CAN304
Computer Systems Security

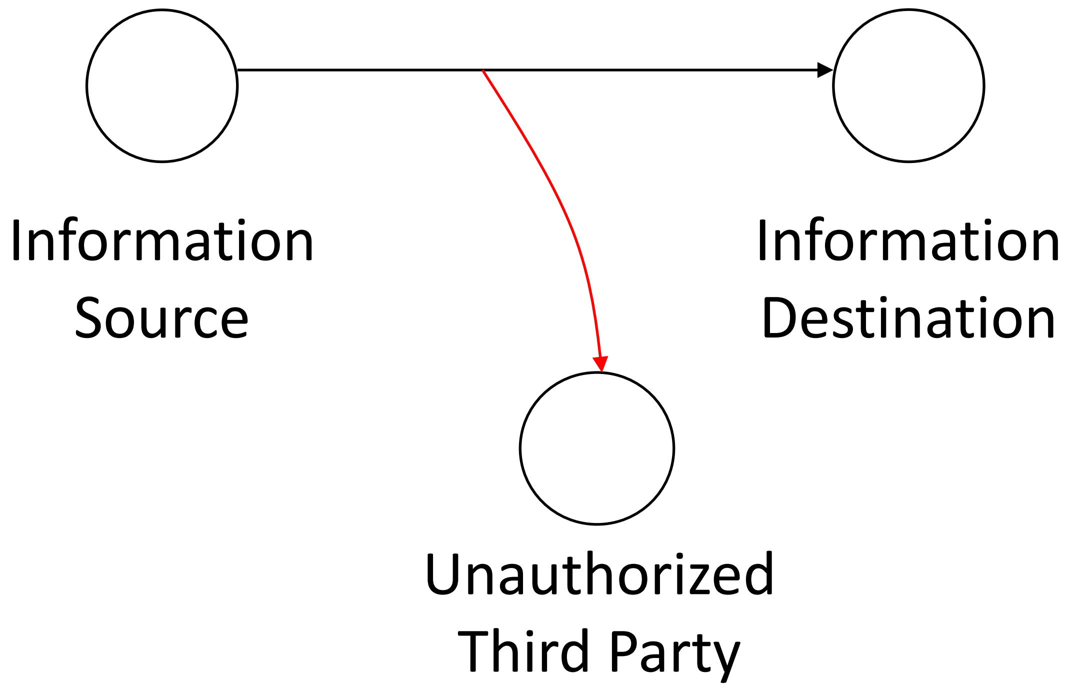
Lecture 2. Fundamentals of cryptography (1)

Week 2: 2022-03-04, 14:00-16:00, Friday

Jie Zhang
Department of Communications and Networking
Email: jie.zhang01@xjtlu.edu.cn
Office: EE522

Review of last week

- Key objectives of security: CIA
- What assets do we need to protect?
- How are those assets threatened?
- What can we do to counter those threats?



Outline

- Classical and modern cryptography
- Symmetric encryption

Learning objetives

- Learn about classical cyphers.
- Understand the basic operation of symmetric block encryption.
- Compare and contrast block encryption and stream encryption.
- Understand why CBC mode is needed.
- Apply symmetric encryption.

1. Classical and modern cryptography

Cryptography (historically)

- "...the art of writing or solving codes..."
  - Concise Oxford English Dictionary

- Historically, cryptography focused exclusively on ensuring secret communication between two parties sharing secret information in advance (aka, codes or private-key encryption)
- Which one of CIA is achieved?

- Application
  - Military organization and governments

Modern cryptography

- Much broader scope!
  - Data integrity
  - User authentication
  - Electronic auction and voting
  - Digital cash
  - …

- "Design, analysis, and implementation of mathematical techniques for securing information, systems, and computation against adversarial attack"

Classical and modern cryptography

- "...the art of writing or solving codes..."
- Historically, cryptography was an art
  - Heuristic, ad hoc design and analysis
  - Schemes proposed, broken, repeat...
- Application
  - Military organization and governments

- Cryptography is now a science
  - Rigorous analysis, firm foundations, deeper understanding, rich theory
- Application
  - Everywhere

Classical cryptography

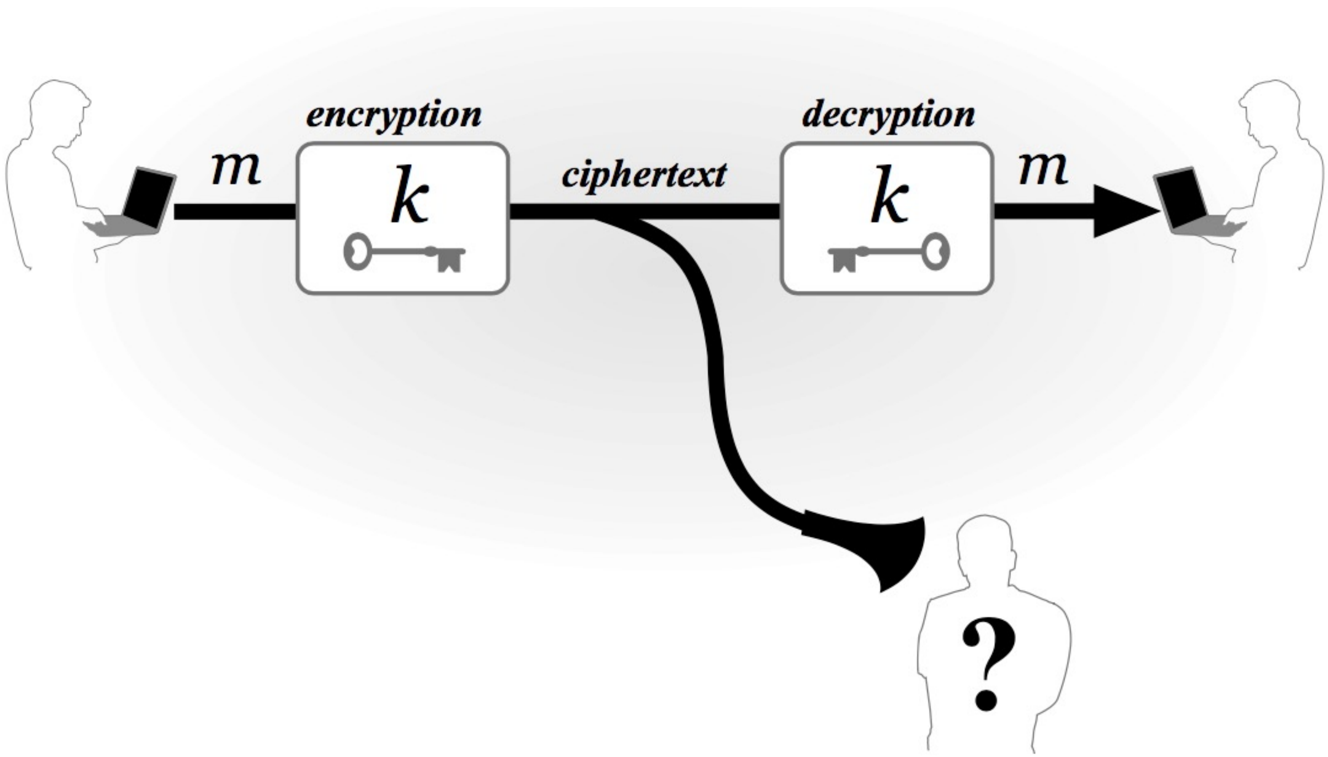
- From at least at least 4000 year ago, ancient Egyptians
- Until the 1970s,

- Exclusively concerned with ensuring secrecy of communication
  - Encryption

- Relied exclusively on secret information (a key) shared between the communicating parties
  - Private-key cryptography
  - AKA secret-key / shared-key / symmetric-key cryptography

