The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

ECE568 Lecture 05: Introduction to Cryptography 2

Wei Huang
Department of Electrical and Computer Engineering
University of Toronto

Outline

- Review of cryptography basics
- Mid-Term information
- AES
- Block cipher
- Side channels

Summary of Previous Lecture

- **Cryptosystem**
  - P: plaintext space
  - C: ciphertext space
  - K: key space
  - $E = \{E_k : k \in K\}$: set of encryption functions
  - $D = \{D_k : k \in K\}$: set of decryption functions

For any $k \in K$, there exists a
$e_k \in E$ and corresponding $d_k \in D$, that satisfy:
for any $e_k : P \rightarrow C$, $d_k : c \rightarrow P$, plaintext $x \in P$, there is

$$d_k(e_k(x)) = x$$

A cryptosystem can be defined as a tuple:

$$(P, C, K, E, D)$$

Summary of Previous Lecture

- Shift Cipher?
  - $P=C=K=Z_{26}$. For $0 \le k \le 25$, $x,y,k \in Z_{26}$, define

$$e_k(x) = x+k \bmod 26$$

$$d_k(y) = y-k \bmod 26$$

Caesar cipher: k=3

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

|K| = ?

Summary of Previous Lecture

- Substitution Cipher?
  - $P=C=K=Z_{26}$
  - K is all possible substitutions π in the set {0,1,…,25}.
  - For any K, we define:

$$e_\pi(x)= \pi(x)$$
$$d_\pi(y)=\pi^{-1}(y)$$

  - Key space |K| = 26! $\approx 7.21 * 10^{26}$

Summary of Previous Lecture

- $R_1=L_0 \oplus F(k_1,R_0)$, $L_1=R_0$;
- $R_2=L_1 \oplus F(k_2,R_1)$, $L_2=R_1$;

……

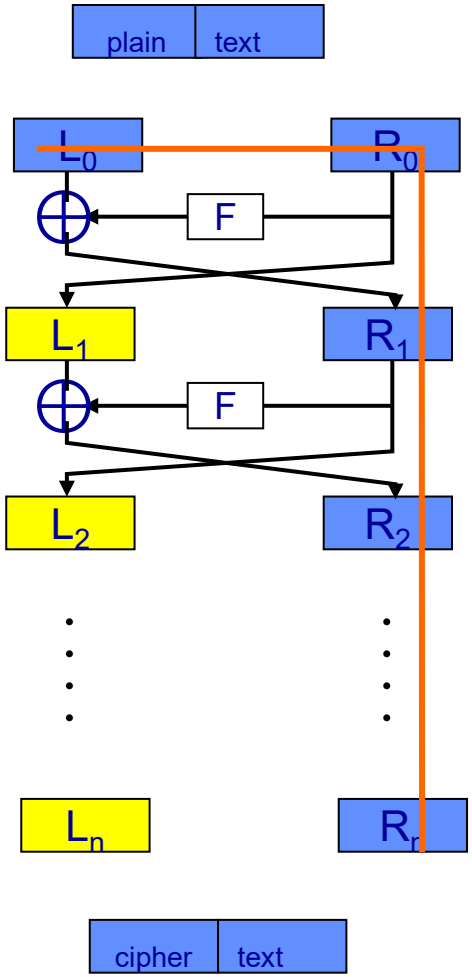
- $R_i=L_{i-1} \oplus F(k_i,R_{i-1})$, $L_i=R_{i-1}$;

……

- $R_n=L_{n-1} \oplus F(k_n,R_{n-1})$, $L_n=R_{n-1}$;

F does not have to be inversable:

- $R_{i-1}=L_i$, $L_{i-1}=R_i \oplus F(k_i,R_{i-1})$

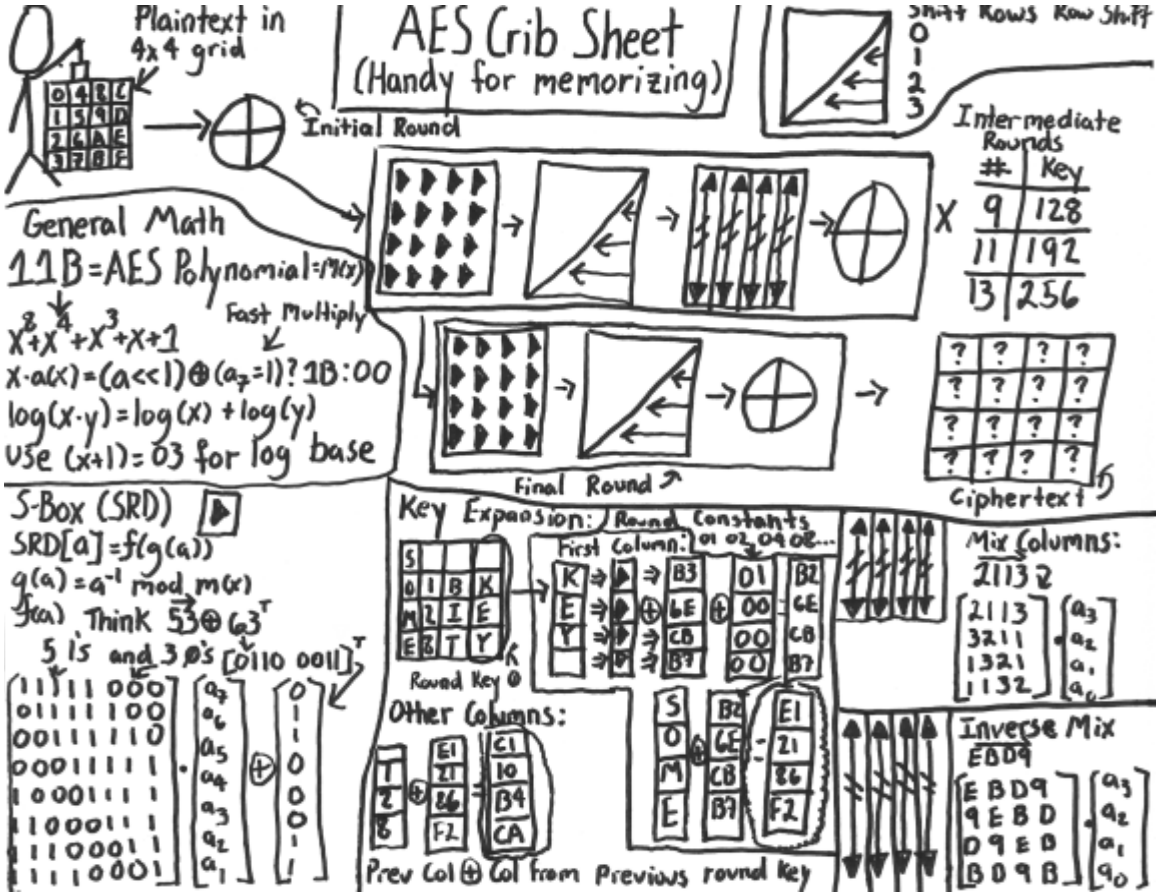


Mid-Term Exam Information

- Time: 2023-10-30 Monday, 18:30-20:30
- Venue: SF 3202
- Past Exams: check website
- Examples

Introduction to AES

- Advanced Encryption Standard



AES History

- In January 1997, NIST sent out a request for a new encryption standard to replace DES which was 20 years old at the time. A competition would be held among proposals to decide the winner. The requirements were:
  - AES should be publicly defined (it's workings would be known to all)
  - AES shall be a symmetric block cipher
  - AES shall be designed so that the key length may be increased
  - AES should be implementable in hardware and software
  - AES shall be freely available

Result of AES Contest

- 1998: NIST announced 15 candidate ciphers
  - Inviting the world help analyze
- 1999 August 20: Final round candidates:
  - MARS, RC6, RIJNDAEL, Serpent, Twofish
- 2000 Oct 2: RIJNADAEL won
  - After rigorous testing among many criteria, Rijndael, created by Joan Daemen and Vincent Rijmen from Belgium was selected as the winner and designated the AES cipher

Details on AES

- Has variable key and block lengths
  - Can take 128, 192 or 256-bit keys.
  - Can operate on 128, 192 or 256-bit blocks. Any combination of key and block length is possible.
  - Extensions exist to allow it to take block and key lengths of 160 and 224 bits.

