Lecture 1

Wednesday, January 30, 2019 10:49 PM

Password Based Attacks

1) Guessing (online or offline)
2) Capture : either by exploit/ malware or intercepting traffic orby breaking in or by social enginnering
3) Backdoor
4) Bypass : buffer overflow
5) Defeat recovery mechanism: weak verification questions or interception of recovery email
6) Reconstruction: Capture partial information or leaked

Why Security is hard

7) Adversary is intelligent and adaptive, and has all the time
8) No rules for adversary to play by, while defender has to follow protocols
9) Defender needs to defend all possible attack points, exploiter only needs to find one
10) Scale of attack
11) Universal connectivity : Always vulnerable
12) Human factors : non compliance or non expert user
13) Errors in design
14) Deployment of protection could introduce new vulnerabilities

The mere fact of capability is sensitive , since an adversary who knows what we can and cannot break is able to elude to your capabilities even without knowing technical details of how the capabilities work.

Security mindset involves thinking about how things can be made to fail as opposed to the traditional thinking of how things should be made to work.
It makes you think like an attacker/ adversary/ criminal.

Lecture 2

Wednesday, January 30, 2019 11:12 PM

Cryptography

Must always think in 3 different perspectives:

1) Defender point of view
2) Attacker point of view
3) Cost point of view to protect and attack

Think about the threat model. Make assumptions about the system and operating environment.

Cryptography Goals:

1) Confidentiality: prevent others from reading data
2) Integrity of data
3) Data origin authentication: If source originated is what it says
4) Entity authentication: is sender who they claim to be
5) Non repudiation: Prevent someone from denying past actions

Attacker Model:

1) Passive vs Active Attacker: In **active** attacks the attacker intercepts the connection and **modifies** the information. Whereas, in a **passive** attack, the attacker intercepts the transit information with the intention of **reading** and analyzing the information not for altering it.
2) What computational resources the attacker has? This could be a factor in how quickly he can break a certain type of encryption.
3) What does the attacker know about the system? What cryptography is used? What protocols are used?
4) What are the assumptions? Encryption keys are shared via secure channels

BruteForce Time (2^55 AES / sec)

AES 256-Bit Key : 509x10^38 trillion years
AES 128-bit Key : 149 trillion years

Virtual Matrix Encryption--> Symmetric encryption with 1MB key

Goals of Encryption

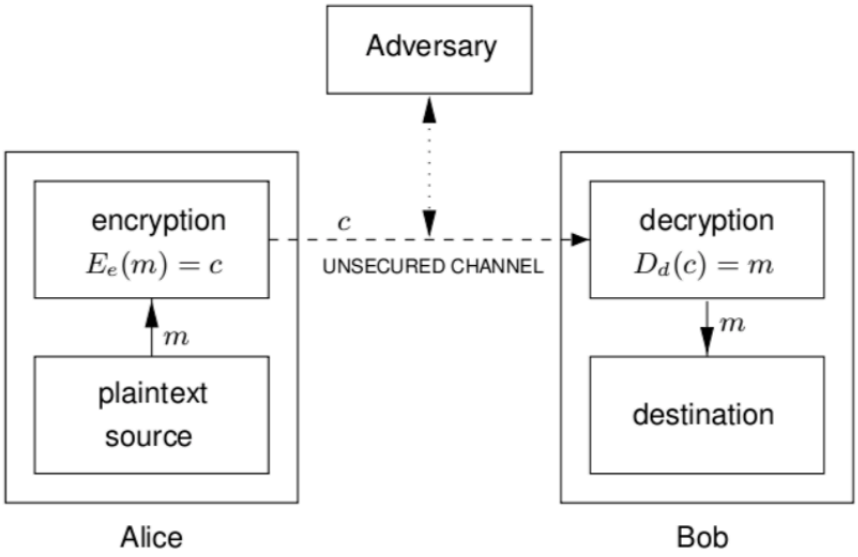
1) Data confidentiality
2) Protect large amount of data with short secret. We say short secret because if the secret is long it becomes inconvenient to share.
3) Make it difficult for those without the secret key, but make it easy for those with the secret key.

Encryption Model

Symmetric Key encryption:

- Given a plaintext "m"
- Private key "e" and Public key "d"
- Cipher E and D
- encrypted message "C" is the such that: Ee(m) = C , and Dd(c) = m .