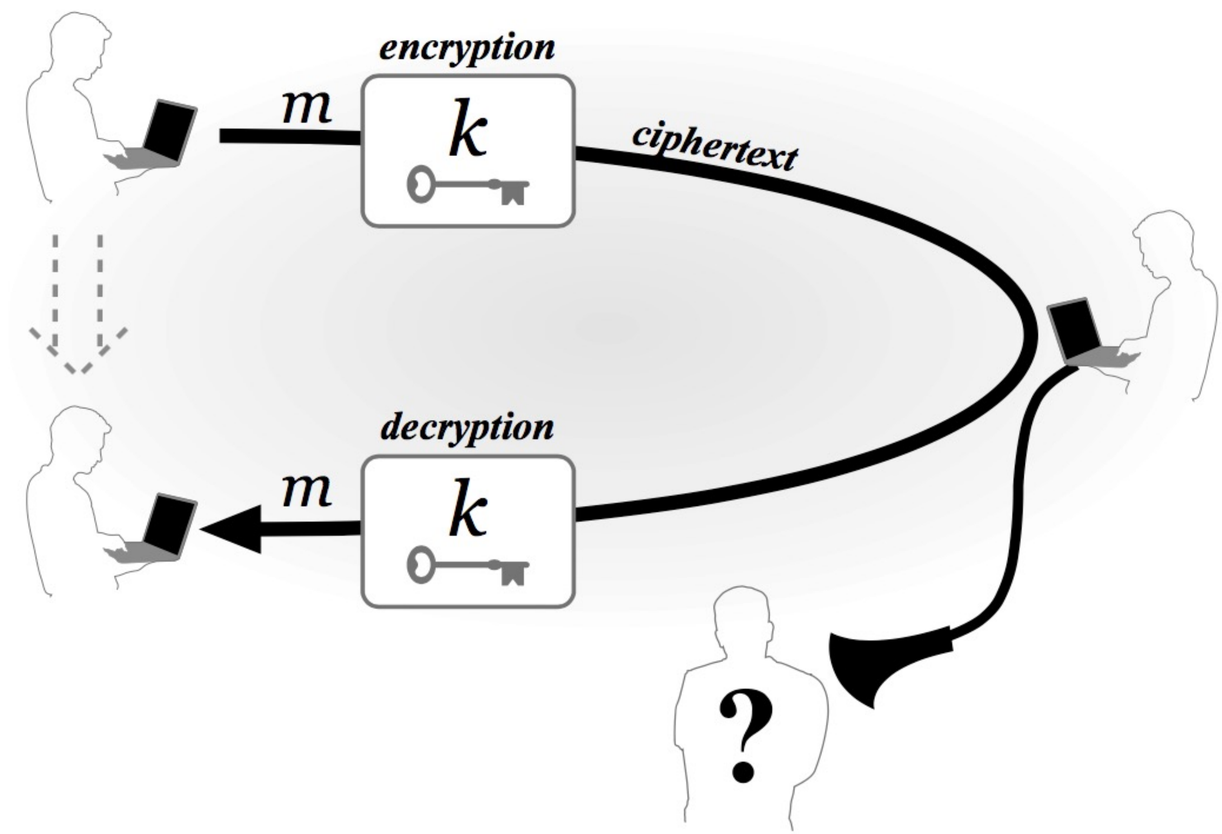
Private-key encryption

- Secure communication
  - Two parties share a key that they use to communicate securely



Private-key encryption

- Secure storage
  - A single user stores data securely over time



Private-key encryption

- A private-key encryption scheme is defined by a message space M and algorithms (Gen, Enc, Dec):
  - Gen (key-generation algorithm): generates k
  - Enc (encryption algorithm): takes key k and message $m \in M$ as input; outputs ciphertext c

$$c \leftarrow Enc_k(m)$$

  - Dec (decryption algorithm): takes key k and ciphertext c as input; outputs m or "error"

$$Dec_k(c) = m$$

    – For all $m \in M$ and k output by Gen,

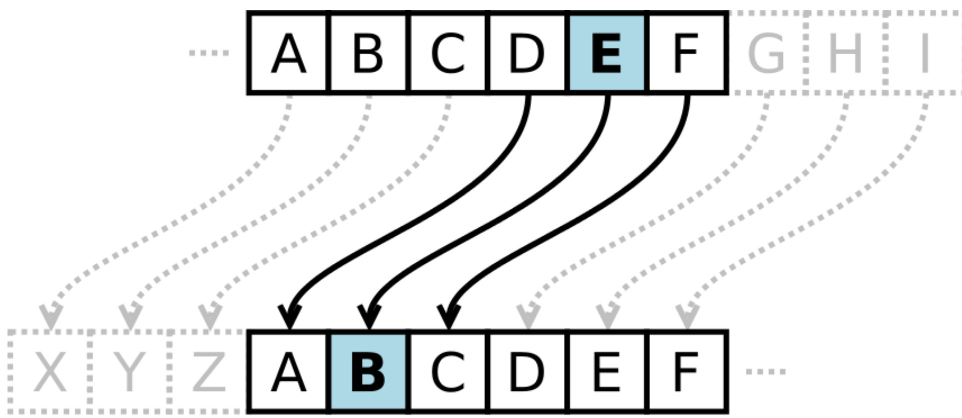
$$Dec_k(Enc_k(m)) = m$$

The shift cipher

- Caesar cipher, used by Julius Caesar
- Consider encrypting English text
- Associate a with 0; b with 1; ...; z with 25
- $k \in \{0, \ldots, 25\}$
- To encrypt using key k, shift every letter of the plaintext by k positions to the right (with wraparound)
- Decryption just reverses the process

The shift cipher

- Example
  - $k = 23$



The shift cipher, formally

- M = {strings over lowercase English alphabet}
- Gen: choose uniform $k \in \{0, \dots, 25\}$
- $Enc_k(m_1 \dots m_t)$: output $c_1 \dots c_t$, where