- Each input block is used to form a matrix of bytes. The matrix always has 4 rows and has a variable number of columns depending on the size of the block. This matrix is called the "state".

Structure of AES

- Like DES, AES is based on rounds:
  - The number of rounds depends on the key length and block size

Block Size \ Key Size	128	192	256
128	10	12	14
192	12	12	14
256	14	14	14

- Each round consists of 4 stages:
  1. Byte Substitution Transformation
  2. Shift Rows Transformation
  3. Mix Column Transformation
  4. Round Key Addition

The Structure of AES

1. Byte Substitution (ByteSub) Transformation:
   - A non-linear byte by byte substitution. The same substitution is performed on every byte.
   - The substitution is based on a mathematical inverse and then an affine function (multiply + addition).
   - In practice, this is usually done via a 128 (2^8) entry lookup table called an S-box.

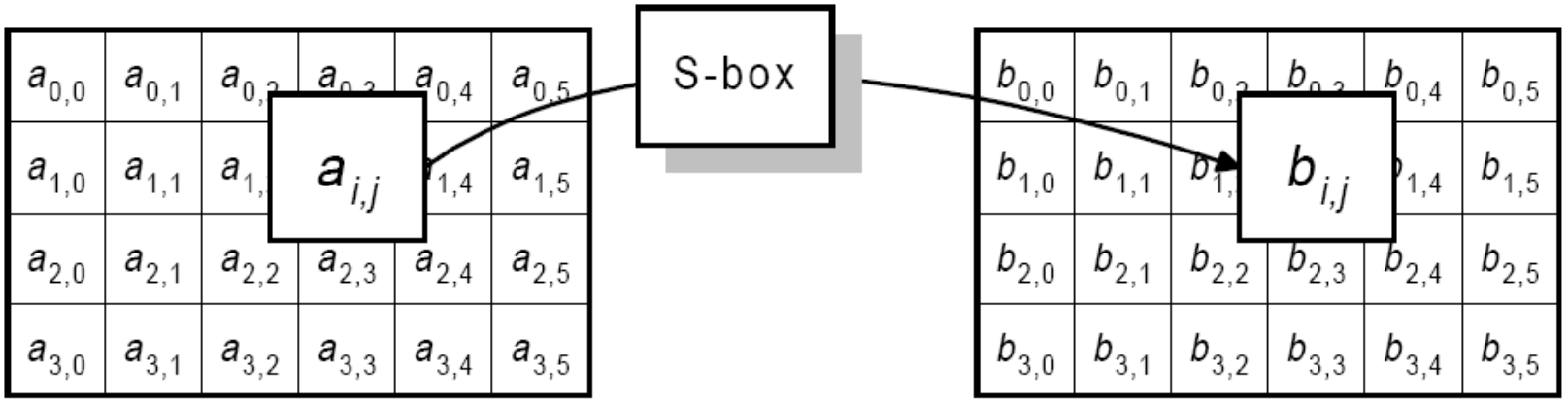
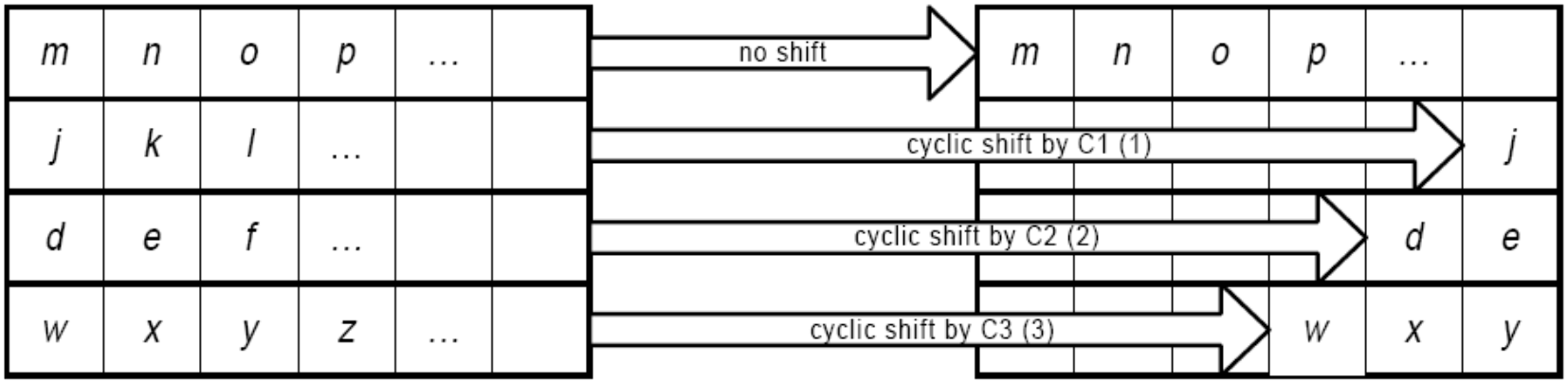


Image from the official AES Spec at NIST

The Structure of AES

2. Shift Row Transformation:
   - Each row in the state is then shifted (cyclically) by a specified offset based on the block size.



Size of shift based on block size (Nb):

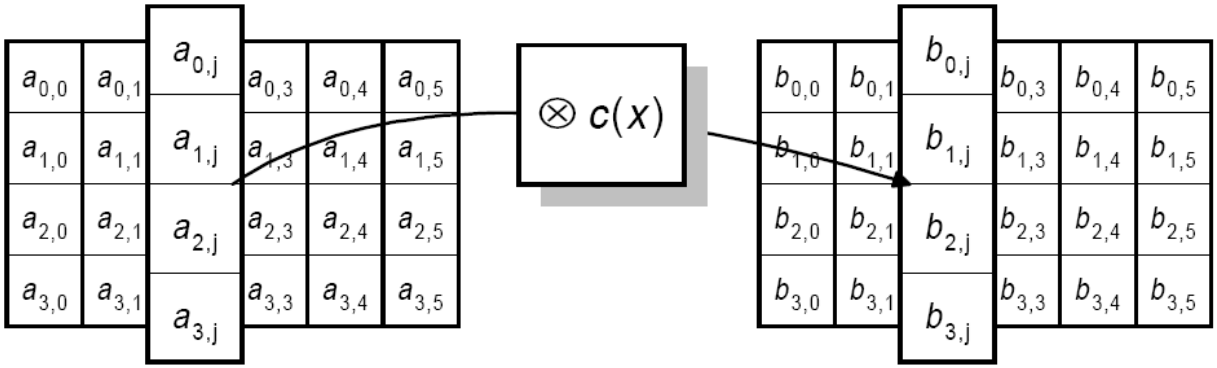
Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

Images from the official AES Spec at NIST

The Structure of AES

3. Mix Column Transformation:
   - Each column in the state is treated as a vector and the following matrix is applied to mix the elements of each column:

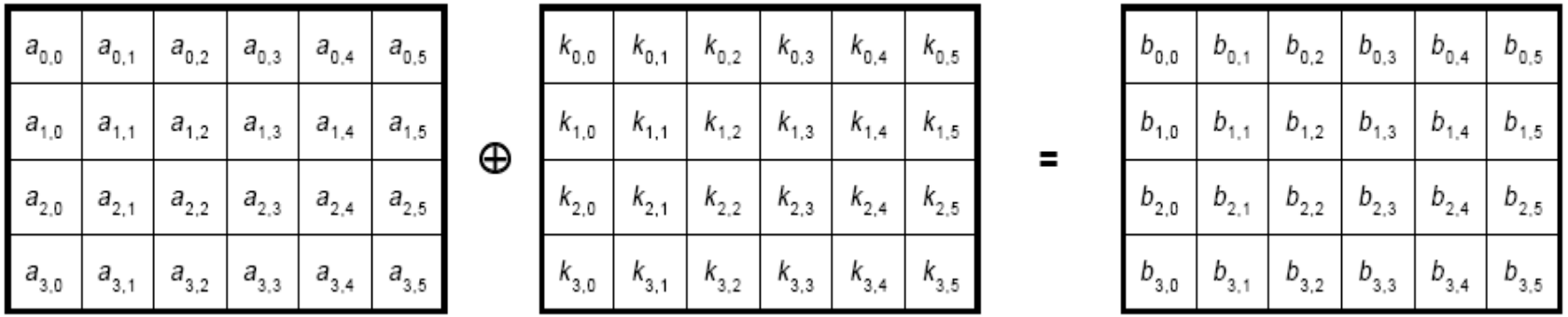
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$



Images from the official AES Spec at NIST

The Structure of AES

4. Round Key Addition
   - Like DES, round keys are generated from the original key for each round via a key schedule algorithm.
   - The round key is XOR'ed with the state to produce the input to the next round:



- Details on AES can be found in the official AES Specification (on Quercus).

