls go in the FedRAMP System Security Plan
- *If we have time: Brief review of Hashing & Digital Signatures*

Symmetric versus asymmetric algorithms

- Symmetric cryptography
 - Use a copied pair of symmetric (identical) secret keys
 - The sender and the receiver use the same key for encryption and decryption functions
 - Confidentiality, but no integrity, authentication nor non-repudiation
- Asymmetric cryptography
 - Also known as “public key cryptography”
 - Use different (“asymmetric”) keys for encryption and decryption
 - One is called the “private key” and the other is the “public key”
 - Confidentiality, but also want authenticity and non-repudiation

Symmetric cryptography

Strengths:

- Much faster (less computationally intensive) than asymmetric systems.
- Hard to break if using a large key size.

Symmetric cryptography is 1,000 times faster than Asymmetric cryptography

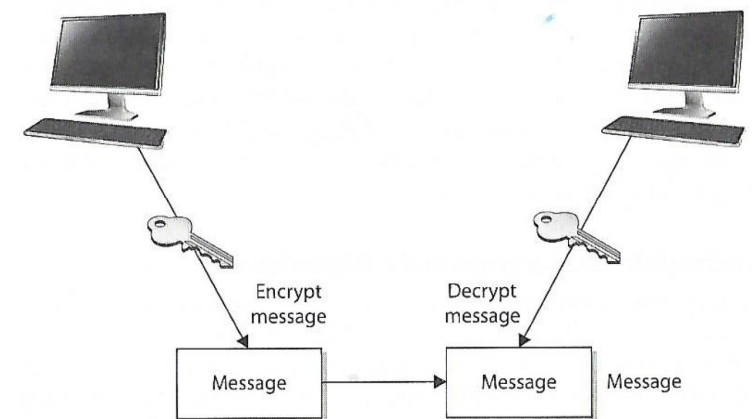
Weaknesses:

- Requires a secure mechanism to deliver keys properly.
- Each pair of users needs a unique key, so as the number of individuals increases, so does the number of keys, possibly making key management overwhelming.
- Provides confidentiality but not authenticity or nonrepudiation.

Two types: Stream and Block Ciphers

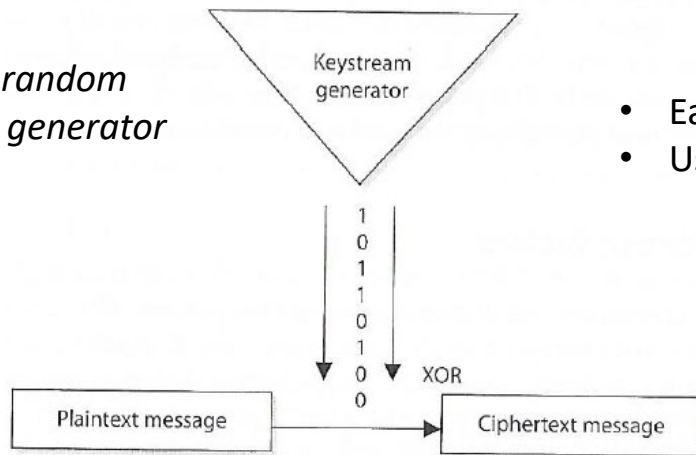
- **Stream Ciphers** treat the message a stream of bits and performs mathematical functions on each bit individually
- **Block Ciphers** divide a message into blocks of bits and transforms the blocks one at a time

Symmetric encryption uses the same keys.



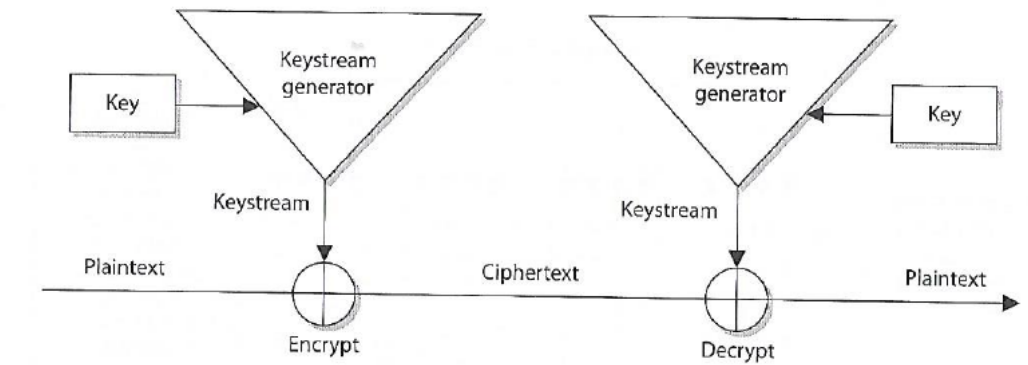
Symmetric Stream Ciphers

*Pseudo-random
number generator
(PRNG)*



- Easy to implement in hardware
- Used in cell phones and Voice Over Internet Protocol

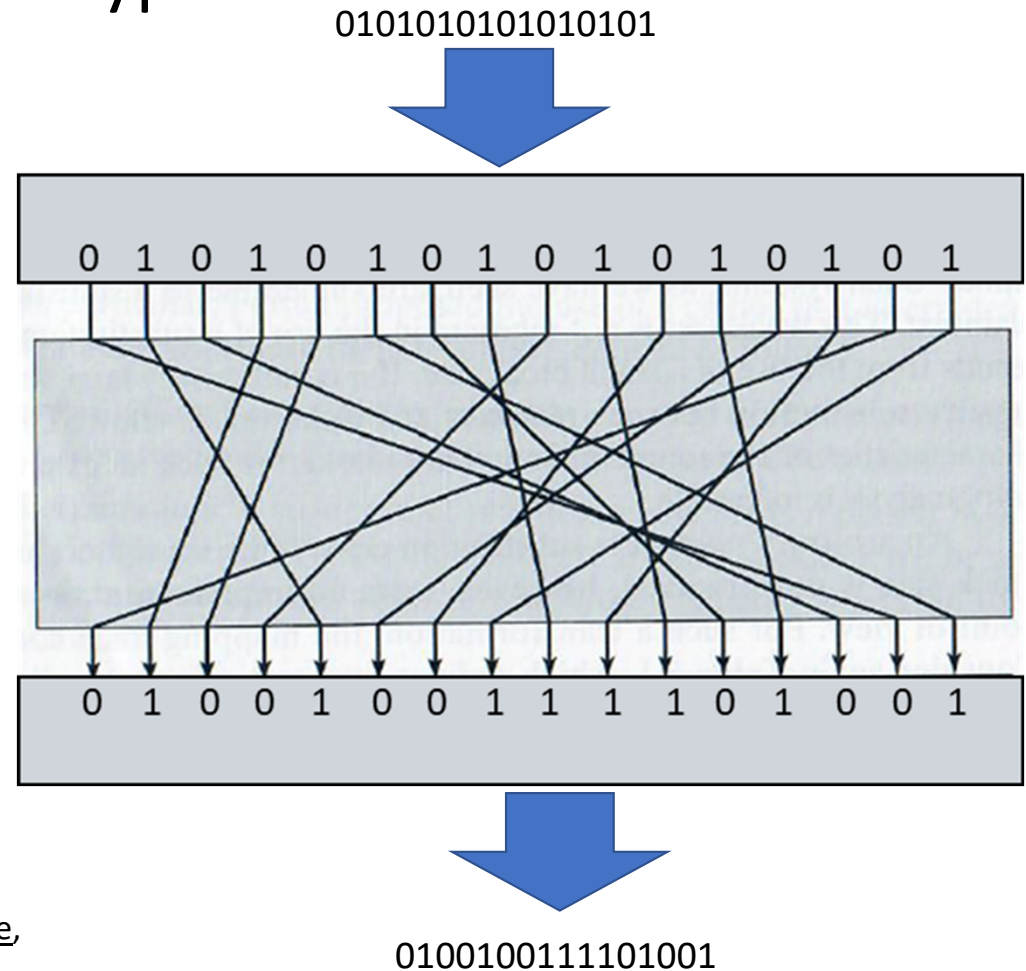
*Wi-Fi access points (like the one on
the classroom ceiling) and cell phone
use stream ciphers to
encrypt/decrypt data they send and
receive*



The sender and receiver must have the same key to generate the same keystream.

2 main attributes combined in a cypher

1. Confusion: usually carried out through substitution
2. **Diffusion:** Usually carried out through transposition



Columnar Transposition Cipher

Description: The message is written into a grid column-wise and then read in a different order.

Example (Key = 3124):

•**Plaintext:** HELLO WORLD

Write in Columns (Key = 3124)

We assume **4 columns** based on the key length (3124):

1st Column	2nd Column	3rd Column	4th Column
H	E	L	L
O	W	O	R
L	D		

Step 2: Reordering the Columns Based on Key

The key = 3124 means that columns will be read in this order:

Key Order	Column Number
3	1st Column
1	2nd Column
2	3rd Column
4	4th Column

Thus, the **columns will be rearranged** as follows:

1. 3rd column (L O) → 1st in order
2. 1st column (H O L) → 2nd in order
3. 2nd column (E W D) → 3rd in order
4. 4th column (L R) → 4th in order

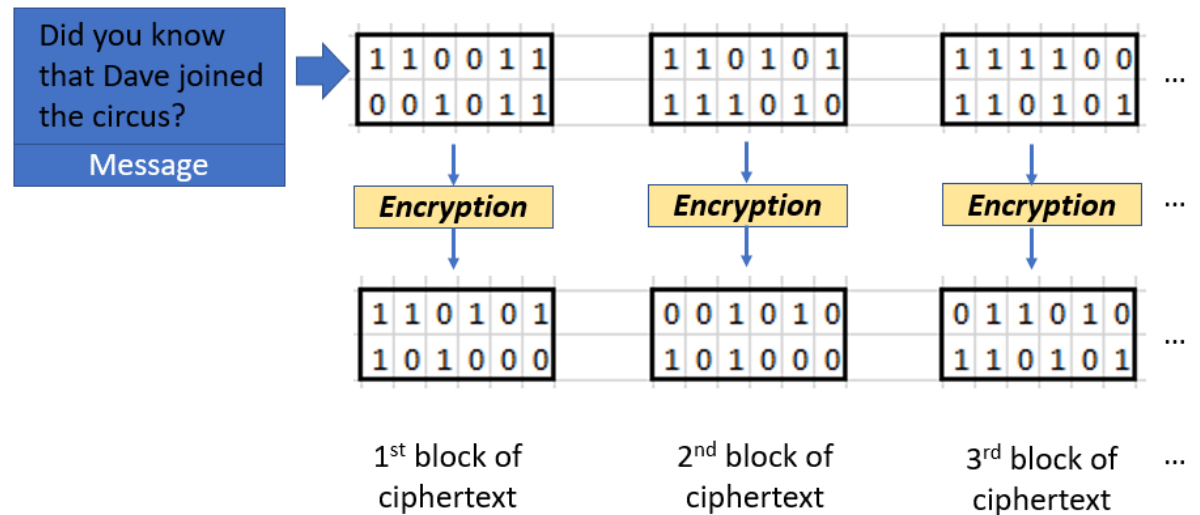
Ciphertext: "LOHOWLDERL"

Block Cyphers (“Cipher”)

- Message is divided into blocks of bits
- Blocks are put through encryption functions 1 block at a time

Suppose you are encrypting a 640-bit long message to send using a block cypher that uses 64 bits

- Your message would be chopped up into 10 blocks each 64 bits long
- Each block, in turn, would be run through a series of encryption functions (substitution and transposition)
- Ending up with 10 blocks of ciphertext

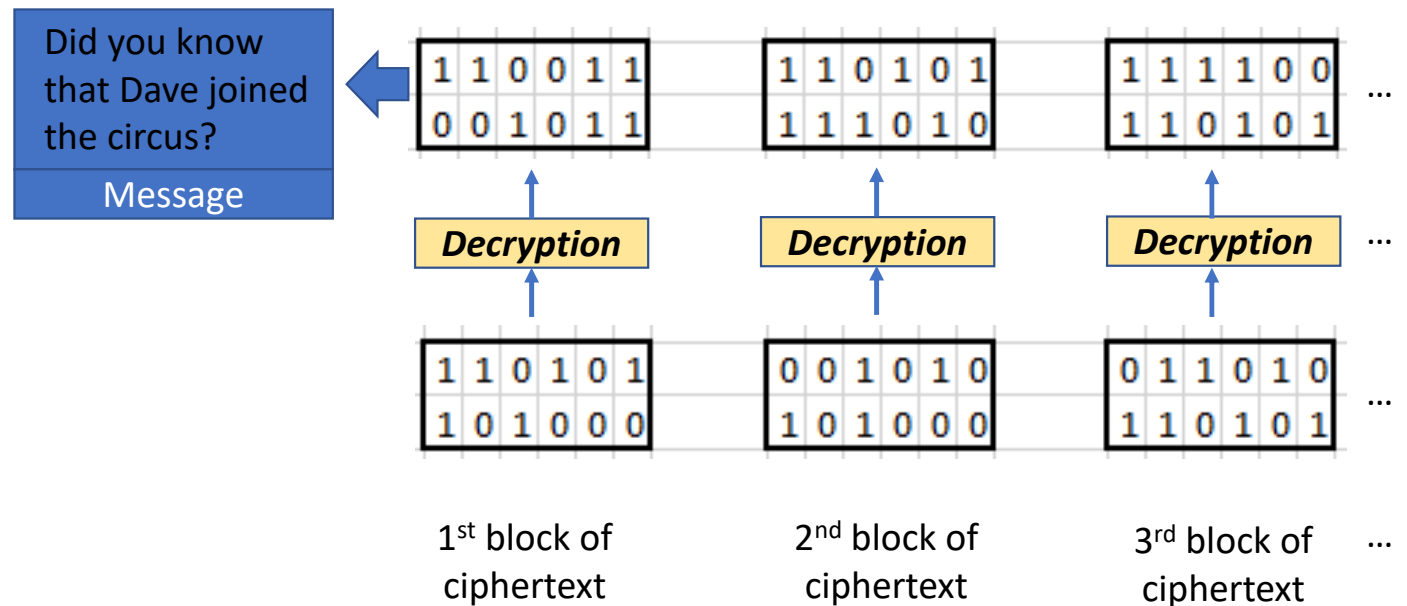


Block Ciphers

- Message is divided into blocks of bits
- Blocks are put through mathematical functions 1 block at a time

