KDK College Of Engineering, Nagpur

Department Of Computer Science & Engineering

Name of Faculty: Prof. V. R. Surjuse

Name Of Subject: Cryptography & Network Security

Branch: CSE

Sem: VII

What is network security?

Confidentiality: only sender, intended receiver should understand message contents

- sender encrypts message
- receiver decrypts message

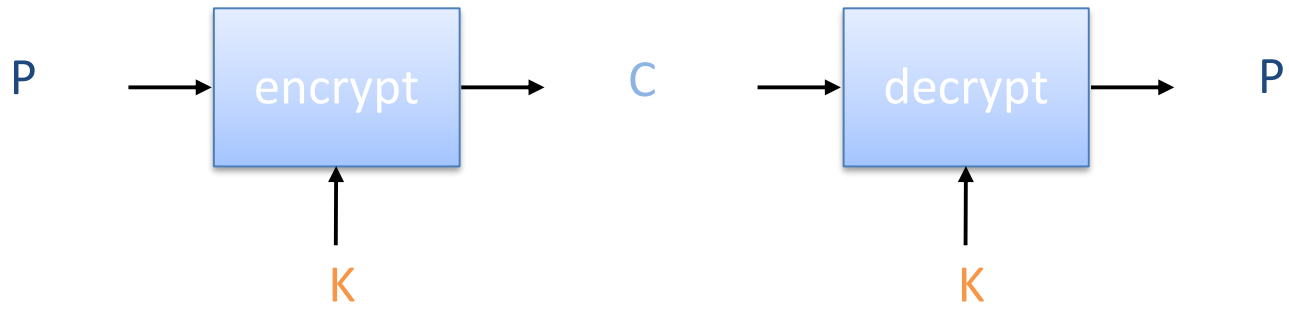
Authentication: sender, receiver want to confirm identity of each other

Message integrity: sender, receiver want to ensure message not altered without detection

Access and availability: services must be accessible and available to users

Symmetric key cryptography

- Scenario
  - A wants to send a message (plaintext P) to B.
  - The communication channel is insecure and can be eavesdropped
  - If A and B have previously agreed on a symmetric encryption scheme and a secret key K, the message can be sent encrypted (ciphertext C)
- **Issues**
  - What is a good symmetric encryption scheme?
  - What is the complexity of encrypting/decrypting?
  - What is the size of the ciphertext, relative to the plaintext?



Substitution Ciphers

- Each letter is uniquely replaced by another.
- There are 26! possible substitution ciphers for English language.

substituting one thing for another

- Simplest one: monoalphabetic cipher:
  - substitute one letter for another (**Caesar Cipher**)

Vigenere Cipher

- Idea: Uses Caesar's cipher with various different shifts, in order to hide the distribution of the letters.
- A key defines the shift used in each letter in the text
- A key word is repeated as many times as required to become the same length

Plain text: I a t t a c k

Key: 2 3 4 2 3 4 2 (key is “234”)

Cipher text: K d x v d g m

Problem of Vigenere Cipher

- Vigenere is easy to break
- Assume we know the length of the key. We can organize the ciphertext in rows with the same length of the key. Then, every column can be seen as encrypted using Caesar's cipher.

- The length of the key can be found using several methods:
  - 1. If short, try 1, 2, 3, ...
  - 2. Find repeated strings in the ciphertext. Their distance is expected to be a multiple of the length. Compute the gcd of (most) distances.
  - 3. Use the index of coincidence.

Substitution Boxes

- Substitution can also be done on binary numbers.
- Such substitutions are usually described by substitution boxes, or S-boxes.

	00	01	10	11
00	0011	0100	1111	0001
01	1010	0110	0101	1011
10	1110	1101	0100	0010
11	0111	0000	1001	1100

(a)

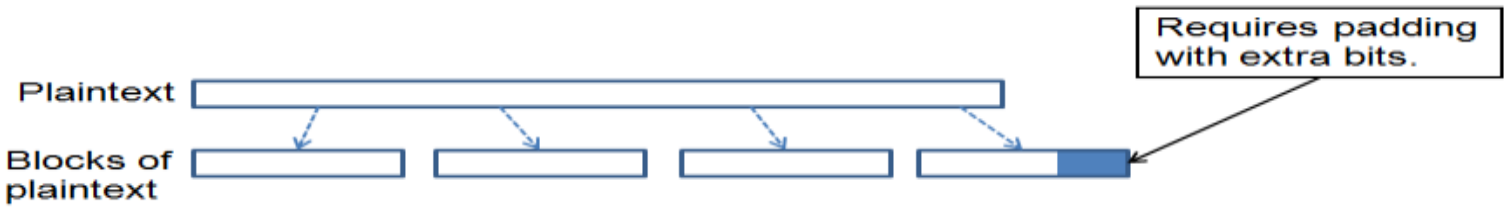
	0	1	2	3
0	3	8	15	1
1	10	6	5	11
2	14	13	4	2
3	7	0	9	12