Image from the official AES Spec at NIST

AES Today

- AES became effective as a standard May 26, 2002
- Today, AES is one of the most popular block cipher algorithms, available in many different encryption packages
- AES is the first publicly accessible and open cipher approved by the NSA for Top Secret information

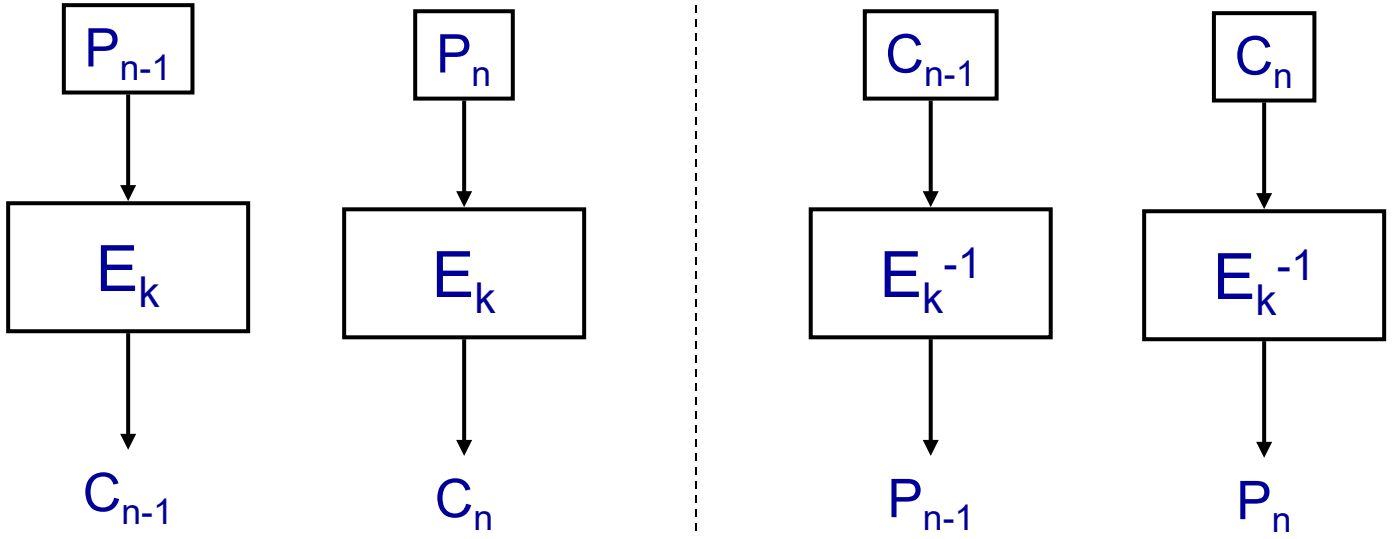
Block Cipher Encryption Modes

- Often these ciphers are used to encrypt data that is larger than a block. How do we use a block cipher to do this safely?

- There are several modes available and they can be compared in the following ways:
  1. Security: What are the security properties?
  2. Performance: What throughput can be supported?
  3. Error Propagation: What is the effect of a bit error during transmission of the cipher text?
  4. Error Recovery: Can we recover from a transmission error?
     - How much has to be retransmitted?

Electronic Codebook (ECB)

- ECB is the simplest mode:
  1. The message is broken into block-sized chunks.
  2. Padding is added to the last block.
  3. Each chunk is encrypted independently.



Electronic Codebook Properties

1. Security: The security of ECB is pretty poor. The adversary can add blocks, delete blocks or reorder them. Plain text blocks always encrypt to the same cipher text blocks revealing the macro-structure of the data.

Original Image

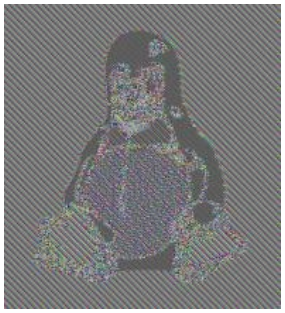


Image encrypted with ECB Mode

Image encrypted with a more secure mode

2. Performance: This mode is highly parallelizable as every block is encrypted independently. The throughput is essentially the speed of the block cipher.

Images from Wikipedia

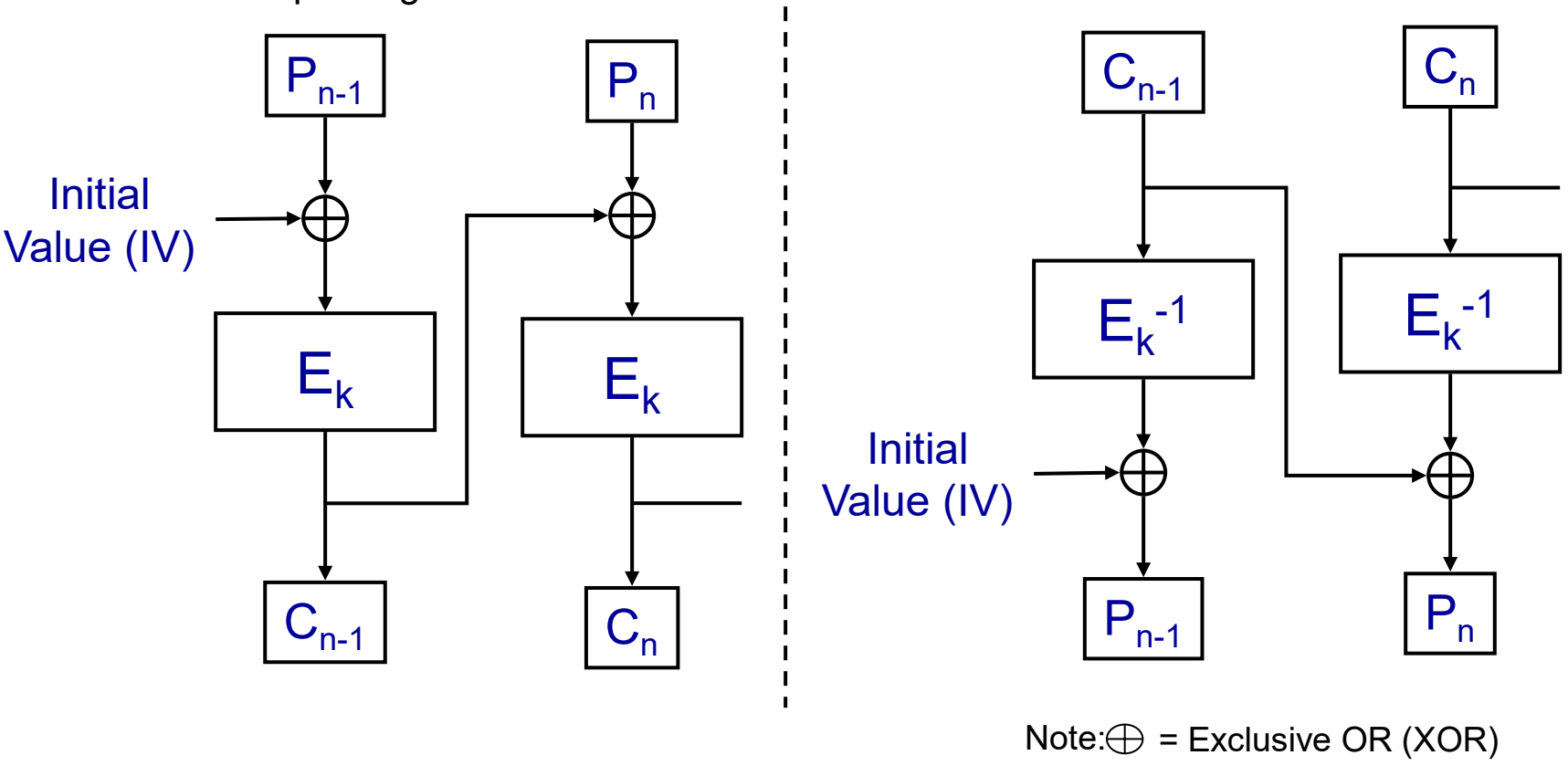
Electronic Codebook Properties

3. Error Propagation: Any transmission errors in the cipher text will only affect the corresponding plain text block. The plain text block will be changed in a completely random way.

