Introduction to Cryptography

Yu Bi

ELE594 – Special Topic on Hardware Security & Trust
University of Rhode Island

Stream and Block Ciphers

a. Stream ciphers

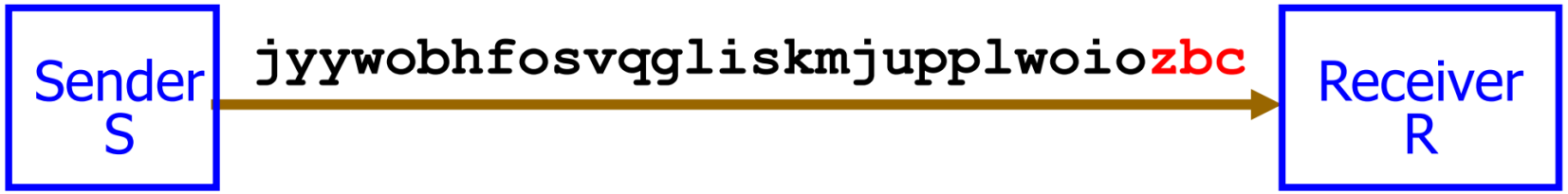
b. Problems with stream ciphers

c. Block ciphers

d. Pros / cons for stream and block ciphers

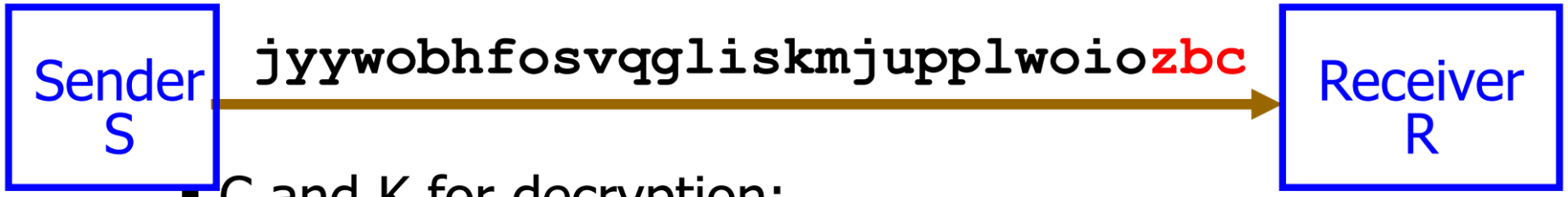
Stream Ciphers

- Stream cipher: 1 char from P → 1 char for C
  - Example: polyalphabetic cipher
    - P and K (repeated '**EXODUS**'):
      YELLOWSUBMARINEFROMYELLOWRIVER
      EXODUSEXODUSEXODUSEXODUSEXODUS
    - Encryption (char after char, using Vigenère Tableaux):
      (1) E(**Y**, **E**) → **c** (2) E(**E**, **X**) → **b** (3) E(**L**, **O**) → **z** . . .
    - C: **cbzoiowlppujmksilgqvsofhbowyyj**
    - C as sent (in the right-to-left order):



Stream Ciphers

- Example: polyalphabetic cipher - cont.
  - C as received (in the right-to-left order):



  - C and K for decryption:

    cbzoiowlppujmksilgqvsofhbowyyj
    EXODUSEXODUSEXODUSEXODUSEXODUS

  - Decryption:
    (1) D(**c**, **E**) → **Y** (2) D(**b**, **X**) → **E** (3) D(**z**, **O**) → **L** **. . .**
  - Decrypted P:

    YEL. . .

Problem with Stream Ciphers

- Problems with stream ciphers
  - Dropping a char from key K results in wrong decryption
  - Example:
    - P and K (repeated '**EXODUS**') with a char in K missing:

YELLOWSUBMARINEFROMYELLOWRIVER
EODUSEXODUSEXODUSEXODUSEXODUSE

missing X in K ! (no errors in repeated K later)

- Encryption (using VT):
  1) E(**Y**,**E**) → **c**
  2) E(**E**,**O**) → **s**
  3) E(**L**,**D**) → **o**

- Ciphertext: **cso...**

C in the order as sent (right-to-left):

...osc

Problem with Stream Ciphers

- C as received (in the right-to-left order):

- C and correct K ('EXODUS') for decryption:

```
cso...
EXO...
```

- Decryption (using VT, applying correct key):
  1) D(c, E) → Y
  2) D(s, X) → V
  3) D(o, O) → A

  ...

