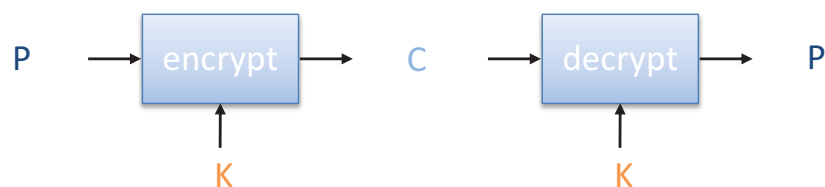


Cryptography

Symmetric Cryptosystem

- **Scenario**
 - Alice wants to send a message (plaintext P) to Bob.
 - The communication channel is insecure and can be eavesdropped
 - If Alice and Bob 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?



Basics

- Notation
 - Secret key K
 - Encryption function $E_K(P)$
 - Decryption function $D_K(C)$
 - Plaintext length typically the same as ciphertext length
 - Encryption and decryption are **permutation functions (bijections)** on the set of all n -bit arrays
- Efficiency
 - functions E_K and D_K should have efficient algorithms
- Consistency
 - Decrypting the ciphertext yields the plaintext
 - $D_K(E_K(P)) = P$

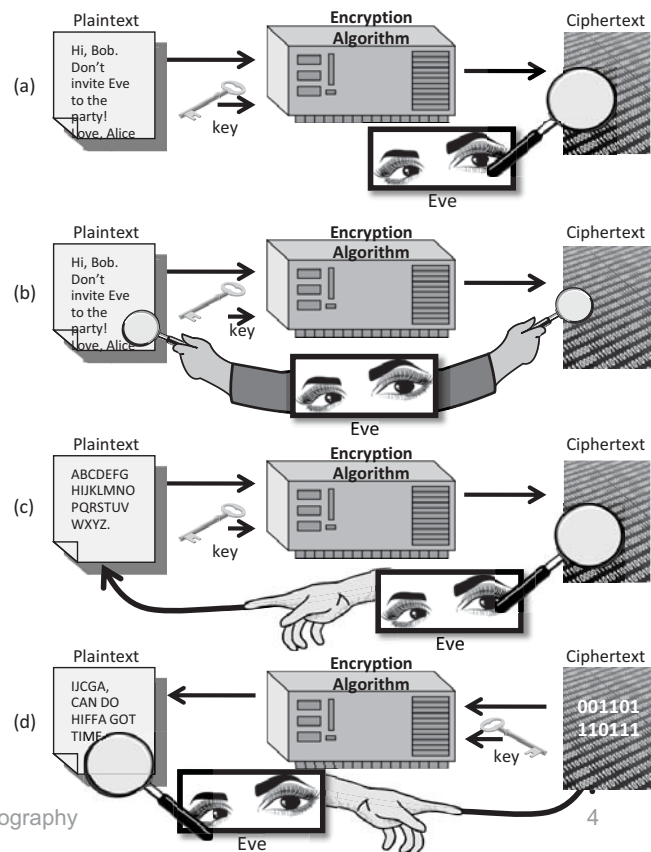
11/22/2010

Cryptography

3

Attacks

- Attacker may have
 - a) collection of ciphertexts (**ciphertext only attack**)
 - b) collection of plaintext/ciphertext pairs (**known plaintext attack**)
 - c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker (**chosen plaintext attack**)
 - d) collection of plaintext/ciphertext pairs for ciphertexts selected by the attacker (**chosen ciphertext attack**)



11/22/2010

Cryptography

4

Brute-Force Attack

- Try all possible keys K and determine if $D_K(C)$ is a likely plaintext
 - Requires some knowledge of the structure of the plaintext (e.g., PDF file or email message)
- Key should be a sufficiently long random value to make exhaustive search attacks unfeasible



