Introduction to Cyber Security

Fall 2017 | Sherman Chow | CUHK IERG 4130

Chapter 5
Overview of Crypto.,
& Symmetric Encryption

This Chapter

- Overview of (Modern) Cryptography
  - What can “encryption” possibly do?
- Classic ciphers (and attack)
- Stream cipher: RC4
- Block cipher: DES (and Feistel network)
- Next chapter: 2DES (and attack), 3DES, AES, and so on

Crypto is Everywhere

main photo from pixelc urse.com (davelei)

Crypto. as a scientific discipline [Shamir]

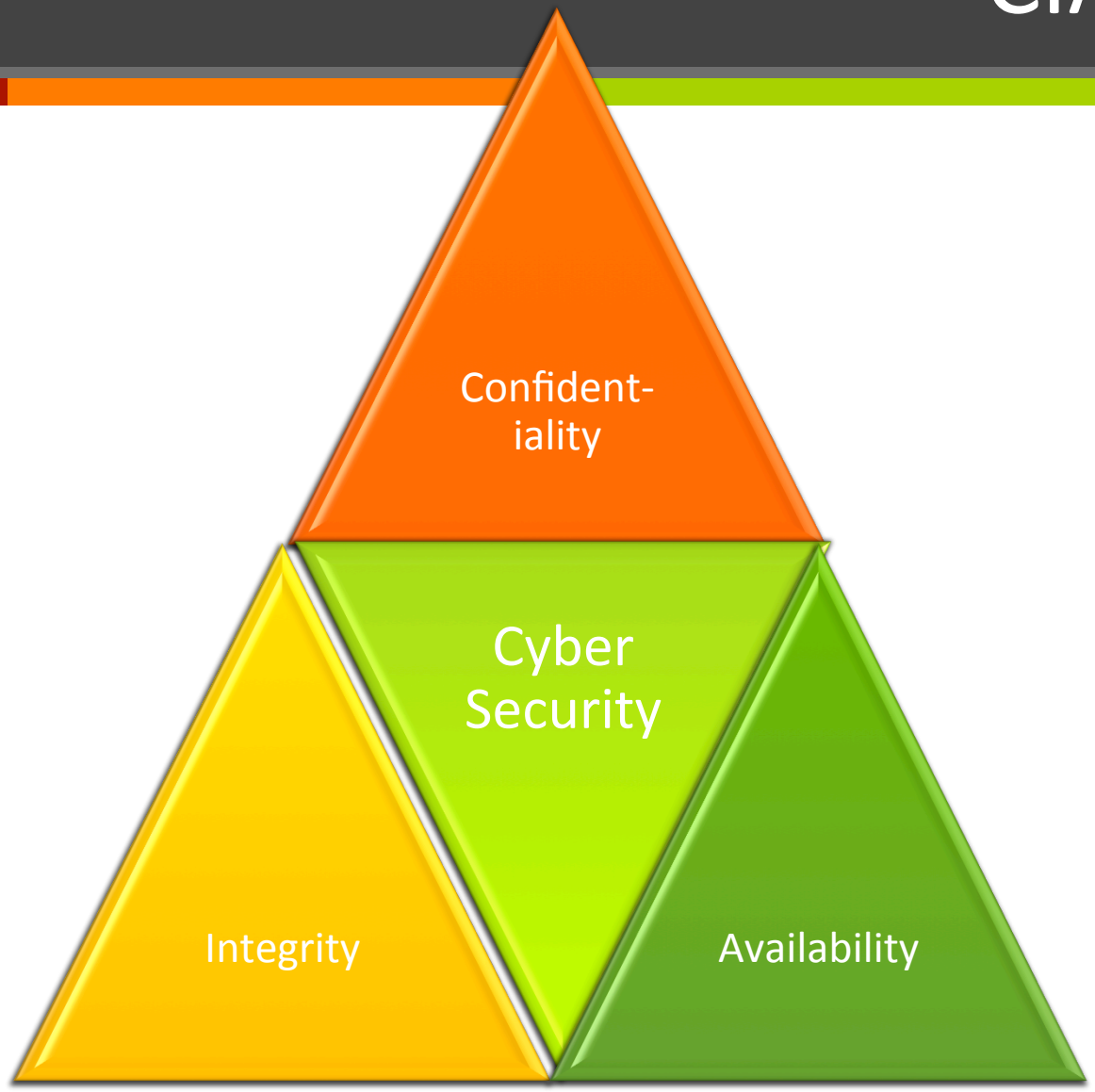
Is thriving as a scientific area of research:

- Taught at most major universities
- Attracts many excellent students
- Discussed at many conferences
- Published in hundreds of papers (e.g., http://eprint.iacr.org)
- Major conferences have >500 attendees
  - (Major trade shows have >10,000 attendees)

Award “outside” of Crypto Community

- Turing award
  - “The ultimate seal of approval” from the general CS community
  - Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, 2002
  - Silvio Micali and Shafi Goldwasser, 2012
- MacArthur Fellowship, or "Genius Grant”
  - to citizens/residents of the United States, working in any field, who “show exceptional merit and promise for continued and enhanced creative work”
  - Craig B. Gentry, 2014, for Fully Homomorphic Encryption
  - Yitang Zhang, 2014, for “properties” of prime number
    - (number theory is useful in cryptography)