4. Error Recovery: Recovery is easy. The error is only limited to the affected block, all blocks before and afterwards will be decrypted properly.
   - Only retransmit affected blocks
   - Does not stop decryption, just skip the bad blocks and keep going.

Cipher Block Chaining (CBC)

- CBC makes every block dependent on the cipher text of the previous block:
  - Note, (Initial Value) IV does not have to be kept secret, but should avoid repeating IV's



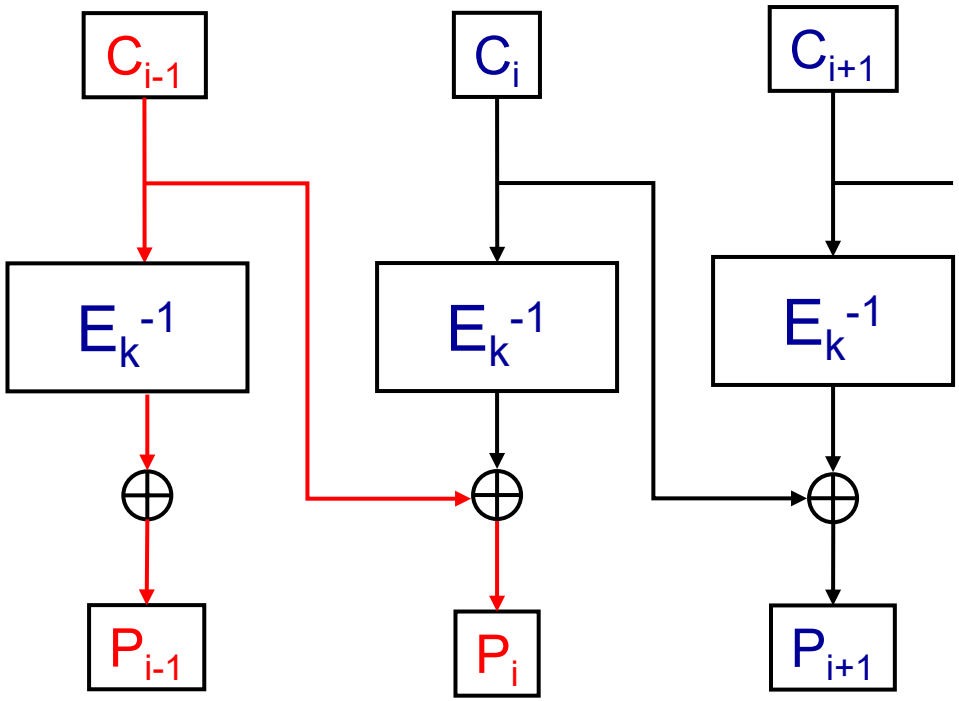
Note: $\oplus$ = Exclusive OR (XOR)

Cipher Block Chaining Properties

1. Security: Cipher Block Chaining has good security. Any **change in the PT** affects all blocks after it. Blocks that have the same plain text have different cipher text blocks.
2. Performance: No parallelism for encryption, must be sequential. Decryption can be parallelized.

3. Error Propagation: A transmission error only affects the current block and the following block.
4. Error Recovery: The receiver can drop the affected blocks and still continue decryption.

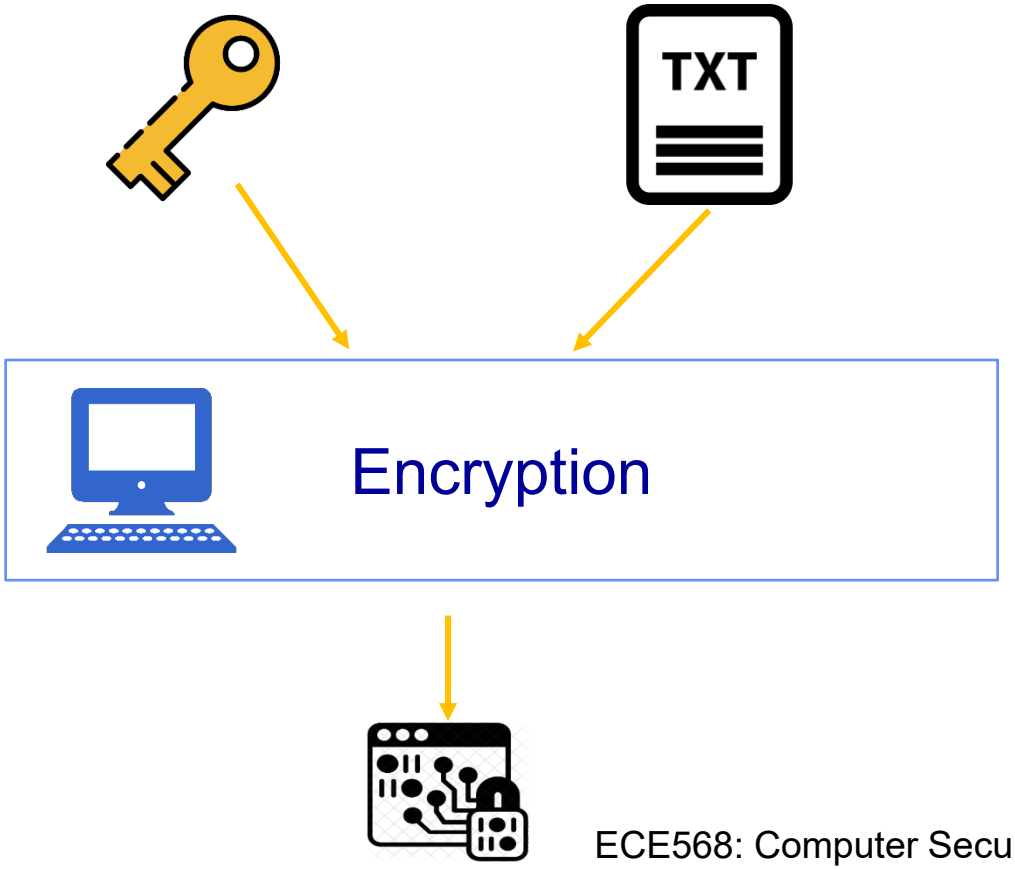
Cipher Block Chaining Properties



3. Error Propagation: A transmission error only affects the current block and the following block.
4. Error Recovery: The receiver can drop the affected blocks and still continue decryption.

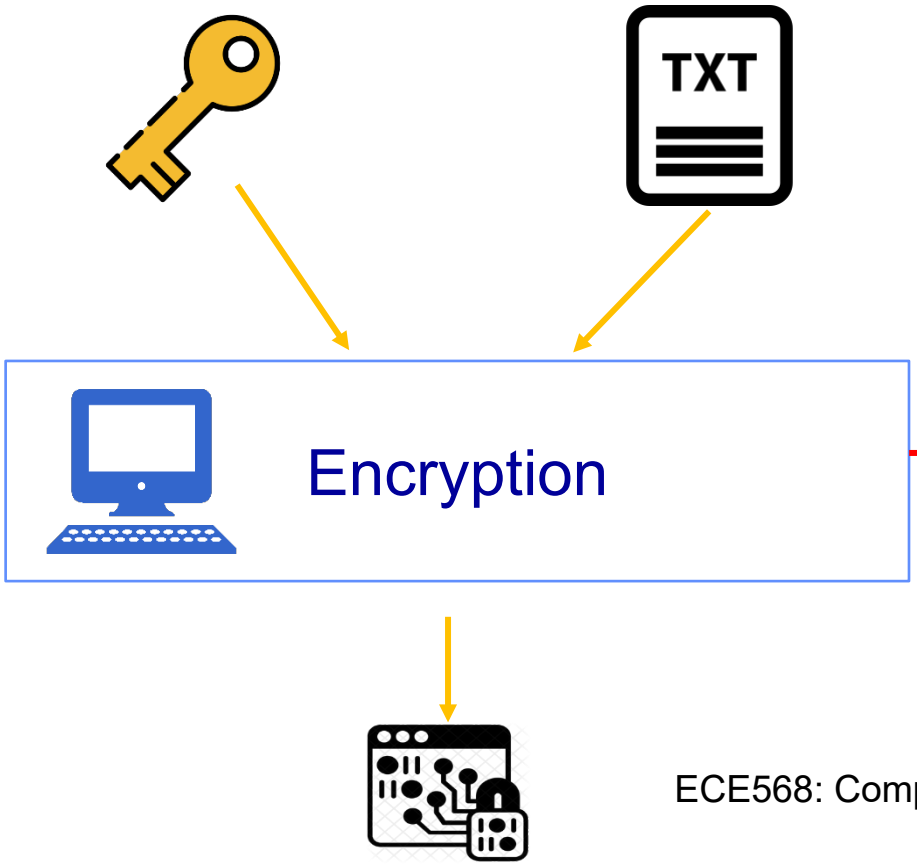
Ways Around Ciphers like AES?