You send the message. Receiver uses the same block cipher and key (symmetric) to decipher the message

- The 10 ciphertext blocks go back through the algorithm in the reverse sequence
- Resulting in original plaintext message



Block Ciphers versus Stream Ciphers

In contrast, block ciphers encrypt a block of bits at a time

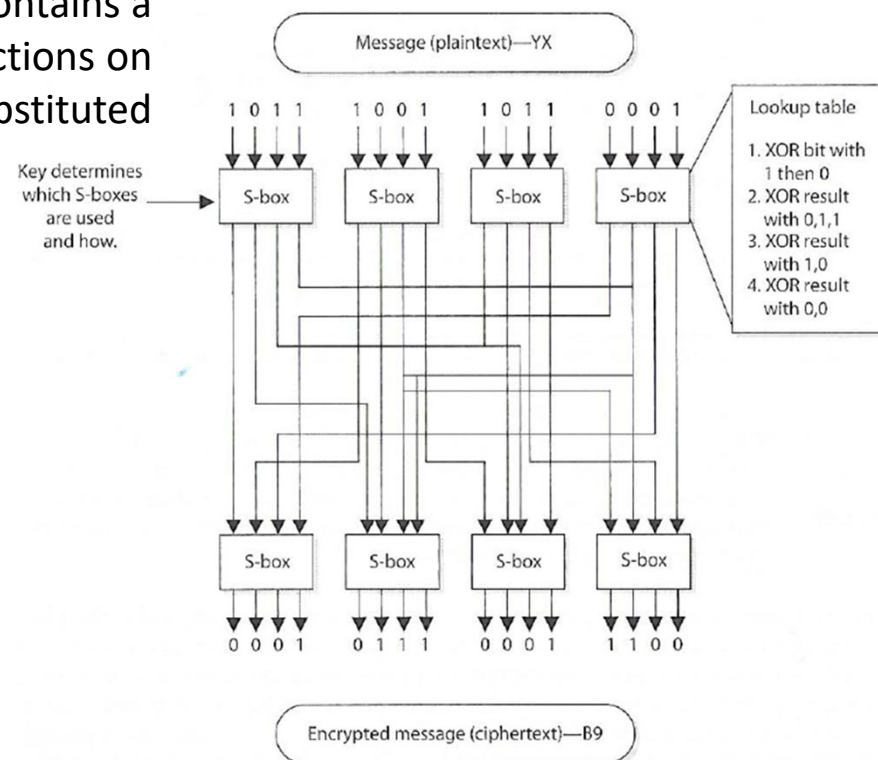
In this example, each Substitution Box (S-box) contains a lookup table used by the algorithm as instructions on how the bits are substituted

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Encryption table

Ciphertext	Plaintext
0000	1110
0001	0101
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Decryption table



...followed by transposition...

Block Cipher Modes of Operation

- Block ciphers (e.g., AES, DES, 3DES) encrypt data in fixed-size blocks (e.g., 128-bit blocks for AES). However, real-world applications require flexibility, so modes of operation modify how blocks are processed.

The five major modes are:

- 1.ECB – Electronic Code Book
- 2.CBC – Cipher Block Chaining
- 3.CFB – Cipher Feedback
- 4.OFB – Output Feedback
- 5.CTR – Counter Mode

ECB – Electronic Code Book mode

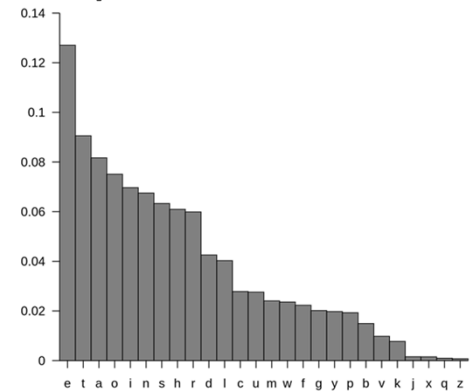
- A data block of a certain size (e.g. 64 bits or 128 bits or...) is entered into the algorithm with the key, and a block of cipher text is produced

$$C_i = \text{Encrypt}(\text{Key}, P_i)$$

for $i = 1, \dots, k$

Where:

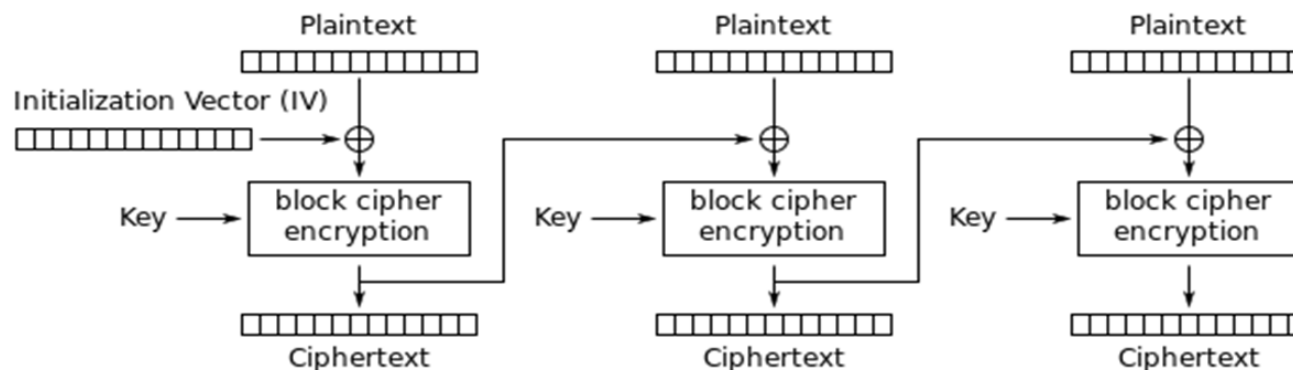
- C_i is block i of ciphertext
- P_i is a block of plaintext



- Encrypts every block the same way every time for a given key
- Why is this a problem?
 - This is a problem because **frequency analysis** of the encrypted text can reveal a lot of information
 - Not enough randomness

CBC – Cipher Block Chaining mode

- Is much more secure
- Does not reveal a pattern of encryption for frequency analysis
- Each block of text, the key, and the value based on the previous block are processed in the algorithm and applied to the next block of text



- XORs a plaintext with the **last** encrypted block before encrypting it. This ensures that the same plaintext is encrypted differently every time.
- Requires an initialization vector (or IV) to get started, since the first block doesn't have a previous encrypted block to XOR against.

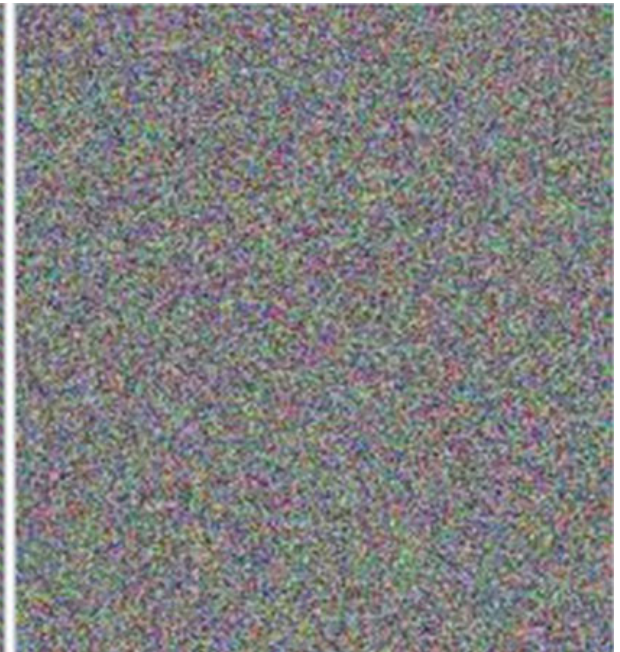


Original Image



Block cipher with ECB
(Electronic Code Book)
encryption

Not good!



Block cipher with CBC
(Cipher Block Chaining) or any
of the other modes of
encryption

These are good!

Cryptanalysis Attacks

- Brute force
 - Trying all key values in the keyspace
- Frequency Analysis
 - Guess values based on frequency of occurrence
- Dictionary Attack
 - Find plaintext based on common words
- Known Plaintext
 - Format or content of plaintext available
- Chosen Plaintext
 - Attack can encrypt chosen plaintext
- Chosen Ciphertext
 - Decrypt known ciphertext to discover key
- Random Number Generator (RNG) Attack
 - Predict initialization vector used by an algorithm
- Social Engineering
 - Humans are the weakest link

Modern Block Ciphers

Use block sizes of 128-bits or greater

- Examples of Block Ciphers that can be used are:

- AES
- Blowfish
- Twofish
- Serpent

Do not use these examples of block ciphers which use 64-bit blocks, which are too small to be secure include:

- DES
- 3DES

Key Entry: DES (ECB)

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 64 bits (effectively 56 bits)

00 00 00 00 00 00 00 00

Key Entry: DES (CBC)

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 64 bits (effectively 56 bits)

00 00 00 00 00 00 00 00

Key Entry: Triple DES (CBC)

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 128 bits (effectively 112 bits)

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Key Entry: Rijndael (AES)

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 128 bits

128 bits
192 bits
256 bits

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Key Entry: Twofish

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 128 bits

128 bits
192 bits
256 bits

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Key Entry: Serpent

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 128 bits

128 bits
192 bits
256 bits

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

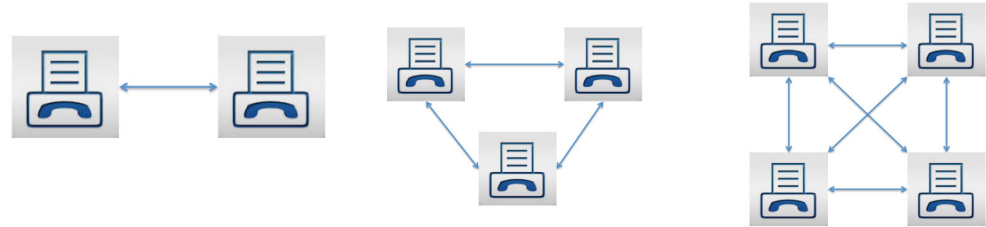
DES Cracker:

- A DES key search machine
- Contains 1,536 chips
- Cost: \$250,000
- Searches 88 billion keys per second
- Won RSA Laboratory's "**DES Challenge II-2**" by successfully finding a DES key in 56 hours

Agenda

- ✓ Cryptography terminology
- ✓ Symmetric Key Cryptography
 - ✓ Symmetric stream cryptography
 - ✓ Symmetric block cryptography
- Key sharing problem
- Public Key Cryptography
 - Diffie-Hellman algorithm: symmetric key generation through asynchronous cryptography
 - RSA algorithm
- Hybrid-Cryptography
 - Perfect Forward Secrecy
- Where do cryptographic controls go in the FedRAMP System Security Plan
- *If we have time: Brief review of Hashing & Digital Signatures*

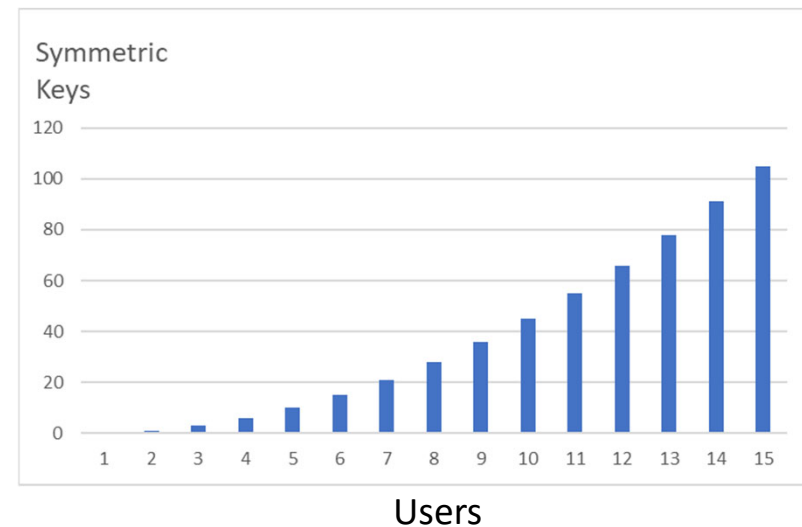
Key sharing problem



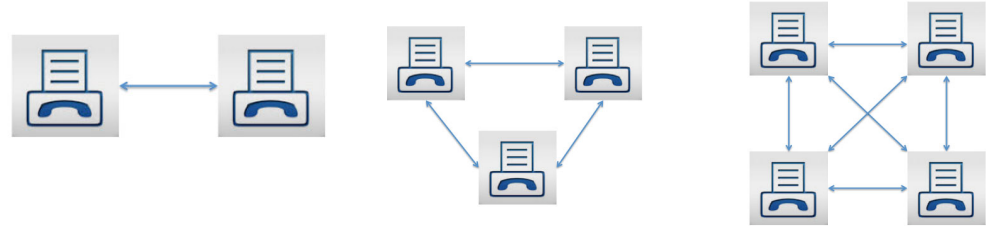
Sharing cryptographic keys has been a problem throughout history

- The number of pairs of keys (“secure network connections”) grows at a near exponential rate (i.e. geometric rate) as the number of users increases

Users	Symmetric Keys
1	0
2	1
3	3
4	6
5	10
6	15
7	21
8	28
9	36
10	45
11	55
12	66
13	78
14	91
15	105
...	...



