CIS 551 / TCOM 401

Computer and Network Security

Spring 2008

Lecture 15

Announcements

- Project 3 available on the web soon.

- Plan for the next couple lectures:
  - Industrial strength crypto: DES / AES / RSA
  - Cryptographic protocols

Computational Security

- Perfect Ciphers are *unconditionally secure*
  - No amount of computation will help crack the cipher (i.e. the *only* strategy is brute force)

- In practice, strive for *computationally security*
  - Given enough power, the attacker could crack the cipher (example: brute force attack)
  - But, an attacker with only *bounded resources* is extremely unlikely to crack it
  - Example: Assume attacker has only polynomial time, then encryption algorithm that can’t be inverted in less than exponential time is secure.

Kinds of Industrial Strength Crypto

- Shared Key Cryptography
- Public Key Cryptography
- Cryptographic Hashes

- All of these aim for computational security
  - Not all methods have been proved to be intractable to crack.

Shared Key Cryptography

- Sender & receiver use the same key
- Key must remain private
- Also called *symmetric* or *secret key* cryptography
- Often are *block-ciphers*
  - Process plaintext data in blocks
- Examples: DES, Triple-DES, Blowfish, Twofish, AES, Rijndael, …

Shared Key Notation

- Encryption algorithm

  E : key x plain → cipher

  Notation: K{msg} = E(K, msg)
- Decryption algorithm

  D : key x cipher → plain
- D inverts E

  D(K, E(K, msg)) = msg
- Use capital “K” for shared (secret) keys
- Sometimes E is the same algorithm as D

Secure Channel: Shared Keys

Alice

Bart

$K_{AB}\{Hello!\}$

$K_{AB}\{Hi!\}$

K_{AB}

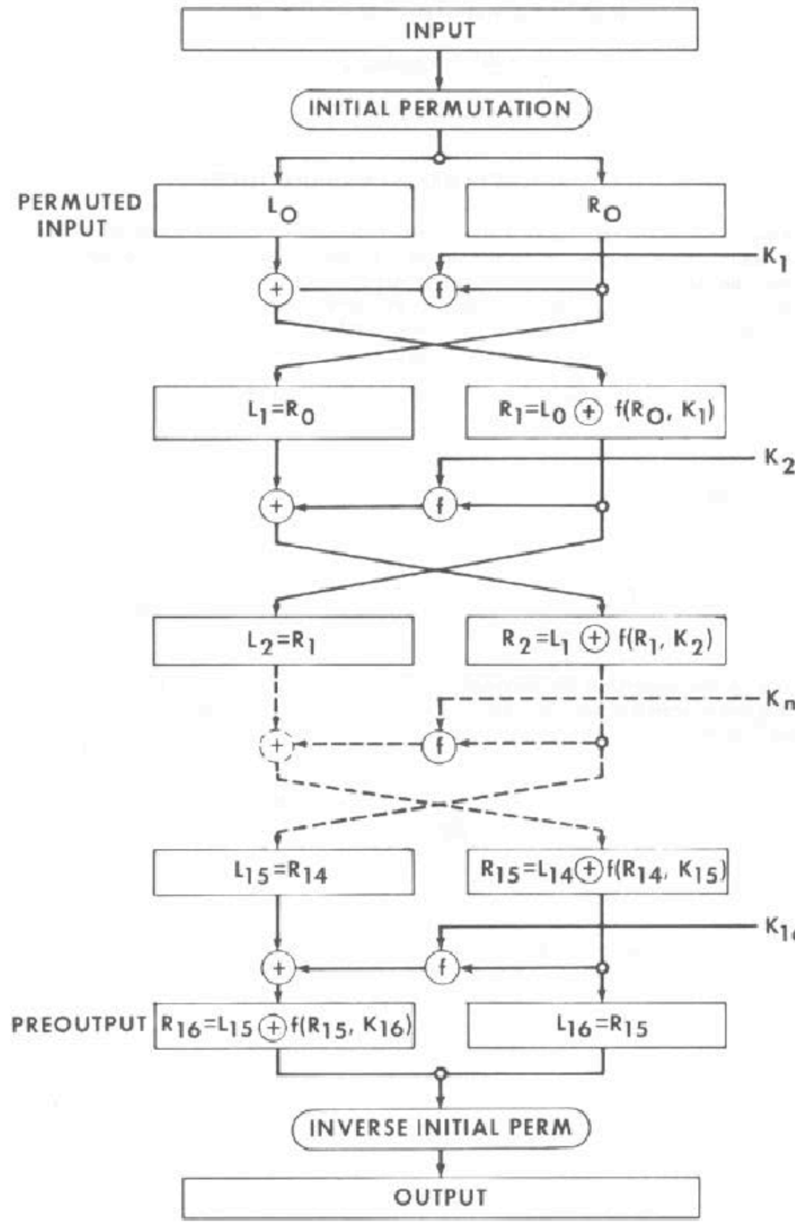
K_{AB}

Data Encryption Standard (DES)