CIA Traid



Cipher: Terminology

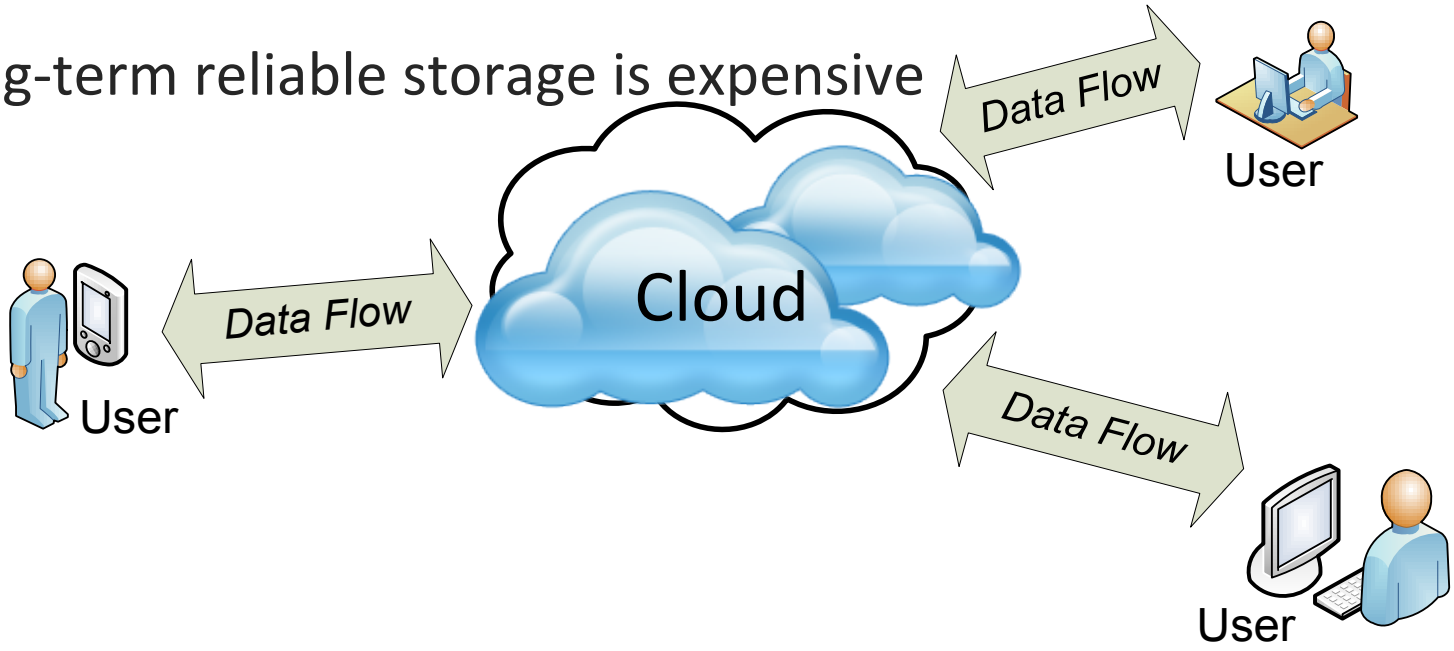
- Encryption: converting plaintext to ciphertext
- Decryption: converting ciphertext to plaintext
  - "Invert" the "encryption function"
- Prevent the disclosure of info. to "unauthorized party"
- Cryptanalysis: to break the code by analyzing the algorithm
- Cipher may refer to the cryptosystem/algorithm or ciphertext
- Encryption may mean a specific algorithm or a system/scheme (encryption algorithm + decryption algorithm) as a whole

Security by Obscurity?

- Encryption: turns a plaintext into a ciphertext with a "key"
- Without the "*secret key*", the ciphertext is not "*useful*"
- Only need to keep the "key" secret, can afford to have the "algorithm" public (also facilitate implementation by the mass)
- It is easy to change the "key", but difficult to design and describe / communicate a new secure algorithm
- *Kerckhoff's Principle*: The security of a cipher MUST NOT depend on anything that cannot be easily changed

Basic Settings of Cloud Storage

- Client stores (large) files with the server
  - Online backup, Software as a Service (SaaS), etc.
- Long-term reliable storage is expensive