(b)

A 4-bit S-box (a) An S-box in binary. (b) The same S-box in decimal. This particular S-box is used in the Serpent cryptosystem, which

Block Ciphers

- In a **block cipher:**
  - Plaintext and ciphertext have fixed length b (e.g., 128 bits)
  - A plaintext of length n is partitioned into a sequence of m **blocks**, P[0], …, P[m–1], where $n \leq bm < n + b$
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks



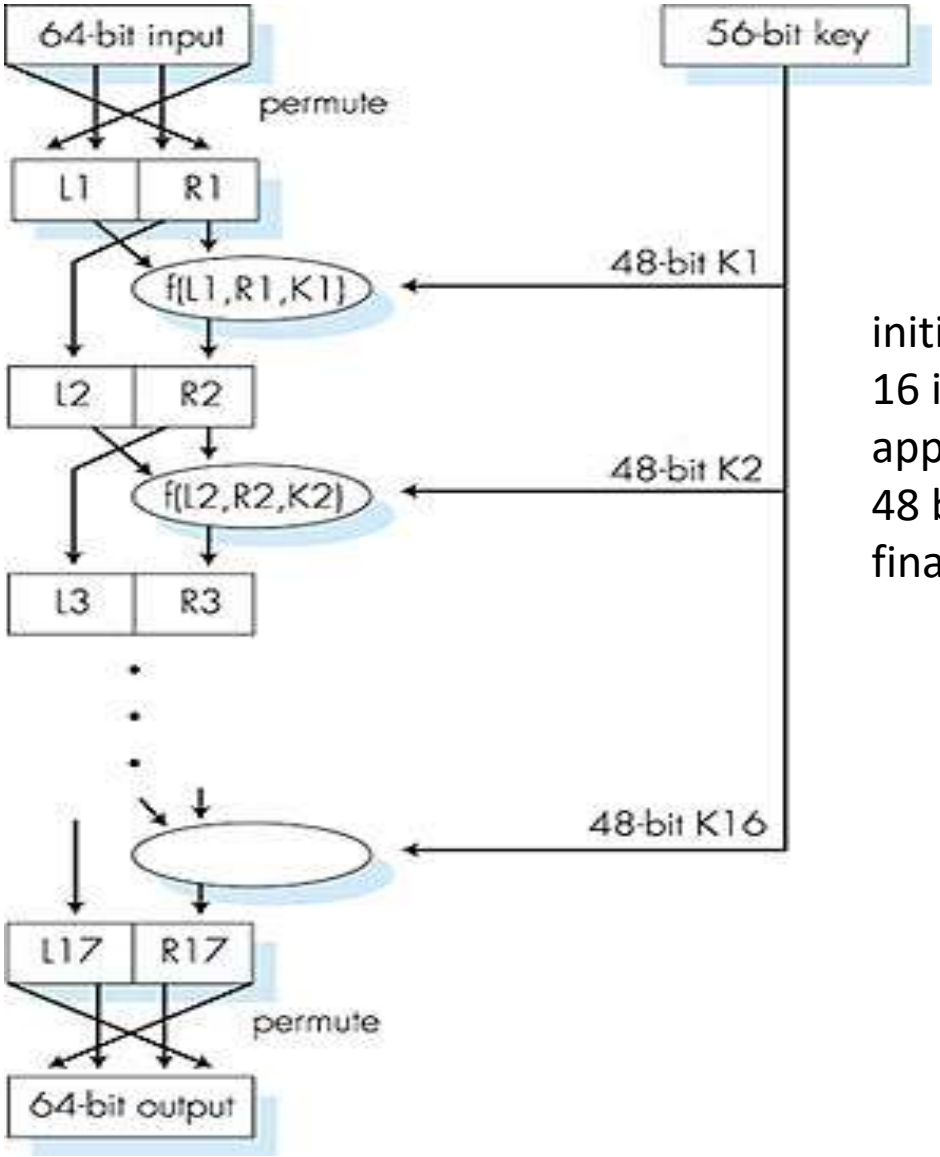
Padding

- Block ciphers require the length n of the plaintext to be a multiple of the block size b
- Padding the last block needs to be unambiguous (cannot just add zeroes)
- When the block size and plaintext length are a multiple of 8, a common padding method (PKCS5) is a sequence of identical bytes, each indicating the length (in bytes) of the padding
- Example for b = 128 (16 bytes)
  - Plaintext: “Roberto” (7 bytes)
  - Padded plaintext: “Roberto999999999” (16 bytes), where 9 denotes the number and not the character
- We need to always pad the last block, which may consist only of padding

Block Ciphers in Practice