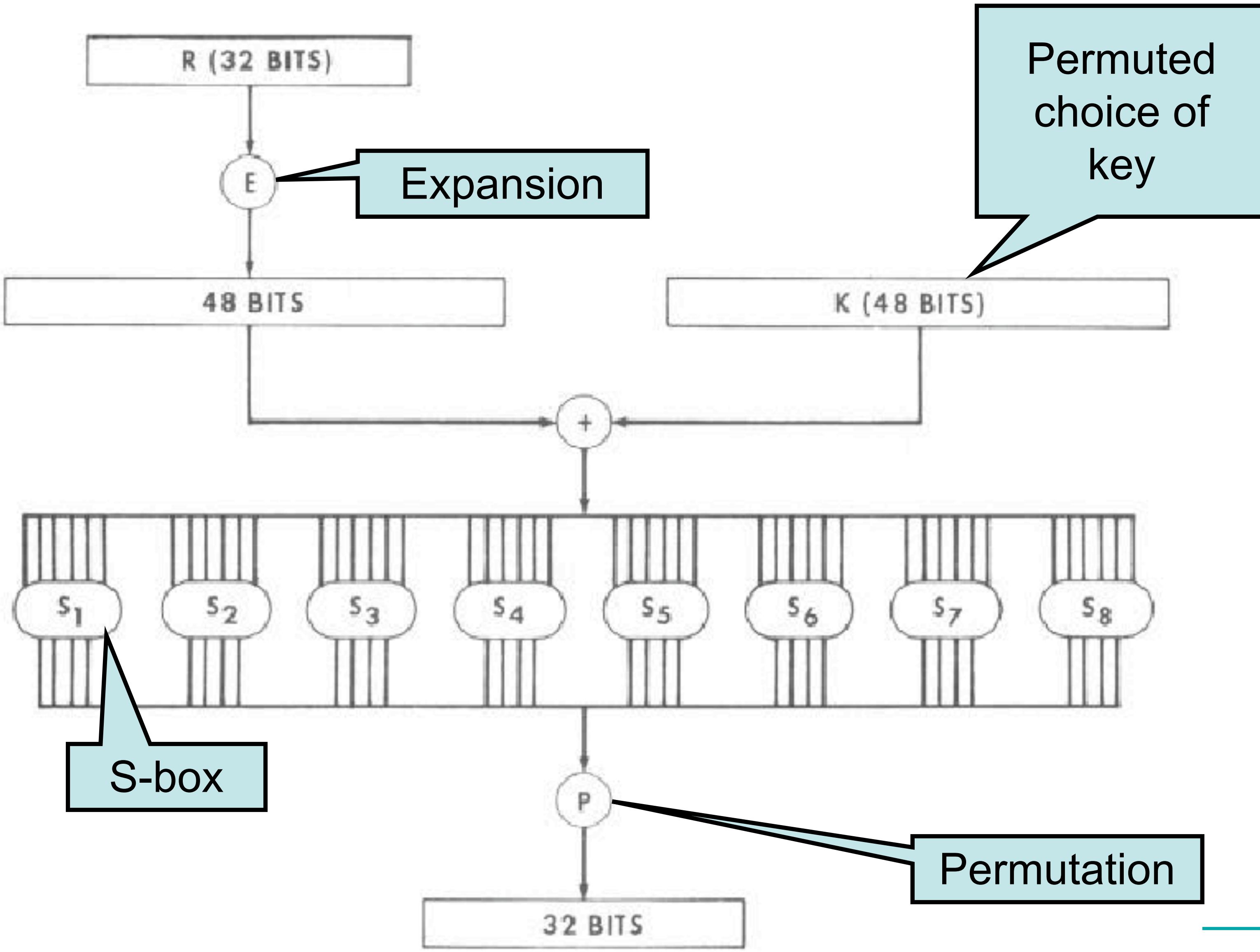
- Adopted as a standard in 1976
- Security analyzed by the National Security Agency (NSA)
  - http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
- Key length is 56 bits
  - padded to 64 bits by using 8 parity bits
- Uses simple operators on (up to) 64 bit values
  - Simple to implement in software or hardware

- Input is processed in 64 bit blocks
- Based on a series of 16 *rounds*
  - Each cycle uses permutation & substitution to combine plaintext with the key

DES Encryption

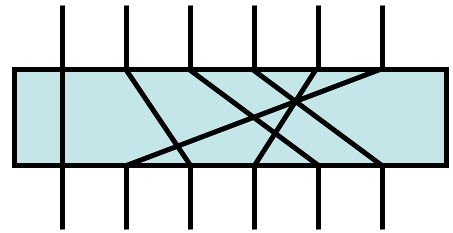


One Round of DES (f of previous slide)