$$c_i := [m_i + k \bmod 26]$$

- $Dec_k\ (c_1 \dots c_t)$: output $m_1 \dots m_t$, where

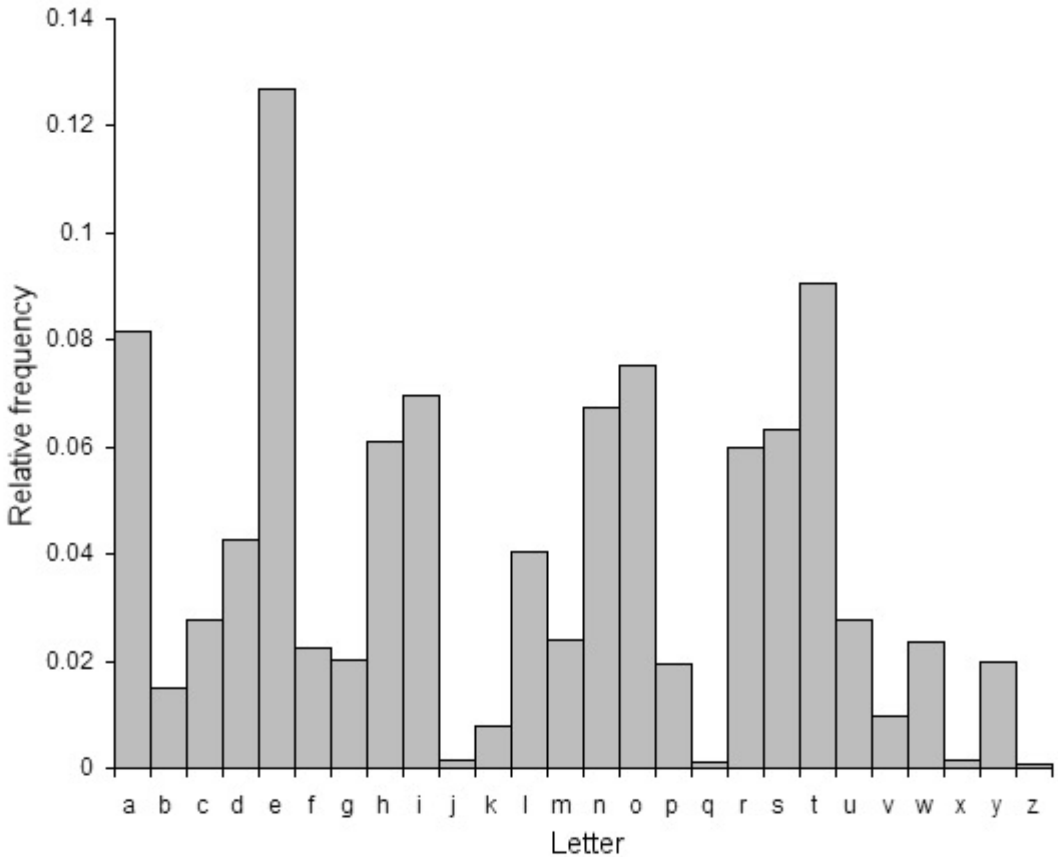
$$m_i := [c_i - k \bmod 26]$$

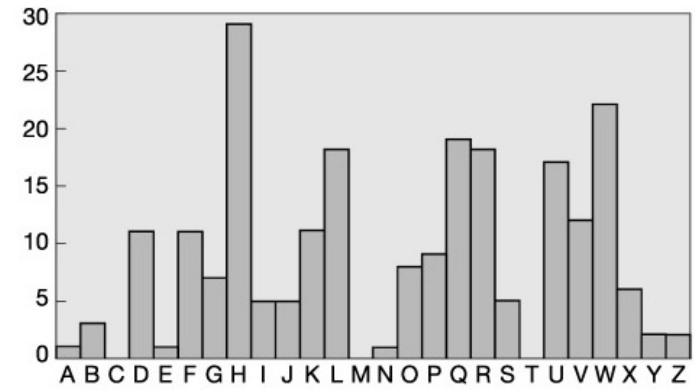
Is the shift cipher secure?

- No, only 26 possible keys!
  - Given a ciphertext, try decrypting with every possible key
  - If ciphertext is long enough (and plaintext is normal English), only one possibility will “make sense”
- Example
  - Ciphertext: uryybjbeyq
  - Try every possible key...
    - $k = 1$: tqxxaiadxp
    - $k = 2$: spwwzhzcwo
    - …
    - $k = 13$: helloworld

Frequency analysis

- Match up the frequency distribution of the letters.





Sufficient key space principle

- The key space should be large enough to prevent “brute-force” exhaustive-search attacks
- If an encryption scheme has a key space that is too small, then it will be vulnerable to exhaustive-search attacks

- Caesar cipher is insecure
  - The key space is 26

The Vigenère cipher

- The key is now a string, not just a character
- To encrypt, shift each character in the plaintext by the amount dictated by the next character of the key
  - Wrap around in the key as needed
- Decryption just reverses the process
- Example
  - k='cafe'

```
tellhimaboutme
cafecafecafeca
veqpjiredozxoe
```

The Vigenère cipher

- Size of key space?
  - If keys are 14-character strings; then key space has size $26^{14} \approx 2^{66}$
- Brute-force search expensive/impossible

- Is the Vigenère cipher secure?
  - (Believed secure for many years...)

Attacking the Vigenère cipher

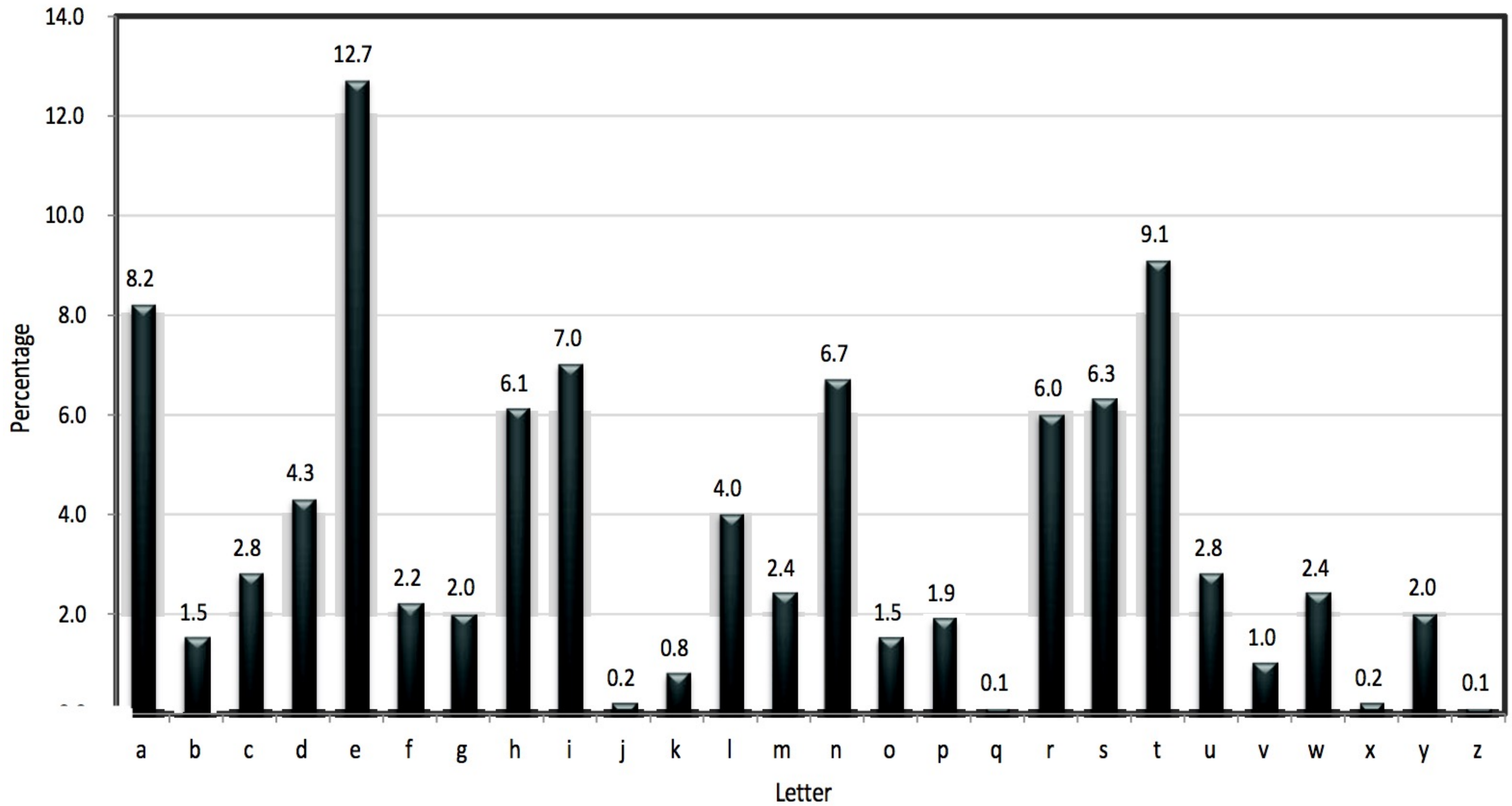
- (Assume a 14-character key)
- Observation: every 14th character is “encrypted” using the same shift

veqpjiredozxoe**u**alpcmsdjqu
iqn**d**nossoscdcusoa**k**jqmxpqr
hyycjq**o**qqodhjcciowie**i**i

- Looking at every 14th character is almost like looking at ciphertext encrypted with the shift cipher

Using plaintext letter frequencies

- English letter frequencies



Attacking the Vigenère cipher

- Look at every 14th character of the ciphertext, starting with the first
- Let α be the most common character appearing in this portion of the ciphertext
- Most likely, this character corresponds to the most common plaintext character (‘e’)
- Guess the first character of the key is $\alpha - \mathrm{e}$
- Repeat for all other positions