Symmetric key encryption: d is easily derived from e
Public key encryption: deriving d from e is not feasible



Encryption Keywords

1) Plaintext : Original message
2) Ciphertext: encrypted message
3) Cipher : Algorithm for encrypting the message and decrypting it
4) Key: info used in the cypher to encrypt and decrypt
5) Encipher (encrypt) : converting plaintext to ciphertext
6) Decipher (decrypt) : converting ciphertext to plaintext
7) Substitution : each element is mapped to another
8) Transposition : rearrangement of elements

Kerckhoff's Principles:

1) Security should depend only on the key. Do not assume enemy won't know the encryption algo. Too expensive to invent new algorithm if it is compromised.
2) Security by obscurity doesn't work . People can disassemble and figure it out

Ways to Attack Encryption

1) Cryptanalysis : Searching for weakness in algorithms. Also possible to get partial knowledge about plaintext or cipher text by analyzing patterns
2) Bruteforce: Try all possible (N) combinations . Takes on average N/2 trials.
3) Malware, Keylogger, Physical and side channel attacks, implementation bugs, backdoors, legal means……

Attacks on Encryption schemes

1) Ciphertext only attack: deduce decryption key or plaintext by observing cyphertext
2) Known plaintext analysis : Using a quantity of plaintext and corresponding cipher text to find patterns
3) Chosen plaintext attack: Choose plaintext and is then given corresponding ciphertext
4) Adaptive chosen plaintext attack: Choice of plaintext may depend on the cipher text received from previous request
5) Chosen cipher text attack: Select the cipher text and then the corresponding plaintext is given
6) Adaptive chosen cipher text attack: Choice of cipher text may depend on plaintext received from previous request

Substitution Ciphers: Each symbol in plaintext is replace by another symbol according to some fixed permutation . Key k (e=d) in K maps plaintext to ciphertext. Keyspace is all possible mapping : 26! = 4.03x10^26

Bruteforce Key search: Success average K/2 keys
For each key "k" in K (set of keys)

- Calculate m' = Dk(c)

If m' looks like the real message --> success , else continue

Language Redundancy and cryptanalysis: Monoalphabetic ciphers retain the relative letter frequencies of the english alphabet. This means that letters like A, E, I, O, T are most often used.

Shift/Caesar Cipher: Simple substitution cipher where each letter is replaced by another

Meaning of Mod: Can be used as a binary operator or congruence relation

1) Binary Operator : Modulus (math) . It is the remainder of a division.
    a. If r=a mod n , then a = (k*n) + r
2) Congruence relation: expresses that 2 arguments have the same remainder w/r to given modulus
    a. A = B (mod n) , where n divides (A-B)
        i. This implies: (A mod n) = (B mod n)

Example: 7 == 4 (mod 3) expresses that both 7 and 4 have a remainder of 1 when divided by 3 .

Polyalphabetic Substitution: Substitution alphabet more resistant to frequency analysis because it uses multiple substitutions.

Transposition Cipher: In a Transposition cipher, the letters are just moved around. The letters or words of the plaintext are reordered in some way, fixed by a given rule.

Drawback: transposition ciphers that operate on characters have the same letter frequency as original plaintext. Solution to this is to do multiple transpositions to eliminate the ease of simply guessing the column order.

Block Cipher: Plaintext message is broken into fixed length blocks before encryption. One block is processed at a time.

Stream Cipher: Block length is one. Requires limited buffering of data.

REVIEW:

HAC

- Chapter 1
  - Sections: 1.2, 1.3.1, 1.4, 1.5.1, 1.5.2, 1.5.4, 1.8.1, 1.8.2, 1.8.4, 1.13
- Chapter 7
  - Sections: 7.2,7.4

1) Public key encryption
2) Hash
3) Message authentication code (MAC)
4) Digital signatures

Lecture 3

Thursday, January 31, 2019 6:00 PM

Stream Ciphers: Simple Classification

A stream cipher nis a symmetric key cipher where plaintext digits are combined with a random cipher digit stream (known as a keystream). In the stream cipher, each plaintext digit is encrypted one at a time with its corresponding digit in the keystream.

1) Vernam Cipher (One time pad): is an encryption technique that cannot be cracked because it employs the use of a one time pressured key the same size or longer than the message being sent. It is defined on the alphabet {0,1} meaning it is a bit (binary message).

2) Syncronous Stream ciphers : The key stream is generated independently of the plaintext and of the cipher

3) Self synchronizing stream cipher: The key stream is generated as a function of the key and a fixed number of previous cipher text digits. The receiver will automatically synchronize with the keystream generator after receiving N ciphertext digits, making it easier to recover if digits are dropped or added to the message stream.

Unconditional Security

One time pad is unconditionally secure against a cipher text only attack. Observation of the cipher text provides no information to an adversary. This means that even **after** looking at the ciphertext, your uncertainty of the plaintext should be the **same as it was before** you saw the ciphertext.

Important key points:

- The key stream should never be reused.
  - **Vulnerability**: Ciphertext 1 (XOR) Ciphertext2 = Plaintext1 (XOR) Plaintext2
- The key stream should always be random: if a attacker can guess the key or limit the number of possibilities, then they could discover the plaintext.