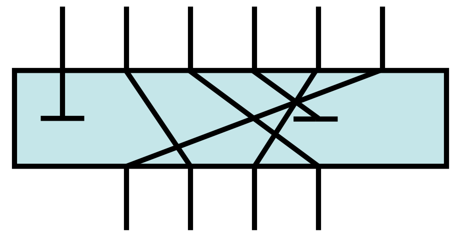


Types of Permutations in DES

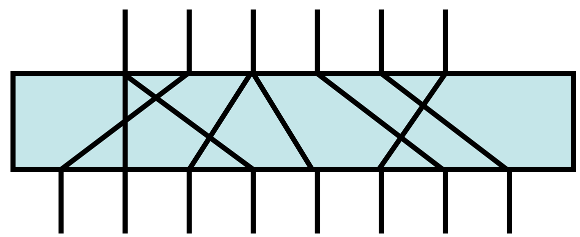
Permutation



Permuted Choice



Expansion Permutation



DES S-Boxes

- Substitution table
- 6 bits of input replaced by 4 bits of output
- Which substitution is applied depends on the input bits

- Implemented as a lookup table
  - 8 S-Boxes
  - Each S-Box has a table of 64 entries
  - Each entry specifies a 4-bit output

DES Decryption

- Use the same algorithm as encryption, but use $k_{16} \ldots k_1$ instead of $k_1 \ldots k_{16}$
- Proof that this works:
  - To obtain round j from j-1:

$$(1) \quad L_j = R_{j-1}$$

$$(2) \quad R_j = L_{j-1} \oplus f(R_{j-1}, k_j)$$

  - Rewrite in terms of round j-1:

$$(1) \quad R_{j-1} = L_j$$

$$\begin{aligned}
(2) \quad & L_{j-1} \oplus f(R_{j-1}, k_j) = R_j \\
& L_{j-1} \oplus f(R_{j-1}, k_j) \oplus f(R_{j-1}, k_j) = R_j \oplus f(R_{j-1}, k_j) \\
& L_{j-1} = R_j \oplus f(R_{j-1}, k_j) \\
& L_{j-1} = R_j \oplus f(L_j, k_j)
\end{aligned}$$

Problems with DES

- Key length too short: 56 bits
  - www.distributed.net broke a DES challenge in 1999 in under 24 hours (parallel attack)
- Other problems
  - Bit-wise complementation of key produces bit-wise complemented ciphertext
  - Not all keys are good (half 0's half 1's)
  - Differential cryptanalysis (1990): Carefully choose pairs of plaintext that differ in particular known ways (e.g. they are complements)
    - But particular choice of S boxes is secure against this (!)

