Symmetric Cryptography

Motivation

- Protect data confidentiality, integrity & availability on insecure network.
- Based on strong math foundation
- Use it every day, i.e. access website

Encryption

- Sequence of transforming a given string into a different one, semantically equivalent.
- It can be transformed back to original one.

Algo: $c = \mathcal{E}(m, k)$ encrypts plaintext m producing a $\boxed{\text{ciphertext } c}$

Algo: $m = D(c, k')$ decrypts ciphertext c producing a $\boxed{\text{plaintext } m}$

Not necessarily $k = k'$, $k = \text{keygen}()$ → randomly gen.

Symmetric Encryption

- Def:

  The same key as one which created ciphertext shall be used to decrypt the ciphertext back

- Goal:

  Confidentiality

- Correctness:

  $D(\mathcal{E}(m, k), k) = m$

  Security:

  $\forall k', \ k' \neq k \rightarrow D(\mathcal{E}(m, k), k') \neq m$

  $\forall \ k' \neq k$ we will not yield initial plaintext!

- Examples:

  Caesar cipher, 3DES, AES
  Typical key length 128/256
  Good performance

- Doing a random permutation with mono-alphabetic substitution is better than Caesar Cypher!
- Caesar cypher space is way too small!
- Can still use symbols by frequency to decrypt mono

Perfect Secrecy

- Vernam cipher or One-time pad
- Idea: use properties of XOR ($\oplus$) to encrypt & decrypt

  $k = \text{Keygen}()$

  $c = \mathcal{E}(m, k) = m \oplus k$

  $m = D(c, k) = m \oplus k \oplus k = m$

- 01010111010111010101010 ← k
- $\mathcal{E}(1101010000111101011, k) =$ 1101010000111101011 $\oplus$
  0101011101011101010
  1000001101100000001

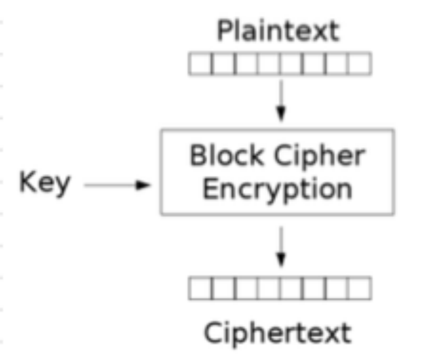
- Knowing the ciphertext tells u nothing about the message
  - ∀ cipher E a key that maps to any plaintext
- Key length is a problem:
  - msg length = key length
- Key can be used only once!!!

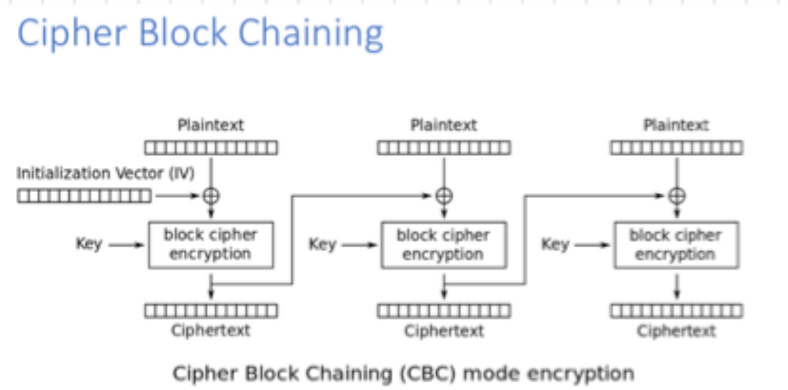
Note!

A OTP is secure against adversaries with unlimited computational power!

Block Cipher

- Recall length problems:
  - Agree on short key & generate fixed-length permutations.
- Key used only once:
  - Use random value on each encryption (initialisation vector)
- Multiple variants of block cipher exists.





Cipher Block Chaining (CBC) mode encryption

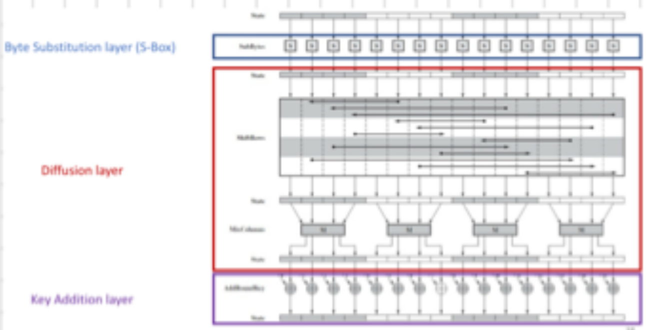
- A block cypher is secure if it is a good pseudorandom perm.
- Output of a secure cipher canno be distinguished from a random permutation!
- Use confusion:

  make connection between cipher & key as complicated as possible

Note! It is not possible to encrypt messages of any size with one call to a block cipher!