Historically…

- Cryptography was an art
  - Heuristic, ad hoc design and analysis

- In the late 1970s and early 1980s, cryptography began to develop into more of a science

2. Symmetric encryption

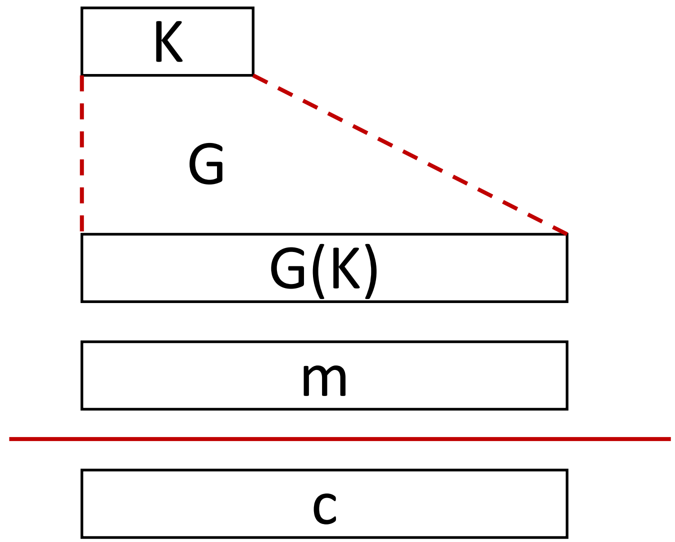
- Block ciphers
- Cryptographic modes
- Uses of symmetric cryptography

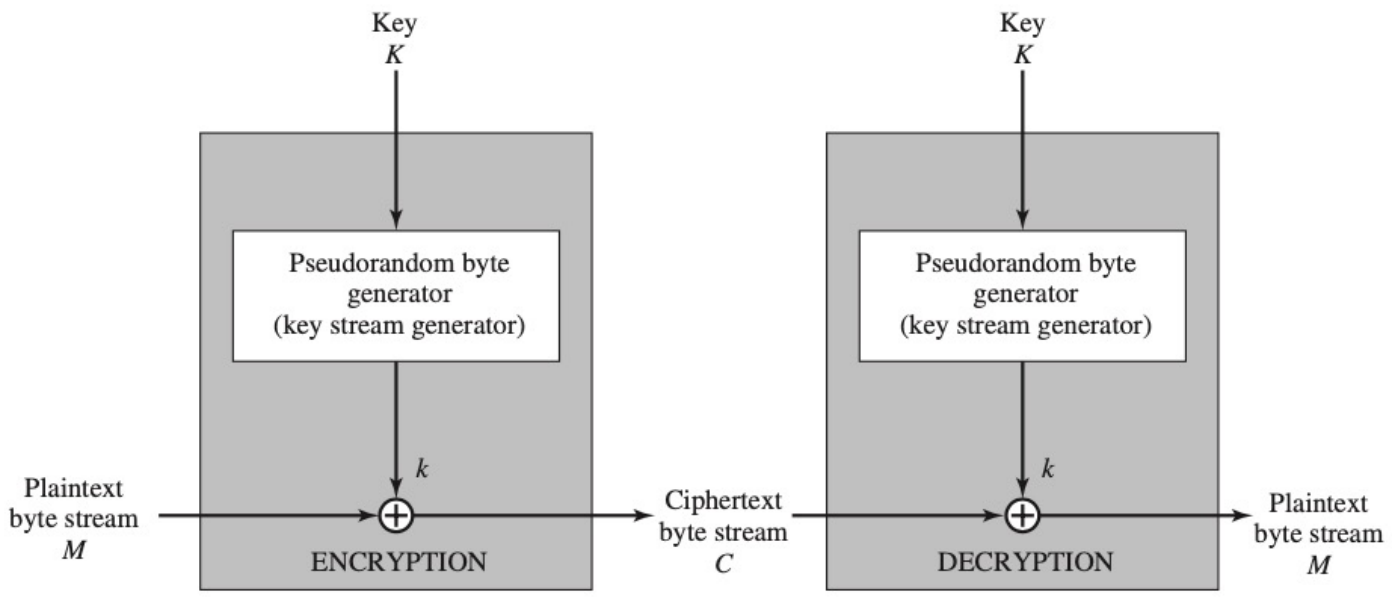
Symmetric ciphers

- Definition
  - A pair of “efficient” algorithms (E, D)
  - defined over (K, M, C)
  - where $E: K \times M \to C, D: K \times C \to M$
  - such that $\forall m \in M, k \in K:\ D\big(k, E(k, m)\big) = m$

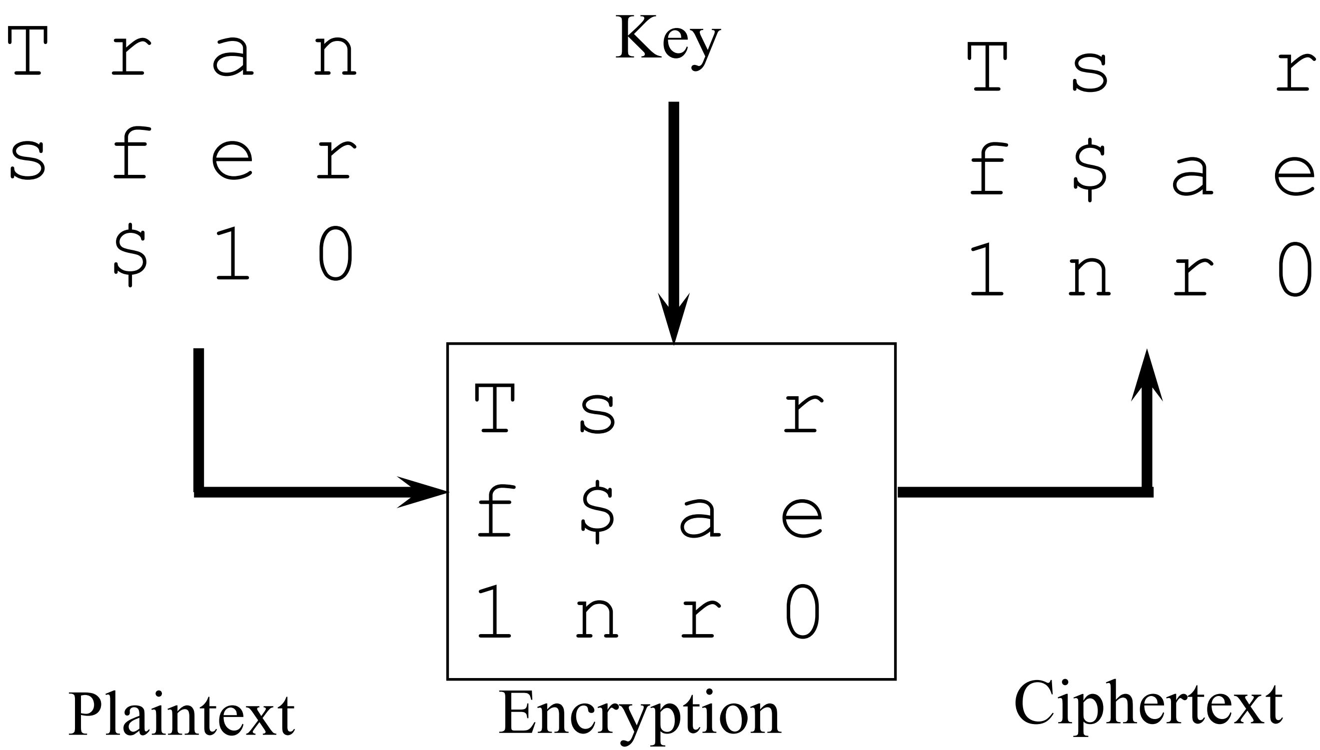
Stream ciphers

- $c \coloneqq E(K, m) = G(K) \oplus m$
- $m \coloneqq D(K, c) = G(K) \oplus c$





Block ciphers



Advantages of block ciphers

- Good diffusion
    - Easier to make a set of encrypted characters depend on each other
- Immunity to insertions
    - Encrypted text arrives in known lengths