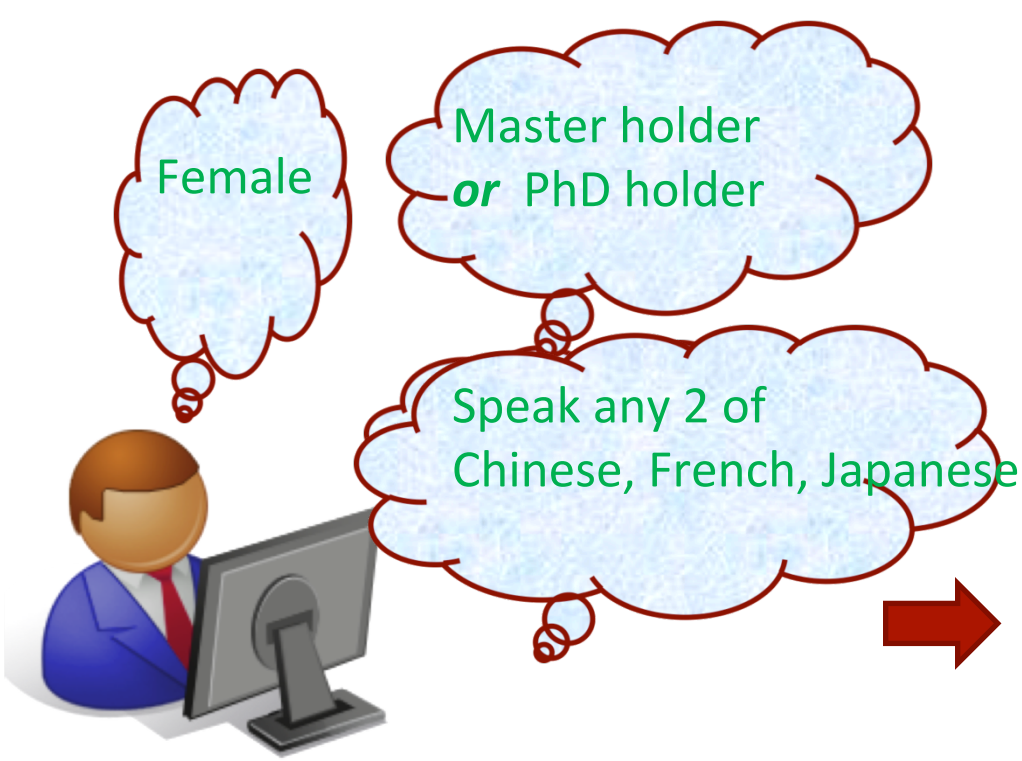


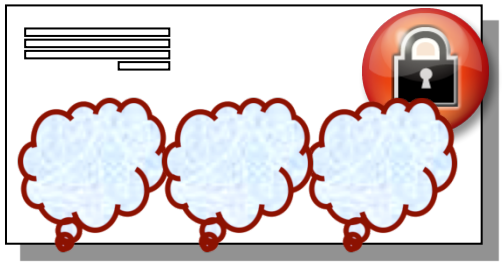
One-time vs. Many-time

- One-time: The key is only used to encrypt one message
- Many-time: Same key used to encrypt multiple messages
- Need more machinery (when compared to one-time key)

- Single-recipient vs. Multi-recipient?
  - Trivial construction: *n* ciphertexts for *n* recipients
  - Broadcast encryption is more efficient (not in 4130 :)

Attribute-Based Encryption





(not a "symmetric-key" encryption,
will not discuss it again in 4130)

Is “full” confidentiality always desirable?

- Consider you want to upload your files to the cloud.
- What do you want your cloud service providers do?
- They cannot do much more than storage.
- How about encrypted e-mail?
- You may want your mobile devices only download e-mails marked with the keyword “urgent” from the server.
- You don’t want the server to know what are the keywords associated with each email.

Retrieval of Encrypted Data

- Download all data, then decrypt
  - O(*N*) communication
  - *N*: number of documents
- Build a local index, then download
  - O(*N*) local storage
- Ideally, O(*n*) complexity (at least at client)
  - *n*: number of matching documents (*n* << *N*)
- "Searchable Encryption" (not in 4130 either)

Yao's Millionaires' Problem

I have $$x$$

I have $$y$$

Is $x > y$?

What constitutes an encryption scheme?

- Encryption: E (m) → c
- Decryption: D (c) → m
- Need to generate a key k
- Key generation algorithm
  - Input: security parameter
  - Output: a key k
- E_k (m) → c, D_k (c) → m
- *Symmetric*-key encryption

Caesar Cipher

- Let's play another game
- "WII QI EJXIV XLMW GPEWW"
- How many W's do we have!?
- How many I's?
- What should be "WII" then?

Images from … images.google.com

More "Serious" Definition…

- Consider the 26 alphabets of English
- Encoded them as a number in [0, 25]
- $E_k(m) \rightarrow m + k \bmod 26$
- $D_k(c) \rightarrow c - k \bmod 26$
- salad -> wepeh ($k = 4$)
- Monoalphabetic
- Frequency analysis

Types of Attack

- Ciphertext only: Given ciphertext only to derive plaintext/key
- Known plaintext attack
  - Given <Plaintext, Ciphertext> pair(s) to derive the key
  - Why plaintext is known!? e.g., “From:”
- Chosen plaintext attack:
  - can inject chosen plaintext and observe the ciphertext outcome
  - e.g., lure the victim to send an email having a certain word

Brute-Force Attack

- Let's enumerate over all the possible keys

- Wait, how many of them are there, say, for Caesar cipher?