Key sharing problem



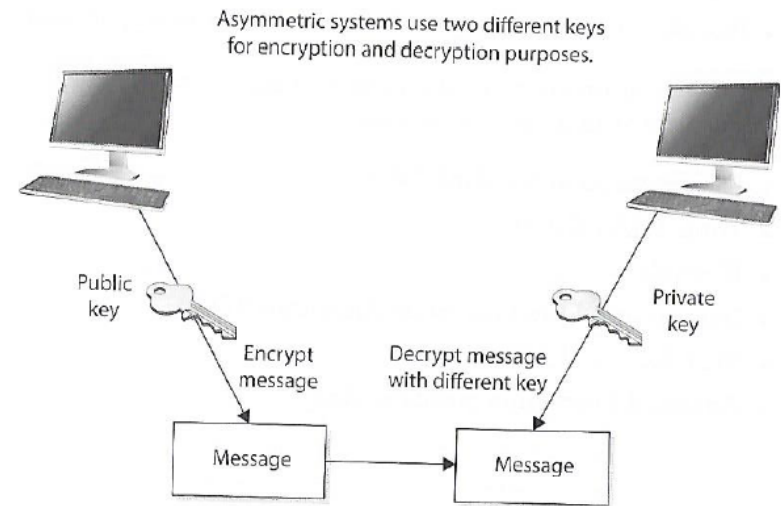
- The number of pairs of keys needed for “n” users is determined by an equation known as [Metcalf’s Law](#)
- Number of key pairs needed for n users = $(n*(n-1))/2$
 - *The reason for the n-1 is that you do not need to communicate with yourself*
- For 22 students how many keys would we need:

$$(22 * 21) / 2 = 231 \text{ keys}$$

Asymmetric cryptography

- **Public and Private** keys are mathematically related
 - Public keys are generated from private key
 - Private keys cannot be derived from the associated public key (if it falls into the wrong hands)
- **Public key** can be known by everyone
- **Private key** must be known and used only by the owner

Examples: **RSA (Rivest-Shamir-Adleman)**, **ECC (Elliptic Curve Cryptography)**, **Diffie-Hellman**, **DSA (Digital Signature Algorithm)**.



Asymmetric cryptography is computationally intensive and much slower (1,000 times slower) than symmetric cryptography

Diffie-Hellman

The Diffie-Hellman (DH) algorithm is a cryptographic protocol used to establish a shared secret key between two parties over an insecure channel without transmitting the key itself. It is primarily used for secure key exchange rather than encryption or decryption of data

- Diffie-Hellman is an asymmetric key exchange algorithm, meaning it does not encrypt or decrypt messages but securely generates a shared secret key.
- Once the shared key is established, symmetric encryption (e.g., AES, DES) can be used for secure communication.
- It is based on modular arithmetic and discrete logarithms, making it computationally difficult to reverse-engineer the key.
- Alone, it does not provide authentication, integrity, or non-repudiation, but these can be added with digital signatures and hashing.
- Used in TLS, VPNs, encrypted messaging, and secure communications.
- Vulnerable to MITM attacks without authentication.
- Elliptic Curve Diffie-Hellman (ECDH) is a more secure modern variant

Diffie-Hellman Algorithm: *Secret symmetric key derivation through public key sharing*

Assumptions:

A prime number is a positive whole number whose only factors (i.e. integer divisors) are 1 and the number itself (e.g. 2, 3, 5, 7, 11, 13, 17, 19, 23, ...). Bob & Alice want to compute a shared secret key to protect confidentiality of their conversation. Eve eavesdrops...

Algorithm:

- Alice and Bob agree on public values: A large prime number (p) and a base (g).
 - Example: $p = 23$, $g = 5$
- Each selects a private key:
 - Alice picks $a = 6$, Bob picks $b = 15$.
- Each computes and exchanges public keys:
 - Alice sends $A = g^a \bmod p = 5^6 \bmod 23 = 8$
 - Bob sends $B = g^b \bmod p = 5^{15} \bmod 23 = 19$
- Each computes the shared secret:
 - Alice computes $S = B^a \bmod p = 19^6 \bmod 23 = 2$
 - Bob computes $S = A^b \bmod p = 8^{15} \bmod 23 = 2$
- Now, both parties have the shared secret key $S = 2$, which they can use for secure communication.
- ***Eve cannot calculate Bob & Alice's shared symmetric secret key from their public keys, p and g alone – even though she knows they are using the Diffie-Hellman algorithm!***

let's use [WolframAlpha](https://www.wolframalpha.com/) online calculator

Diffie-Hellman Demonstration - Visualization of the Diffie-Hellman Key Exchange Protocol



Set public
parameters

Choose
secrets

Create
shared keys

Exchange
shared keys

Generate common
Session Key

Close

Public parameters:

Prime module p: 16272850338688309819305277492836853453982568378006455706374812013890

Generator g: 94080696475816174992882457680992982563522890618958310947886463671545

Alice

Secret

a: xx

Calculate

A: 325651096213018193878032

Calculate

S: 628539455234403405666395

Bob

Secret

b: x

Calculate

B: 649451625061931705121547

Calculate

S: 628539455234403405666395



Show introduction dialog

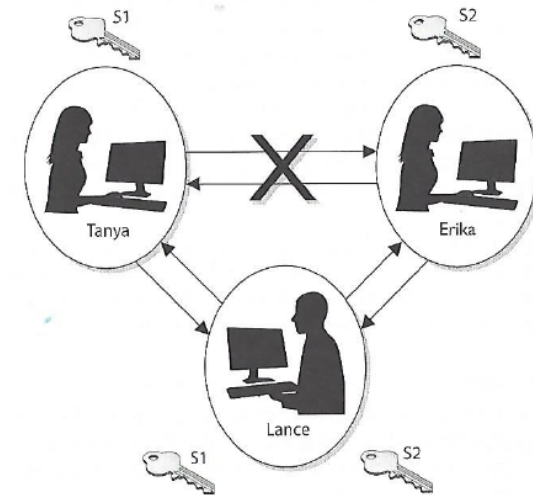


Show information dialogs



Diffie-Hellman was vulnerable to man-in-the-middle attack, because no authentication occurs before public keys are exchanged

1. Tanya sends her public key to Erika, but Lance grabs the key during transmission so it never makes it to Erika
2. Lance spoofs Tanya's identity and sends over his public key to Erika. Erika now thinks she has Tanya's public key
3. Erika sends her public key to Tanya, but Lance grabs the key during transmission so it never makes it to Tanya
4. Lance spoofs Erika's identity and sends over his public key to Tanya. Tanya now thinks she has Erika's public key
5. Tanya combines her private key and Lance's public key and creates a symmetric key S1
6. Lance combines his private key and Tanya's public key and creates symmetric key S1
7. Erika combines her private key and Lance's public key and creates symmetric key S2
8. Lance combines his private key and Erika's public key and creates symmetric key S2
9. Now Tanya and Lance share a symmetric key (S1) and Erika and Lance share a different symmetric key (S2). Tanya and Erika think they are sharing a key between themselves and do not realize Lance is involved
10. Tanya writes a message to Erika, and uses her symmetric key (S1) to encrypt the message, and sends it
11. Lance grabs the message and decrypts it with symmetric key S1, reads or modifies the message and re-encrypts it with symmetric key S2, and then sends it to Erika
12. Erika takes symmetric key S2 and uses it to decrypt and read the message....



Agenda

- ✓ Cryptography terminology
- ✓ Symmetric key cryptography
 - ✓ Symmetric stream cryptography
 - ✓ Symmetric block cryptography
- ✓ Key sharing problem
- Public Key Cryptography
 - Diffie-Hellman algorithm: symmetric key generation through asynchronous cryptography
 - RSA algorithm
- Hybrid-cryptography
 - Perfect Forward Secrecy
- Hashing revisited

Quick review

1. If a symmetric key is encrypted with a receiver's public key, what security service is provided?
 - **Confidentiality**: only the receiver's private key can be used to decrypt the symmetric key, and only the receiver should have access to this private key

Quick review

2. If data is encrypted with the sender's private key, what security services are provided?
 - **Authenticity** of the sender and nonrepudiation. If the receiver can decrypt the encrypted data with the sender's public key, then receiver knows the data was encrypted with the sender's private key

Quick review

3. Why do we encrypt the message with the symmetric key rather than the asymmetric key?
 - **Because the asymmetric key algorithm is too slow**



Leonard **A**dleman

Adi **S**hamir

Ron **R**ivest

RSA

Public Key algorithms

Public Key Algorithms, also known as asymmetric encryption algorithms, use two mathematically linked keys:

- **Public Key:** Shared openly and used for encryption or signature verification.
- **Private Key:** Kept secret and used for decryption or signing.

Common **public key algorithms**:

1. RSA (Rivest-Shamir-Adleman)
2. ECC (Elliptic Curve Cryptography)
3. DSA (Digital Signature Algorithm)
4. Diffie-Hellman Key Exchange
5. ElGamal Encryption

- Fundamental security elements in cryptosystems, applications and protocols
- Assure confidentiality, authenticity and non-repudiation of electronic communications and data storage
- Provide:
 - Key distribution and secrecy (e.g. Diffie–Hellman key exchange)
 - Digital signatures (e.g. Digital Signature Algorithm)
 - Both: key distribution and secrecy and digital signatures (e.g., RSA, ECC)

RSA Public Key Algorithm

RSA (Rivest-Shamir-Adleman) is a widely used asymmetric cryptographic algorithm that relies on the mathematical difficulty of factoring large prime numbers. It is primarily used for secure data transmission, encryption, and digital signatures. The RSA algorithm is based on a **public key** and a **private key**, where the public key is used for encryption and the private key is used for decryption.

- Most popular worldwide standard, that can be used for:
 - Asymmetric encryption/decryption
 - Key exchange (i.e. used to encrypt AES symmetric key)
 - Digital signatures
- In one direction, RSA provides:
 - Confidentiality through encryption
 - Authentication and non-repudiation through signature verification
- In the inverse direction, RSA provides:
 - Confidentiality through decryption
 - Authentication and non-repudiation through signature generation

RSA Public Key Algorithm

- Based on factoring large numbers into their prime numbers
 - A prime number is a positive whole number whose only factors (i.e. integer divisors) are 1 and the number itself
 - E.g. 2, 3, 5, 7, 11, 13, 17, 19, 23, ...
- Prime number factoring is
 - Easy when you know the result and one of the factors
 - $6,700,283 = 1889 * 3547$
 - Difficult when you do not know the factors, and the result is large
 - $6,700,283 = \text{prime1} * \text{prime2}$

RSA Demonstration

RSA using the private and public key -- or using only the public key

☒ Choose two prime numbers p and q. The composite number $N = pq$ is the public RSA modulus, and $\phi(N) = (p-1)(q-1)$ is the Euler totient. The public key e is freely chosen but must be coprime to the totient. The private key d is then calculated such that $d = e^{-1} \pmod{\phi(N)}$.

☐ For data encryption or certificate verification, you will only need the public RSA parameters: the modulus N and the public key e.

Prime number entry

Prime number p: 211

Prime number q: 233

Generate prime numbers...

RSA parameters

RSA modulus N: 49163 (public)

$\phi(N) = (p-1)(q-1)$: 48720 (secret)

Public key e: $2^{16}+1$

Private key d: 44273

Update parameters

RSA encryption using e / decryption using d (alphabet size: 256)

Input as: ☒ text ☐ numbers

Alphabet and number system options...

Input text

This is a Test of RSA Encryption

The Input text will be separated into segments of Size 1 (the symbol '#' is used as separator).

T # h # i # s # # # i # s # # a # # T # e # s # t # # o # f # # R # S # A # # E # n # c # r # y # p # t # i # f

Numbers input in base 10 format.