- Decrypted P:
  YVA... - Wrong!

What if message is corrupted in a noisy area?

Problem with Stream Ciphers

- The problem might be recoverable
  - Example:

    If R had more characters decoded, R might be able to detect that S dropped a key char, and R could recover
    - E.g., suppose that R decoded:

      YELLOW SUBMAZGTR
      - R could guess, that the 2nd word should really be:

        SUBMARINE
      - => R would know that S dropped a char from K after sending "**SUBMA**"
      - => R could go back 4 chars, drop a char from K ("recalibrate K with C"), and get "resynchronized" with S

Block Ciphers

- We can do better than using recovery for stream ciphers
  - Solution: use block ciphers

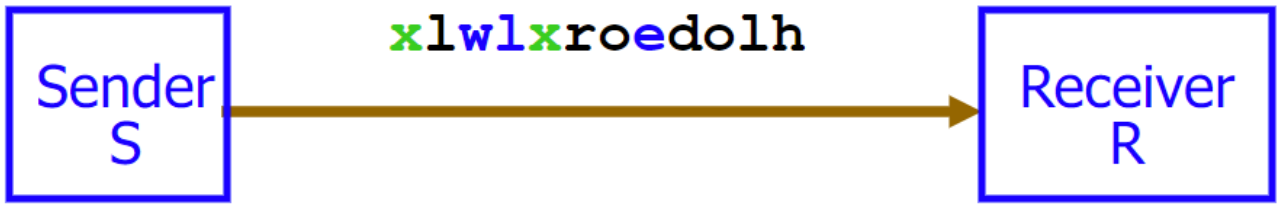
- Block cipher:
  1 *block* of chars from P → 1 *block* of chars for C
  - Example of block cipher: columnar transposition
  - Block size = "o(message length)" (informally)

Block Ciphers

- Why block size = "o(message length)" ?
  - Because R must wait for "almost" the entire C before R can decode some characters near beginning of P
  - E.g., for P = '**HELLO WORLD**', block size is "o(10)"
  - Suppose that Key = 3 (3 columns):

    HEL
    LOW
    ORL
    DXX

  - C as sent (in the right-to-left order):



Block Ciphers

- C as received (in the right-to-left order): **xlwlxroedolh**
- R knows: K = 3, block size = 12 (=> 4 rows)

123	a=10
456	b=11
789	c=12
abc	

=> R knows that characters wil be sent in the order:
1st-4th-7th-10th--2nd-5th-8th-11th--3rd-6th-9th-12th

- R must wait for at least:
  - 1 char of C to decode 1st char of P ('**h**')
  - 5 chars of C to decode 2nd char of P ('**he**')
  - 9 chars of C to decode 3rd, 4th, and 5th chars of P ('**hello**')
  - 10 chars of C to decode 6th, 7th, and 8th chars of P ('**hello wor**')
  - etc.

Block Ciphers

- *Informally*, we might call ciphers like the above example columnar transposition cipher "weak-block" ciphers
  - R can get some (even most) but not all chars of P before entire C is received
    - R can get one char of P immediately
      - » the 1st-after 1 of C (delay of 1 - 1 = 0)
    - R can get some chars of P with "small" delay
      - » e.g., 2nd-after 5 of C (delay of 5 - 2 = 3)
    - R can get some chars of P with "large" delay
      - » e.g., 3rd-after 9 of C (delay of 9 – 3 = 6)
- There are block ciphers when R cannot even start decoding C before receiving the entire C
  - *Informally*, we might call them "strong-block" ciphers

Pros and Cons

- Pros / cons for stream ciphers
  - + Low delay for decoding individual symbols
    - Can decode as soon as received
  - + Low error propagation
    - Error in $E(c_1)$ does not affect $E(c_2)$
  - - Low diffusion
    - Each char separately encoded => carries over its frequency info
  - - Susceptibility to malicious insertion / modification
    - Adversary can fabricate a new msg from pieces of broken msgs, even if he doesn't know E (just broke a few msgs)