Probabilities:

1. Key bit being a 0 or 1 = 0.5
2. Plaintext bit being 0 or 1 = (1-x) if the other is x
3. Ciphertext bit being 0 or 1 = 0.5x + 0.5(1-x) = 0.5

Generating Random keys (Keystreams)

Most cryptographic ciphers take a fixed length key called a seed and produces an unbounded bit stream. It must generate statistically random numbers , meaning it passes a standard test for randomness.

Definition of Crytographically secure: It should be computational impossible to predict the next bit given a complete history of past bits generated . This is known as the **next bit test**. As well, the number of 0 and 1 should be equal/ **balanced** given a statistically significant sample size.

Definition of Computationally secure: It is computationally infeasible for your adversary to defeat the encryption given his computational resources given the best attack known.

RC4 Encryption has a variable length key : 40-2048 bits .

RC4 Algorithm

Initialization

```
for i from 0 -> 255:
        S[i] : = i
        T[i] : = key[ i mod length(key)]
endfor
j : = 0
for i from 0 -> 255
        j : = (j + S[i] + T[i]) mod 256
        swap(S[i] , S[j] )
endfor
```

Keystream

```
i : = 0;
j : = 0;
while GeneratingOutput:
        i : = (i+1) mod 256
        j : = (j + S[i]) mod 256
        swap(S[i] , S[j])
        k : = S[(S[i] + S[j]) mod 256 ]
        output k
endwhile
```

RC4 Vulnerabilities: WEP attacks (aircrack-ptw) can break 104 bit RC4 keys of WEP-128 in less than a minute.

ChaCha20 Stream Cipher :
ChaCha20 Stream Cipher : If you never use the same nonce and key twice, it can be treated as a one time pad

- 256 bit key
- 96 bit nonce
- 32bit block counter
- Outputs 64byte block of keystrea
  Outputs 64byte block of key stream and increments block counter each time
- Plaintext is XOR'ed with keystream

Encryption: Ciphertext = Plaintext (XOR) ChaChaStream(key, nonce, counter)

Decryption: Plaintext = Ciphertext (XOR) ChaChaStream(key, nonce, counter)

Entropy Sources: Source for the randomness of the nonce

1) Software (random number generator) and Hardware (HD seek time)

2) Sound/video input (microphone atmospheric noise into bits)
3) Elapsed time between particle emission during radioactive decay...

Lecture 4: Block ciphers

Sunday, February 17, 2019 10:16 PM

Used commonly for :
1) Data confidentiality
2) MAC Addresses
3) PRNG (Random number generators)
4) Authentication

Efficient software and hardware implementations can be used in low resource devices

Block Cipher Definition

A function that maps n-bit plaintext blocks --> n bit cipher text blocks
where n=block length
Ex: DES = 64 bits, AES = 128 bits

It is a parameterized by k-bit key K (chosen at random)

Encryption Algorithm: C = E(M,K)
Decryption Algorithm: M = D(C,K)

If K is k bits long , **number of keys in 2^K** therefore if you have a 3 digit pin, the number of keys is 2^3 = 8

** Important points about block-ciphers:
1) Encryption function must be one-to-one because the encryption function is a bijection. Each key potentially defines a different bijection
2) If each k-bit key is equiprobable, each key defines a different bijection , where the entropy of key space is k.

Rule for E^K : Must be a bijection as the process must be reversible (decrypted)

Bijection: One to one && Onto
One to one: Each element in X maps to **at most** one element in Y where X=plaintext and Y=ciphertext
Onto: each element in Y maps to at least one element in X

Practicality (An attacker has access to all transmitted ciphertext)
A block cipher is **totally broken** if the key is recoverable
A block cipher is **partially broken** if some plaintext is recoverable

Complexity of Attacks: Types

1) Data Complexity: expected number of required input data units (cipher text blocks)
2) Storage Complexity: Expected number of required storage units
3) Processing complexity: Expected number of required operations on data

Computationally secure if:
1) Data complexity is 2^n (depending on block size)
2) Processing complexity is 2^k (depending on key size)

Modes of Operation

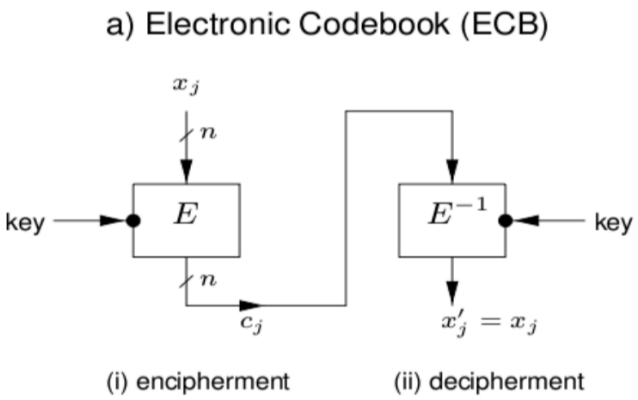
1) **Dividing messages**
2) **Padding the last block**

Basic Modes: ECB, CBC, OFB, CFB

Initialization Vector (IV) : A block of data used in addition to the input message, which randomizes the encryption process

ECB(Electronic Code Book):

- Encryption: $c_j \leftarrow E_K(x_j)$
- Decryption: $x_j \leftarrow E^{-1}{}_K(c_j)$



Identical plaintext under the same key results in the identical cipher text . Blocks are encrypted independently of other blocks. Bit errors in single cipher text affect decryption of that block only.