084 # 104 # 105 # 115 # 032 # 105 # 115 # 032 # 097 # 032 # 084 # 101 # 115 # 116 # 032 # 111 # 102 #

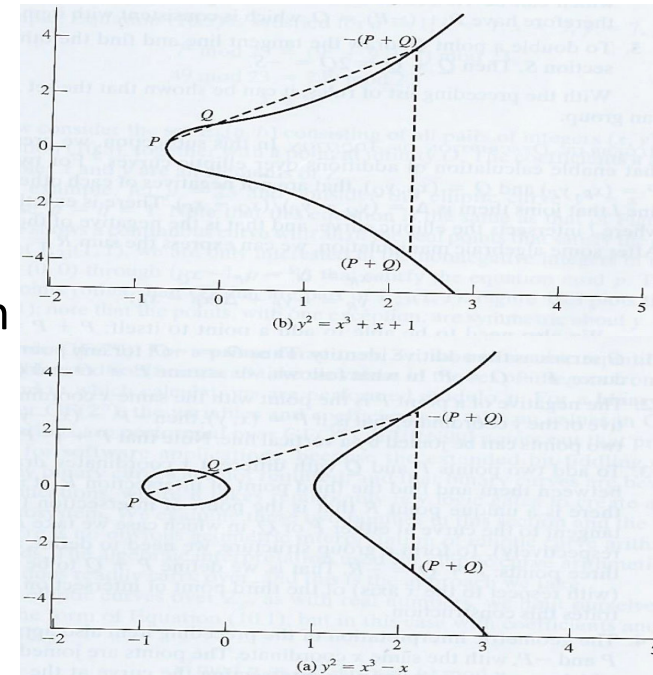
Encryption into ciphertext $c[i] = m[i]^e \pmod{N}$

00500 # 23366 # 20714 # 06205 # 09394 # 20714 # 06205 # 09394 # 18504 # 09394 # 00500 # 07428 # 01

Encrypt Decrypt Close

Elliptic-curve cryptography (ECC)

- Alternate approach to public-key cryptography based on algebraic structure of elliptic curves (based on Galois fields)
- Provides much the same security functionality as Diffie-Hellman and RSA:
 - Encryption/decryption (confidentiality)
 - Secure key distribution (authenticity, confidentiality)
 - Digital signatures (authenticity, non-repudiation)
- ECC's is much more efficient than RSA and the other asymmetric algorithms
 - Requires less bits and smaller keys than RSA for achieving the same level of security in its calculations and other algorithms
 - ECC's efficiency makes it very good for wireless devices and cellular phones with limited processing capacity, storage, power supply and bandwidth

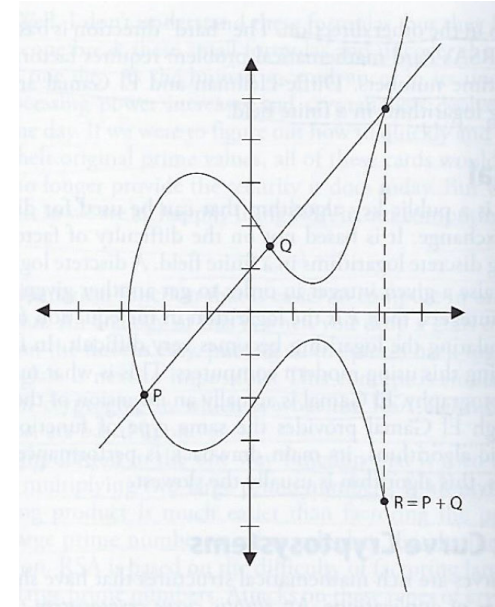


Examples of an elliptic curves

Elliptic-curve cryptography (ECC)

Elliptic-curve Diffie–Hellman (ECDH) is a variant of the Diffie–Hellman protocol using elliptic-curve cryptography

- A key agreement protocol that allows two parties, each having an elliptic-curve public–private key pair, to establish a shared secret over an insecure channel
- The shared secret may be directly used as a key, or to derive another key
- The key, or the derived key, can then be used to encrypt subsequent communications using a symmetric-key cipher



Example of an elliptic curve

Public Key Infrastructure

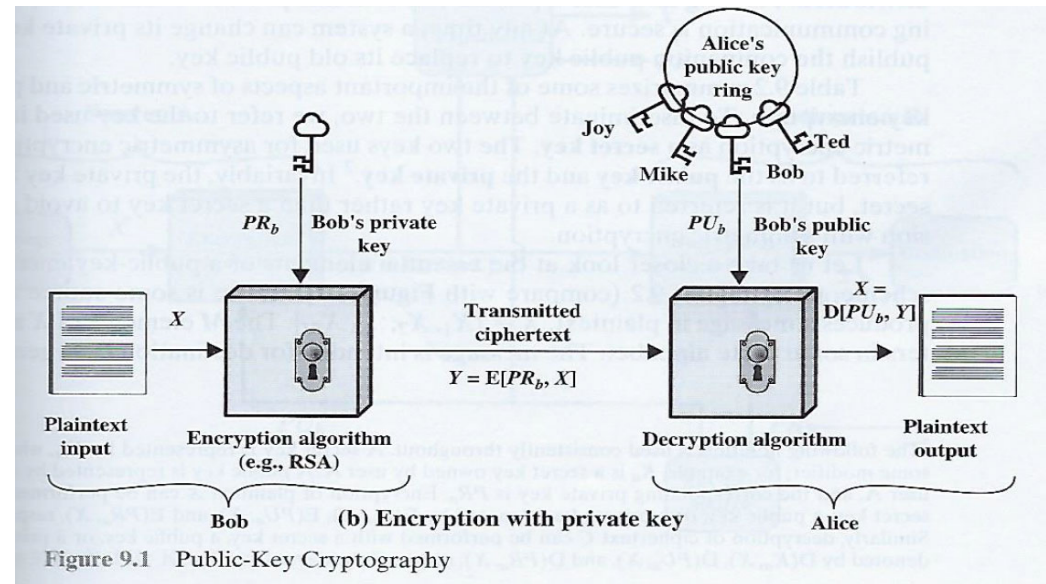
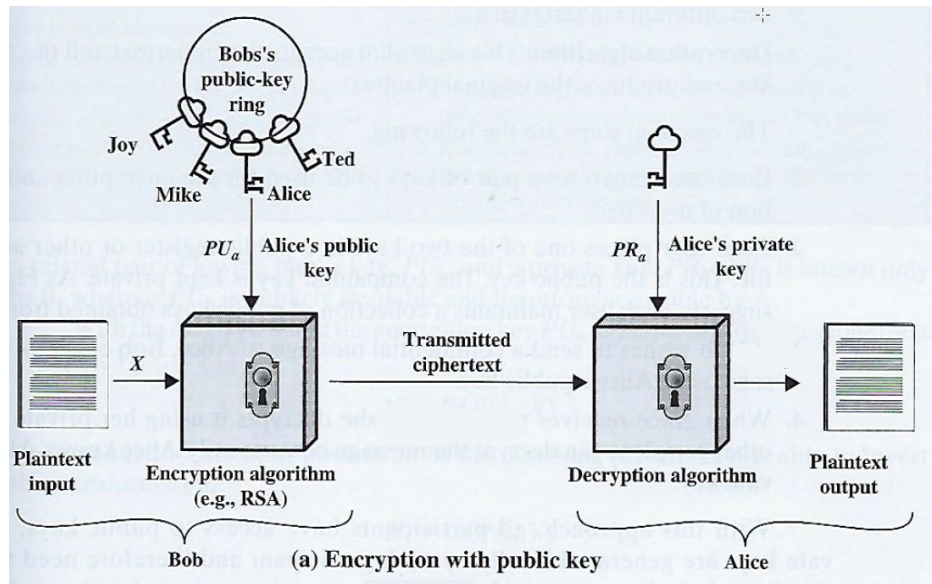
What is PKI (Public Key Infrastructure)?

Public Key Infrastructure (PKI) is a framework that enables secure digital communications by using asymmetric cryptography. PKI provides mechanisms for creating, distributing, managing, and revoking digital certificates that link public keys to entities (such as people, organizations, or devices). It is widely used for ensuring confidentiality, integrity, authentication, and non-repudiation in digital communications.

PKI relies on:

- 1. Public and Private Key Pair** – Each entity has a unique public-private key pair.
- 2. Certificate Authority (CA)** – Issues and verifies digital certificates.
- 3. Registration Authority (RA)** – Validates entity requests before the CA issues certificates.
- 4. Certificate Revocation List (CRL) / Online Certificate Status Protocol (OCSP)** – Used to check if a certificate is revoked.

Public Key Management



Stallings, W. (2014) Cryptography and Network Security

Agenda

- ✓ Cryptography terminology
- ✓ Symmetric Key Cryptography
 - ✓ Symmetric stream cryptography
 - ✓ Symmetric block cryptography
- ✓ Key sharing problem
- ✓ Public Key Cryptography
 - ✓ Diffie-Hellman algorithm: symmetric key generation through asynchronous cryptography
 - ✓ RSA algorithm
- Hybrid-Cryptography
 - Perfect Forward Secrecy
- Where do cryptographic controls go in the FedRAMP System Security Plan
- *If we have time: Brief review of Hashing & Digital Signatures*

Hybrid Encryption (a.k.a. “digital envelope”)

Hybrid encryption is a **cryptographic method** that combines the strengths of **symmetric encryption** (fast and efficient) and **asymmetric encryption** (secure key exchange). This approach is commonly used in real-world applications like **TLS/SSL, PGP, and secure messaging protocols**.

How Hybrid Encryption Works

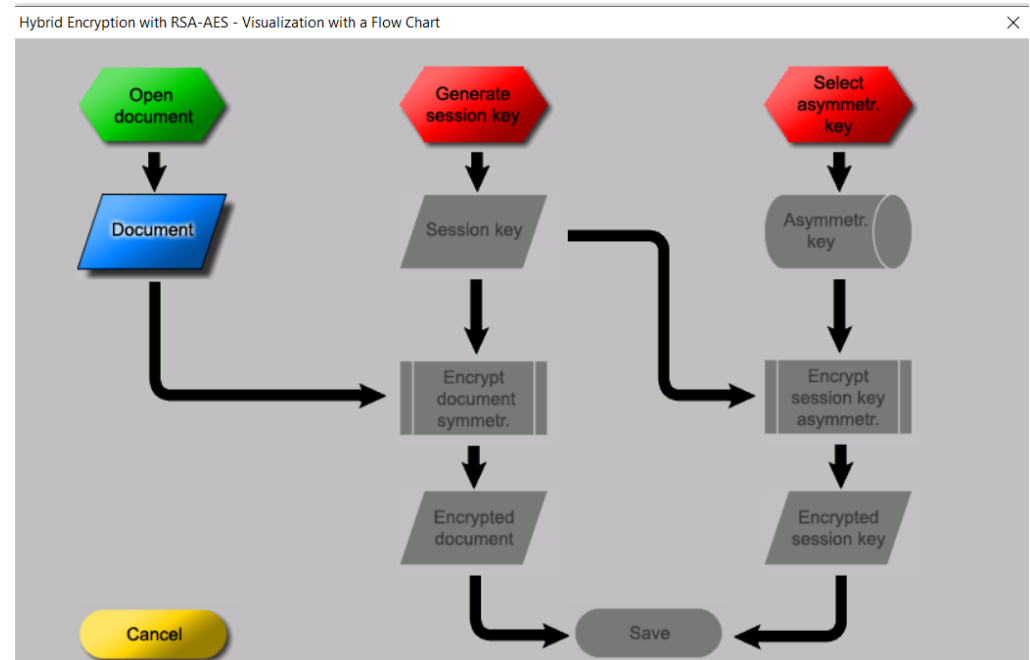
Hybrid encryption follows a two-step process:

1. Asymmetric Encryption (Key Exchange)

1. A sender generates a **random symmetric key** (called the session key).
2. The session key is **encrypted using the recipient’s public key**.
3. The encrypted session key is sent to the recipient.

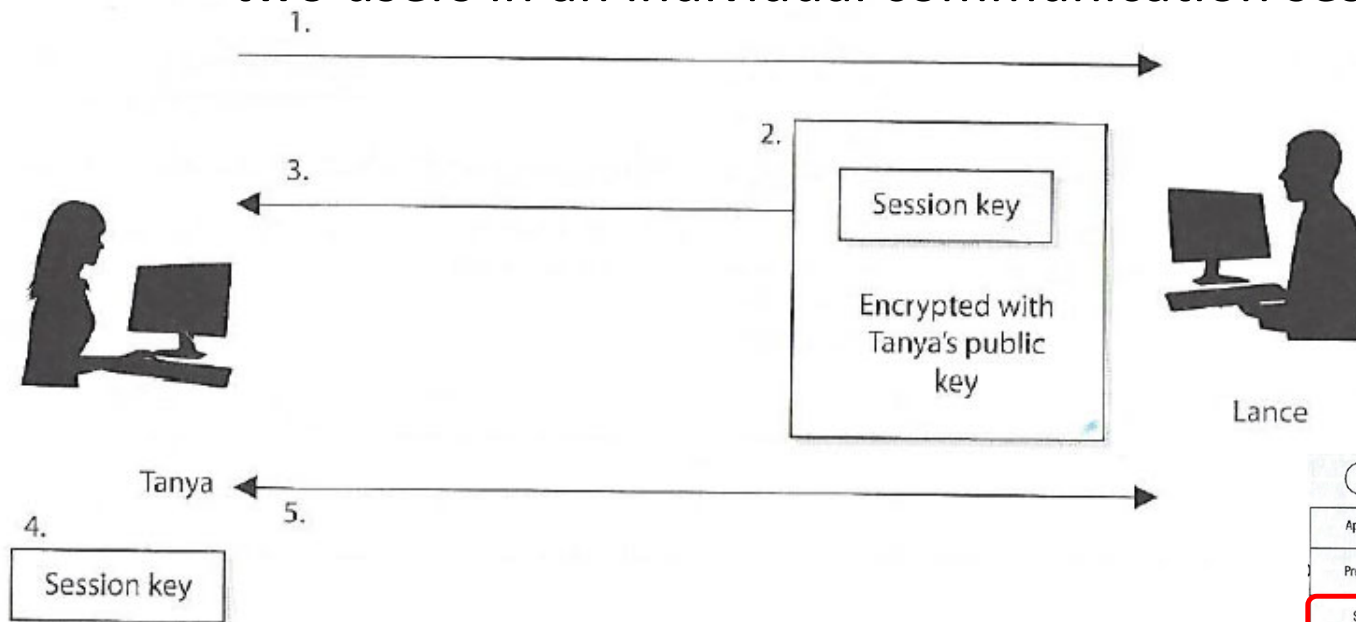
2. Symmetric Encryption (Data Encryption)

1. The sender uses the **session key** to encrypt the actual data.
2. The recipient decrypts the session key using their **private key**.
3. The decrypted session key is then used to **decrypt the actual data**.