11/22/2010

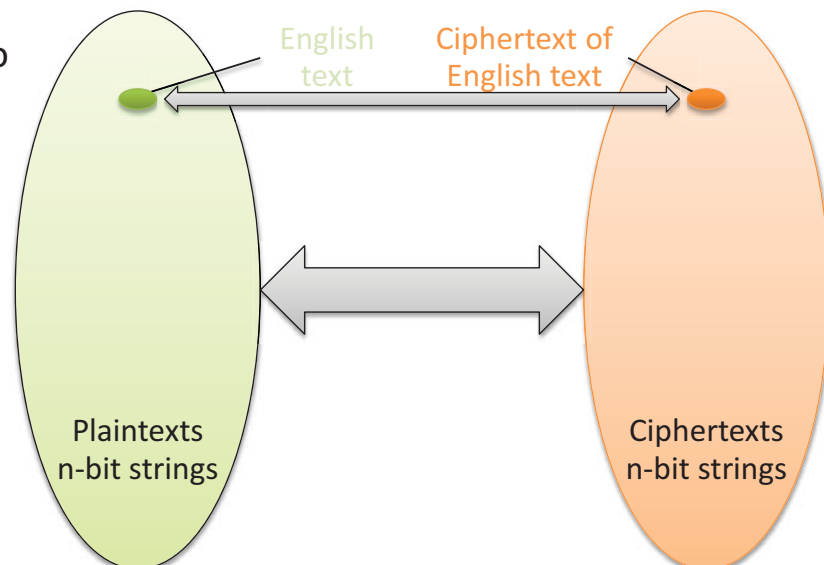
5

Image by Michael Cote from http://commons.wikimedia.org/wiki/File:Bingo_cards.jpg

Encrypting English Text

- English text typically represented with 8-bit ASCII encoding
- A message with t characters corresponds to an n -bit array, with $n = 8t$

- Redundancy due to repeated words and patterns
 - E.g., “th”, “ing”
- English plaintexts are a very small subset of all n -bit arrays



11/22/2010

Cryptography

6

Entropy of Natural Language

- Information content (**entropy**) of English: 1.25 bits per character
- t-character arrays that are English text:

$$(2^{1.25})^t = 2^{1.25 t}$$

- n-bit arrays that are English text:

$$2^{1.25 n/8} \approx 2^{0.16 n}$$

- For a natural language, constant $\alpha < 1$ such that there are $2^{\alpha n}$ messages among all n-bit arrays
- Fraction (probability) of valid messages

$$2^{\alpha n} / 2^n = 1 / 2^{(1-\alpha)n}$$

- Brute-force decryption
 - Try all possible 2^k decryption keys
 - Stop when valid plaintext recognized

- Given a ciphertext, there are 2^k possible plaintexts
- Expected number of valid plaintexts

$$2^k / 2^{(1-\alpha)n}$$

- Expected unique valid plaintext, (no spurious keys) achieved at **unicity distance**

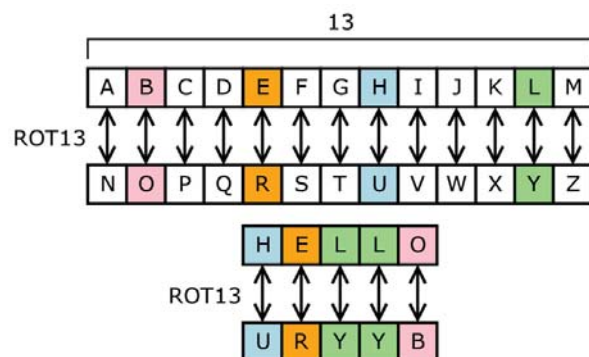
$$n = k / (1-\alpha)$$

- For English text and 256-bit keys, unicity distance is 304 bits

Substitution Ciphers

- Each letter is uniquely replaced by another.
- There are $26!$ possible substitution ciphers.
- There are more than 4.03×10^{26} such ciphers.

- One popular substitution “cipher” for some Internet posts is ROT13.



Frequency Analysis

- Letters in a natural language, like English, are not uniformly distributed.
- Knowledge of letter frequencies, including pairs and triples can be used in cryptologic attacks against substitution ciphers.

a: 8.05%	b: 1.67%	c: 2.23%	d: 5.10%
e: 12.22%	f: 2.14%	g: 2.30%	h: 6.62%
i: 6.28%	j: 0.19%	k: 0.95%	l: 4.08%
m: 2.33%	n: 6.95%	o: 7.63%	p: 1.66%
q: 0.06%	r: 5.29%	s: 6.02%	t: 9.67%
u: 2.92%	v: 0.82%	w: 2.60%	x: 0.11%
y: 2.04%	z: 0.06%		

Letter frequencies in the book *The Adventures of Tom Sawyer*, by

11/22/2010 Twain.