Weakness of ECB:

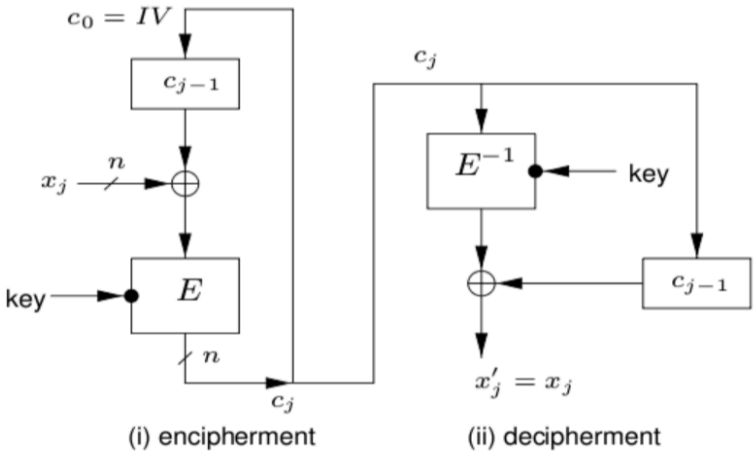
- Does not hide data patterns
- Malicious substitution of cipher text is possible

Best not to use when:

- Multi-block messages
- Keys are reused for more than a single block

CBC (Cipher Block Chaining):

- Encryption: $c_o \leftarrow IV, c_j \leftarrow E_K(c_{j-1} \oplus x_j)$
- Decryption: $c_o \leftarrow IV, x_j \leftarrow c_{j-1} \oplus E^{-1}{}_K(c_j)$



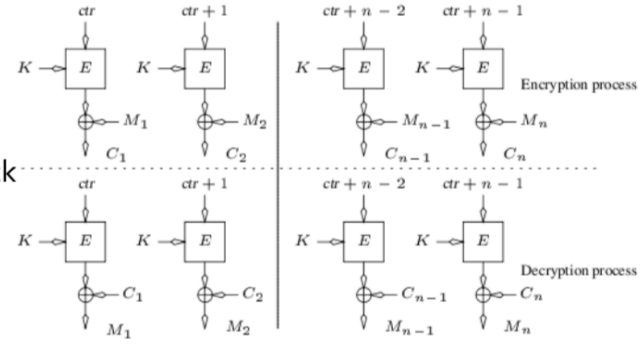
Properties of CBC:

- Chaining causes cipher text to depend on all preceding plaintext
- Same key and IV and plaintext results in identical ciphertext
- Random access to encrypted data not possible
- IV Must be integrity protected otherwise attackers may make predictable bit changes to the 1st block

- Single bit error in cipher text (Cj) affects decryption of blocks Cj , Cj+1
- It is self synchronizing: errors (including loss of blocks) in Cj but not in Cj+i therefore Xj+2 correctly decrypted

CTR (Counter Mode)

- Included with AES in 2001
- Proposed: Diffie & Hellman in 1979
- CTR must be different for each block
- An IV/nonce value is also used with the counter for uniqueness (concat/addition/xor)



- It is software and hardware efficient, meaning it can encrypt different blocks in parallel on a multiprocessor CPU
- Preprocessing exists, meaning the encryption can be done offline and when message is known, do the XOR
- Random access: Decryption of block can be done in random order, which is very useful or HD encryption

Block cipher - Design Components

1) **Avalanche effect:** A slight change in the input (ex: flipping single bit) has significant changes to the output. This is a desired property of block ciphers and cryptographic hash functions
2) **Substitution cipher:** Plaintext characters are replaced by other characters, where there is mapping between plaintext and ciphertext
3) **Transposition Cipher**: With a fixed period (t) , encryption involves grouping plaintext into blocks of t characters and applying to each block a single permutation (e) on the numbers 1--> t
    - Transposition adds **diffusion** (spreading out bits so redundancy in plaintext is spread over ciphertext)
    - Substitution adds **confusion** (makes relationship between key and cipher text as complex as possible)
4) **Product Cipher:** Execution of 2 or more simple ciphers in sequence such that the final product is cryptographically stronger than its individual components (Best when alternating between substitution & transposition)