- Most common Internet crypto are done with block ciphers

Disadvantages of block ciphers

- Slower
  - Need to wait for block of data before encryption/decryption starts
- Worse error propagation
  - Errors affect entire blocks

2.1 Block ciphers

Sample block ciphers

- The Data Encryption Standard
- The Advanced Encryption Standard
- There are many others

The Data Encryption Standard

- Well known symmetric cipher
- Developed in 1977, still much used
  - Shouldn’t be, for anything serious
- Block encryption, using substitutions, permutations, table lookups
  - With multiple rounds
  - Each round is repeated application of operations
- Only serious problem based on short key

The Advanced Encryption Standard

- A relatively new cryptographic algorithm
- Intended to be the replacement for DES
- Chosen by NIST
  - Through an open competition
- Chosen cipher was originally called Rijndael
  - Developed by Dutch researchers
  - Uses combination of permutation and substitution

AES Internals

- Process blocks of 128 bits using a secret key of 128, 192, or 256 bits
- View 16-byte plaintext as a two-dimensional array of bytes: s

s_{00}	s_{01}	s_{02}	s_{03}
s_{10}	s_{11}	s_{12}	s_{13}
s_{20}	s_{21}	s_{22}	s_{23}
s_{30}	s_{31}	s_{32}	s_{33}

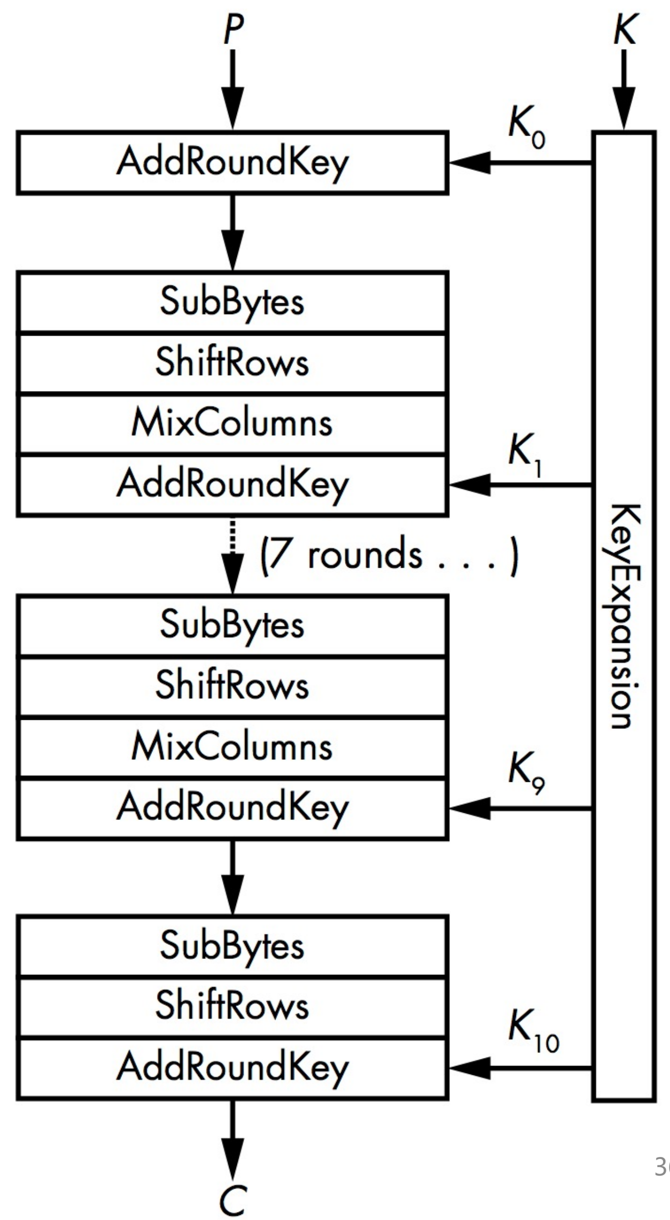
The internal state of AES viewed as a 4 × 4 array of 16 bytes.

  - This array is called the internal state

- AES transforms the bytes, columns, and rows of this array to produce a final value that is the ciphertext.

Substitution–permutation network (SPN)

- In order to transform its state, AES uses an SPN structure, with 10 rounds for 128-bit keys, 12 for 192-bit keys, and 14 for 256-bit keys.
- Four building blocks
  - AddRoundKey
  - SubBytes
  - ShiftRows
  - MixColumns



Substitution–permutation network (SPN)

- AddRoundKey
  - XORs a round key to the internal state
- SubBytes
  - Replaces each byte $(s_{00}, s_{01}, \ldots, s_{33})$ with another byte according to an S-box (next slides).

S-box

x \ y	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	B9	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

Substitution–permutation network (SPN)

- ShiftRows
  - Shifts the ith row of i positions to the left, for i ranging from 0 to 3

$$\begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33} \end{bmatrix} \xrightarrow{\text{shiftRows( )}} \begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{11} & S_{12} & S_{13} & S_{10} \\ S_{22} & S_{23} & S_{20} & S_{21} \\ S_{33} & S_{30} & S_{31} & S_{32} \end{bmatrix}$$

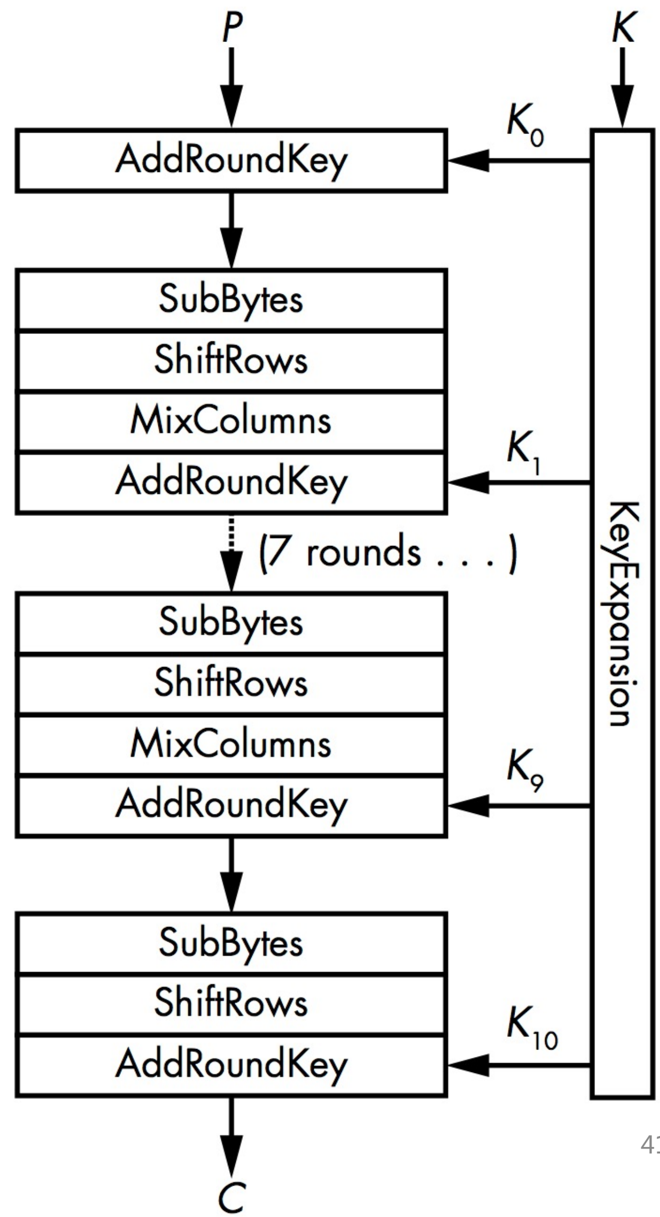
Substitution–permutation network (SPN)

- MixColumns
  - Each column of four bytes is now transformed using a special mathematical function.
  - This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column.
  - The result is another new matrix consisting of 16 new bytes.