- Data Encryption Standard (DES)
  - Developed by IBM and adopted by NIST in 1977
  - 64-bit blocks and 56-bit keys
  - Small key space makes exhaustive search attack feasible since late 90s
- Triple DES (3DES)
  - Nested application of DES with three different keys KA, KB, and KC
  - Effective key length is 168 bits, making exhaustive search attacks unfeasible
  - $C = E_{KC}(D_{KB}(E_{KA}(P)))$; $P = D_{KA}(E_{KB}(D_{KC}(C)))$
  - Equivalent to DES when KA=KB=KC (backward compatible)
- Advanced Encryption Standard (AES)
  - Selected by NIST in 2001 through open international competition and public discussion
  - 128-bit blocks and several possible key lengths: 128, 192 and 256 bits
  - Exhaustive search attack not currently possible
  - AES-256 is the symmetric encryption algorithm of choice

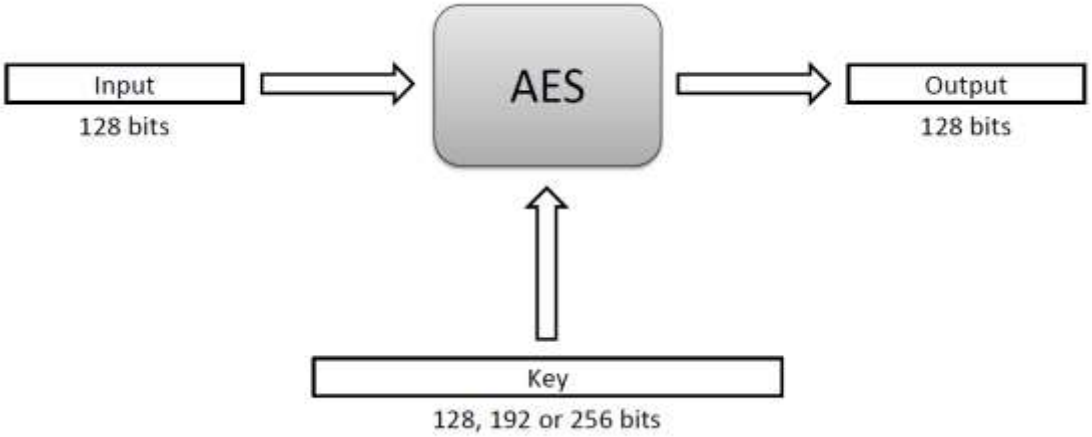
Symmetric key crypto: DES



initial permutation
16 identical “rounds” of function application, each using different 48 bits of key
final permutation

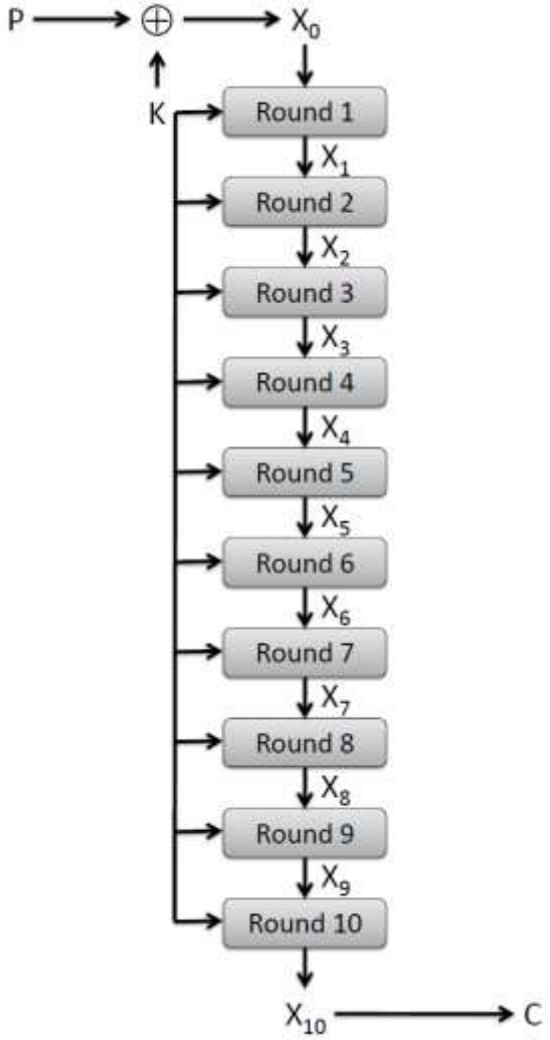
The Advanced Encryption Standard (AES)