Block cipher design approaches:
1) A round function is used so that it has a strong avalanche effect and is bijective
2) Repeated multiple times (N rounds)

Substitution Permutation (SP) Network

- Permutation is a simple transposition
- SP is faster than Feistel in a multi processor CPU

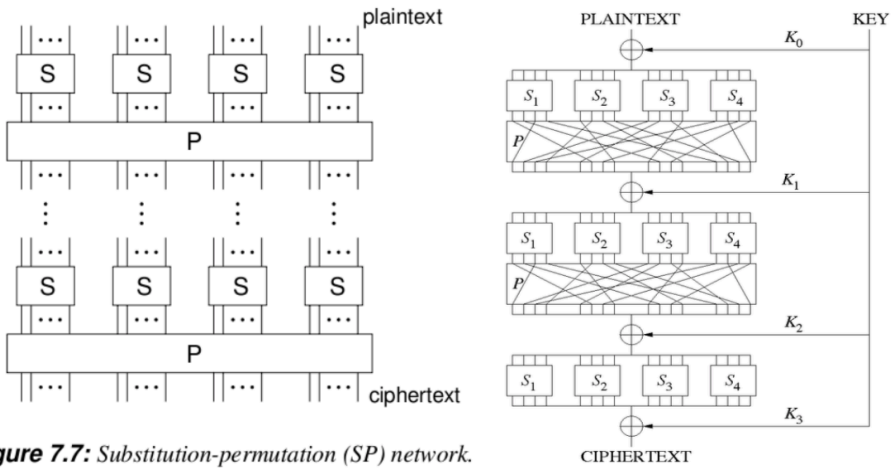


Figure 7.7: *Substitution-permutation (SP) network.*

Feistel Structure

It consists of a number of identical rounds of processing where in each round, a substitution is performed on half of the plaintext block, followed by a permutation that interchanges two halves
The original key is expanded so a different key is used in each round.

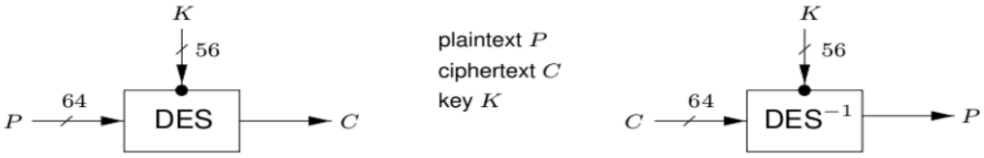
Structure parameters

1) Block size and key size : Larger means greater security, but may slow down the encryption and decryption speed
2) Number of rounds : More rounds means more strength . Typically 16 rounds.
3) Subway generation algorithm and round factor: Greater complexity should lead to greater cryptanalysis resistance

DES (Data encryption Standard)

It is the most widely used symmetric cipher, and it see the festal structure.

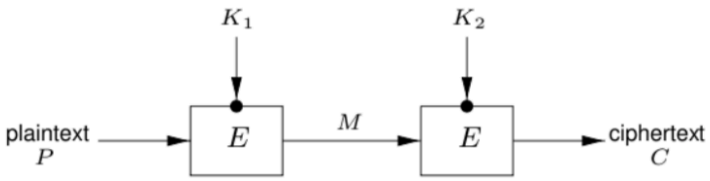
- Block length = 64 , Key size = 56, Number of rounds = 16



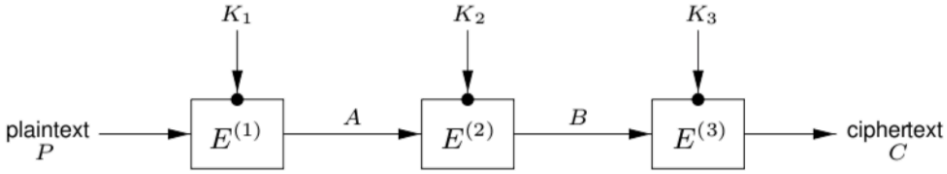
Controversy:

1) Relatively short key length at the time (56 bits too short for today)
2) Design criteria for internal structure of DES was classified

(a) double encryption



(b) triple encryption ($K_1 = K_3$ for two-key variant)



Meet in the Middle Attacks (DES)

- For double encryption, it reduces the key search from 2^2k to 2^k at the expense of additional storage 2^k
- It requires only a few plaintext-cipher text pairs