AES

- Advance Encryption Standard
- Block length = 128 bits
  - msg not multiples are padded.
- Key length = 128, 192, 256 bits
- # of rounds = 10, 12, 14 → each round consists of layers
- Focus on AES128



$AES(K,M)$ where $|M|=128$, $|K|=128$

$(k_0,\dots,k_{10}) = KeySchedule(K)$ where $|k_i| = 128$

$s = M \oplus k_0$

Byte Substitution

For $r=1$ to 10

$s = S(s)$

$s = ShiftRows(s)$

if $r \le 9$:

$s = MixCols(s)$

Diffusion Layer

Key additional layer

$s = s \oplus k_r$

return s

Byte Substitution Layer (S-BOX)

^ S-Box has the following properties

- **Identical** same S-boxes per round
- **Nonlinear** $S(s) + S(s') \neq S(s + s')$
- **Bijective** $\exists$ 1 1-to-1 mapping of input & output bytes

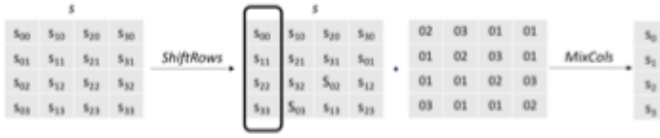
- Implemented as a lookup table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Diffusion Layer

^ Diffusion over all input state bits

- ShiftRows provides permutation of data
- Linear $SR(s) + SR(s') = SR(s + s')$
- MixCols provides mix of blocks



Recap

^ S-Boxes provide confusion

^ ShiftRows & MixCols provide diffusion

^ Key additional layer protects against inverting attacks

^ AES is secure because:

- Pseudorandom permutation is very good
- Got no serious cryptoanalysis attack so far!

Note!

AES has no known practical attacks against it

Hash and MAC

^ Hash Functions:

- Common building blocks of security:
  - * Compare by hash, virus protection, CIP, storing passwords, ingredient many crypto
- DEF:

  A function H that takes an arbitrary block of data & returns a fix-size bit string.

- Goal:

  Integrity.

E.g. $\mathcal{H}$('fox') = b99c21513df8309c021977902526e2f3881758a1
E.g. $\mathcal{H}$('The red fox jumps over the blue dog') = 0504e140d01c8c8cad73ac18873fd7944e236f90
E.g. $\mathcal{H}$('The red fox bumps over the blue dog')= 78e883a20497df7af2ba0d4dff062a26137c024d
E.g. $\mathcal{H}$('The red fox jumps over the blue dogs')= 8ee7cb3ea20307bbb68bee60fd1c3068aa28b455

^ Cryptographic Hash Function requirements:

- Pre-image resistance (one-way):
  - Given $h = H(m)$ is infeasible to find m.
- Second pre-image:
  - Given m_1 is infeasible to find $m_2 \neq m_1$ if $H(m_1) = H(m_2)$

This mens if: $h_1 = h_2 \rightarrow m_1 = m_2$

- Collision resistance:
  - It is infeasible to find $H(m_1) = H(m_2)$ & $m_1 \neq m_2$
  - This implies second pre-image.

^ Industry standards: SHA3

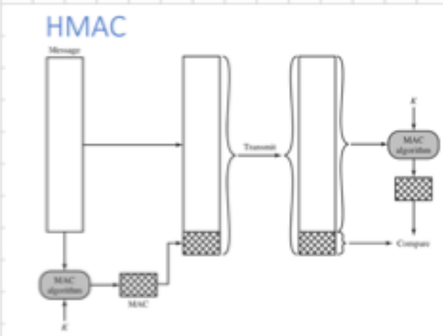
- If a DB is compromised:
  - If people have the same password we can see that!!!
  - That's why we concatenate a random string: SALT

UserID	$\mathcal{H}$(password)
brun	1977902526e2f3881758a1
rosg	73ac18873fd7944e236f90
rikj	ba0d4dff062a26137c024d
maca	68bee60fd1c3068aa28b45
debois	ba0d4dff062a26137c024d

UserID	Salt	$\mathcal{H}$(password|salt)
brun	4738295	3881758a11977902526e2f
rosg	3727283	jej48929dj38d833838ddj39
rikj	3838759	dkkeoe33392lj39d84939dk
maca	9048040	4849dj29d9ke93304kf94k4
debois	2872900	48d83jj9d2kk334449dk9s9

Message Auth Codes (MAC)

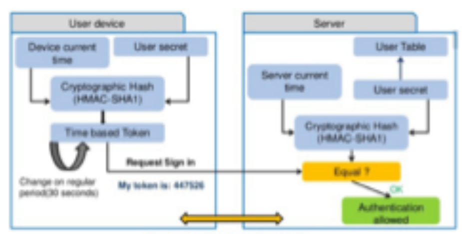
- Goal: integrity + authenticity ~~confidentiality~~
- Algo: tag = mac(m,k)
- Algo: d = ver(tag, m, k)
- Correctness: ver(mac(m,k), m, k) = true



MAC = $\mathcal{H}((k \oplus const1) \mid \mathcal{H}(k \oplus const2 \mid m))$

- const1 and const2 constants and public

Smart Token



Note!