- AES is a block cipher that operates on 128-bit blocks. It is designed to be used with keys that are 128, 192, or 256 bits long, yielding ciphers known as AES-128, AES-192, and AES-256.



AES Round Structure

- The 128-bit version of the AES encryption algorithm proceeds in ten rounds.
- Each round performs an invertible transformation on a 128-bit array, called **state**.
- The initial state X_0 is the XOR of the plaintext P with the key K:
- $X_0 = P \text{ XOR } K$.
- Round i (i = 1, …, 10) receives state X_{i-1} as input and produces state X_i.
- The ciphertext C is the output of the final round: $C = X_{10}$.



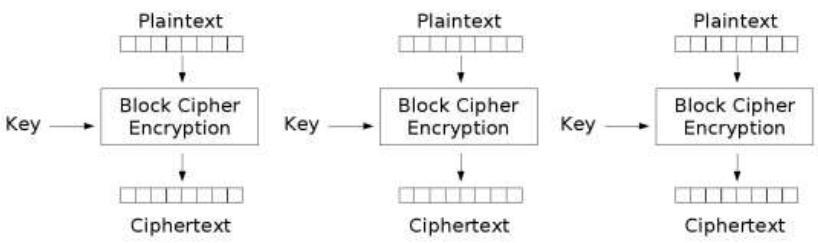
AES Rounds

- Each round is built from four basic steps:

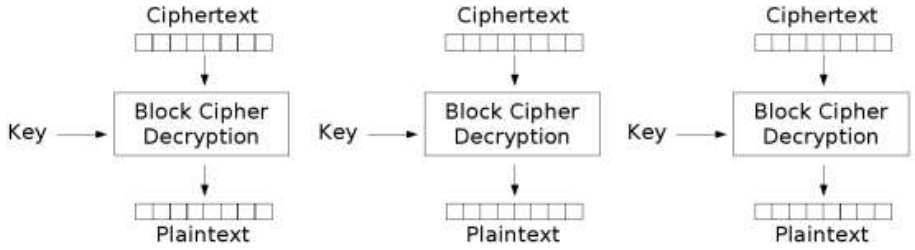
1. **SubBytes step**: an S-box substitution step
2. **ShiftRows step**: a permutation step
3. **MixColumns step**: a matrix multiplication step
4. **AddRoundKey step**: an XOR step with a **round key** derived from the 128-bit encryption key

Block Cipher Modes

- A block cipher mode describes the way a block cipher encrypts and decrypts a sequence of message blocks.
- Electronic Code Book (ECB) Mode (is the simplest):
  - Block P[i] encrypted into ciphertext block $C[i] = E_K(P[i])$
  - Block C[i] decrypted into plaintext block $M[i] = D_K(C[i])$



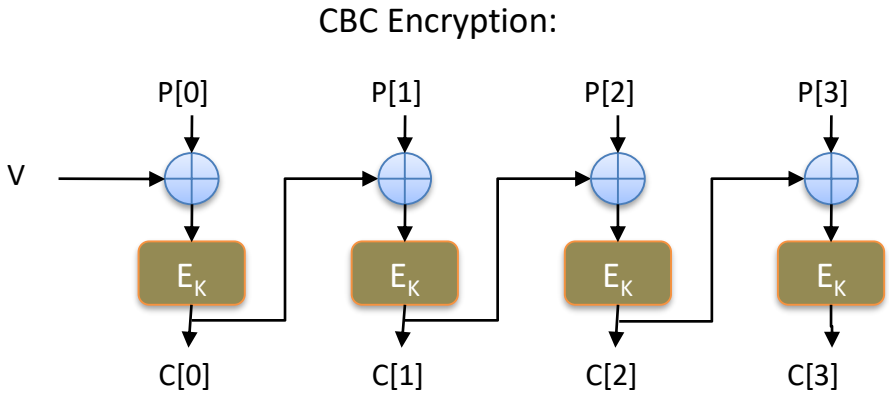
Electronic Codebook (ECB) mode encryption