- $C = E_{K_2}(E_{K_1}(P))$
- $M = E_{K_1}(P) = D_{K_2}(C)$

- For Double-DES, assume we have (P,C)
  - Compute & store $E_{K_1}(P)$ for all possible K1 (2^{56} keys)
  - Compute $D_{K_2}(C)$ for all possible K2 (2^{56} keys)
  - Check the result with the previous step
  - If match occurs (K1, K2) are kept as candidate keys
  - False positives (2^{48} matches: $2^{112}/ 2^{64}$)
  - A second pair (P,C) reduces the false positive to 2^{-16}

Triple DES

- $C = E_{K_3}(D_{K_2}(E_{K_1}(P))$

Brute-force key search costs on the order of 2^112 operations, where 2 keys K1=K3

Alternatives to DES

1) Doubles DES : No security advantage
2) Triple DES : Secure but slow
3) AES (Advanced Encryption Standard) - Block length of 128 bits and key bit length of 128, 192, and 256 bits are used
    - AES-128 (n=10), AES-192 (n=12) , AES-256 (n=14)

AES (Advanced Encryption Standard) - High level description

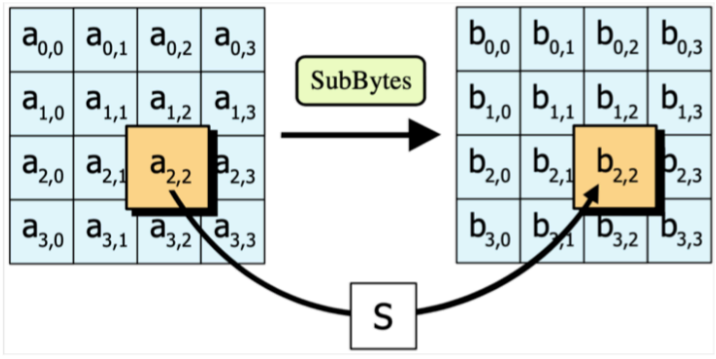
Input block is 128bits in length. Look at it like a 4x4 square matrix of bytes where 4x4=16 bytes and 1bytes=8 bits . Arranged from B0-> B15 . Uses round keys that are result of a key expansion. Each round key is 128 bits.

1) State = X (input) , KeyExpansion (round keys are derived)
2) Initial round
    a. AddRoundKey(State, Key0) : State (XOR) the expanded key
3) Rounds (Nr -1 times)
    a. SubBytes(State, S-box) : A substitution step, where each byte is replaced with another according to lookup table
    b. ShiftRows(State) : A transposition step, where each row of the state is shifted cyclically a number of steps
    c. MixColumns(State): A mixing operation where it operates on the columns of the state, combining the 4 bytes in each column
    d. AddRoundKey(State, KeyR)
4) Final Round (No MixColumns)
    a. SubBytes(State, S-Box)
    b. ShiftRow(State)
    c. AddRoundKey(State, KeyNR)
5) Y (output) = State

Operations

SubBytes: Each byte in the state is replaced with its entry in a fixed 8-bit lookup table S (Substitution box or S-Box). This operation provides non-linearity therefore makes it resistant to linear and differential cryptanalysis

- Minimizes the correlation between linear transformations of input/output bits
- Minimize the difference propagation probability



(AES) S- Box: A 16x16 matrix is known as an S-Box. It is based on modular arithmetic with polynomials. It can be defined algebraically, not randomly.
Forward S-Box = Encryption
Inverse S-Box = Decryption

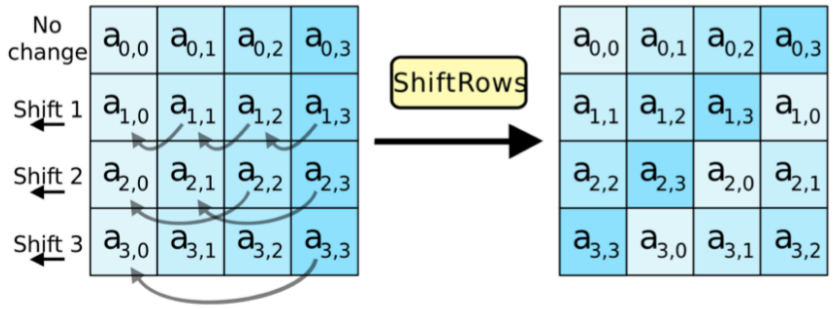
AES S-Box. The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value 0x9a is converted into 0xb8.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

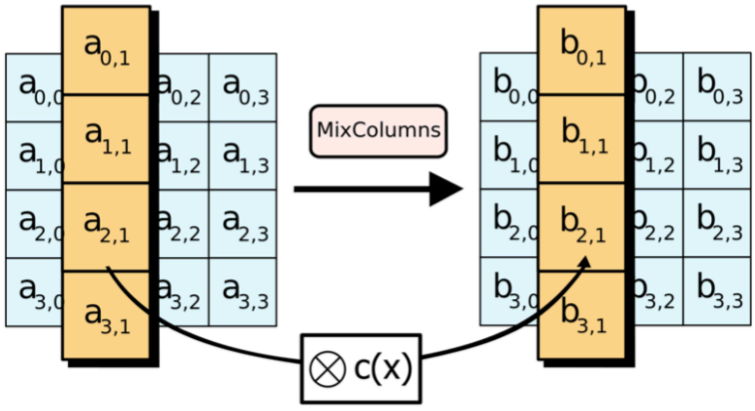
Inverse S-Box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