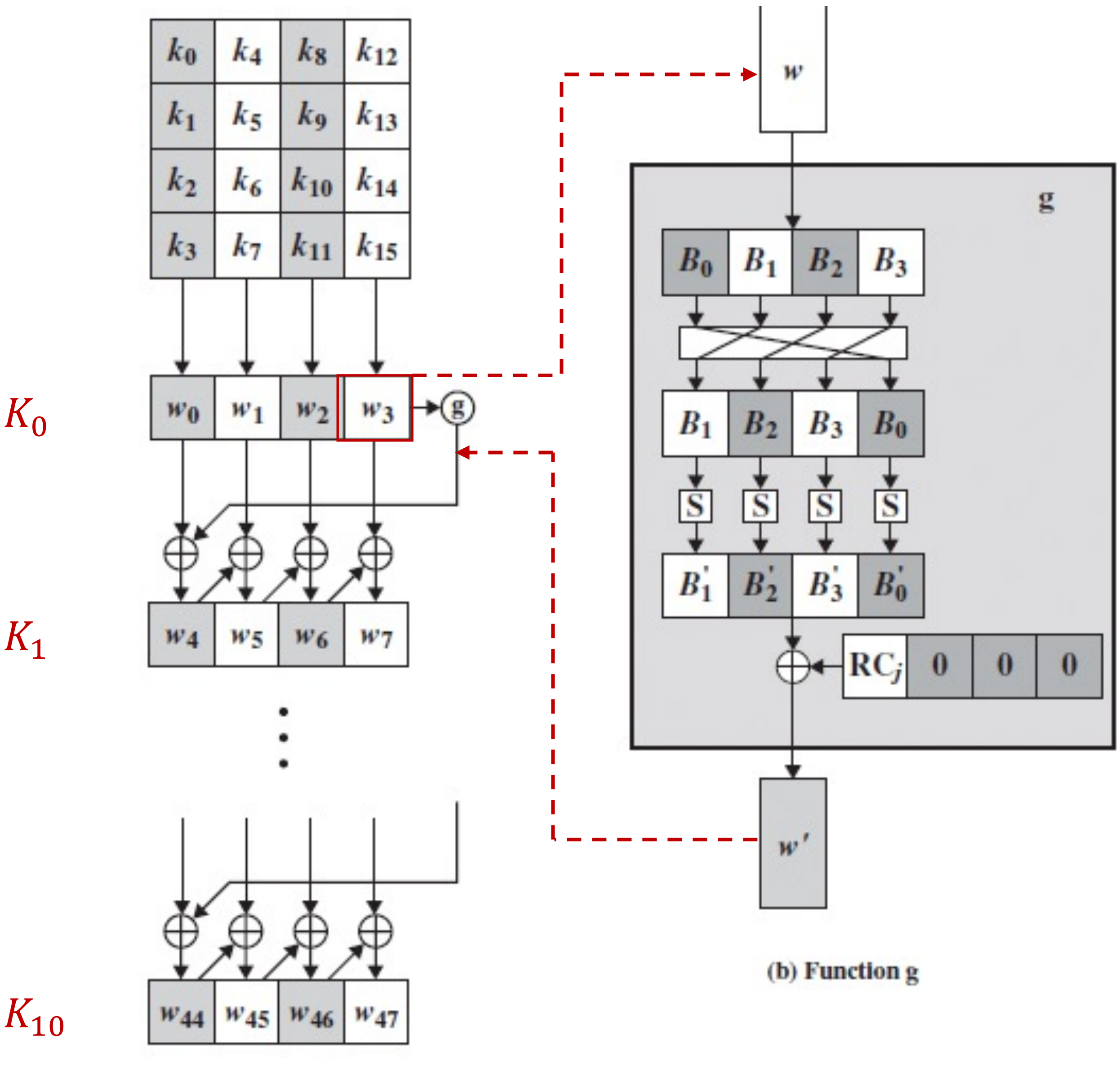
$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} = \begin{bmatrix} s'_{00} & s'_{01} & s'_{02} & s'_{03} \\ s_{10} & s'_{11} & s_{12} & s'_{13} \\ s'_{20} & s'_{21} & s'_{22} & s'_{23} \\ s'_{30} & s'_{31} & s'_{32} & s'_{33} \end{bmatrix}$$

Key schedule function

- KeyExpansion
  - Create 11 round keys (K0, K1, . . . , K10) of 16 bytes each from the 16-byte initial key, using the same S-box as SubBytes and a combination of XORs



Key schedule function



(a) Overall algorithm

(b) Function g

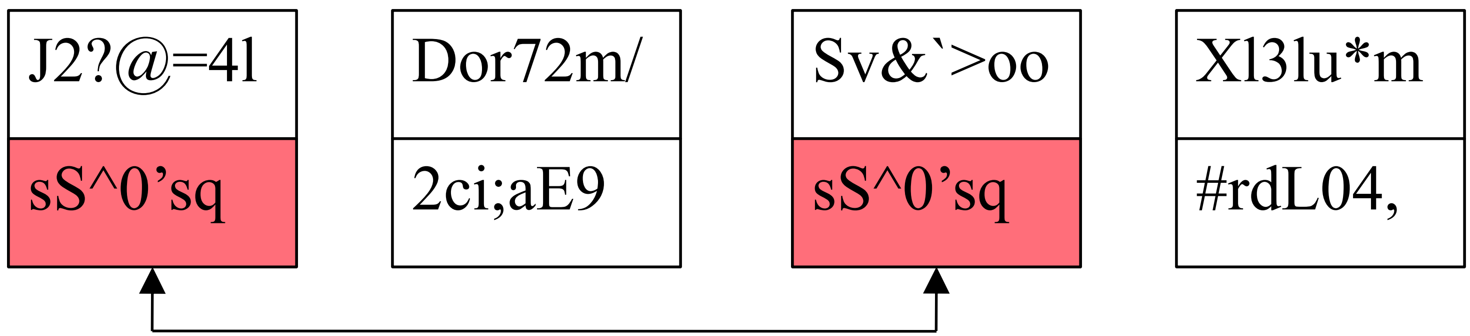
$RC_1 = 01$
$RC_2 = 02$
$RC_3 = 04$
$RC_4 = 08$
$RC_5 = 10$
$RC_6 = 20$
$RC_7 = 40$
$RC_8 = 80$
$RC_9 = 1B$
$RC_{10} = 36$

Is AES secure?

- AES is as secure as a block cipher can be
  - All output bits depend on all input bits in some complex, pseudorandom way.
- But there’s no proof that AES is immune to all possible attacks.
  - e.g., the new side-channel attacks

2.2 Cryptographic modes

The basic situation



- Let’s say our block cipher has a block size of 7 characters and we use the same key for all
- Now let’s encrypt
- There’s something odd here . . .
  - *Why did it happen?*
  - *Is this good?*

Problem with this approach

- What if these are transmissions representing deposits into bank accounts?