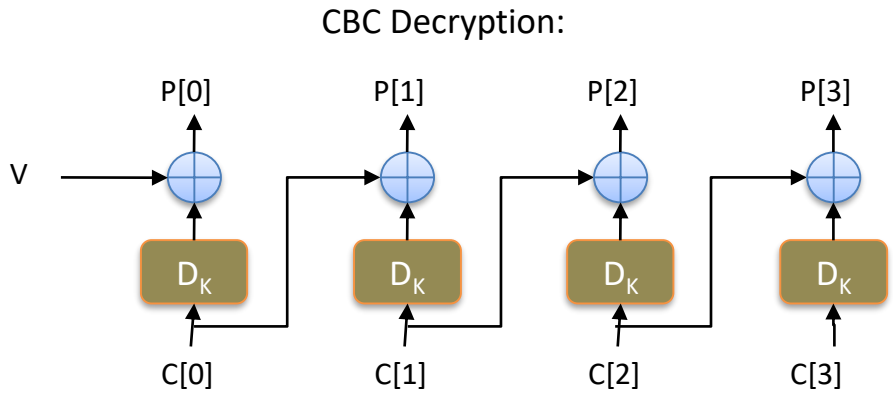


Electronic Codebook (ECB) mode decryption

Cipher Block Chaining (CBC) Mode

- In Cipher Block Chaining (CBC) Mode
  - The previous ciphertext block is combined with the current plaintext block $C[i] = E_K(C[i-1] \oplus P[i])$
  - $C[-1] = V$, a random block separately transmitted encrypted (known as the initialization vector)
  - Decryption: $P[i] = C[i-1] \oplus D_K(C[i])$





Strengths and Weaknesses of CBC

- Strengths:
  - Doesn’t show patterns in the plaintext
  - Is the most common mode
  - Is fast and relatively simple
- Weaknesses:
  - CBC requires the reliable transmission of all the blocks sequentially
  - CBC is not suitable for applications that allow packet losses (e.g., music and video streaming)

Hill Cipher: a cipher based on matrix multiplication

- Message P =“ACTDOG”, use m=3
  - Break into two blocks: “ACT”, and “DOG”
  - 'A' is 0, 'C' is 2 and 'T' is 19, “ACT” is the vector: x= $\begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix}$
  - Encryption key is a 3*3 matrix: K= $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$
  - The cipher text of the first block is:

    c = K·x $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} = \begin{pmatrix} 67 \\ 222 \\ 319 \end{pmatrix} \equiv \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \pmod{26} = \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$

    c =‘POH’

Hill Cipher

- If the first block plaintext is ‘CAT’
  - x = $\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$
  - c=K · x $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix} \equiv \begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix} \equiv \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix} \pmod{26}$
  - c= ‘FIN’
  - The Hill cipher has achieved Shannon's diffusion, and an n-dimensional Hill cipher can diffuse fully across n symbols at once.
  - This and the previous slide’s examples are from Wikipedia http://en.wikipedia.org/wiki/Hill_cipher

Hill Cipher Decryption

- x= $K^{-1} \cdot c$

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}^{-1} \equiv \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \pmod{26}$$

- Its features:
  - Interpret letters as numbers
  - Linear algebra
- Both features have been used in DES and AES

Hill Cipher to Realize Transposition

- Transposition can be realized by special Hill cipher
- Special key K = $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Then it corresponds to permutation:
  - π: (1,2,3) → (2,1,3)

Stream Cipher

- Key stream
  - Pseudo-random sequence of bits S = S[0], S[1], S[2], …
  - Can be generated on-line one bit (or byte) at the time
- Stream cipher
  - XOR the plaintext with the key stream C[i] = S[i] $\oplus$ P[i]
  - Suitable for plaintext of arbitrary length generated on the fly, e.g., media stream
- Synchronous stream cipher
  - Key stream obtained only from the secret key K
    - Independent with plaintext and ciphertext
  - Works for high-error channels if plaintext has packets with sequence numbers
  - Sender and receiver must synchronize in using key stream
  - If a digit is corrupted in transmission, only a single digit in the plaintext is affected and the error does not propagate to other parts of the message.