Pros and Cons

- Pros / cons for block ciphers
  - + High diffusion
    - Frequency of a *char* from P diffused over (a few chars of) a *block* of C
  - + Immune to insertion
    - Impossible to insert a char into a block without easy detection (block size would change)
    - Impossible to modify a char in a block without easy detection (if checksums are used)

Pros and Cons

- Pros / cons for block ciphers — Part 2
  - - High delay for decoding individual chars
    - See example for '**hello worldxx**' above
      - For some E can't decode even the 1st char before whole k chars of a block are received
  - - High error propagation
    - It affects the block, not just a single char

Cryptanalysis

- What cryptanalysts do when confronted with unknown?

  Four possible situations w.r.t. available info:

  1) C available
  2) Full P available
  3) Partial P available
  4) E available (or D available)

- (1) – (4) suggest 5 different approaches

Cryptanalysis

- Cryptanalyst approaches
  1) Ciphertext-only attack
     - We have shown examples for such attacks
       - E.g., for Caesar's cipher, columnar transposition cipher
  2) Known plaintext attack
     - Analyst have C and P
       - Needs to deduce E such that C=E(P), then finds D
  3) Probable plaintext attack
     - Partial decryption provides partial match to C
       - This provides more clues

Cryptanalysis

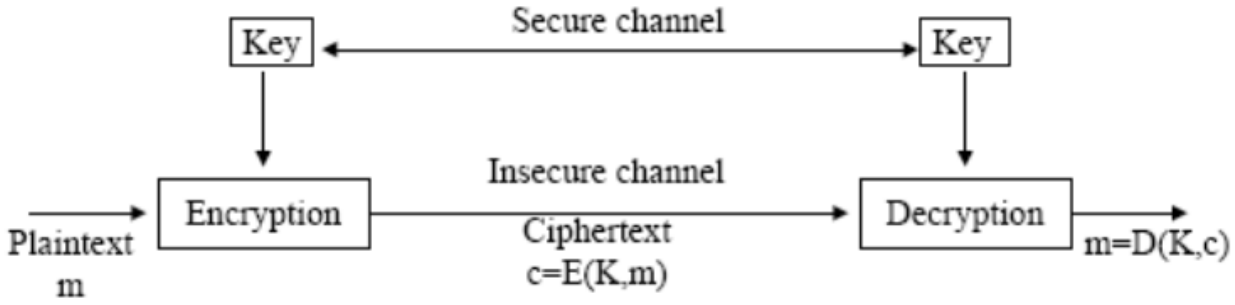
- Cryptanalyst approaches – cont.

4) Chosen plaintext attack
   - Analyst able to fabricate encrypted msgs
     - Then observe effects of msgs on adversary's actions
       - » This provides further hints

5) Chosen ciphertext attack
   - Analyst has both E and C
   - Run E for many candidate plaintexts to find P for which E(P) = C
     - Purpose: to find K_E

Symmetric and Asymmetric Crypt

- Symmetric encryption = secret key encryption
  - $K_E = K_D$ — called a secret key or a private key
  - Only sender S and receiver R know the key



[cf. J. Leiwo]

  - As long as the key remains secret, it also provides authentication (= proof of sender's identity)

Symmetric and Asymmetric Crypt

- Asymmetric encryption = public key encryption (PKE)
  - $K_E \neq K_D$ — public and private keys
- PKE systems eliminate symmetric encryption problems
  - Need no secure key distribution channel
    - => easy key distribution

Symmetric and Asymmetric Crypt

- One PKE approach:
  - R keeps her private key K_D
  - R can distribute the correspoding public key K_E to anybody who wants to send encrypted msgs to her
    - No need for secure channel to send K_E
    - Can even post the key on an open Web site — it is public!
  - Only private K_D can decode msgs encoded with public K_E!
    - Anybody (K_E is public) can encode
    - Only owner of K_D can decode

DES History

- Early 1970's - NBS (Nat'l Bureau of Standards) recognized general public's need for a secure crypto system

  NBS – part of US gov't / Now: NIST – Nat'l Inst. of Stand's & Technology

  - "Encryption for the masses" [A. Striegel]
  - Existing US gov't crypto systems were not meant to be made public
    - E.g. DoD, State Dept.
  - Problems with proliferation of commercial encryption devices
    - Incompatible
    - Not extensively tested by independent body