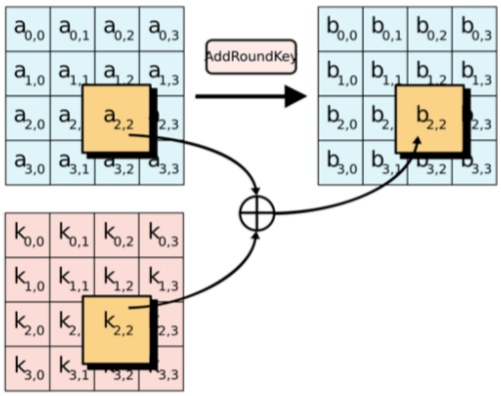
ShiftRows: Each column of the output state ShiftRows is composed of bytes from each column of the input state. This step is done to avoid the columns being encrypted independently, in which case AES degenerates into four independent block ciphers.



MixColumns: Each column of the state is multiplied with a fixed polynomial C(x) . Each input byte affects all four output bytes. Together (ShiftRows, MixColumns) provides **diffusion.**



AddRoundKey: The round sub-key is combined with the state using bitwise XOR. For each round, a sub-key is derived from the main key using Rijndael's key schedule. Each subway is the same size as the state.



Possible Attacks on AES:

1) 9, 10, 11 round versions . However full AES-256 uses 14 rounds.
2) Key recovery Attack: faster than brute force by 4x

**** READ Lecture 4.5

Lecture 5: Domain Fronting

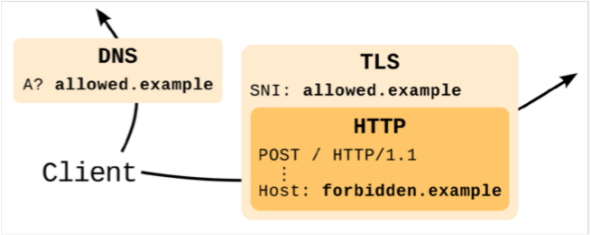
Sunday, February 17, 2019 11:40 PM

Definition: Domain fronting circumvents internet censorship by obfuscating the domain of a HTTPS connection. This allows the user to connect to a blocked service via DNS, IP or deep packet inspection.

It uses different domain names at different layers of communication. In HTTPS requests, the destination name appears in the following:

1) DNS query
2) TLS Server Name Indication extension (SNI)
3) HTTP Host header

For regular connections, the same domain name is used. However for domain fronted connections, HTTP Host has the true destination domain.
The domain fronting uses different domain names at different layers. This is because the **front domain** which is allowed is used at the plaintext layers **visible to the censor** (DNS, TLS SNI). However, the **actual destination** is at the HTTP Layer, which is **not visible to the censor,** but visible to the frontend server that is receiving the HTTPS request.



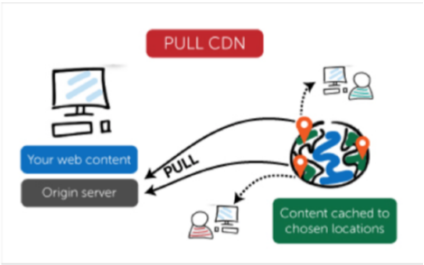
Fronting Techniques

1) With front domain name: needs TLS SNI.
   a. Server name indication (SNI) is a TLS extension for a client to indicate which hostname is attempting to connect. Multiple HTTPS sites are hosted from the same IP address, with different certificates.
2) Domain-less fronting: HTTPS target is an IP Address from a well known block

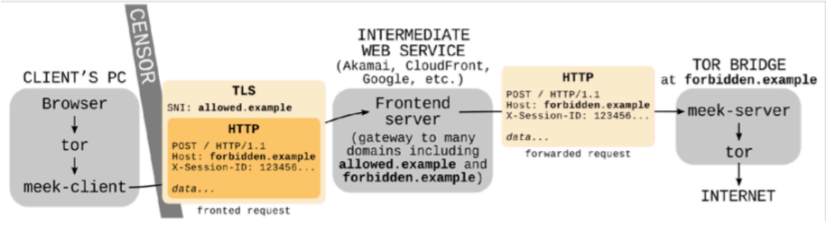
CDNs (Content Delivery Network) and Fronting

CDNs are essential for domain fronting because the CDN frontend server (edge server) on receiving a request for a resource not already cached, forwards the request to the domain found in the host header (origin server), where the origin server must be hostel from the CDN.