Stream Cipher

- Self-synchronizing stream cipher
  - Key stream obtained from the secret key and N previous ciphertexts
  - the receiver will automatically synchronize with the keystream generator after receiving *N* ciphertext digits, making it easier to recover if digits are dropped or added to the message stream.
  - Lost packets cause a delay of q steps before decryption resumes
  - Single-digit errors are limited in their effect, affecting only up to *N* plaintext digits.

Key Stream Generation

- RC4
  - Designed in 1987 by Ron Rivest for RSA Security
  - Trade secret until 1994
  - Uses keys with up to 2,048 bits
  - Simple algorithm
- Block cipher in counter mode (CTR)
  - Use a block cipher with block size b
  - The secret key is a pair (K,t), where K is key and t (counter) is a b-bit value
  - The key stream is the concatenation of ciphertexts

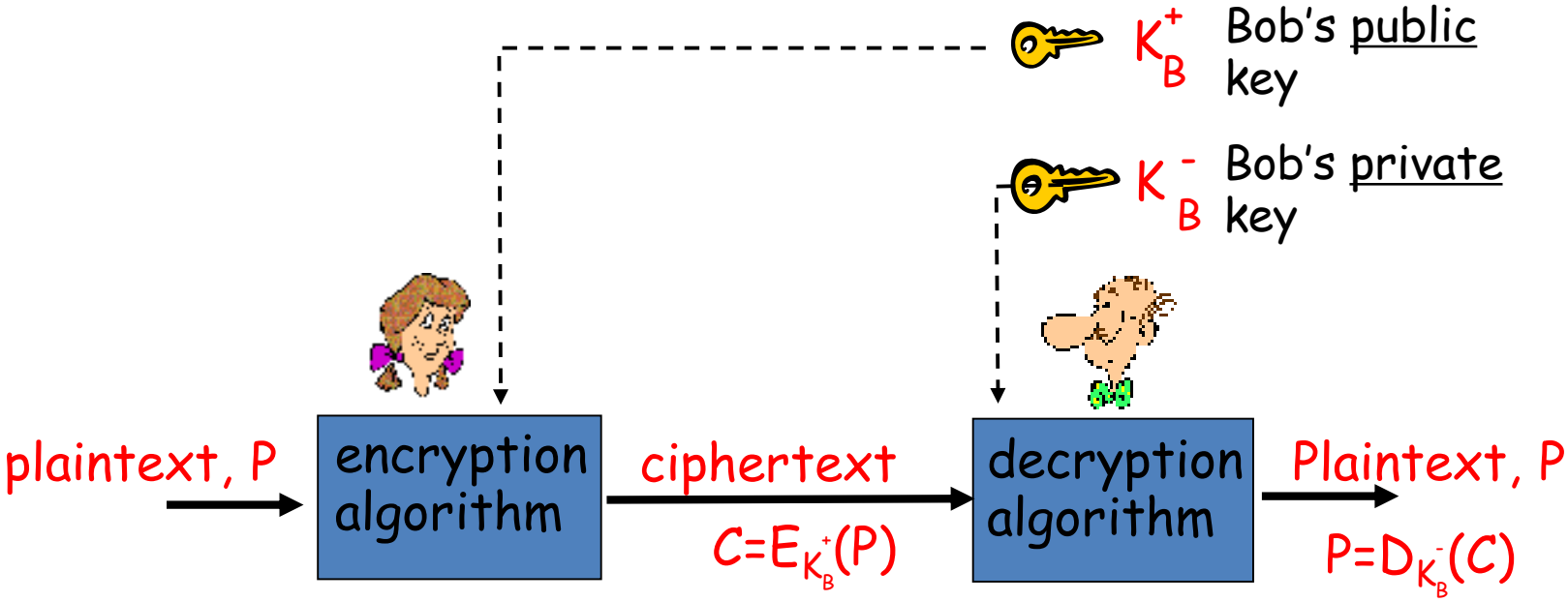
$$E_K(t),\ E_K(t+1),\ E_K(t+2),\ \ldots$$

  - Can use a shorter counter concatenated with a random value
  - Synchronous stream cipher

Attacks on Stream Ciphers

- Repetition attack
  - if key stream reused, attacker obtains XOR of two plaintexts (why?)