1840326	5610993	3370259	6840924
$100.00	$550.00	$100.00	$225.00

Xl3lu*m
#rdL04,

Dor72m/
2ci;aE9

1840326	450
2201568	5000
3370259	8900
5610993	**1579**
6840924	2725
8436018	10

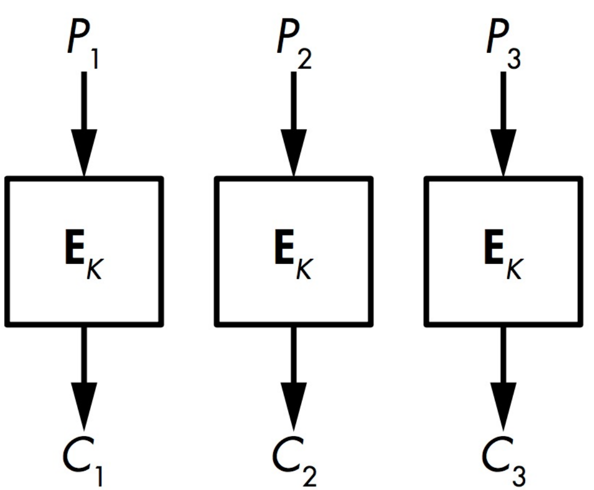
What if account 5610993 belongs to him?

So far, so good . . .

What caused the problem?

- Each block of data was independently encrypted
  - With the same key
- So two blocks with identical plaintext encrypt to the same ciphertext
- Not usually a good thing

- We used the wrong cryptographic mode
  - Electronic Codebook (ECB) Mode



Cryptographic modes

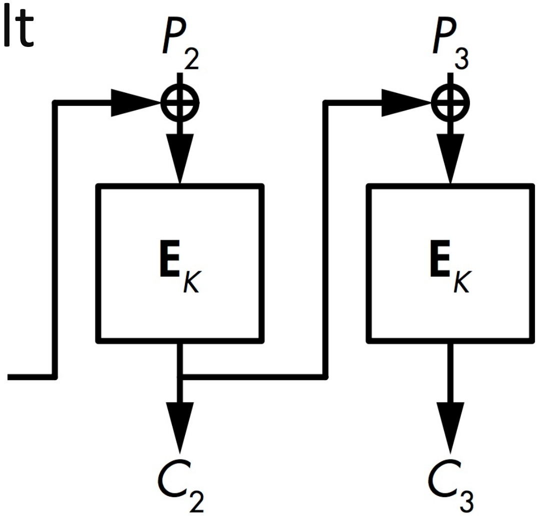
- A cryptographic mode is a way of applying a particular cipher
  - Block or stream
- The same cipher can be used in different modes

So, what mode should we have used?

- Cipher Block Chaining (CBC) mode might be better
- Ties together a group of related encrypted blocks
- Hides that two blocks are identical

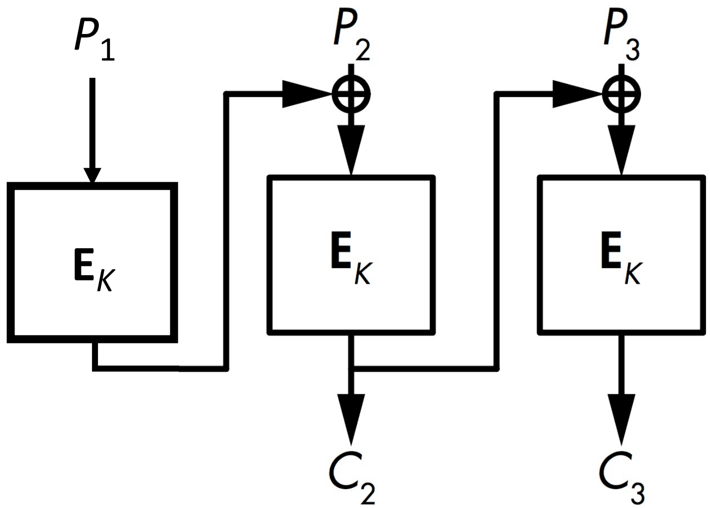
Cipher block chaining mode

- Adds feedback into encryption process
- The encrypted version of the previous block is used to encrypt this block
- For block X+1, XOR the plaintext with the ciphertext of block X
  - Then encrypt the result



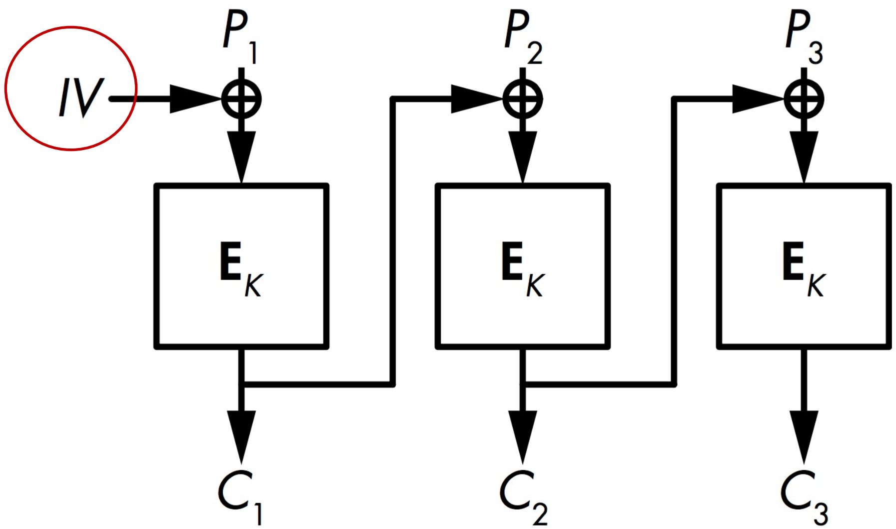
- Each block's encryption depends on all previous blocks' contents
- Decryption is similar

What about the first block?



- If we send the same first block in two messages with the same key,
  - Won’t it be encrypted the same way?
- Might easily happen with message headers or standardized file formats
- CBC as described would encrypt the first block of the same message sent twice the same way both times

Initialization vectors



- A technique used with CBC
  - And other crypto modes
  - Abbreviated IV
- Ensures that encryption results are always unique
  - Even for duplicate message using the same key
- XOR a random string with the first block
  - $P_1 \oplus IV$
  - Then do CBC for subsequent blocks

Encrypting with an IV

First block of message

1	1	0	1	0	0	0	1

Initialization vector

0	1	0	0	1	1	0	0

XOR IV and message

1	0	0	1	1	1	0	1

Encrypt message and send IV plus message

0	0	1	1	0	1	1	1

Second block of message

0	0	0	1	1	0	0	0

Use previous msg for CBC

Apply CBC (XOR P2 with C1)

0	0	1	0	1	1	1	1

Encrypt and send second block of message

1	0	0	1	1	1	1	0

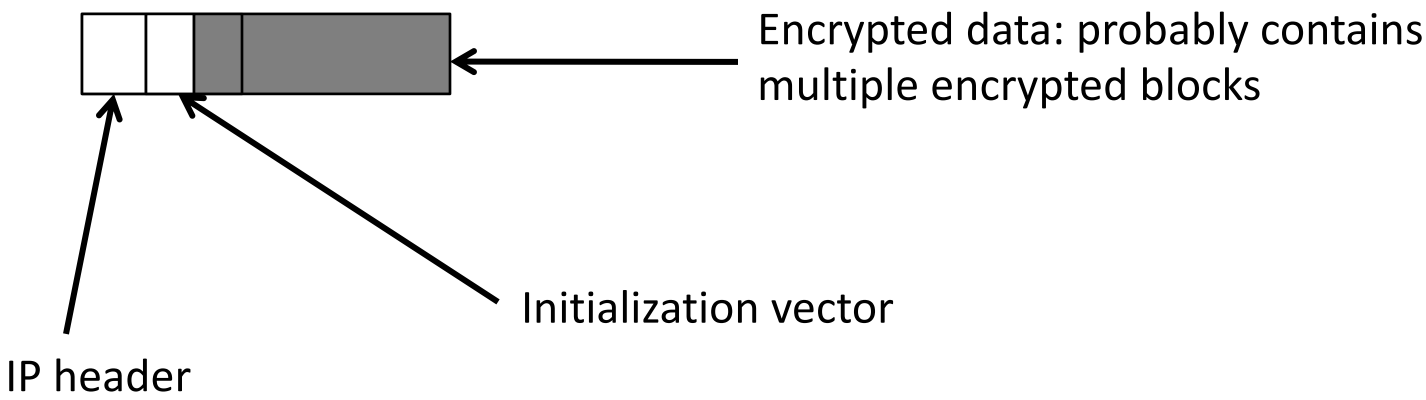
How to decrypt with initialization vectors?