- Attackers always cheat!



Side Channels

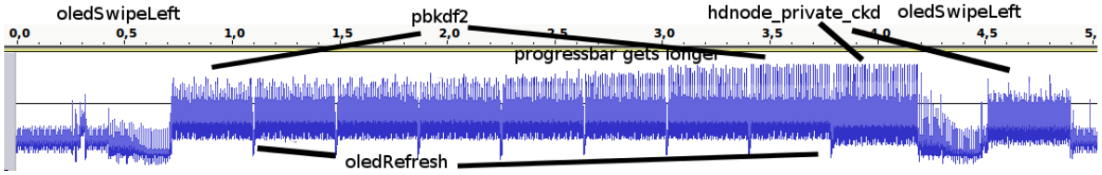
Side-channel cryptanalysis is any attack on a cryptosystem requiring information emitted as a byproduct of the physical implementation



- Time
- Power
- Noise
- Heat
- EM radiation
- …

Power Channel

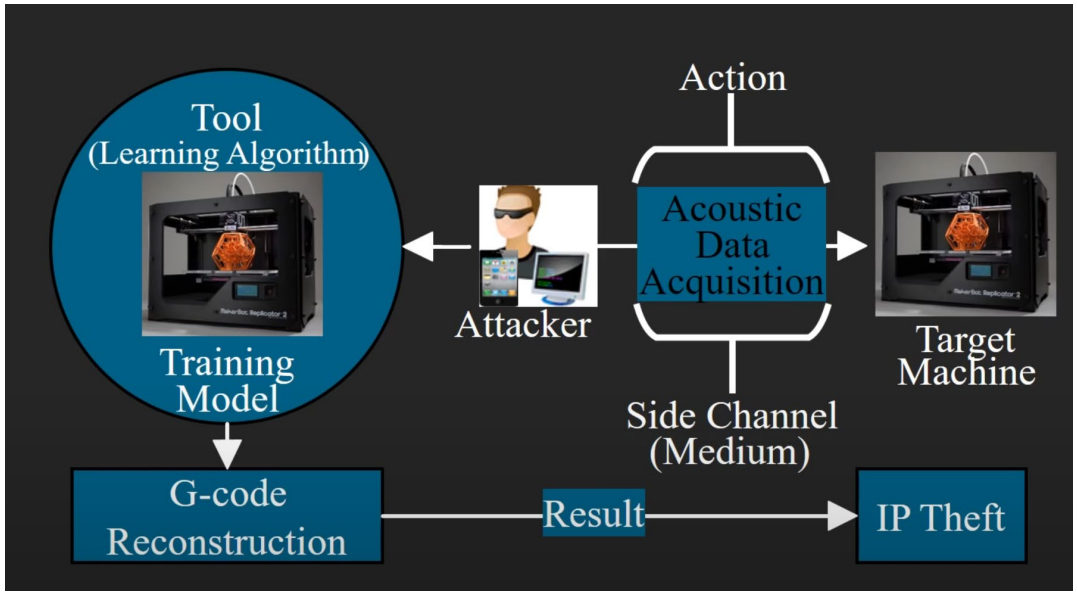
- Extracting private keys from hardware by analyzing its power usage

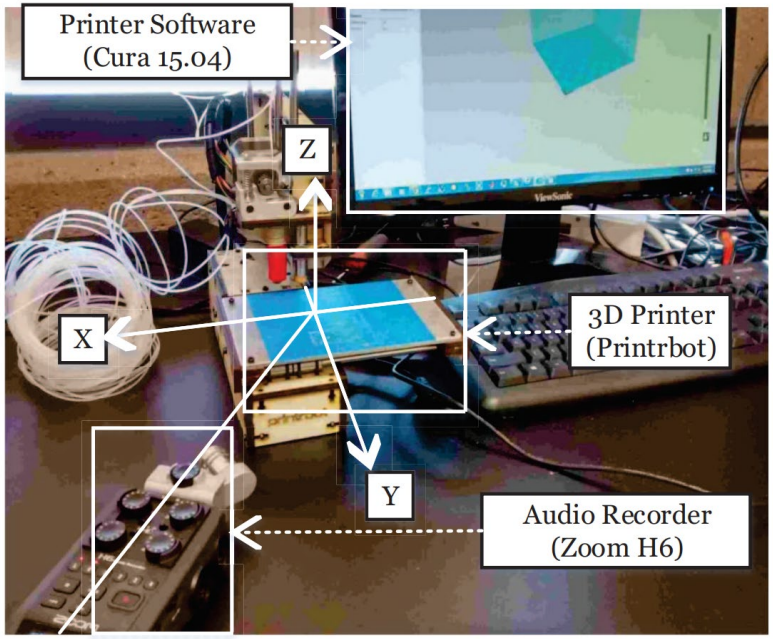


https://johoe.mooo.com/trezor-power-analysis/

Noise (Acoustic) Side Channel

- Extracting 3D-print model by recording printer noise

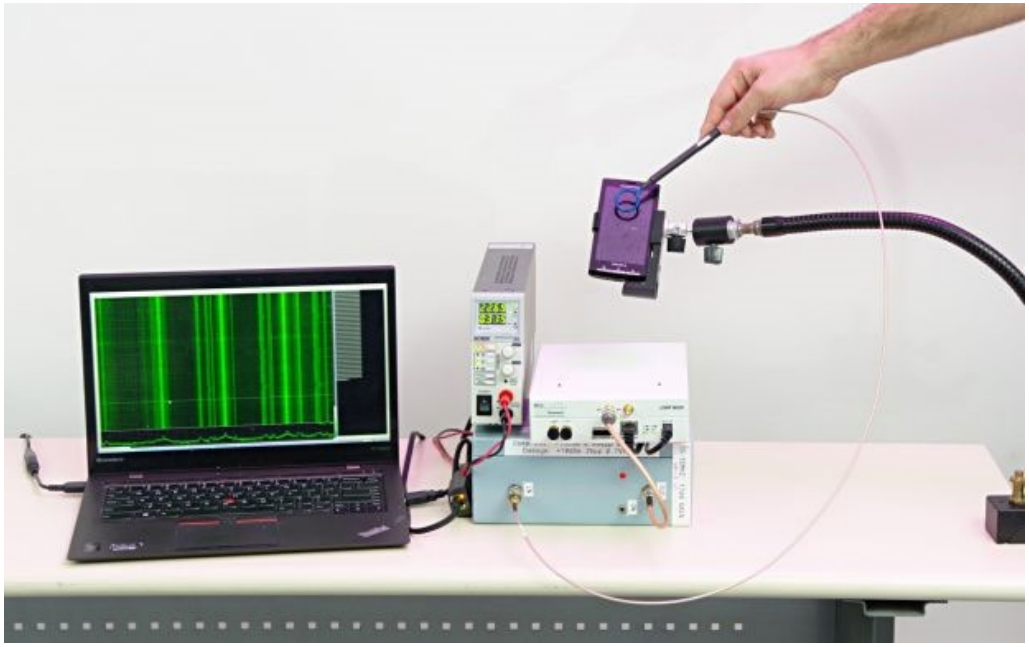




Paper: [Acoustic Side-Channel Attacks on Additive Manufacturing Systems]