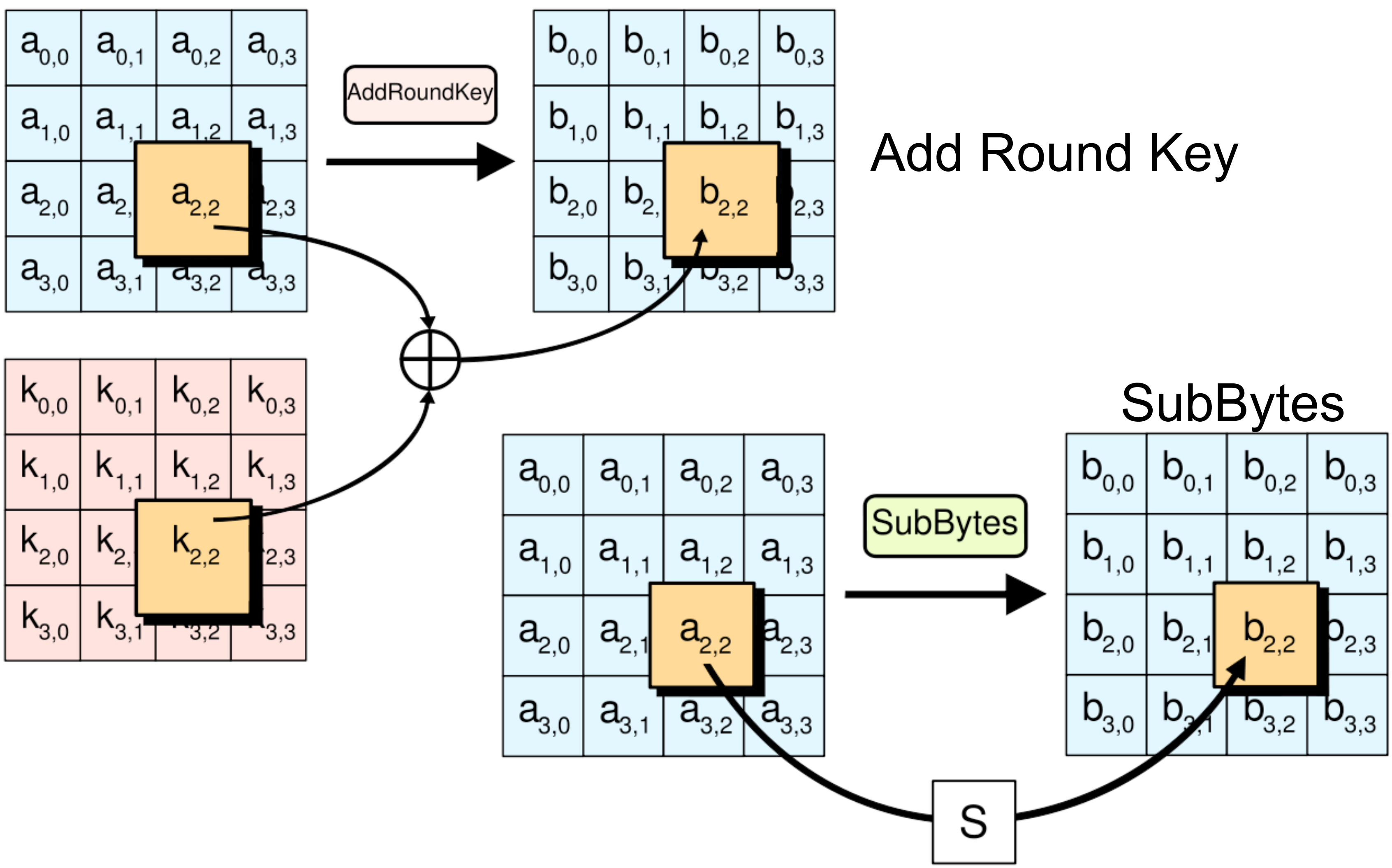
Block Cipher Performance

Algorithm	Key Length	Block Size	Rounds	Clks/Byte
Twofish	variable	128	16	18.1
Blowfish	variable	64	16	19.8
Square	128	128	8	20.3
RC5-32/16	variable	64	32	24.8
CAST-128	128	64	16	29.5
DES	56	64	16	43
Serpent	128,192,256	128	32	45
SAFER (S)K-128	128	64	8	52
FEAL-32	64, 128	64	32	65
IDEA	128	64	8	74
Triple-DES	112	64	48	116

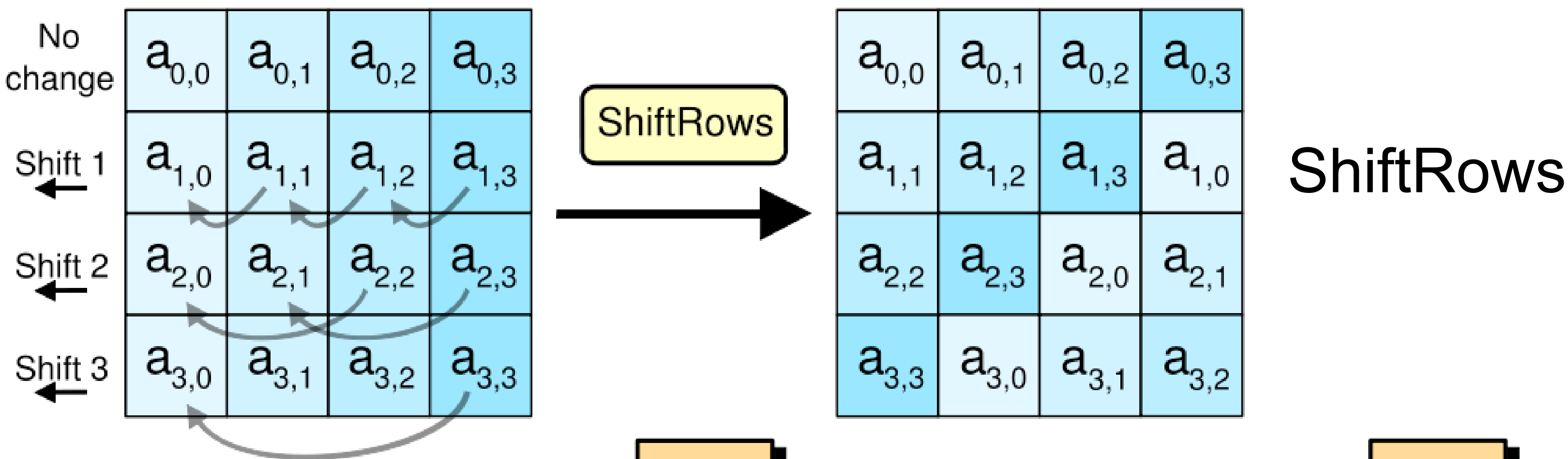
Advanced Encryption Standard (AES)