DES History

- 1972 - NBS calls for proposals for a *public* crypto system
  - Criteria:
    - Highly secure / easy to understand / publishable / available to all / adaptable to diverse app's / economical / efficient to use / able to be validated / exportable
    - In truth: Not *too* strong (for NSA, etc.)
- 1974 – IBM proposed its Lucifer
  - DES *based* on it
  - Tested by NSA (Nat'l Security Agency) and the general public
- Nov. 1976 – DES adopted as US standard for *sensitive but unclassified* data / communication
  - Later adopted by ISO (Int'l Standards Organization)

DES Overview

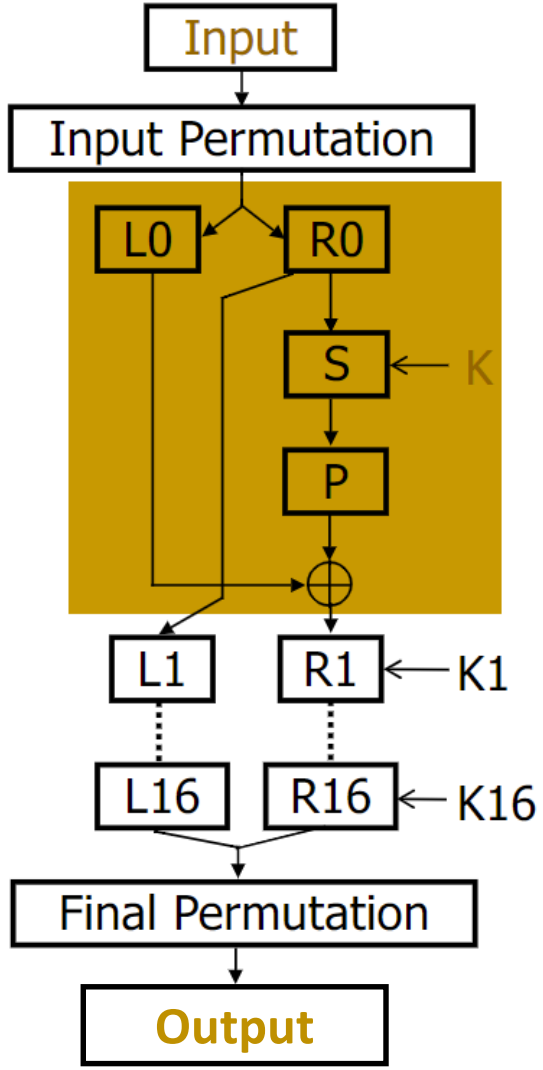
- DES - a block cipher
  - a product cipher
  - 16 rounds (iterations) on the input bits (of P)
    - substitutions (for confusion) and permutations (for diffusion)
  - Each round with a *round key*
    - Generated from the user-supplied key
- Easy to implement in S/W or H/W

- There are 72,000,000,000,000,000 (72 quadrillion) or more possible encryption keys that can be used.
- For each given message, the key can be chosen at random from among this enormous number of keys.