Public key cryptography



Facts About Numbers

- Prime number $\boldsymbol{p}$:
  - $\boldsymbol{p}$ is an integer
  - $\boldsymbol{p} \geq 2$
  - The only divisors of $\boldsymbol{p}$ are 1 and $\boldsymbol{p}$
- Examples
  - $2, 7, 19$ are primes
  - $-3, 0, 1, 6$ are not primes
- Prime decomposition of a positive integer $\boldsymbol{n}$:

$$\boldsymbol{n} = \boldsymbol{p_1}^{\boldsymbol{e}_1} \times \ldots \times \boldsymbol{p_k}^{\boldsymbol{e}_k}$$

- Example:
  - $200 = 2^3 \times 5^2$

Fundamental Theorem of Arithmetic

The prime decomposition of a positive integer is unique

Greatest Common Divisor

- The greatest common divisor (GCD) of two positive integers $\boldsymbol{a}$ and $\boldsymbol{b}$, denoted $\gcd(\boldsymbol{a}, \boldsymbol{b})$, is the largest positive integer that divides both $\boldsymbol{a}$ and $\boldsymbol{b}$
- The above definition is extended to arbitrary integers
- Examples:

$$\gcd(18, 30) = 6 \qquad \gcd(0, 20) = 20$$
$$\gcd(-21, 49) = 7$$

- Two integers a and b are said to be relatively prime if

$$\gcd(\boldsymbol{a}, \boldsymbol{b}) = 1$$

- Example:
  - Integers 15 and 28 are relatively prime

Modular Arithmetic

- Modulo operator for a positive integer n

  $$r = a \bmod n$$

  equivalent to

  $$a = r + kn$$

  and

  $$r = a - \lfloor a/n \rfloor n$$

- Example:

  $29 \bmod 13 = 3$ $\quad 13 \bmod 13 = 0$ $\quad -1 \bmod 13 = 12$

  $29 = 3 + 2\times 13$ $\quad 13 = 0 + 1\times 13$ $\quad 12 = -1 + 1\times 13$

  For a<0, we first add a large kn to a such that it becomes positive

- Modulo and GCD:

  $$\gcd(a, b) = \gcd(b, a \bmod b)$$

- Example:

  $\gcd(21, 12) = 3$ $\quad \gcd(12, 21 \bmod 12) = \gcd(12, 9) = 3$

Euclid's GCD Algorithm

- Euclid's algorithm for computing the GCD repeatedly applies the formula

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

- Example
  - $\gcd(412, 260) = 4$

```
Algorithm EuclidGCD(a, b)
    Input integers a and b
    Output gcd(a, b)

    if b = 0
        return a
    else
        return EuclidGCD(b, a mod b)
```

a	412	260	152	108	44	20	4
b	260	152	108	44	20	4	0

RSA: Choosing keys

1. Choose two large prime numbers *p*, *q*. (e.g., 1024 bits each)

2. Compute *n* = *pq*, *z* = (*p*-1)(*q*-1)

3. Choose *e* (with *e*<*n*) that has no common factors with z. (*e*, *z* are "relatively prime").

4. Choose *d* such that *ed*-1 is exactly divisible by *z*. (in other words: *ed* mod *z* = 1).

5. *Public* key is $\underbrace{(n,e)}_{K_B^+}$. *Private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

1. To encrypt bit pattern, m, compute

   $c = m^e \bmod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

   $m = c^d \bmod n$ (i.e., remainder when c^d is divided by n)

Magic happens!

$$m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$$

RSA example:

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=24$.

$e=5$ (so e, z relatively prime).

$d=29$ (so $ed-1$ exactly divisible by z).

encrypt:

letter	m	m^e	$c = m^e \bmod n$
l	12	1524832	17

decrypt:

c	c^d	$m = c^d \bmod n$	letter
17	48196857210675091509141182522307169 7	12	l

Computational extensive

RSA: Why is that $m = (m^e \bmod n)^d \bmod n$

Useful number theory result: If p,q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{E_{K^+}(E_{K^-}(P))}_{\text{use public key first, followed by private key}} = P = \underbrace{E_{K^-}(E_{K^+}(P))}_{\text{use private key first, followed by public key}}$$

Result is the same!