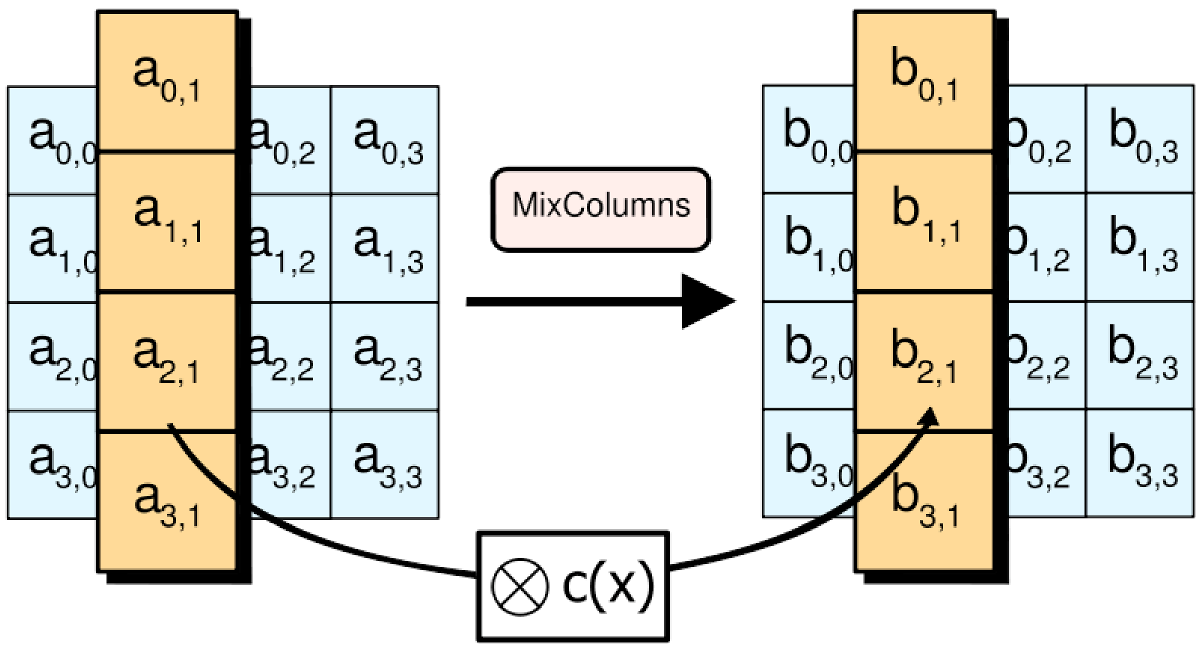
- National Institute of Standards & Technology NIST
  - Computer Security Research Center (CSRC)
  - http://csrc.nist.gov/
  - http://www.esat.kuleuven.ac.be/~rijmen/rijndael/
- Uses the Rijndael algorithm
  - Invented by Belgium researchers
    Dr. Joan Daemen & Dr. Vincent Rijmen
  - Adopted May 26, 2002
  - Key length: 128, 192, or 256 bits
  - Block size: 128, 192, or 256 bits

AES Operations



AES Operations





MixColumns

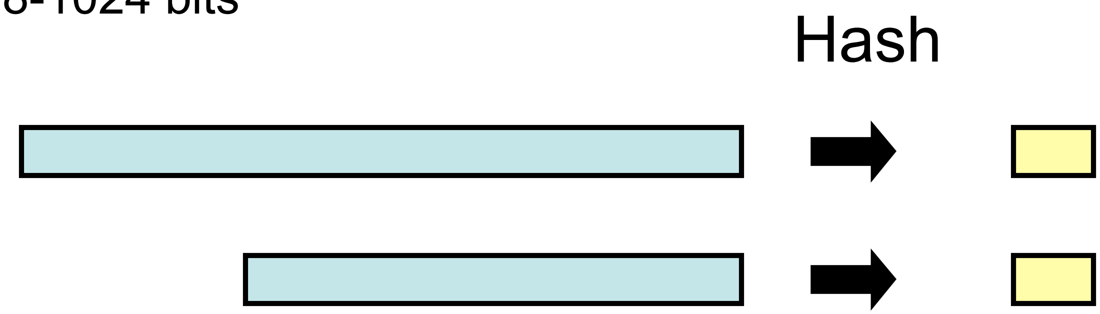
$c(x) = 3x^2 + x^2 + x + 2$

Problems with Shared Key Crypto

- Compromised key means interceptors can decrypt any ciphertext they've acquired.
  - Change keys frequently to limit damage
- Distribution of keys is problematic
  - Keys must be transmitted securely
  - Use couriers?
  - Distribute in pieces over separate channels?
- Number of keys is $O(n^2)$ where n is # of participants
- Potentially easier to break?