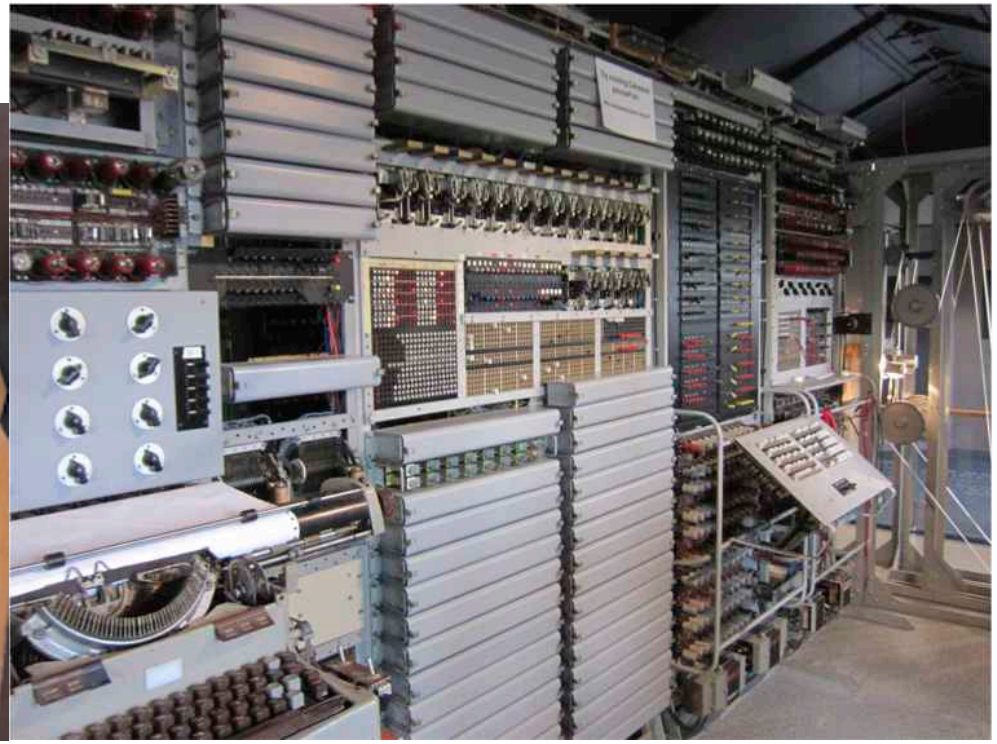
- We need to know when to stop

Vigenère Cipher

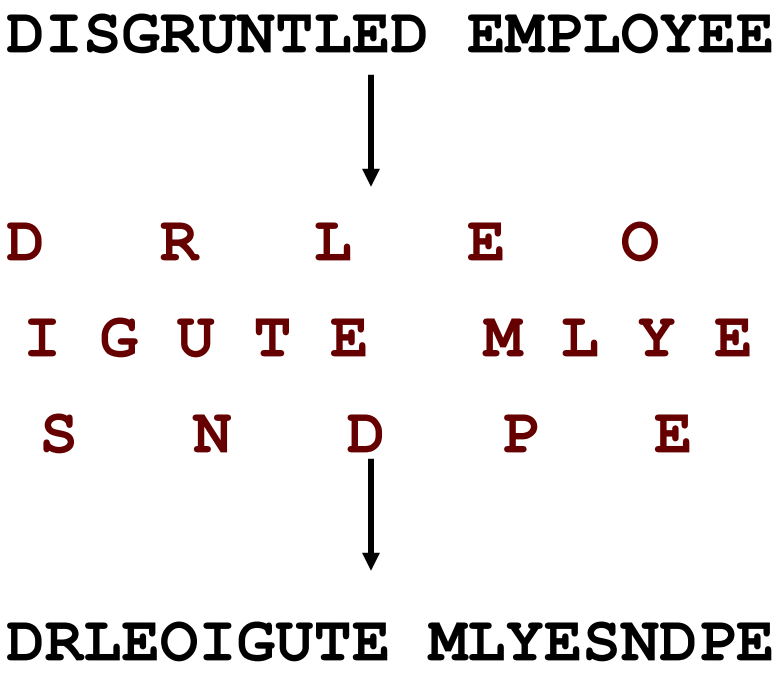
- From 16th century, Rome (variants of Caeser Cipher)
- Idea: not always map a plaintext to the same ciphertext
- Plaintext: AttackAtDawn (case insensitive)
- Key: Lemon
- Key "Sequence": LEMONLEMONLE
- Ciphertext: LXFOPVEFRNHR
- Again, how will you attack it? (No, I don't mean do it at dawn…)

Enigma

- Vigenère Ciphers and Enigma are both polyalphabetic
- Based on Substitution
- So does Enigma (a “Rotor”)



“Rail-Fence” Cipher via Transposition



Basic Techniques

- Permutations (aka Transposition)
  - *rearrange* bits, or bytes or characters in the plaintext
- Substitution
  - replace bits, or bytes, or characters, or block of characters with *substitute* value
- What is “absolutely secure”?
  - One-Time Pad, proposed by Miller/Vernam, proven by Shannon
  - *Pad* a plaintext w/ a single-use key as long as the plaintext
  - The key can only be used for *one-time*
- Enough old-school topics?

eXclusive OR (XOR)

- One-time pad: $C_i = M_i \oplus K_i$
- Stream cipher uses K to generate S
- $C_i = M_i \oplus S_i$
- How to decrypt?
- $(M_i \oplus S_i) \oplus S_i$
  - $X \oplus X = 0 \quad \forall X$
  - $Y \oplus 0 = Y \quad \forall Y$

XOR	0	1
0	0	1
1	1	0

Block Cipher vs. Stream Cipher

- Process the message block by block of constant size, e.g., 64 bits
- Each block goes through multiple rounds of permutation and substitution
- Mixed operators, data or key dependent rotation/shifting :P
- Key-dependent substitution ("S-box") :S
- More complex key scheduling
  - part of the key is used to generate the "per-round key" for each round