- First block received decrypts to
  - P = Dec(k, message)
  - plaintext = P $\oplus$ IV

- No problem if receiver knows IV
  - Typically, IV is sent in the message

- Subsequent blocks use standard CBC
  - So can be decrypted that way

An example of IV decryption



0	1	0	0	1	1	0	0

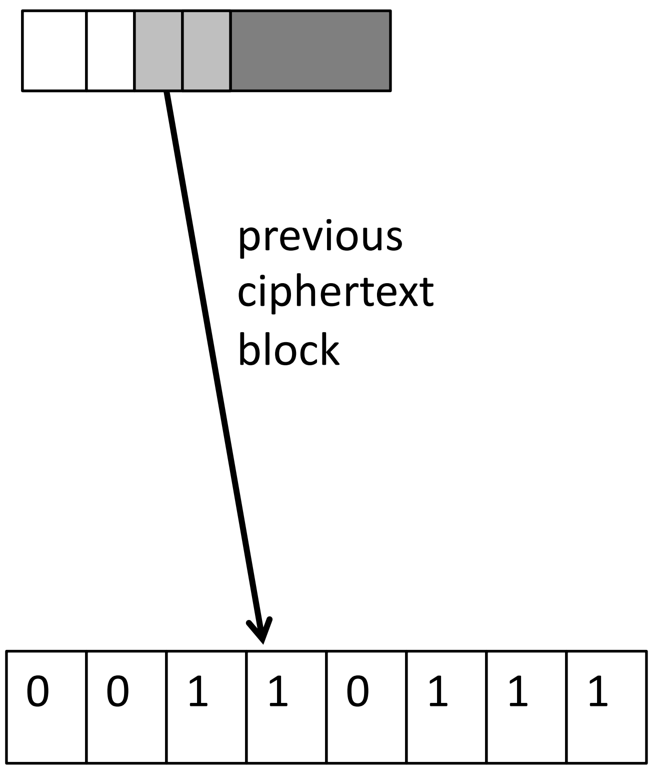
Now decrypt the message

1	0	0	1	1	1	0	1

And XOR with the plaintext IV

1	1	0	1	0	0	0	1

For subsequent blocks



1	1	0	0	0	0	1	1

Now decrypt the message

0	0	1	0	1	1	1	1

And XOR with the previous ciphertext block

0	0	0	1	1	0	0	0

2.3 Uses of symmetric cryptography

Uses of symmetric cryptography

- What can we use symmetric cryptography for?
- Lots of things
  - Secrecy (confidentiality)
  - Authentication
  - Prevention of alteration (integrity)

Symmetric cryptography and secrecy

- Pretty obvious
- Only those knowing the proper keys can decrypt the message
  - Thus preserving secrecy

Symmetric cryptography and authentication

- How can I prove to you that I created a piece of data?

- I give you the data in encrypted form?
  - Using a key only you and I know
- Then only you or I could have created it
  - Unless one of us told someone else the key . . .

Using symmetric cryptography for authentication

- Problems with non-repudiation
  - e.g., I create a data and encrypt it using a secret key known only by you and me; later, I deny that the data is created by me.

- What if three parties want to share a key?
  - No longer certain who created anything
  - Public key cryptography can solve this problem

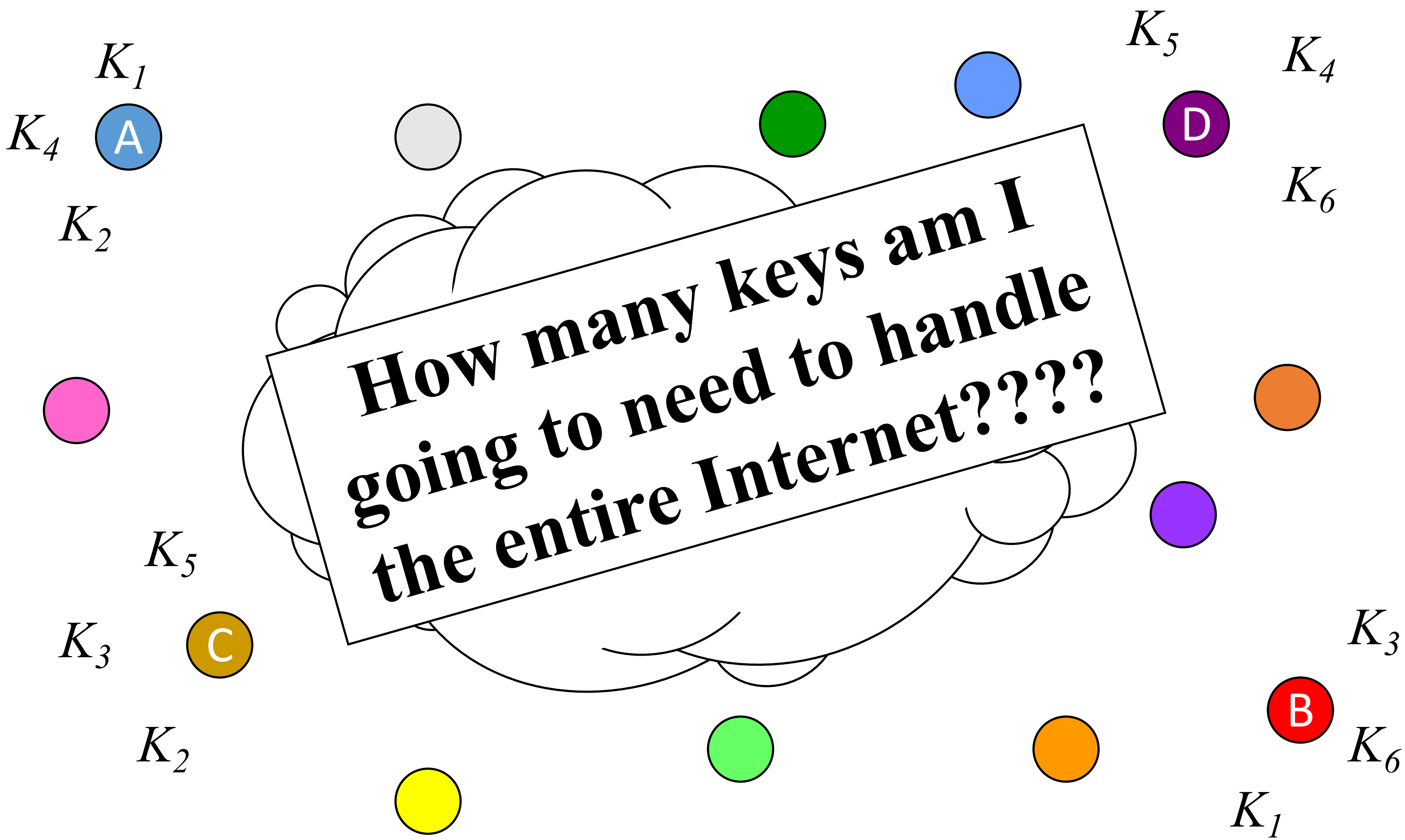
- What if I want to prove authenticity without secrecy?
  - Encryption is not necessary.

Symmetric cryptography and non-alterability

- Changing one bit of an encrypted message completely garbles it
  - For many forms of cryptography
  - ciphertext → plaintext (meaningless, unreadable)

- If a checksum is part of encrypted data, that's detectable
- If you don't need secrecy, can get the same effect
  - By encrypting only the checksum

Scaling problems of symmetric cryptography



Summary

- Classical and modern cryptography
- Symmetric encryption
  - Block ciphers
  - Cryptographic modes
  - Uses of symmetric cryptography