9

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)

Figure 8.3: A 4-bit S-box (a) An S-box in binary. (b) The same S-box in decimal.

One-Time Pads

- There is one type of substitution cipher that is absolutely unbreakable.
 - The **one-time pad** was invented in 1917 by Joseph Mauborgne and Gilbert Vernam
 - We use a block of shift keys, (k_1, k_2, \dots, k_n) , to encrypt a plaintext, M , of length n , with each shift key being chosen uniformly at random.
- Since each shift is random, every ciphertext is equally likely for any plaintext.

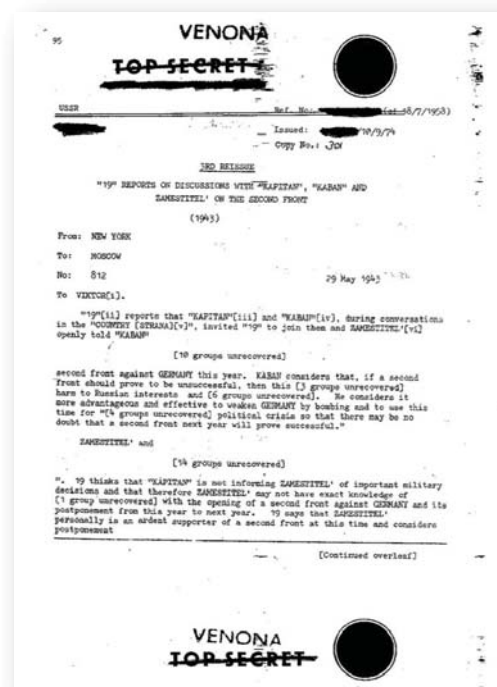
11/22/2010

Cryptography

11

Weaknesses of the One-Time Pad

- In spite of their perfect security, one-time pads have some weaknesses
- The key has to be as long as the plaintext
- Keys can never be reused
 - Repeated use of one-time pads allowed the U.S. to break some of the communications of Soviet spies during the Cold War.



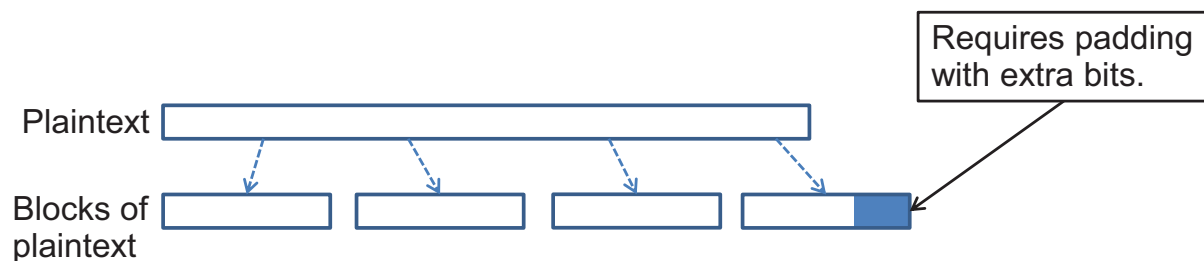
11/22/2010

Cryptography

12

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**, $P[0], \dots, P[m-1]$, where $n \leq bm < n + b$
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks.