- Process the message bit by bit (as a stream) or byte-by-byte
- Typically have a **stream key**, its (pseudo) randomness destroys any statistical properties in the message
- Combined ("⊕" XOR) with plaintext bit by bit
- It could be simpler and faster (and can process "online")
- $C_i = M_i \oplus S_i$
  - X_i: i-th bit of ctxt/p.txt/stream key

Design of Stream Cipher

- Secret key is used to generate the "seed" of pseudo random number generator (PRNG) which outputs the random stream key
- Seed should be "large enough" to avoid exhaustive enumeration
- The PRNG should have
  - long period with no repetitions
  - statistical randomness
  - "correlation immunity" [Siegenthaler '84] [**]
- *"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."* joked by [John von Neumann]
- *"Random numbers should not be generated with a method chosen at random"* [Donald Knuth]

RC4, an example of stream cipher

- Rivest Cipher / Ron's Code
- a proprietary cipher (trade-secret) owned by RSA DSI (1987)
- Leaked to the public in 1994 via an Internet posting
- Variable key size (1 to 256-byte long), byte-oriented stream cipher
- Widely used in practice (Web SSL/TLS, Wireless LAN WEP)
- Key forms random permutation of all 8-bit values
- Use permutation to scramble input info. processed a byte at a time

RC4 Key-Scheduling Algorithm

```
// input: a key k of length L bytes

for i = 0 to 255 do // forms internal state S
    S[i] = i // starts with an array of S numbers: 0..255
j = 0
// use key to well and truly shuffle
for i = 0 to 255 do
    j = (j + S[i] + k[i mod L]) (mod 256)
    swap (S[i], S[j])
```

RC4-KSA Toy Example

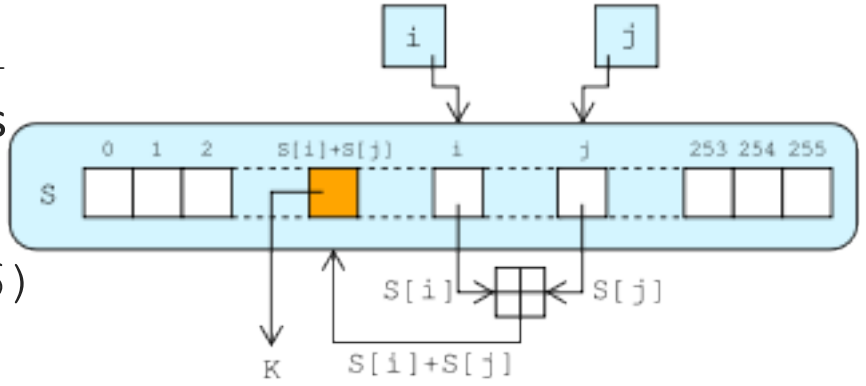
- Consider key length = 4, i.e., S = {0, 1, 2, 3}, call it S'
- Suppose K = {0, 9, 1, 2}
- i = 0: j ← (j + S[i] + K[i]) = (j + S[0] + K[0]) = 0 // No swap!
- i = 1: j ← (j + S[1] + K[1]) = (0 + 1 + 9) (mod 4) = 2
- Now S = {0, 2, 1, 3}
- i = 2: j ← (2 + S[2] + K[2]) = (2 + 1 + 1) (mod 4) = 0 // S ← {1, 2, 0, 3}
- i = 3: j ← (0 + S[3] + K[3]) = (0 + 3 + 2) (mod 4) = 1 // S ← {1, 3, 0, 2}
- Q: Do we gain anything for allowing K[i] to be as large as 9 in this process?

RC4 PRGA

// pseudo-random generation algorithm (PRGA)

```
i = j = 0
for each message byte M_i
```

(from Wikipedia)




```
    // continues shuffling array values
    i = (i + 1) (mod 256)
    j = (j + S[i]) (mod 256)
    swap(S[i], S[j])
    // sum of shuffled pair selects "stream key" value
    t = (S[i] + S[j]) (mod 256)
    C_i = M_i XOR S[t]
```
// bitwise op., e.g., 3 XOR 2 = 11 XOR 10 = 01

RC4-PRGA Toy Example

- Recall S became {1, 3, 0, 2}
- i ← i + 1 = 1
- j ← j + S[i] = 0 + 3 (mod 4) = 3
- Output S[S[1] + S[j]] = S [3 + 2 (mod 4)] = S[1] = 3

- (Demo was available at blog.markloiseau.com/2012/07/rc4-tutorial)

RC4 Security

- claimed secure against known attacks
  - demonstrated some analytical attacks, none practical
  - result is "very" non-linear
- MUST throw away the 1st few hundred bytes of the key-stream to avoid the "weak-key" problem of RC4
  1. Bias in initial output: Pr[2nd byte = 0] = 2/256
  2. Pr[(0,0)] is biased, turns out to be $1/256^2 + 1/256^3 >$ (the expected) $1/256^2$
- have a concern with WEP in 802.11, but due to key handling, not RC4 itself
- especially vulnerable to "related-key attacks" [**]
- Fluhrer-Mantin-Shamir. "Weakness in the Key Scheduling Algorithm of RC4" (http://www.crypto.com/papers/others/rc4_ksaproc.pdf) [**]
- Check Wikipedia entry to see what happened in 2014 and 2015.