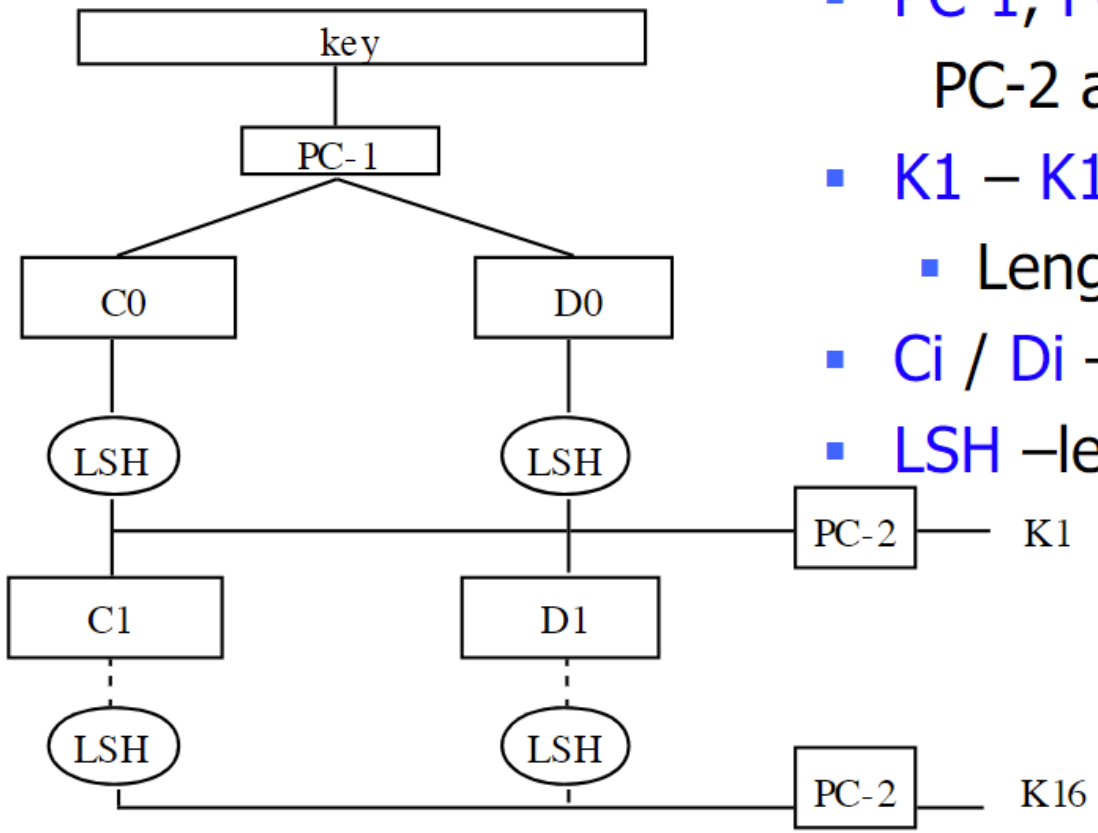
DES Structure

- Input: 64 bits (a block)
- Li/Ri– left/right half of the input block for iteration i (32 bits) – subject to substitution S and permutation P
- K - user-supplied key
- Ki - round key:
  - 56 bits used +8 unused

    (unused for E but often used for error checking)
- Output: 64 bits (a block)
- Note: Ri becomes L(i+1)
- All basic op's are simple logical ops
  - Left shift / XOR



Round Keys Generation



- key – user-supplied key (input)
- PC-1, PC-2 – permutation tables
  PC-2 also extracts 48 of 56 bits
- K1 – K16 – round keys (outputs)
  - Length(Ki) = 48
- Ci / Di – confusion / diffusion (?)
- LSH –left shift (rotation) tables

Problems with DES

- Diffie, Hellman 1977 prediction: "In a few years, technology would allow DES to be broken in days."
- Key length is fixed (= 56)
  - 2^{56} keys ~ 10^{15} keys
  - "Becoming" too short for faster computers
    - 1997: 3,500 machines – 4 months
    - 1998: special "DES cracker" h/w – 4 days
- Design decisions not public
  - Suspected of having backdoors
    - Speculation: To facilitate government access?

Double DES

- Double DES:
  - Use double DES encryption

    C = E(k2, E(k1, P))

  - Expected to multiply difficulty of breaking the encryption
    - Not true!
      - In general, 2 encryptions are not better than one

        [Merkle, Hellman, 1981]

      - Only doubles the attacker's work

Triple DES

- Triple DES:
  - Is it C = E(k3, E(k2, E(k1, P)) ?
  - Not soooo simple!