Hash Algorithms

- Take a variable length string
- Produce a fixed length digest
  - Typically 128-1024 bits



- (Noncryptographic) Examples:
  - Parity (or byte-wise XOR)
  - CRC (cyclic redundancy check) used in communications
  - Ad hoc hashes used for hash tables
- Realistic Example
  - The NIST Secure Hash Algorithm (SHA) takes a message of less than 2^{64} bits and produces a digest of 160 bits

Cryptographic Hashes

- Create a hard-to-invert summary of input data
- Useful for integrity properties
  - Sender computes the hash of the data, transmits data and hash
  - Receiver uses the same hash algorithm, checks the result
- Like a check-sum or error detection code
  - Uses a cryptographic algorithm internally
  - More expensive to compute
- Sometimes called a Message Digest
- History:
  - Message Digest (MD4 -- invented by Rivest, MD5)
  - Secure Hash Algorithm - 1993 - (SHA-0)
  - Secure Hash Algorithm (SHA-1)
  - SHA-2   (actually a family of hash algorithms with varying output sizes)

- Attacks have been found against both SHA-0 and SHA-1

Uses of Hash Algorithms

- Hashes are used to protect *integrity* of data
  - Virus Scanners
  - Program fingerprinting in general
  - Modification Detection Codes (MDC)
- Message Authenticity Code (MAC)
  - Includes a cryptographic component
  - Send (msg, hash(msg, key))
  - Attacker who doesn’t know the key can’t modify msg (or the hash)
  - Receiver who knows key can verify origin of message
- Make digital signatures more efficient (we'll see this later)

Desirable Properties

- The probability that a randomly chosen message maps to an n-bit hash should ideally be $(\frac{1}{2})^n$.
  - Attacker must spend a lot of effort to be able to modify the source message without altering the hash value

- Hash functions *h* for cryptographic use as MDC's fall in one or both of the following classes.
  - *Collision Resistant Hash Function:* It should be computationally infeasible to find two distinct inputs that hash to a common value (ie. $h(x) = h(y)$).
  - *One Way Hash Function:* Given a specific hash value y, it should be computationally infeasible to find an input x such that $h(x)=y$.

Secure Hash Algorithm (SHA)

- Pad message so it can be divided into 512-bit blocks, including a 64 bit value giving the length of the original message.
- Process each block as 16 32-bit words called *W(t)* for *t* from 0 to 15.
- Expand from these 16 words to 80 words by defining as follows for each t from 16 to 79:
  - $W(t) := W(t-3) \oplus W(t-8) \oplus W(t-14) \oplus W(t-16)$
- Constants H0, …, H5 are initialized to special constants
- Result is final contents of H0, … , H5

```
for each 16-word block begin
    A := H0; B := H1; C := H2; D := H3; E := H4
    for I := 0 to 19 begin
        TEMP := S(5,A) + ((B ∧ C) ∨ (¬ B ∧ D)) + E + W(I) + 5A827999;
        E := D; D := C; C := S(30,B); B := A; A := TEMP
    end
                                        Chaining Variables
    for I := 20 to 39 begin
        TEMP := S(5,A) + (B ⊕ C ⊕ D) + E + W(I) + 6ED9EBA1;
        E := D; D := C; C := S(30,B); B := A; A := TEMP
    end
    for I := 40 to 59 begin
        TEMP := S(5,A) + ((B ∧ C) ∨ (B ∧ D) ∨ (C ∧ D)) + E + W(I) + 8F1BBCDC;
        E := D; D := C; C := S(30,B); B := A; A := TEMP
    end
                                   Shift A left 5 bits
    for I := 60 to 79 begin
        TEMP := S(5,A) + (B ⊕ C ⊕ D) + E + W(I) + CA62C1D6;
        E := D; D := C; C := S(30,B); B := A; A := TEMP
    end
    H0 := H0+A; H1 := H1+B; H2 := H2+C; H3 := H3+D; H4 := H4+E
end
```