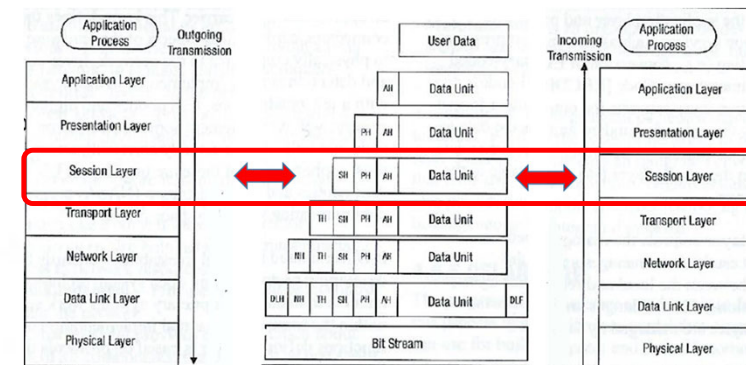


Session keys

Single-use symmetric keys used to encrypt messages between two users in an individual communication session



This is how secure web client applications communicate with server-side services



- 1) Tanya sends Lance her public key.
- 2) Lance generates a random session key and encrypts it using Tanya's public key.
- 3) Lance sends the session key, encrypted with Tanya's public key, to Tanya.
- 4) Tanya decrypts Lance's message with her private key and now has a copy of the session key.
- 5) Tanya and Lance use this session key to encrypt and decrypt messages to each other.

Perfect Forward Secrecy (PFS) or Forward Secrecy (FS)

Designed to prevent the compromise of a long-term secret key from affecting the confidentiality of past conversations

- Protects encrypted data recorded in past sessions against future attacks and compromises of private or secret keys
- Diffie-Hellman and RSA are used together to protect encrypted communications and sessions recorded in the past from being retrieved and decrypted in the future if long-term secret or private keys are compromised in the future

<https://www.wired.com/2016/11/what-is-perfect-forward-secrecy/>

Example of a simple instant messaging protocol employing forward secrecy:

1. Alice and Bob each generate a pair of long-term, asymmetric public and private keys, verification establishes confidence that the claimed owner of a public key is the actual owner
2. Alice and Bob use a key exchange algorithm such as Diffie–Hellman, to securely agree on a short-term symmetric session key
 - *They use the asymmetric keys from step 1 only to authenticate one another during this process*
3. Alice sends Bob a message, encrypting it with a symmetric cipher using the session key negotiated in step 2
4. Bob decrypts Alice's message using the key negotiated in step 2
5. The symmetric session key exchange process repeats for each new message sent, starting from step 2 (switching Alice and Bob's roles as sender/receiver as appropriate)
 - *Step 1 is never repeated*
- Forward secrecy is achieved by generating new session keys for each message
 - It ensures that past communications cannot be decrypted if one of the keys generated in an iteration of step 2 is compromised, since such a key is only used to encrypt a single message
 - It also ensures that past communications cannot be decrypted if the long-term private keys from step 1 are compromised
- However, masquerading as Alice or Bob would be possible going forward if this occurred, possibly compromising all future messages

Perfect Forward Secrecy

- Forward secrecy is present in several major protocol implementations:
 - IPsec (RFC 2412) as an optional feature
 - Transport Layer Security (TLS)
 - Cipher suites based on Diffie–Hellman key exchange (DHE-RSA, DHE-DSA)
 - Elliptic curve Diffie–Hellman key exchange (ECDHE-RSA, ECDHE-ECDSA)
 - OpenSSL supports forward secrecy using elliptic curve Diffie–Hellman since V1.0
 - Off-the-Record Messaging, a cryptography protocol and library for many instant messaging clients

Home - Temple University Portal x +

tuportal5.temple.edu

Ambler 51°F Harrisburg 53°F Tokyo 49°F

TUportal TEMPLE UNIVERSITY

Home

SEARCH

Enter Keyword For...

TUAPPLICATIONS

Canvas Self-Service Banner

ANNOUNCEMENTS

Announcements

2019-2020 Self Study

Security overview

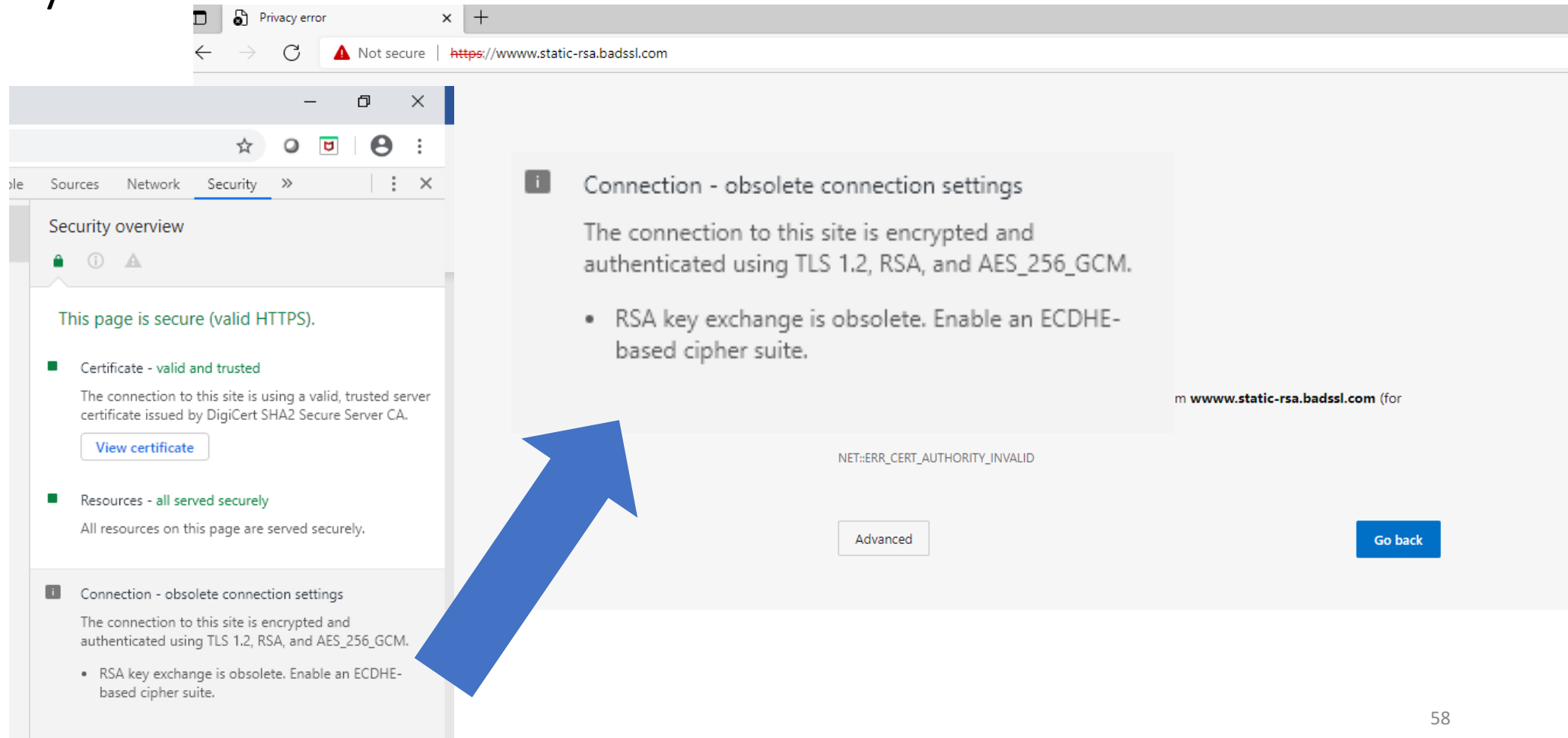
This page is secure (valid HTTPS).

- Certificate - valid and trusted
The connection to this site is using a valid, trusted server certificate issued by GlobalSign Extended Validation CA - SHA256 - G3.
[View certificate](#)
- Connection - secure connection settings
The connection to this site is encrypted and authenticated using TLS 1.2, ECDHE_RSA with P-256, and AES_128_GCM.
- Resources - all served securely
All resources on this page are served securely.

Elliptic-Curve Diffie Hellman for fast symmetric encryption key generation with forward security and RSA for authentication

The screenshot shows a web browser window with the Temple University Portal. A security overlay is displayed in the center, stating: "Connection - secure connection settings. The connection to this site is encrypted and authenticated using TLS 1.2, ECDHE_RSA with P-256, and AES_128_GCM." A red arrow points from this overlay to the "Security" tab in the browser's developer tools. The "Security" tab shows a "Security overview" section with three items: "Certificate - valid and trusted", "Connection - secure connection settings", and "Resources - all served securely". The "Connection - secure connection settings" item is highlighted, showing the same text as the overlay. The browser's address bar shows "tuportal5.temple.edu". The page header includes the Temple University logo and navigation links. The page footer includes a search bar and a list of applications.

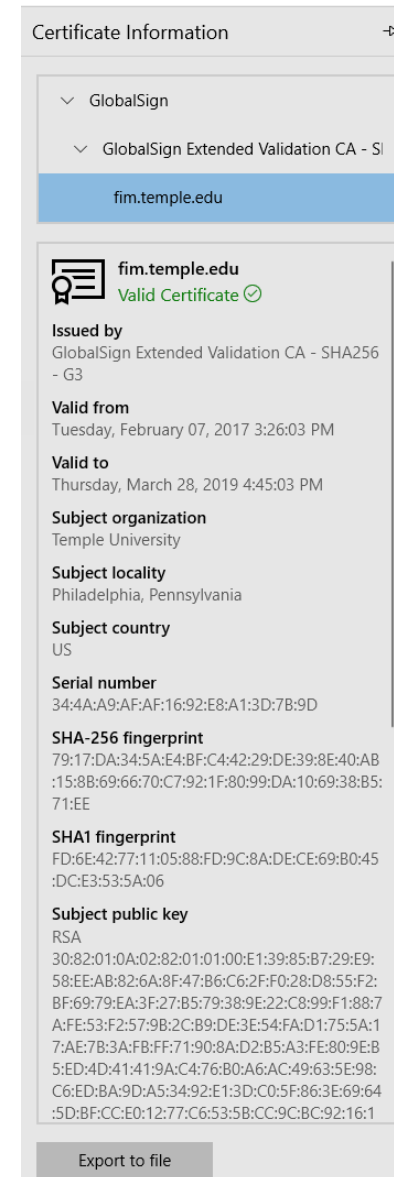
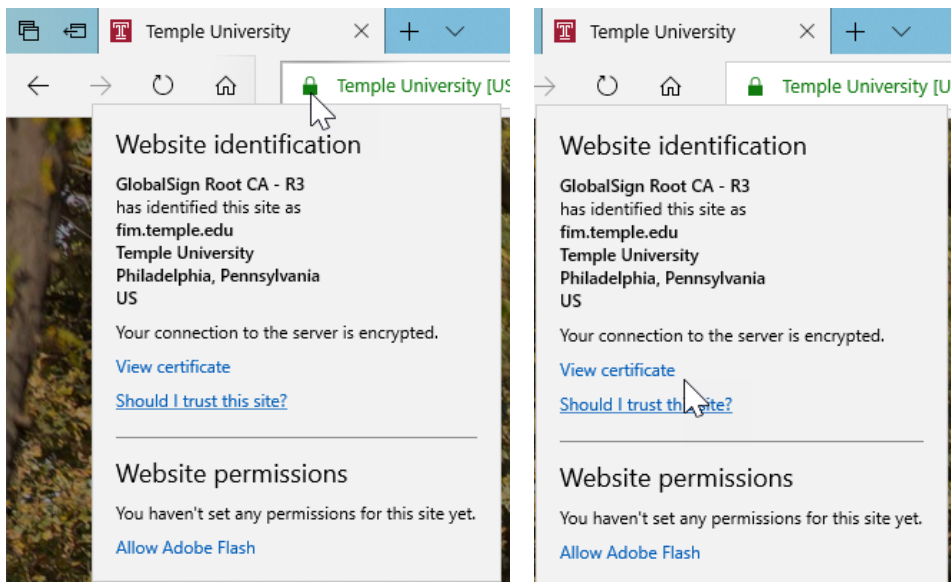
Insecure detection of the use of a static RSA encryption key



Viewing Digital Certificate

Microsoft Edge

1. Click the padlock on the URL bar
2. Click View certificate link



Services of cryptosystems

- ✓ **Confidentiality** – Renders information unintelligible except by authorized entities
- ✓ **Authentication** – Verifies the identity of the user or system that created, requested or provided the information
- ✓ **Nonrepudiation** – Ensure the sender cannot deny sending the information
- **Integrity** – Data has not been altered in an unauthorized manner since it was created, transmitted, or stored


Agenda

- ✓ Cryptography terminology
- ✓ Symmetric Key Cryptography
 - ✓ Symmetric stream cryptography
 - ✓ Symmetric block cryptography
- ✓ Key sharing problem
- ✓ Public Key Cryptography
 - ✓ Diffie-Hellman algorithm: symmetric key generation through asynchronous cryptography
 - ✓ RSA algorithm
- ✓ Hybrid-Cryptography
 - ✓ Perfect Forward Secrecy
- Where do cryptographic controls go in the FedRAMP System Security Plan
- *If we have time: Brief review of Hashing & Digital Signatures*

Where in the FedRAMP System Security Plan would you look for information to help you assess the security of the Titan Information System?