Origin Pull:



Integration with TOR (Meek Plugin) : Everything between the meek client and the meek server is an opaque data transport to TOR. The allowed host actually receives no traffic.



Meek Obfuscation:

1) Encodes the TOR traffic into HTT specifying the host name of the reflection server
2) Wraps that HTTP traffic in legitimate TLS connection to a server hosted in the same CDN cloud
3) The CDN server receives the connection, decrypts the TLS traffic, identifies the hostname in the HTTP header and redirects the traffic to the reflection server.
4) The reflection server reconstructs the original TOR traffic from the HTTP stream and sends the traffic to the TOR network, which routes it back to its true destination.

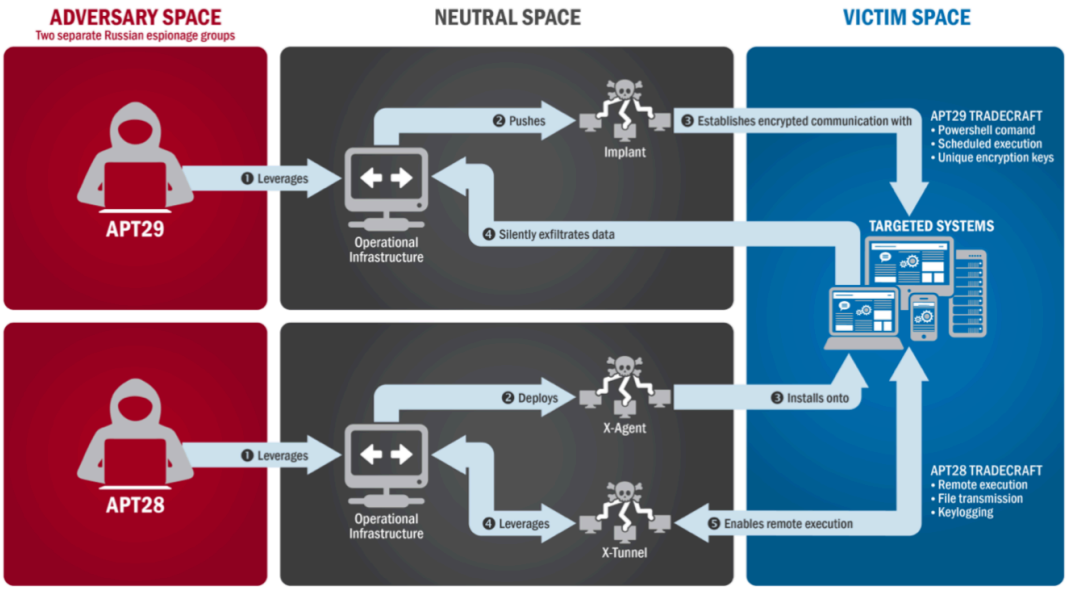
What the censor sees: Outgoing network connection that contains normal HTTPS POST requests for google.com on a google owned IP

What actually happens: Discretely passes client traffic through the reflection server to the TOR network

Weaknesses of Domain Fronting

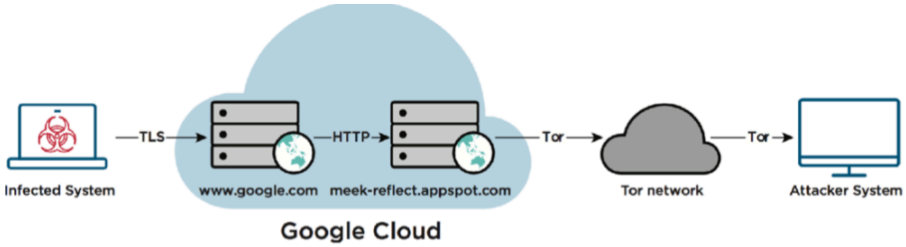
1) Relies on the censor not being crazy, and blocking the front domain. This is because the front domain is too important to block.
2) Needs the participation of a "friendly" CDN
3) TLS Interceptors can cause trouble
4) Traffic analysis (packet length, connection duration) can uncover true intentions
5) Who is paying for the bandwidth?

APT29: Russian Hacker group using domain fronting for their own advantage to hack political targets across the world.



The group uses domain fronting to do "Safe Data Exfiltration" which means hacking and extracting data without getting caught in the redirection of the data back to its attacker system, because domain fronting is used to mask the final destination.

The group infected systems and then the backdoor used Tor to provide secure, discrete remote access. It used Meek plugin to hide traffic, and then forwarded tor traffic to local ports. It was able to modify the registry to allow Remote Desktop Protocol on Windows.



Moral of the story: Security tools are often double edged swords that can be used for good but also for evil.