Electromagnetic Side Channel

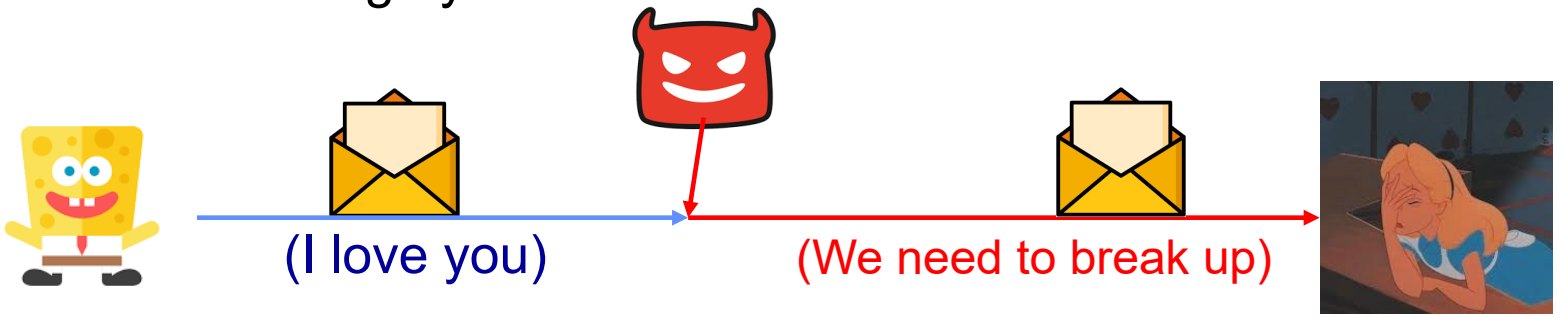
- Recovering secret keys of ECDSA on Android's standard cryptographic library by collecting EM data



Paper: ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels

Example of Timing Channel

- MAC
  - Message Authentication Code
  - Protect integrity



- MACs to verify updates for Xbox 360
  - Implementation allowed different rejection times (2.2 ms)
  - Attackers were able to exploit it and load pirate games

Exploiting a Naïve MAC Timing Channel

```
int naive_mac_check(char * test, char * correct) {
        return (strcmp(test, correct) == 0);
}

int strcmp(char * s1, char * s2) {
        while (*s1 != '\0' && *s1 == *s2) {
                s1++;
                s2++;
        }

        return ( *s1 - *s2 );
}
```

Revisit AES Implementation

- An AES Round
  - $(X^i, K^i) \rightarrow X^{i+1}$

$$
\begin{aligned}
T &:= \text{SubBytes}(X^i) \\
T &:= \text{SiftRows}(T) \\
T &:= \text{MixColumns}(T) \\
X^{i+1} &:= T \oplus K^i
\end{aligned}
$$

T-Table Details of AES

- The actual round computation in software, as proposed with Rijndael and now widely used:

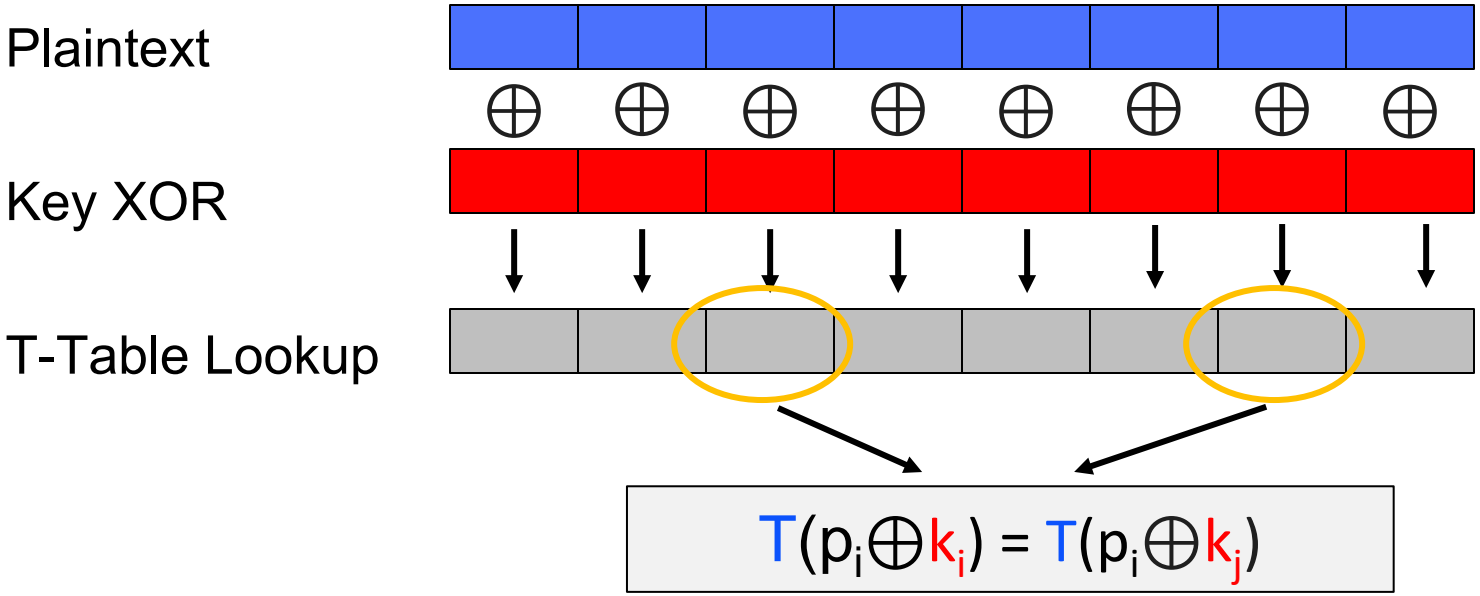
$$
\begin{aligned}
X^{i+1} = \{&T_0[x_0^i\ ] \oplus T_1[x_5^i\ ] \oplus T_2[x_{10}^i] \oplus T_3[x_{15}^i] \oplus \{k_0^i\ , k_1^i\ , k_2^i\ , k_3^i\ \},\\
&T_0[x_4^i\ ] \oplus T_1[x_9^i\ ] \oplus T_2[x_{14}^i] \oplus T_3[x_3^i\ ] \oplus \{k_4^i\ , k_5^i\ , k_6^i\ , k_7^i\ \},\\
&T_0[x_8^i\ ] \oplus T_1[x_{13}^i] \oplus T_2[x_2^i\ ] \oplus T_3[x_7^i\ ] \oplus \{k_8^i\ , k_9^i\ , k_{10}^i, k_{11}^i\},\\
&T_0[x_{12}^i] \oplus T_1[x_1^i\ ] \oplus T_2[x_6^i\ ] \oplus T_3[x_{11}^i] \oplus \{k_{12}^i, k_{13}^i, k_{14}^i, k_{15}^i\}\}.
\end{aligned}
$$

T-Table Implementation in AES

```
t0 =   Te0[(s0 >> 24)        ] ^
       Te1[(s1 >> 16) & 0xff] ^
       Te2[(s2 >>  8) & 0xff] ^
       Te3[(s3      ) & 0xff] ^
       rk[0];
t1 =   Te0[(s1 >> 24)        ] ^
       Te1[(s2 >> 16) & 0xff] ^
       Te2[(s3 >>  8) & 0xff] ^
       Te3[(s0      ) & 0xff] ^
       rk[1];
t2 =   Te0[(s2 >> 24)        ] ^
       Te1[(s3 >> 16) & 0xff] ^
       Te2[(s0 >>  8) & 0xff] ^
       Te3[(s1      ) & 0xff] ^
       rk[2];
t3 =   Te0[(s3 >> 24)        ] ^
       Te1[(s0 >> 16) & 0xff] ^
       Te2[(s1 >>  8) & 0xff] ^
       Te3[(s2      ) & 0xff] ^
       rk[3];
```

```
static const u32 Te0[256] =
{
  0xc66363a5U, 0xf87c7c84U, 0xee777799U, 0xf67b7b8dU,
  0xfff2f20dU, 0xd66b6bbdU, 0xde6f6fb1U, 0x91c5c554U,
  0x60303050U, 0x02010103U, 0xce6767a9U, 0x562b2b7dU,
  0xe7fefe19U, 0xb5d7d762U, 0x4dababe6U, 0xec76769aU,
  ...
  0x824141c3U, 0x299999b0U, 0x5a2d2d77U, 0x1e0f0f11U,
  0x7bb0b0cbU, 0xa85454fcU, 0x6dbbbbd6U, 0x2c16163aU,
};
```