FedRAMP SSP Section	Relevant Cryptographic Controls	Purpose
Section 9: Information System Components	Data encryption methods for storage and transmission	Specifies encryption requirements for data protection in system components
Section 13: System Security Controls	SC-12 (Cryptographic Key Establishment and Management)	Ensures proper key management procedures
	SC-13 (Cryptographic Protection)	Defines encryption mechanisms for system security
	SC-17 (Public Key Infrastructure Certificates)	Manages digital certificates and authentication
	SC-28 (Protection of Information at Rest)	Requires encryption for stored sensitive data
	SC-29 (Protection of Information in Transit)	Mandates encryption for data transmitted over networks
Section 14: Identification and Authentication (IA)	IA-7 (Cryptographic Module Authentication)	Ensures cryptographic modules comply with FIPS 140-2/140-3
Section 15: Risk Assessment (RA) and Continuous Monitoring (CM)	Assessment and validation of cryptographic implementations	Ensures ongoing compliance and security effectiveness

This table helps identify where cryptographic measures should be documented in a FedRAMP System Security Plan (SSP). 

Where do you document this information in your SSP?



TABLE OF CONTENTS

1. INFORMATION SYSTEM NAME/TITLE	1
2. INFORMATION SYSTEM CATEGORIZATION	1
2.1. Information Types	1
2.2. Security Objectives Categorization (FIPS 199)	3
2.3. Digital Identity Determination	3
3. INFORMATION SYSTEM OWNER	4
4. AUTHORIZING OFFICIALS	4
5. OTHER DESIGNATED CONTACTS	4
6. ASSIGNMENT OF SECURITY RESPONSIBILITY	5
7. INFORMATION SYSTEM OPERATIONAL STATUS	6
8. INFORMATION SYSTEM TYPE	7
8.1. Cloud Service Models	7
8.2. Cloud Deployment Models	8
8.3. Leveraged Authorizations	8
9. GENERAL SYSTEM DESCRIPTION	9
9.1. System Function or Purpose	9
9.2. Information System Components and Boundaries	9
9.3. Types of Users	10
9.4. Network Architecture	11
10. SYSTEM ENVIRONMENT AND INVENTORY	12
10.1. Data Flow	12
10.2. Ports, Protocols and Services	14
11. SYSTEM INTERCONNECTIONS	15
12. LAWS, REGULATIONS, STANDARDS AND GUIDANCE	17
12.1. Applicable Laws and Regulations	17
12.2. Applicable Standards and Guidance	17
13. MINIMUM SECURITY CONTROLS	18
13.1. Access Control (AC)	25
AC-1 Access Control Policy and Procedures Requirements (H)	25
AC-2 Account Management (H)	26
AC-2 (1) Control Enhancement (M) (H)	27
AC-2 (2) Control Enhancement (H)	28
AC-2 (3) Control Enhancement (H)	29
AC-2 (4) Control Enhancement (H)	30
AC-2 (5) Control Enhancement (H)	31
AC-2 (7) Control Enhancement (H)	31
AC-2 (9) Control Enhancement (H)	32
AC-2 (10) Control Enhancement (M) (H)	33
AC-2 (11) Control Enhancement (H)	34
AC-2 (12) Control Enhancement (H)	35



FEDRAMP SYSTEM SECURITY PLAN (SSP) HIGH BASELINE TEMPLATE

Cloud Service Provider Name
Information System Name
Version #
Version Date



CONTROLLED UNCLASSIFIED INFORMATION

FEDRAMP SYSTEM SECURITY PLAN (SSP) HIGH BASELINE TEMPLATE

CSP Name | Information System Name

Version # | Date

SA-9 (5) Control Enhancement (M) (H).....	315
SA-10 Developer Configuration Management (M) (H)	316
SA-10 (1) Control Enhancement (M) (H).....	317
SA-11 Developer Security Testing and Evaluation (M) (H).....	318
SA-11 (1) Control Enhancement (M) (H).....	319
SA-11 (2) Control Enhancement (M) (H).....	320
SA-11 (8) Control Enhancement (M) (H).....	321
SA-12 Supply Chain Protection (H)	322
SA-15 Development Process, Standards, and Tools (H).....	322
SA-16 Developer-Provided Training (H).....	324
SA-17 Developer Security Architecture and Design (H).....	324
13.16. System and Communications Protection (SC).....	325
SC-1 System and Communications Protection Policy and Procedures (H)	325
SC-2 Application Partitioning (M) (H).....	326
SC-3 Security Function Isolation (H)	327
SC-4 Information in Shared Resources (M) (H).....	328
SC-5 Denial of Service Protection (L) (M) (H).....	329
SC-6 Resource Availability (M) (H).....	329
SC-7 Boundary Protection (L) (M) (H).....	330
SC-7 (3) Control Enhancement (M) (H).....	331
SC-7 (4) Control Enhancement (H).....	332
SC-7 (5) Control Enhancement (M) (H).....	333
SC-7 (7) Control Enhancement (M) (H).....	334
SC-7 (8) Control Enhancement (M) (H).....	335
SC-7 (10) Control Enhancement (H).....	335
SC-7 (12) Control Enhancement (H).....	336
SC-7 (13) Control Enhancement (H).....	337
SC-7 (18) Control Enhancement (M) (H).....	338
SC-7 (20) Control Enhancement (H).....	339
SC-7 (21) Control Enhancement (H).....	339
SC-8 Transmission confidentiality and Integrity (M) (H).....	340
SC-8 (1) Control Enhancement (M) (H).....	341
SC-10 Network Disconnect (H)	342
SC-12 Cryptographic Key Establishment & Management (L) (M) (H)	343
SC-12 (1) Control Enhancement (H).....	344
SC-12 (2) Control Enhancement (M) (H).....	344
SC-12 (3) Control Enhancement (M) (H).....	345
SC-13 Use of Cryptography (L) (M) (H)	346
SC-15 Collaborative Computing Devices (M) (H)	347
SC-17 Public Key Infrastructure Certificates (M) (H).....	348
SC-18 Mobile Code (M) (H).....	349
SC-19 Voice Over Internet Protocol (M) (H)	350
SC-20 Secure Name / Address Resolution Service (Authoritative Source) (L) (M) (H).....	351
SC-21 Secure Name / Address Resolution Service (Recursive or Caching Resolver) (L) (M) (H).....	352
SC-22 Architecture and Provisioning for Name / Address Resolution Service (L) (M) (H).....	353
SC-23 Session Authenticity (M) (H).....	353

FedRAMP 0100110010001010000100101000100100100101010010001001110101 | xiii

Controlled Unclassified Information

SC-12 Cryptographic Key Establishment & Management (L) (M) (H)

The organization establishes and manages cryptographic keys for required cryptography employed within the information system in accordance with [Assignment: organization-defined requirements for key generation, distribution, storage, access, and destruction].

SC-12 Additional FedRAMP Requirements and Guidance:

Guidance: Federally approved and validated cryptography.

SC-12	Control Summary Information
Responsible Role:	
Parameter SC-12:	
Implementation Status (check all that apply): <input type="checkbox"/> Implemented <input type="checkbox"/> Partially implemented <input type="checkbox"/> Planned <input type="checkbox"/> Alternative implementation <input type="checkbox"/> Not applicable	
Control Origination (check all that apply): <input type="checkbox"/> Service Provider Corporate <input type="checkbox"/> Service Provider System Specific <input type="checkbox"/> Service Provider Hybrid (Corporate and System Specific) <input type="checkbox"/> Configured by Customer (Customer System Specific) <input type="checkbox"/> Provided by Customer (Customer System Specific) <input type="checkbox"/> Shared (Service Provider and Customer Responsibility) <input type="checkbox"/> Inherited from pre-existing FedRAMP Authorization for Click here to enter text. , Date of Authorization	

SC-12 What is the solution and how is it implemented?

FEDRAMP SYSTEM SECURITY PLAN (SSP) HIGH BASELINE TEMPLATE

Cloud Service Provider Name
Information System Name
Version #
Version Date



CONTROLLED UNCLASSIFIED INFORMATION

FEDRAMP SYSTEM SECURITY PLAN (SSP) HIGH BASELINE TEMPLATE

CSP Name | Information System Name

Version # | Date

SA-9 (5) Control Enhancement (M) (H).....	315
SA-10 Developer Configuration Management (M) (H)	316
SA-10 (1) Control Enhancement (M) (H).....	317
SA-11 Developer Security Testing and Evaluation (M) (H).....	318
SA-11 (1) Control Enhancement (M) (H).....	319
SA-11 (2) Control Enhancement (M) (H).....	320
SA-11 (8) Control Enhancement (M) (H).....	321
SA-12 Supply Chain Protection (H)	322
SA-15 Development Process, Standards, and Tools (H).....	322
SA-16 Developer-Provided Training (H).....	324
SA-17 Developer Security Architecture and Design (H).....	324
13.16. System and Communications Protection (SC).....	325
SC-1 System and Communications Protection Policy and Procedures (H)	325
SC-2 Application Partitioning (M) (H).....	326
SC-3 Security Function Isolation (H)	327
SC-4 Information in Shared Resources (M) (H)	328
SC-5 Denial of Service Protection (L) (M) (H).....	329
SC-6 Resource Availability (M) (H)	329
SC-7 Boundary Protection (L) (M) (H)	330
SC-7 (3) Control Enhancement (M) (H).....	331
SC-7 (4) Control Enhancement (H).....	332
SC-7 (5) Control Enhancement (M) (H).....	333
SC-7 (7) Control Enhancement (M) (H).....	334
SC-7 (8) Control Enhancement (M) (H).....	335
SC-7 (10) Control Enhancement (H).....	335
SC-7 (12) Control Enhancement (H).....	336
SC-7 (13) Control Enhancement (H).....	337
SC-7 (18) Control Enhancement (M) (H).....	338
SC-7 (20) Control Enhancement (H).....	339
SC-7 (21) Control Enhancement (H).....	339
SC-8 Transmission confidentiality and Integrity (M) (H).....	340
SC-8 (1) Control Enhancement (M) (H).....	341
SC-10 Network Disconnect (H)	342
SC-12 Cryptographic Key Establishment & Management (L) (M) (H)	343
SC-12 (1) Control Enhancement (H)	344
SC-12 (2) Control Enhancement (M) (H).....	344
SC-12 (3) Control Enhancement (M) (H).....	345
SC-13 Use of Cryptography (L) (M) (H)	346
SC-15 Collaborative Computing Devices (M) (H)	347
SC-17 Public Key Infrastructure Certificates (M) (H).....	348
SC-18 Mobile Code (M) (H)	349
SC-19 Voice Over Internet Protocol (M) (H)	350
SC-20 Secure Name / Address Resolution Service (Authoritative Source) (L) (M) (H).....	351
SC-21 Secure Name / Address Resolution Service (Recursive or Caching Resolver) (L) (M) (H)	352
SC-22 Architecture and Provisioning for Name / Address Resolution Service (L) (M) (H)	353
SC-23 Session Authenticity (M) (H).....	353



SC-12 (1) CONTROL ENHANCEMENT (H)

The organization maintains availability of information in the event of the loss of cryptographic keys by users.

SC-12 (2) CONTROL ENHANCEMENT (M) (H)

The organization produces, controls, and distributes symmetric cryptographic keys using [FedRAMP Selection: NIST FIPS-compliant] key management technology and processes.

SC-12 (3) CONTROL ENHANCEMENT (M) (H)

The organization produces, controls, and distributes asymmetric cryptographic keys using [Selection: NSA-approved key management technology and processes; approved PKI Class 3 certificates or prepositioned keying material; approved PKI Class 3 or Class 4 certificates and hardware security tokens that protect the user's private key].