RSA Cryptosystem

- Setup:
  - $n = pq$, with p and q primes
  - e relatively prime to $\phi(n) = (p - 1)(q - 1)$
  - d inverse of e in $Z_{\phi(n)}$
    - $ed \bmod z = 1$
- Keys:
  - Public key: $K_E = (n, e)$
  - Private key: $K_D = d$
- Encryption:
  - Plaintext M in Z_n
  - $C = M^e \bmod n$
- Decryption:
  - $M = C^d \bmod n$

- Example
  - Setup:
    - $p = 7,\ q = 17$
    - $n = 7 \cdot 17 = 119$
    - $\phi(n) = 6 \cdot 16 = 96$
    - $e = 5$
    - $d = 77$
  - Keys:
    - public key: $(119, 5)$
    - private key: 77
  - Encryption:
    - $M = 19$
    - $C = 19^5 \bmod 119 = 66$
  - Decryption:
    - $C = 66^{77} \bmod 119 = 19$

Complete RSA Example

- Setup:
  - $p = 5$, $q = 11$
  - $n = 5 \cdot 11 = 55$
  - $\phi(n) = 4 \cdot 10 = 40$
  - $e = 3$
  - $d = 27$ ($3 \cdot 27 = 81 = 2 \cdot 40 + 1$)

- Encryption
  - $C = M^3 \bmod 55$
- Decryption
  - $M = C^{27} \bmod 55$

- Pre-compute lookup table (size of n-1, M should not be 0) **Why?**

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
C	1	8	27	9	15	51	13	17	14	10	11	23	52	49	20	26	18	2
M	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
C	39	25	21	33	12	19	5	31	48	7	24	50	36	43	22	34	30	16
M	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
C	53	37	29	35	6	3	32	44	45	41	38	42	4	40	46	28	47	54

Security

- Security of RSA based on difficulty of factoring of $n=pq$
  - Widely believed
  - Best known algorithm takes exponential time
- RSA Security factoring challenge (discontinued)
- In 1999, 512-bit challenge factored in 4 months using 35.7 CPU-years
  - 160 175-400 MHz SGI and Sun
  - 8 250 MHz SGI Origin
  - 120 300-450 MHz Pentium II
  - 4 500 MHz Digital/Compaq
- In 2005, a team of researchers factored the RSA-640 challenge number using 30 2.2GHz CPU years
- In 2004, the prize for factoring RSA-2048 was $200,000
- Current practice is 2,048-bit keys
- Estimated resources needed to factor a number within one year

Length (bits)	PCs	Memory
430	1	128MB
760	215,000	4GB
1,020	342×10^6	170GB
1,620	1.6×10^{15}	120TB

Hash Functions

- A hash function h maps a plaintext x to a fixed-length value x = h(P) called hash value or digest of P
  - Usually x is much smaller in size compared to P.
  - A collision is a pair of plaintexts P and Q that map to the same hash value, h(P) = h(Q)
  - Collisions are unavoidable
  - For efficiency, the computation of the hash function should take time proportional to the length of the input plaintext

Simplex Example of Hash Functions

- Parity bit: map a binary bit stream to ‘1’ or ‘0’
  - Hash value space is only 2.

- Repeated addition of n-byte chunks without considering carry-on bits
  - Hash value space is 2^{8n}

Cryptographic Hash Functions

- A cryptographic hash function satisfies additional properties
  - Preimage resistance (aka one-way)
    - Given a hash value x, it is hard to find a plaintext P such that h(P) = x
  - Second preimage resistance (aka weak collision resistance)
    - Given a plaintext P, it is hard to find a plaintext Q such that h(Q) = h(P)
  - Collision resistance (aka strong collision resistance)
    - It is hard to find a pair of plaintexts P and Q such that h(Q) = h(P)
- Collision resistance implies second preimage resistance
- Hash values of at least 256 bits recommended to defend against brute-force attacks

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x.
- SHA-1 is also used.
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest
- There are many hash functions, but most of them do not satisfy cryptographic hash function requirements
  - example: checksum

Message-Digest Algorithm 5 (MD5)

- Developed by Ron Rivest in 1991
- Uses 128-bit hash values
- Still widely used in legacy applications although considered insecure
- Various severe vulnerabilities discovered
- Chosen-prefix collisions attacks found by Marc Stevens, Arjen Lenstra and Benne de Weger
  - Start with two arbitrary plaintexts P and Q
  - One can compute suffixes S1 and S2 such that P||S1 and Q||S2 collide under MD5 by making 250 hash evaluations
  - Using this approach, a pair of different executable files or PDF documents with the same MD5 hash can be computed