Attacks against SHA-1

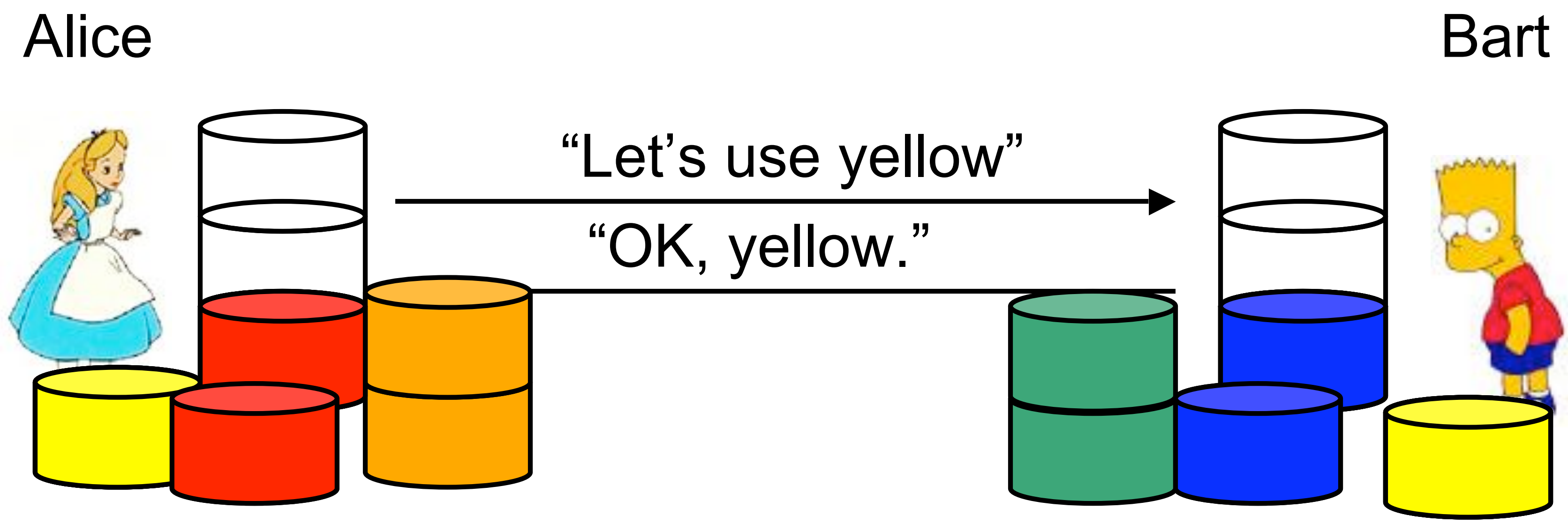
- In early 2005, Rijmen and Oswald published an attack on a reduced version of SHA-1 (53 out of 80 rounds) which finds collisions with a complexity of fewer than 2^{80} operations.
- In February 2005, an attack by Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu was announced. The attacks can find collisions in the full version of SHA-1, requiring fewer than 2^{69} operations (brute force would require 2^{80}.)
- In August 2005, same group lowered the threshold to 2^{63}.

- May lead to more attacks…

Diffie-Hellman Key Exchange

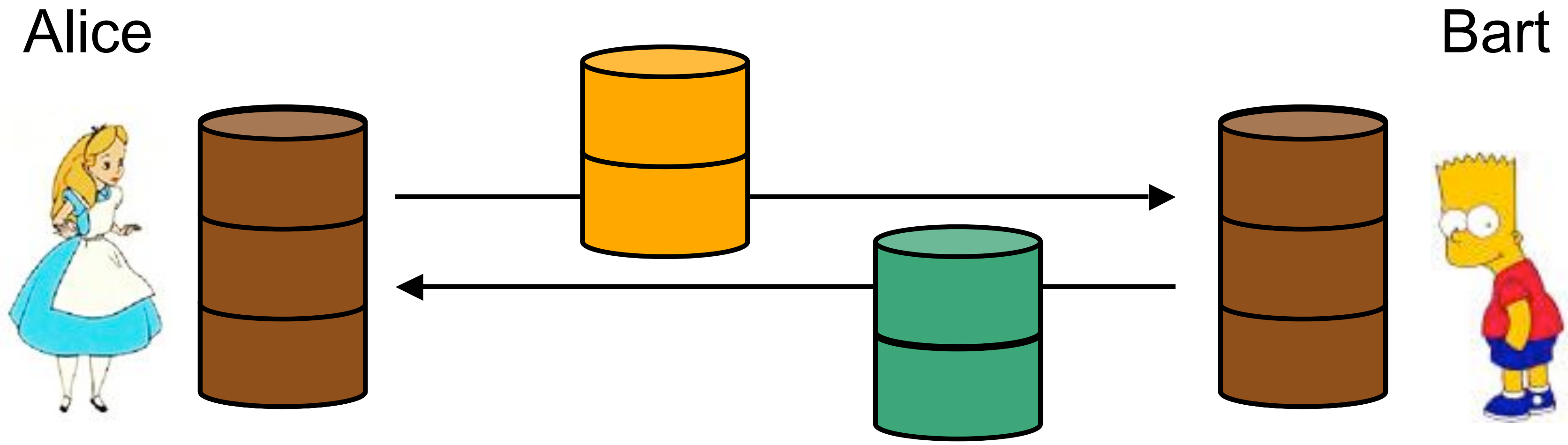
- Problem with shared-key systems:
  Distributing the shared key
- Suppose that Alice and Bart want to agree on a secret (i.e. a key)
  - Communication link is public
  - They don't already share any secrets

Diffie-Hellman by Analogy: Paint



1. Alice & Bart decide on a public color, and mix one liter of that color.
2. They each choose a random secret color, and mix two liters of their secret color.
3. They keep one liter of their secret color, and mix the other with the public color.