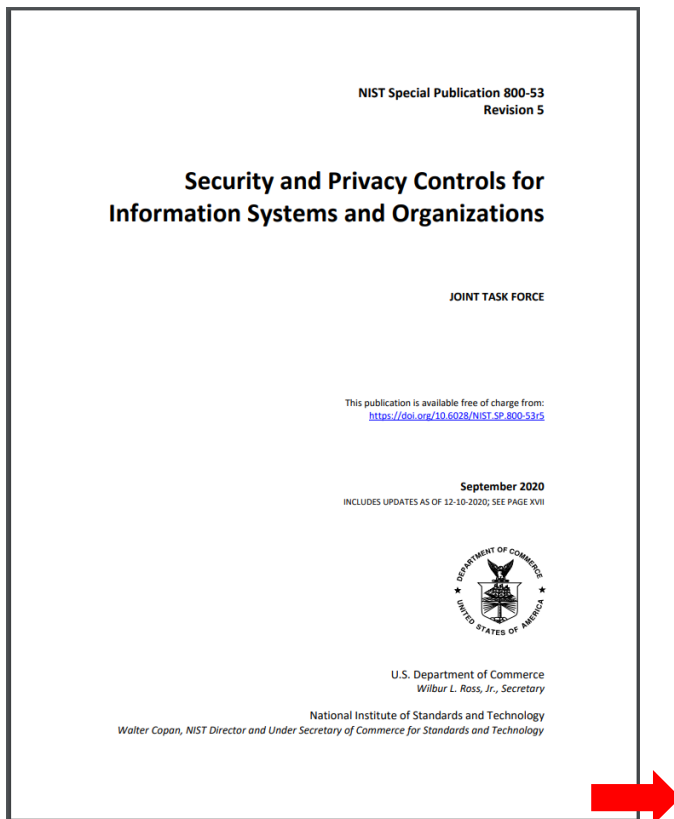
SC-13 Use of Cryptography (L) (M) (H)

The information system implements [FedRAMP Assignment: FIPS-validated or NSA-approved cryptography] in accordance with applicable federal laws, Executive Orders, directives, policies, regulations, and standards.

FedRAMP 01001100100010100001001001001000010011010100100000100110101 | xiii

Controlled Unclassified Information

Where do you look for encryption related controls that could help Titan?



MIS5214 Security Architecture

NIST 800-53 Control Family	Control ID	Control Name	Control Category	Purpose
Access Control (AC)	AC-17(2)	Remote Access – Protection of Confidentiality and Integrity	Technical	Requires encryption for remote access connections
	AC-19(5)	Access Control for Mobile Devices – Encryption	Technical	Ensures mobile device data is encrypted
Audit and Accountability (AU)	AU-09(3)	Protection of Audit Information – Cryptographic Protection	Technical	Encrypts audit records to prevent tampering
Identification and Authentication (IA)	IA-07	Cryptographic Module Authentication	Technical	Requires authentication using cryptographic modules (FIPS 140-2/140-3)
System and Communications Protection (SC)	SC-12	Cryptographic Key Establishment and Management	Technical	Defines key generation, distribution, and lifecycle management
	SC-13	Cryptographic Protection	Technical	Specifies encryption for system security and data protection
	SC-17	Public Key Infrastructure (PKI) Certificates	Technical	Manages issuance and revocation of PKI certificates
	SC-28	Protection of Information at Rest	Technical	Requires encryption for stored sensitive data
	SC-29	Protection of Information in Transit	Technical	Ensures encryption for data transmitted over networks
	SC-31	Covert Channel Analysis	Technical	Analyzes covert channels that could bypass encryption
System and Information Integrity (SI)	SI-07	Software, Firmware, and Information Integrity	Technical	Uses cryptographic methods to verify integrity

In NIST SP 800-53, cryptographic controls primarily fall under the **Technical** control category, as they involve encryption, cryptographic key management, data protection, and authentication mechanisms.

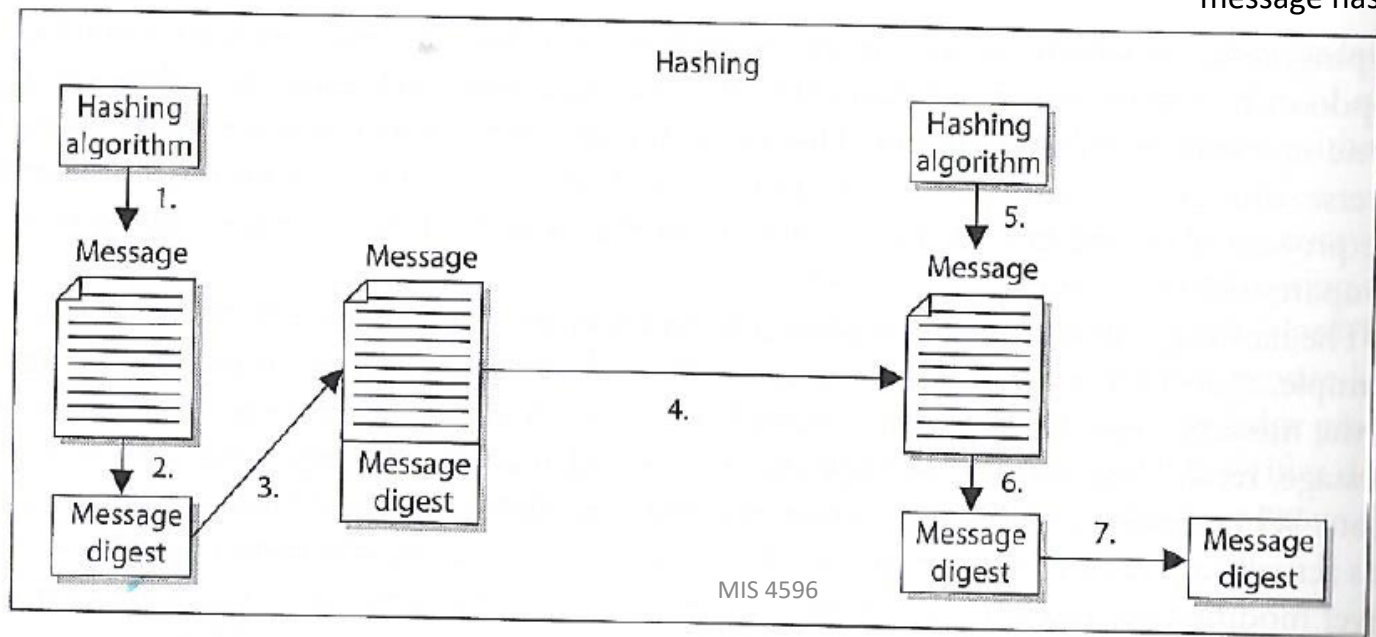
Agenda

- ✓ Cryptography terminology
- ✓ Symmetric Key Cryptography
 - ✓ Symmetric stream cryptography
 - ✓ Symmetric block cryptography
- ✓ Key sharing problem
- ✓ Public Key Cryptography
 - ✓ Diffie-Hellman algorithm: symmetric key generation through asynchronous cryptography
 - ✓ RSA algorithm
- ✓ Hybrid-Cryptography
 - ✓ Perfect Forward Secrecy
- ✓ Where do cryptographic controls go in the FedRAMP System Security Plan
- *If we have time: Brief review of Hashing & Digital Signatures*

Quick Review: One-way Hash

- Assures message **integrity**
- A function that takes a variable-length string (i.e. message) and produces a fixed-length value called a hash value
- Does not use keys

1. Sender puts message through hashing function
2. Message digest generated
3. Message digest appended to the message
4. Sender sends message to receiver
5. Receiver puts message through hashing function
6. Receiver generates message digest value
7. Receiver compares the two message digests values. If they are the same, the message has not been altered




Note: Hashing results in fixed-sized output

- Names for the output of a hashing functions include “hash” and a *message digest (md)*, because a hash “digests” an input of any size down to a **fixed-sized output**
- No matter the size of the input, the out put is the same, for example the md5 hash function’s output:
 - Letter ‘a’ in binary: 01000001 => md5 hash => 32-character string
 - Blu-ray disk digest => md5 hash => 32-character string
 - 6 TB hard drive digest => md5 hash => 32-character string

One-way hash example...

Testing the integrity of a file (e.g. program) downloaded from the internet...

Secure | <https://www.kali.org/downloads/>



BlogDownloadsTrainingDocumentationCommunityAbout Us

Kali Linux Downloads

Download Kali Linux Images

We generate fresh Kali Linux image files every few months, which we make available for download. This page provides the links to **download Kali Linux** in its latest official release. For a release history, check our [Kali Linux Releases](#) page. Please note: You can find unofficial, untested weekly releases at <http://cdimage.kali.org/kali-weekly/>.

Image Name	Download	Size	Version	sha256sum
Kali 64 bit	HTTP Torrent	2.8G	2017.2	4556775bfb981ae64a3cb19aa0b73e8dcac6e4ba524f31c4bc14c9137b99725d
Kali 32 bit	HTTP Torrent	2.9G	2017.2	7f5000d8f55469264399a8bb7358fc22bec87fb1dc8a51b87f26876634e3effc
Kali 64 bit Light	HTTP Torrent	0.8G	2017.2	369a29deff40dff4f53fb47a6015d41d4ada8833a0b6e159657d2f223670f8b
Kali 32 bit Light	HTTP Torrent	0.8G	2017.2	f6ee21b2880501cae8aa47960e8f424ab5fae1a13ba4b4e02d45152b6acd0d
Kali 64 bit e17	HTTP Torrent	2.6G	2017.2	20dee81d9891aa6dcfe505a68692f98f981b43a14234d18d9ed92373d6ed6ab
Kali 64 bit Mate	HTTP Torrent	2.8G	2017.2	9c99a2cc52b1d48875681d12e1fcf6b0b003d44f7ceb610438b5bea148414810
Kali 64 bit Xfce	HTTP Torrent	2.7G	2017.2	9ecf6a054de1e3ad04d4063e3d347efb31326078c104ec2e78ab456fc4d2a578
Kali 64 bit LXDE	HTTP Torrent	2.7G	2017.2	c832df6b7a8e7074a5d7f5a50245b840a3df72fdd4d19a5d1f647beebb4f299
Kali armhf	HTTP Torrent	0.6G	2017.2	a7f3e648ce9784589245c18084e2273eb1f4ec1b78244a2c6d4465f3744c9198

MIS 4596

Follow us on Twitter

Follow @kalinix155K followers

Follow @offsecraining125K followers

Follow @exploitdb128K followers

f


in

v

o

u

Ready for the OSCP?



Join the ever growing group of well trained and highly skilled **Offensive Security Certified Professionals**. Learn hands-on, real world **penetration testing** from the creators of Kali Linux.

One-way hash example...

Image Name	Download	Size	Version	sha256sum
Kali 64 bit	HTTP Torrent	2.8G	2017.2	4556775bfb981ae64a3cb19aa0b73e8dcac6e4ba524f31c4bc14c9137b99725d

```
Windows PowerShell

PS C:\Users\tue87168> cd Downloads
PS C:\Users\tue87168\Downloads> dir *.iso

Directory: C:\Users\tue87168\Downloads

Mode                LastWriteTime         Length Name
----                -
-a----            8/10/2017 10:55 AM        674803712 CSET_8.0 (1).iso
-a----            8/10/2017 11:03 AM        674803712 CSET_8.0 (2).iso
-a----            6/12/2017 10:29 AM        674803712 CSET_8.0.iso
-a----            9/27/2017  3:03 PM       2421987328 en_project_professional_2016_x86_x64_dvd_6962236.iso
-a----            10/3/2017  8:49 PM       2421987328 en_vision_professional_2016_x86_x64_dvd_6962139.iso
-a----           11/11/2016 11:45 AM       1469054976 Fedora-Live-Workstation-x86_64-23-10.iso
-a----           11/9/2017  2:31 PM       3020619776 kali-linux-2017.2-amd64.iso

PS C:\Users\tue87168\Downloads> Get-FileHash kali-linux-2017.2-amd64.iso | Format-List

Algorithm : SHA256
Hash      : 4556775BFB981AE64A3CB19AA0B73E8DCAC6E4BA524F31C4BC14C9137B99725D
Path      : C:\Users\tue87168\Downloads\kali-linux-2017.2-amd64.iso

PS C:\Users\tue87168\Downloads> _
```

One-way hash example...

```
Windows PowerShell
PS C:\Users\tue87168\Downloads> dir *.txt

Directory: C:\Users\tue87168\Downloads

Mode                LastWriteTime         Length Name
----                -
-a-----         11/9/2017   3:04 PM             15 MIS5206-IsGood.txt

PS C:\Users\tue87168\Downloads> type MIS5206-IsGood.txt
MIS5206 is good
PS C:\Users\tue87168\Downloads> Get-FileHash MIS5206-IsGood.txt | Format-List

Algorithm : SHA256
Hash      : E6F053ADE3857C0EDC2896B229D0B91D4752B2D9D8C9BD4B2A45A4ACCB3999DD
Path      : C:\Users\tue87168\Downloads\MIS5206-IsGood.txt

PS C:\Users\tue87168\Downloads> type MIS5206-IsGood.txt
MIS5206 is goop
PS C:\Users\tue87168\Downloads> Get-FileHash MIS5206-IsGood.txt | Format-List

Algorithm : SHA256
Hash      : 877B45EA5D40D98FF8D1ABD919E154F446FEA11387DBB13DDEE448F9932928A5
Path      : C:\Users\tue87168\Downloads\MIS5206-IsGood.txt

PS C:\Users\tue87168\Downloads>
```

Notice the amount of confusion and diffusion resulting from a 1 character change!