DES: The Data Encryption Standard

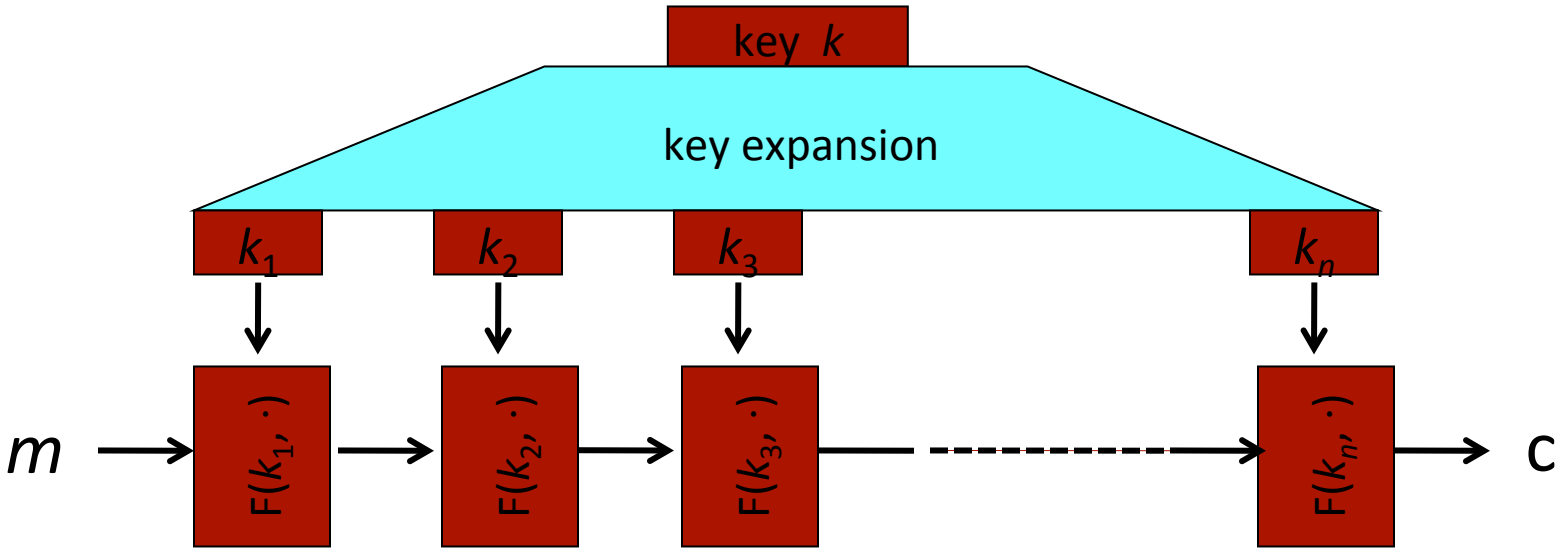
- Designed in the 1970's by IBM
  - with inputs from the NSA (National Security Agency) of US
- The most widely used encryption standard in the world
- 64 bits per block
- Use 56-bit key (7x8); for each 7-bit key-segment, a parity bit is added as checksum to form an octet (which contains redundant info). This results in a 64-bit word to be fed into the algorithm
- The same hardware can be used for both encryption and decryption based on the elegant "Feistel" network structure
- Designed to facilitate hardware implementation WHILE making software implementation much slower

Chronology of DES

- Early 1970s: Horst Feistel designs Lucifer at IBM
  - key-len = 128 bits, block-len = 128 bits
- ’73: National Bureau of Standards asks for block cipher proposals IBM submits variant of Lucifer
- ’76: NBS adopts DES as a federal standard
  - key-len = 56 bits, block-len = 64 bits
- In common use for over 20 years
  - Over 130 validated implementations
  - Widely used for Federal and banking (automated clearing house)
  - Indeed, *de facto* world-wide standard

Block Cipher by Iterations

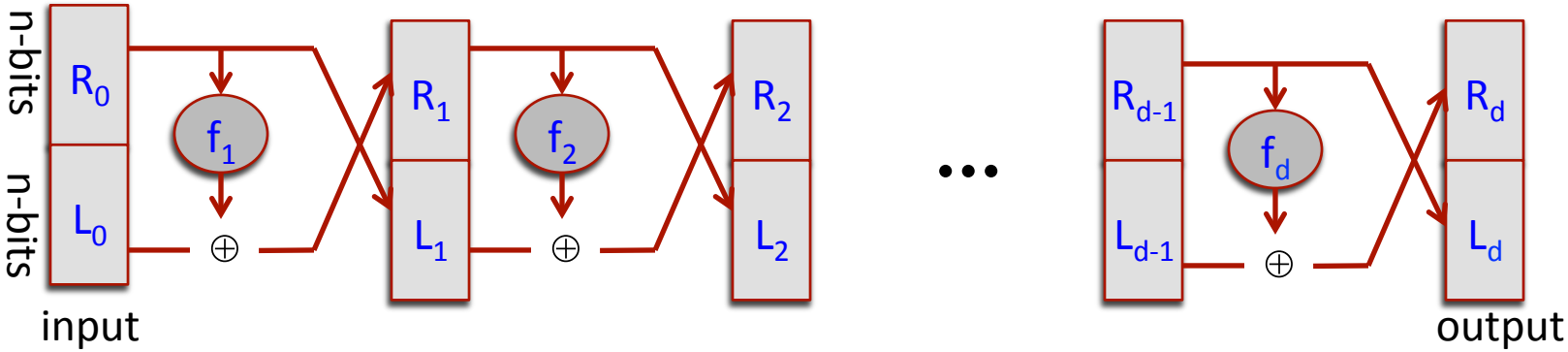
- F(k, m) is called a round function
- Challenge: least number of rounds (n)