Triple DES

- Triple DES: *Is it C=E(k3, E(k2, E(k1, P))?*
  - Tricks used:

    D not E in the 2nd step, k1 used twice (in steps 1 & 3)
  - It is:

    $$C = E(k1, D(k2, E(k1, P)))$$

    and

    $$P = D(k1, E(k2, D(k1, C)))$$
  - Doubles the effective key length
    - 112-bit key is quite strong
      - Even for today's computers
      - For all feasible known attacks

Security of DES

- So, is DES insecure?
- No, not yet
  - 1997 attack required a lot of cooperation
  - The 1998 special-purpose machine is still very expensive
  - Triple DES still beyond the reach of these 2 attacks
- But ...
  - In 1995, NIST (formerly NBS) began search for new strong encryption standard

AES Contest

- 1997 – NIST calls for proposals NIST (Nat'l Institute of Standards and Technology)
  - Criteria:
    - Unclassifed code
    - Publicly disclosed
    - Royalty-free worldwide
    - Symmetric block cipher for 128-bit blocks
    - Usable with keys of 128, 192, and 256 bits
- 1998 – 15 algorithms selected

AES Contest

- 1999 – 5 finalists [cf. J. Leiwo]
  - MARS by IBM
  - RC6 by RSA Laboratories
  - Rijndael (RINE-dahl) by Joan Daemen and Vincent Rijmen
  - Serpent by Ross Anderson, Eli Biham and Lars Knudsen
  - Twofish by Bruce Schneier, John Kelsey, Doug Whiting, Dawid Wagner, Chris Hall and Niels Ferguson
- Evaluation of finalists
  - Public and private scrutiny
  - Key evaluation areas:

    security / cost or efficiency of operation /

    ease of software implementation

AES Contest

- 2001- … and the winner is …

  Rijndael (RINE-dahl)

  Authors: Vincent Rijmen + Joan Daemen (Dutchmen)

- Adopted by US gov't as

  Federal Info Processing Standard 197 (FIPS 197)

Overview of AES

- Similar to DES – cyclic type of approach
  - 128-bit blocks of P
  - # of iterations based on key length
    - 128-bit key => 9 "rounds" (called rounds, not cycles)
    - 192-bit key => 11 rounds
    - 256-bit key => 13 rounds
- Basic ops for a round:
  - Substitution – byte level (confusion)
  - Shift row (transposition) – depends on key length (diff.)
  - Mix columns – LSH and XOR (confusion +diffusion)
  - Add subkey – XOR used (confusion)

AES Strengths

- Extensive cryptanalysis by US gov't and independent experts
- Dutch inventors have no ties to NSA or other US gov't bodies (less suspicion of trapdoor)
- Solid math basis
  - Despite seemingly simple steps within rounds

DES vs. AES

	DES	AES
Date	1976	1999
Block size [bits]	64	128
Key length [bits]	56 (effect.)	128, 192, 256, or more
Encryption Primitives	substitution, permutation	substitution, shift, bit mixing
Cryptographic Primitives	confusion, diffusion	confusion, diffusion
Design	open	open
Design Rationale	closed	open
Selection process	secret	secret, but accepted public comments
Source	IBM, enhan-ced by NSA	independent Dutch cryptographers

DES vs. AES

- Weaknesses in AES?
  - 20+ yrs of experience with DES eliminated fears of its weakness (intentional or not)
    - Might be naïve…
  - Experts pored over AES for 2-year review period

DES vs. AES

- Longevity of AES?
  - DES is nearly 40 yrs old (1976)
    - DES-encrypted message can be cracked in days
  - Longevity of AES more difficult to answer
    - Can extend key length to > 256 bits (DES: 56)
      - 2 * key length => 4 * number of keys
    - Can extend number of rounds (DES: 16)
  - Extensible AES seems to be significantly better than DES, but..
    - Human ingenuity is unpredicatble!

    => Need to incessantly search for better and better encryption algorithms

RSA

- RSA = Rivest, Shamir, and Adelman (MIT), 1978
- **RSA** is one of the first practical public-key cryptosystems and is widely used for secure data transmission.
- Underlying hard problem:
  - Number theory – determining prime factors of a given (large) number (ex. factoring of small #: 5 → 5, 6 → 2 *3)
  - Arithmetic modulo n
- How secure is RSA?
  - So far remains secure (after all these years...)
  - Will quantum computing break it? TBD

RSA

- In RSA:

  $P = E(D(P)) = D(E(P))$ (order of D/E does not matter)

  – More precisely: $P = E(k_E, D(k_D, P)) = D(k_D, E(k_E, P))$

- Encryption: $C = P^e \bmod n$ $K_E = e$

  – Given C, it is very difficult to find P without knowing K_D

- Decryption: $P = C^d \bmod n$ $K_D = d$