Analysis of T-Table Lookups



$$T(p_i \oplus k_i) = T(p_i \oplus k_j)$$

$$p_i \oplus k_i = p_i \oplus k_j$$

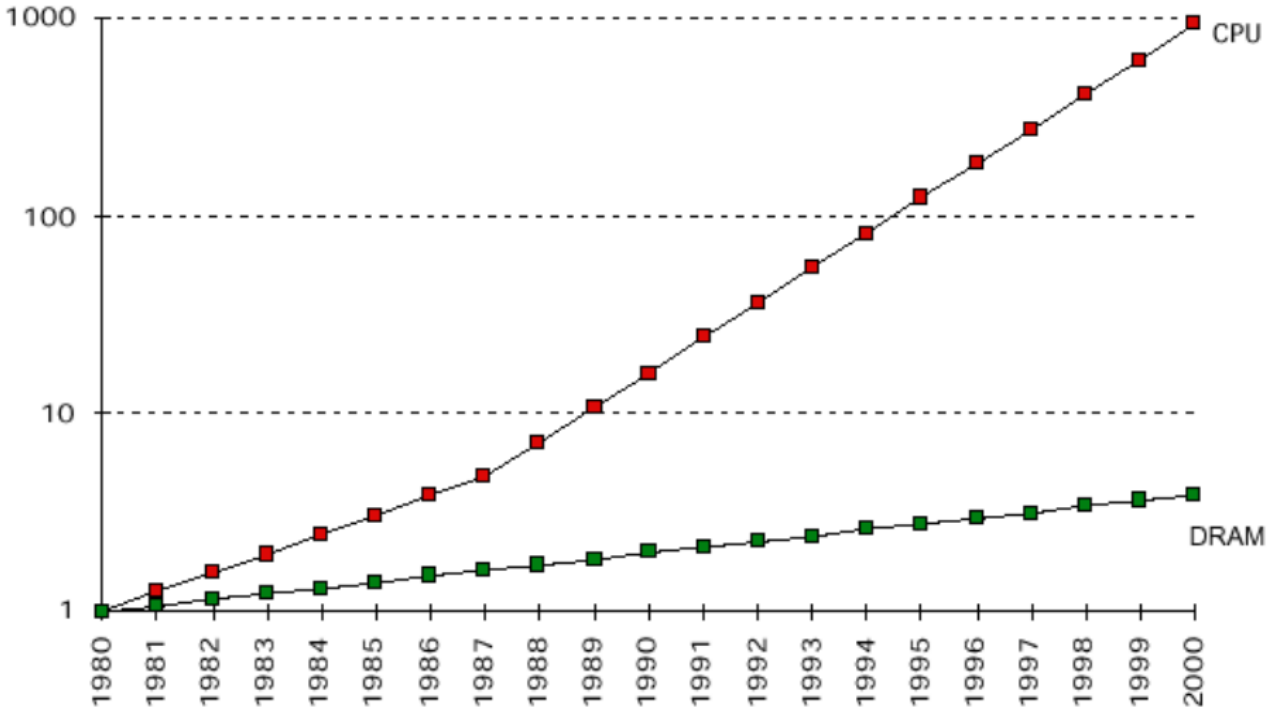
$$p_i \oplus p_i = k_i \oplus k_j$$

Analysis of T-Table Lookups

- How would we know when **x=y** in **T(x)**, **T(y)**
- Exploit cache timing channel