Source: Dan Boneh @ Stanford

DES Core Idea – Feistel Network

- Given functions $f_1, \ldots, f_d$: $\{0,1\}^{32} \longrightarrow \{0,1\}^{32}$
  - We assume k_i is the implicit input of f_i
- Goal (for decryption): build invertible function F: $\{0,1\}^{64} \longrightarrow \{0,1\}^{64}$
- $R_i = f_i(R_{i-1}) \oplus L_{i-1}$, $L_i = R_{i-1}$

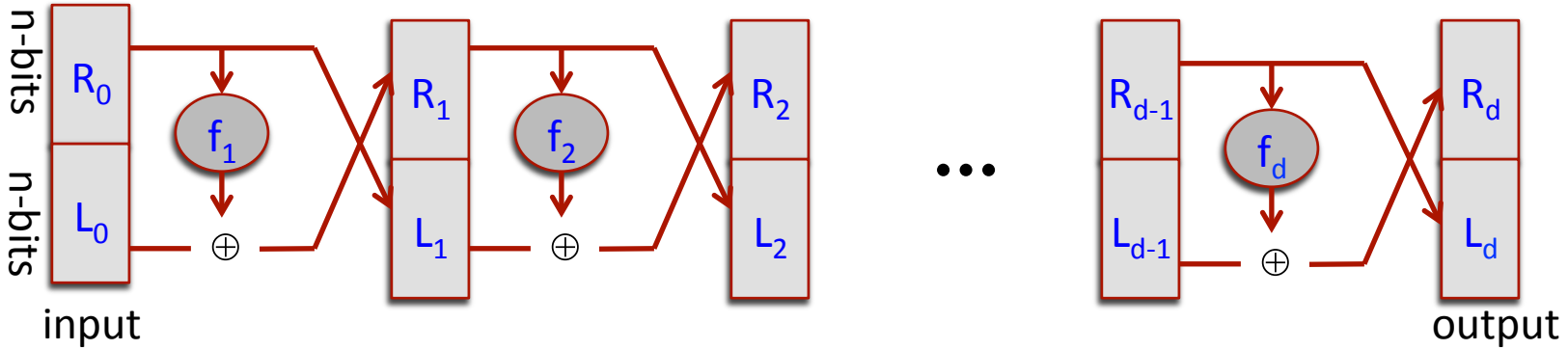


- How to construct $F^{-1}()$? Straightforward solution: construct $f^{-1}()$

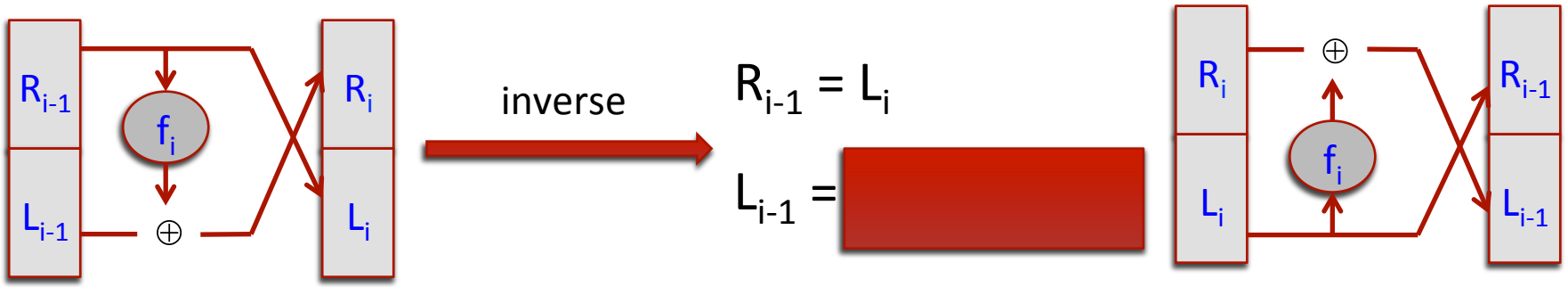
Source: Dan Boneh @ Stanford

How to construct $F^{-1}()$?