Diffie-Hellman by Analogy: Paint

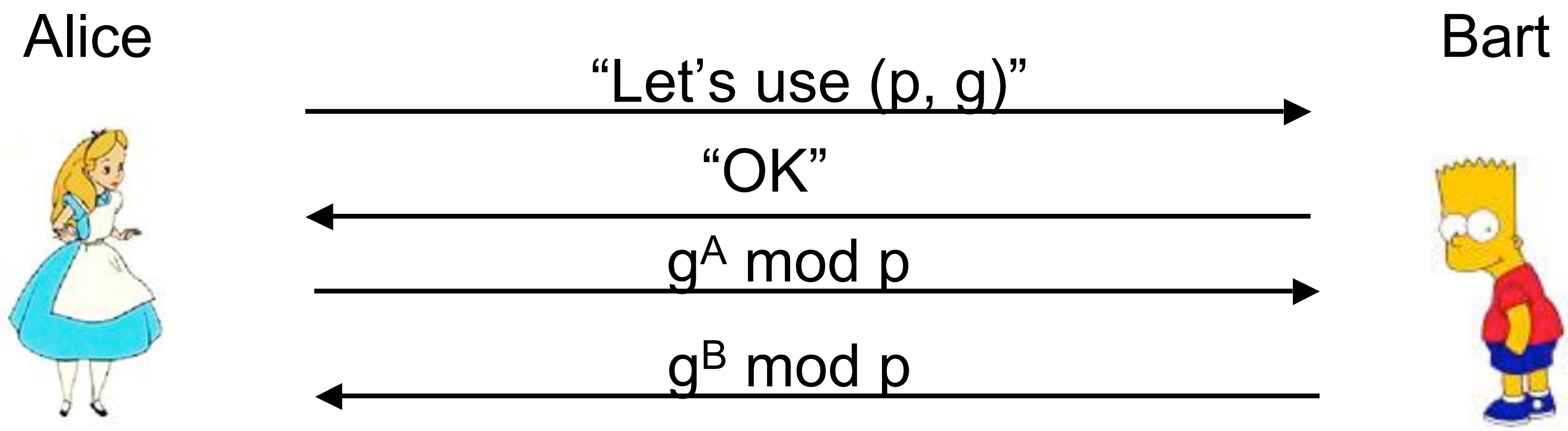


4. They exchange the mixtures over the public channel.
5. When they get the other person's mixture, they combine it with their retained secret color.
6. The secret is the resulting color: Public + Alice's + Bart's

Diffie-Hellman Key Exchange

- Choose a prime p (publicly known)
  - Should be about 512 bits or more
- Pick g < p (also public)
  - g must be a *primitive root* of p.
  - A primitive root *generates* the finite field p.
  - Every n in {1, 2, …, p-1} can be written as g^k mod p
  - Example: 2 is a primitive root of 5
  - $2^0 = 1$ $2^1 = 2$ $2^2 = 4$ $2^3 = 3$ (mod 5)
  - Intuitively means that it's hard to take logarithms base g because there are many candidates.

Diffie-Hellman



1. Alice & Bart decide on a public prime p and primitive root g.
2. Alice chooses secret number A. Bart chooses secret number B
3. Alice sends Bart $g^A \bmod p$.
4. The shared secret is $g^{AB} \bmod p$.

Details of Diffie-Hellman

- Alice computes g^{AB} mod p because she knows A:
  - g^{AB} mod p = $(g^B$ mod p$)^A$ mod p
- An eavesdropper gets g^A mod p and g^B mod p
  - They can easily calculate g^{A+B} mod p but that doesn't help.
  - The problem of computing discrete logarithms (to recover A from g^A mod p is hard.

Example

- Alice and Bart agree that $q=71$ and $g=7$.
- Alice selects a private key $A=5$ and calculates a public key $g^A \equiv 7^5 \equiv 51 \pmod{71}$. She sends this to Bart.
- Bart selects a private key $B=12$ and calculates a public key $g^B \equiv 7^{12} \equiv 4 \pmod{71}$. He sends this to Alice.
- Alice calculates the shared secret:
  $S \equiv (g^B)^A \equiv 4^5 \equiv 30 \pmod{71}$
- Bart calculates the shared secret
  $S \equiv (g^A)^B \equiv 51^{12} \equiv 30 \pmod{71}$

Why Does it Work?

- Security is provided by the difficulty of calculating discrete logarithms.
- Feasibility is provided by
  - The ability to find large primes.
  - The ability to find primitive roots for large primes.
  - The ability to do efficient modular arithmetic.
- Correctness is an immediate consequence of basic facts about modular arithmetic.