Cryptanalysis Attack

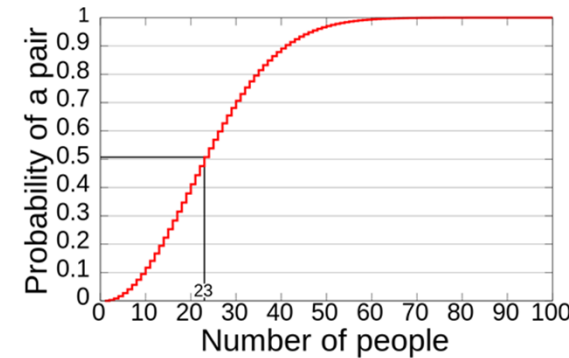
Collisions

- *Two different messages with the same hash value*
- Based on the “birthday paradox”
- Hash algorithms should be resistant to this attack

The birthday paradox, also known as the birthday problem, states that in a random group of 23 people, there is about a 50 percent chance that two people have the same birthday.

Is the Birthday Attack Real?

- There are multiple reasons why this seems like a paradox
- One is that when in a room with 22 other people, if a person compares his or her birthday with the birthdays of the other people it would make for only 22 comparisons—only 22 chances for people to share the same birthday.





When all 23 birthdays are compared against each other, it makes for much more than 22 comparisons. How much more? Well, the first person has 22 comparisons to make, but the second person was already compared to the first person, so there are only 21 comparisons to make. The third person then has 20 comparisons, the fourth person has 19 and so on. If you add up all possible comparisons ($22 + 21 + 20 + 19 + \dots + 1$) the sum is 253 comparisons, or combinations. Consequently, each group of 23 people involves 253 comparisons, or 253 chances for matching birthdays.

MD5 (Message Digest 5)

- A 128-bit hash algorithm, still in common use
- Has been broken
- 128-bit hash, but only need $2^{128/2} = 2^{64}$ to find a collision
- Not strong enough for modern computers

Example of an MD5 hash collision:

Name	Date modified	Type	Size
 ProgramA	1/27/2020 4:08 PM	Application	6 KB
 ProgramB	1/27/2020 4:08 PM	Application	6 KB

```
Hello, world!  
(press enter to quit)
```

ProgramA run

```
This program is evil!!!  
Erasing hard drive...1Gb...2Gb... just kidding!  
Nothing was erased.  
(press enter to quit)_
```

ProgramB run

```
Windows PowerShell
PS C:\Users\Dave\Desktop\MD5-Hash-Collision-Example> get-filehash ProgramA.exe -Algorithm MD5

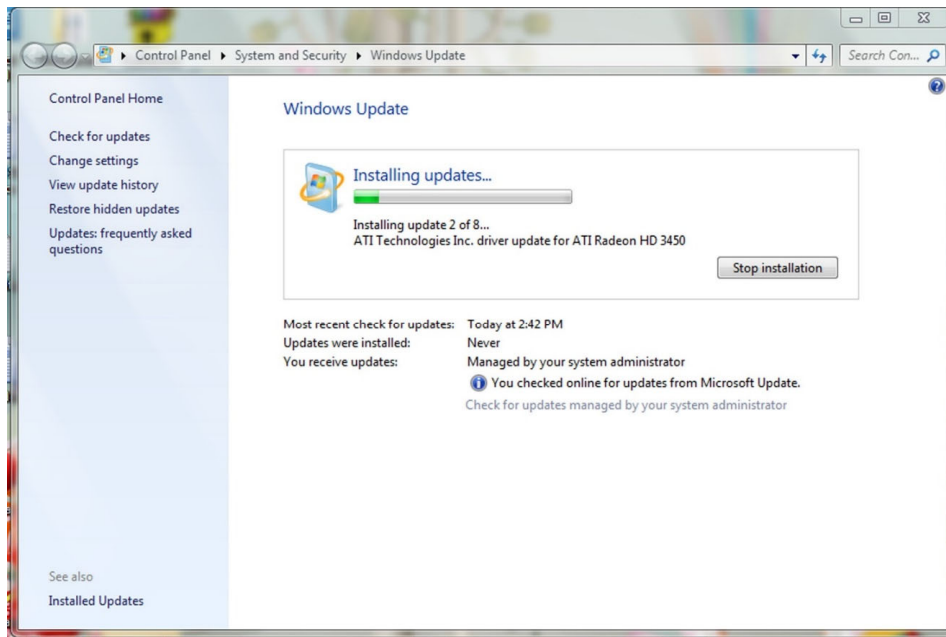
Algorithm      Hash                                     Path
-----
MD5            CDC47D670159EEF60916CA03A9D4A007      C:\Users\Dave\Desktop\MD5-Hash-Collision-Example\ProgramA.exe

PS C:\Users\Dave\Desktop\MD5-Hash-Collision-Example> get-filehash ProgramB.exe -Algorithm MD5

Algorithm      Hash                                     Path
-----
MD5            CDC47D670159EEF60916CA03A9D4A007      C:\Users\Dave\Desktop\MD5-Hash-Collision-Example\ProgramB.exe
```

In 2012 malware Flame used a MD5 hash collision to hijack Microsoft Windows Update and spread itself across networks

- Flame collected **audio, keystrokes, screenshots** which it sent to a malicious server
- Found a collision within a single millisecond
- Cost ~\$200k computing time just for 1ms
- Attributed to advanced persistent threat group [Equation Group](#)
- Used in espionage attacks on countries



SHA -1 (Security Hash Algorithm 1)

- A 160-bit hash algorithm, still in common use
- Has been broken
- 160-bit hash, but only need $2^{160/2} = 2^{80}$ to find a collision
- No longer strong enough for modern computers

SHA-2 uses 224, **256**, 384, and 512-bit hashes

- But... it is built using the design of SHA-1, and prone to the same weaknesses
- It's believed to be a matter of time before SHA-2 is also exploited

SHA-3 was recently ratified by NIST, the U.S. National Institute of Standards and Technology

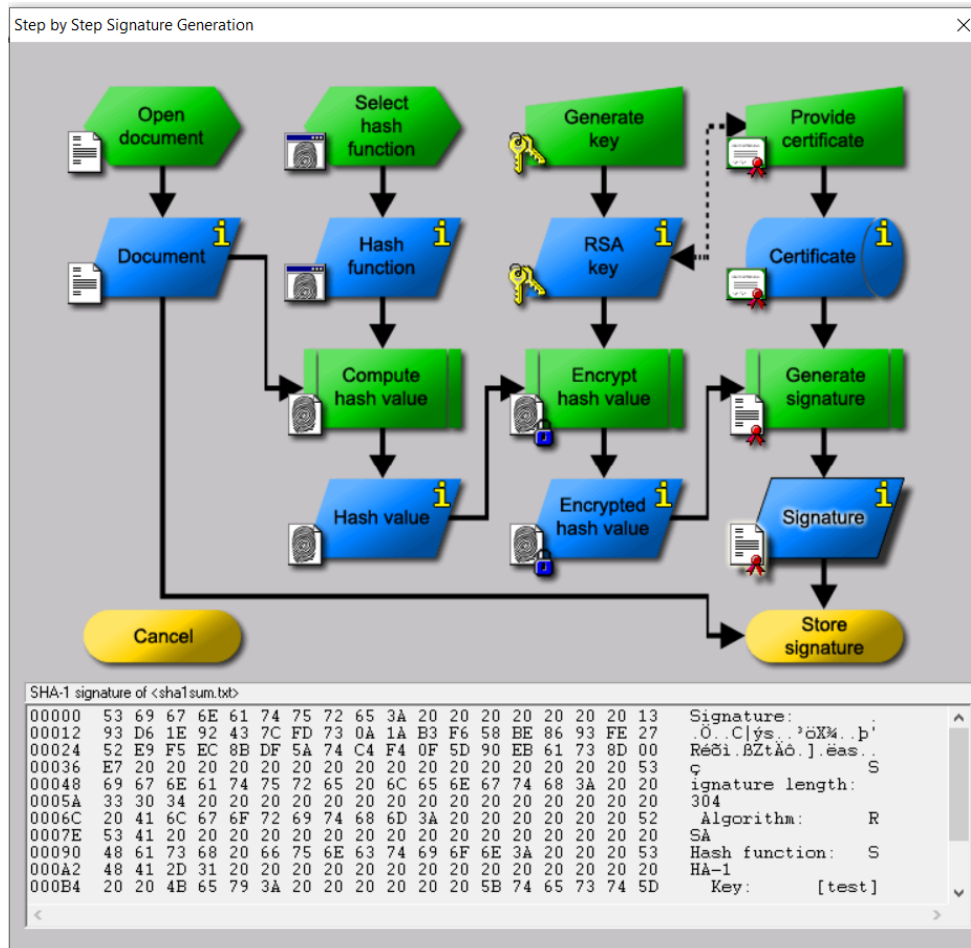
- It was the result of a six-year hashing competition. Also uses 224-, 256-, 384-, 512-bit hashes

Why does this matter for businesses?

Business needs a reliable way to prove integrity of data, files, programs, that can be trusted

Digital Signature

The act of signing means encrypting the message's hash value with the private key



Services of cryptosystems

- ✓ **Confidentiality** – Renders information unintelligible except by authorized entities
- ✓ **Authentication** – Verifies the identity of the user or system that created, requested or provided the information
- ✓ **Nonrepudiation** – Ensure the sender cannot deny sending the information
- ✓ **Integrity** – Data has not been altered in an unauthorized manner since it was created, transmitted, or stored

Summary of some characteristics of cryptographic algorithms

Feature / Algorithm	Hash	Symmetric	Asymmetric
No. of Keys	0	1	2
NIST recommended Key length	256 bits	128 bits	2048 bits
Commonly used	SHA	AES	RSA
Key Management/Sharing	N/A	Big issue	Easy & Secure
Effect of Key compromise	N/A	Loss of both sender & receiver	Only loss for owner of Asymmetric key
Speed	Fast	Fast	Relatively slow
Complexity	Medium	Medium	High
Examples	SHA-224, SHA-256, SHA-384 or SHA-512	AES, Blowfish, Serpent, Twofish, 3DES, and RC4	RSA, DSA, ECC, Diffie-Hellman

SHA – Secure Hash Algorithm

AES – Advanced Encryption Standard

RSA – Public key cryptosystem named after Ron Rivest, Adi Shamir and Leonard Adleman

<https://www.cryptomathic.com/news-events/blog/differences-between-hash-functions-symmetric-asymmetric-algorithms>

Agenda

- ✓ Team Project – It is not too early to get started...
- ✓ Case Study 1
- ✓ Cryptography terminology
- ✓ Symmetric Key Cryptography
 - ✓ Symmetric stream cryptography
 - ✓ Symmetric block cryptography
- ✓ Key sharing problem
- ✓ Public Key Cryptography
 - ✓ Diffie-Hellman algorithm: symmetric key generation through asynchronous cryptography
 - ✓ RSA algorithm
- ✓ Hybrid-Cryptography
 - ✓ Perfect Forward Secrecy
- ✓ Where do cryptographic controls go in the FedRAMP System Security Plan
- ✓ Brief review: Hashing & Digital Signatures

Quiz

Which control is the BEST way to ensure that the data in a file have not been changed during transmission?

- a) Reasonableness check
- b) Parity bits
- c) Hash values
- d) Check digits

The PRIMARY reason for using digital signatures is to ensure data:

- a) confidentiality
- b) integrity
- c) availability
- d) Timeliness

Which of the following provides the GREATEST assurance for database password encryption?

- a) Secure hash algorithm-256 (SHA-256)
- b) Advanced encryption standard (AES)
- c) Secure Shell (SSH)
- d) Triple data encryption standard (3DES)

Email message authenticity and confidentiality is BEST achieved by signing the message using the:

- a) Sender's private key and encrypting the message using the receiver's public key
- b) Sender's public key and encrypting the message using the receiver's private key
- c) Receiver's private key and encrypting the message using the sender's public key
- d) Receiver's public key and encrypting the message using the sender's private key

Quiz

Which control is the BEST way to ensure that the data in a file have not been changed during transmission?

- a) Reasonableness check
- b) Parity bits
- c) Hash values
- d) Check digits

The PRIMARY reason for using digital signatures is to ensure data:

- a) confidentiality
- b) integrity
- c) availability
- d) Timeliness

Which of the following provides the GREATEST assurance for database password encryption?

- a) Secure hash algorithm-256 (SHA-256)
- b) Advanced encryption standard (AES)
- c) Secure Shell (SSH)
- d) Triple data encryption standard (3DES)

Email message authenticity and confidentiality is BEST achieved by signing the message using the:

- a) Sender's private key and encrypting the message using the receiver's public key
- b) Sender's public key and encrypting the message using the receiver's private key
- c) Receiver's private key and encrypting the message using the sender's public key
- d) Receiver's public key and encrypting the message using the sender's private key