- Recall our F is:

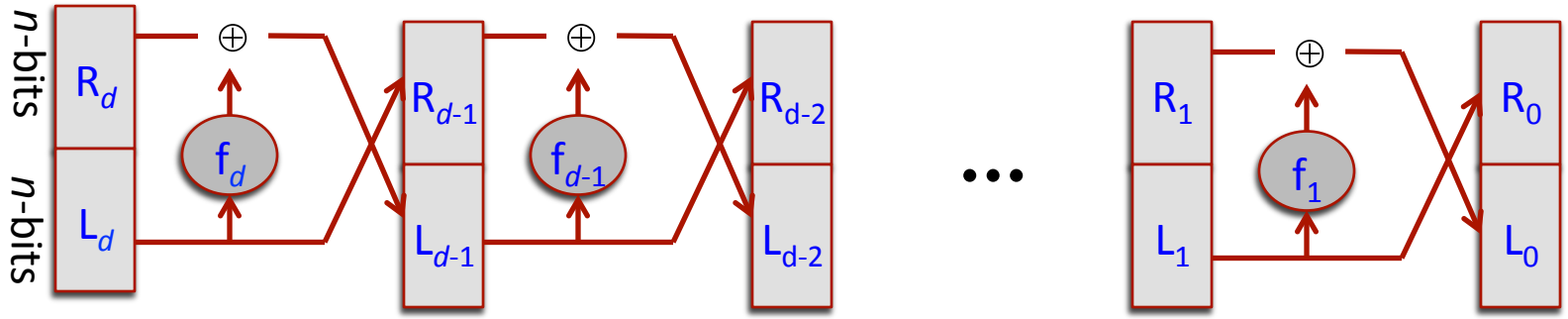


- No matter what f_i's are, we know how to invert F



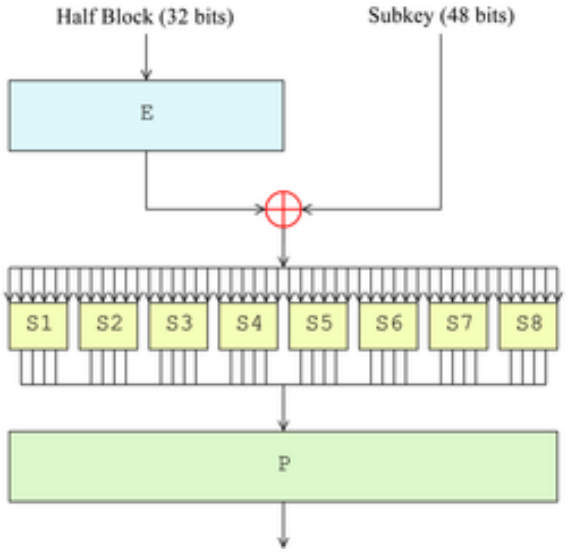
DES Decryption

- Inversion is basically the same circuit,
  with $f_1, \ldots, f_d$ applied in reverse order (but no need its inverse)
- Simply swapping the left and right halves of the input
  (no need to run the algorithm backward)
- General method for building (invertible) block ciphers
- Used in many block ciphers … but not AES



Mangler Function (f_i)

- Now we need to "mangle" the half-block of 32 bits
- Using a "sub-key" of length 48 bits for each round
  - Sub-keys are from a "key schedule"
- 32 ≠ 48? First "expand" 32-bit input to 48-bit
- i.e., Duplicate half of the bits, then key mixing "⊕"
- S-box is a 6-bit to 4-bit decoder



- Using 8 S-boxes: 8x6 = 48 bits are "decoded" to 8x4 = 32 bits
- P-box permutes these 32 bits: bits of the output of an S-box on *one round* of DES affects the input of *multiple* S-boxes on the *next round*

Some Discussions on DES

- Initial & final permutation before & after 16-round Feistel network
  - (not discussed in the previous slides, but exist in the real DES)
  - Do not add security value!
  - Merely to slow down software implementation of the algorithm
  - (recall it is a h/w-oriented design, e.g, same h/w for enc. & dec.)
  - Tried to limit the spread of the technology?
- Poor choice of the S-boxes and P-box (e.g., chosen at random) would result in an insecure block cipher
  - the S-boxes were all chosen by NSA and have special structure

Controversy behind DES

- It is commonly believed that NSA will not recommend an algorithm which it cannot break. People were worried about the existent of “backdoor” within the algorithm
- 56-bit key is too small to start with
  - (It seems like the original design was for 64-bit key and was forced to shorten the key to 56-bit, so that … NSA can break it ??)
- The rationale behind the design was never disclosed
  - The designers knew a lot of advanced cryptanalysis and DES was designed to protect against all these known ways of attack
  - Explaining the design rationale would teach “the enemy” about those advanced cryptanalysis techniques ??

Fundamental Tenet of Cryptography

- If lots of smart people have failed to solve a problem (break the code), then it probably won't be solved (broken) (soon).

- "Cryptography is the study of human stupidity" [Chow]

Cryptographic Misconceptions [Shamir]

- By Policy Makers: crypto is dangerous, but:
  - weak crypto is not a solution
  - controls can't stop the inevitable
- By Researchers: A provably secure system is secure but:
  - proven false by indirect attack
  - can be based on false assumption
  - requires careful choice of parameters
- By Implementers: Cryptography solves everything, but:
  - only basic ideas are successfully deployed
  - only simple attacks are avoided
  - bad crypto can provide a false sense of security

The Three Laws of Security [Shamir]

- Absolutely secure systems do not exist
- To halve your vulnerability,
  you have to double your expenditure
- Cryptography is typically bypassed,
  not penetrated

http://xkcd.com/538