Memory Structure

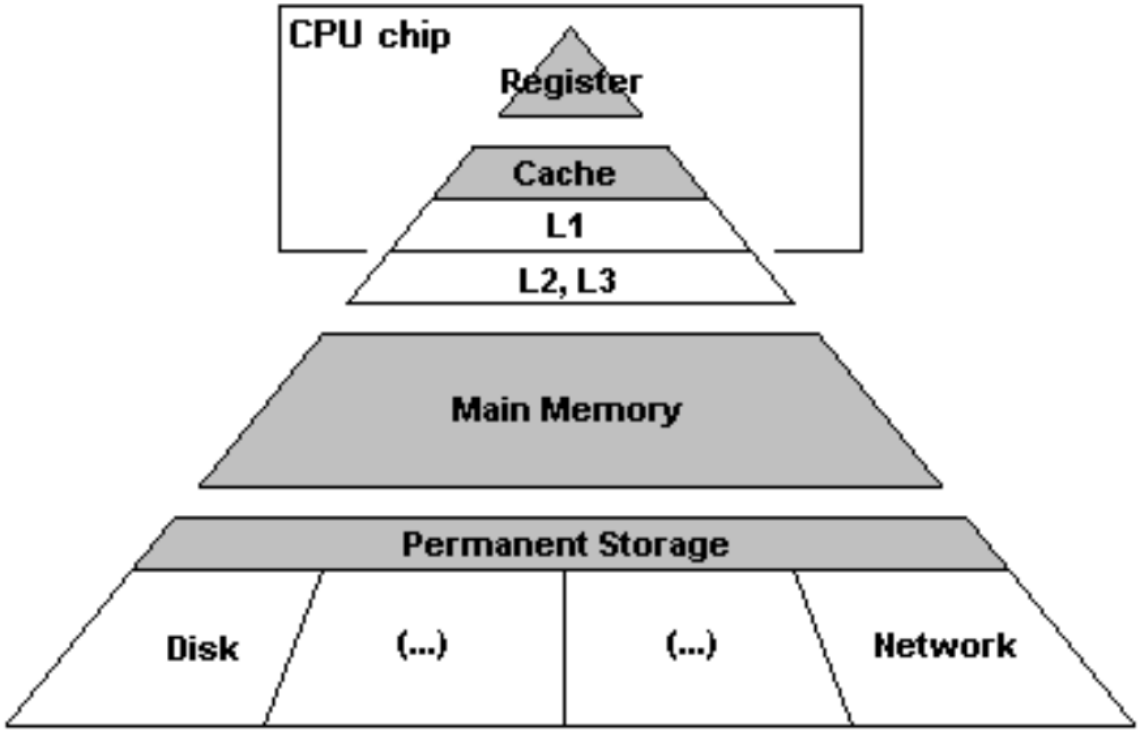
- Processor DRAM Gap



Carlos Carvalho: The Gap between Processor and Memory Speeds

Memory Structure

- Typical modern memory hierarchy



Carlos Carvalho: The Gap between Processor and Memory Speeds

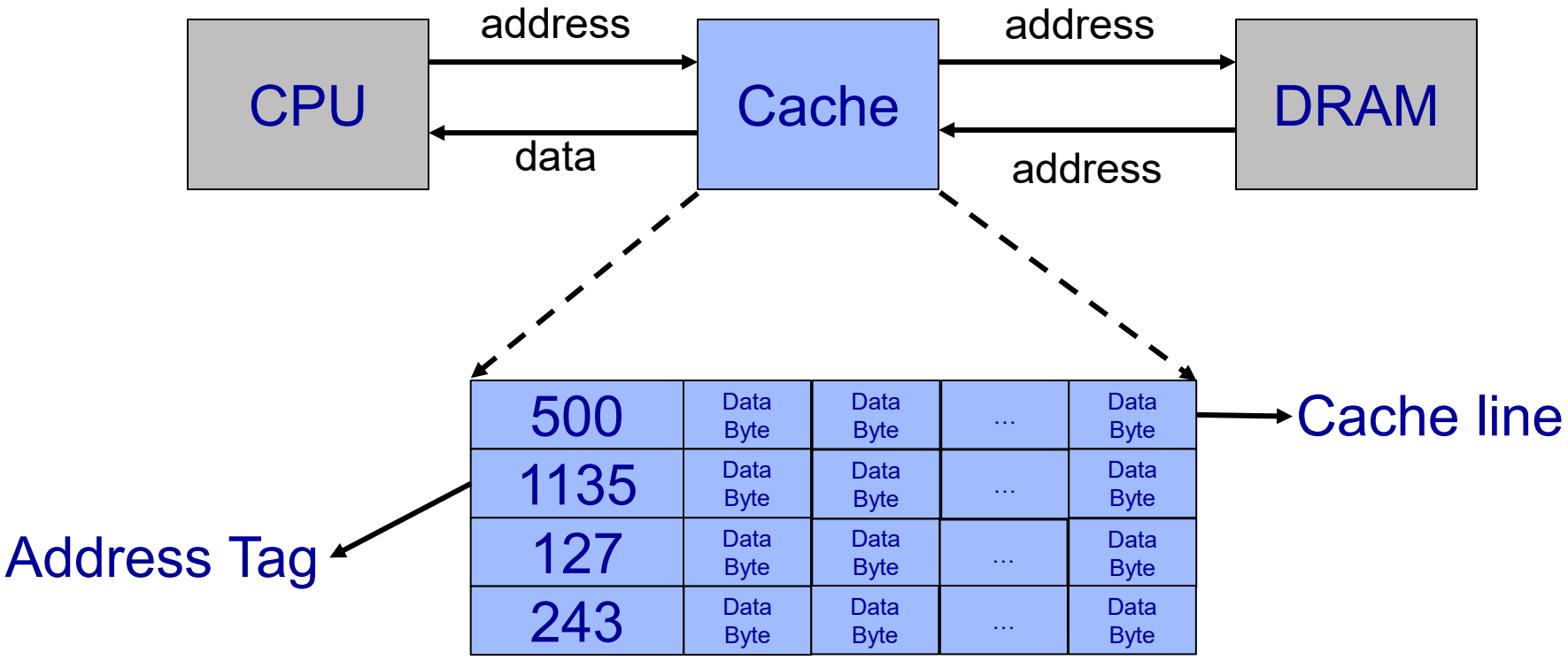
Cache Structure

- Cache: On-chip storage component with fast access

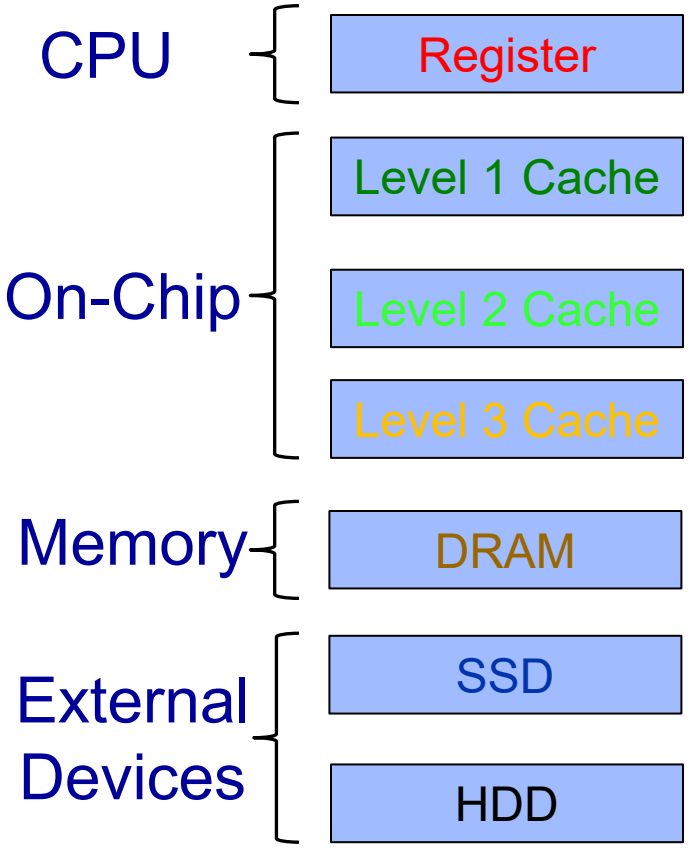
- Cache hit: Data for this address is in cache, return quickly
- Cache miss: Data for this address not in cache, ask DRAM for data

Cache Structure

- Cache: On-chip storage component with fast access

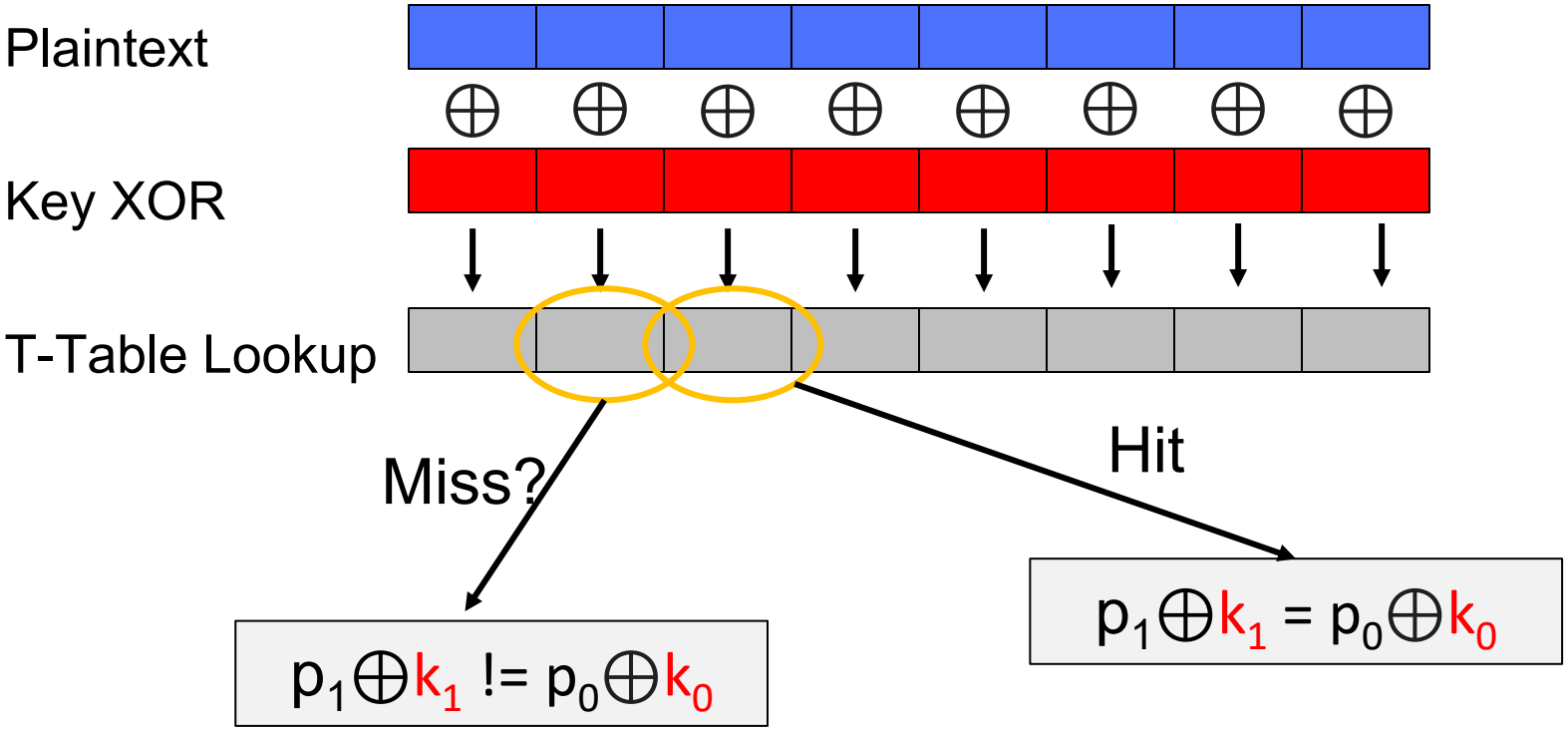


A Typical Memory Structure



Access Time	Capacity
1 cycle	1 KB
2-4 cycles	32 KB
10 cycles	256 KB
40 cycles	10 MB
200 cycles	10 GB
10-100 us	100 GB
10ms	1 TB

Back to AES T-Table



With large amount of tests, key space can be greatly reduced!

Implementation of Cache Attack