11/22/2010

Cryptography

13

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

11/22/2010

Cryptography

14

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 K_A , K_B , and K_C
 - Effective key length is 168 bits, making exhaustive search attacks unfeasible
 - $C = E_{K_C}(D_{K_B}(E_{K_A}(P)))$; $P = D_{K_A}(E_{K_B}(D_{K_C}(C)))$
 - Equivalent to DES when $K_A=K_B=K_C$ (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

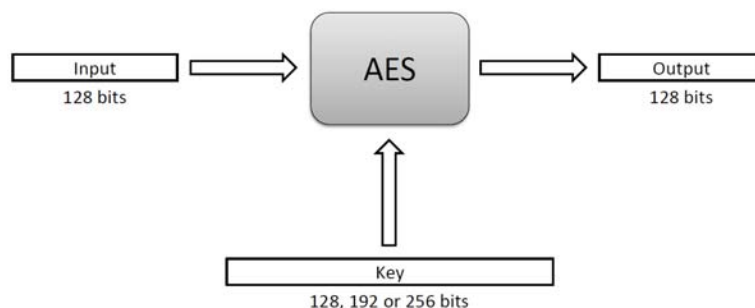
11/22/2010

Cryptography

15

The Advanced Encryption Standard (AES)

- In 1997, the U.S. National Institute for Standards and Technology (NIST) put out a public call for a replacement to DES.
- It narrowed down the list of submissions to five finalists, and ultimately chose an algorithm that is now known as 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.

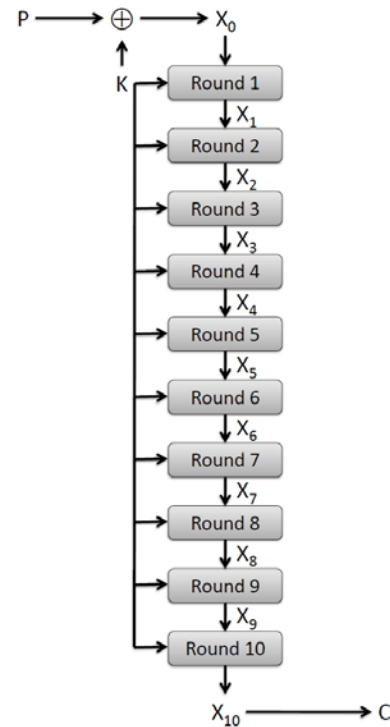


11/22/2010

16

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 ($i = 1, \dots, 10$) receives state X_{i-1} as input and produces state X_i .
- The ciphertext C is the output of the final round: $C = X_{10}$.

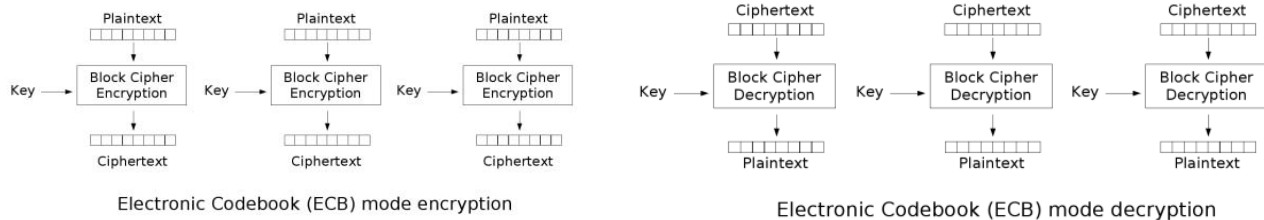


AES Rounds

- 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 $M[i] = D_K(C[i])$



11/22/2010

Cryptography

19

Public domain images from http://en.wikipedia.org/wiki/File:Ecb_encryption.png and http://en.wikipedia.org/wiki/File:Ecb_decryption.png

Problems with ECB

- ECB mode works well with random strings (e.g., keys and initialization vectors) and strings that fit in one block
- Documents and images are not suitable for ECB encryption since patterns in the plaintext are repeated in the ciphertext
- Example of image encrypted in ECB :

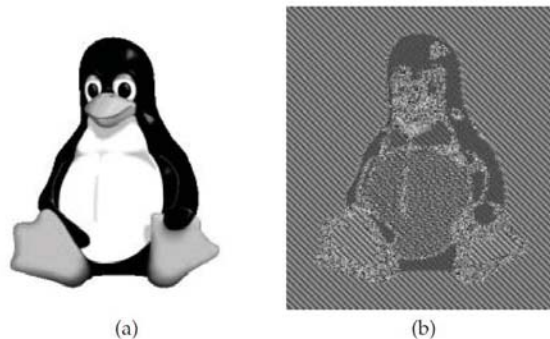


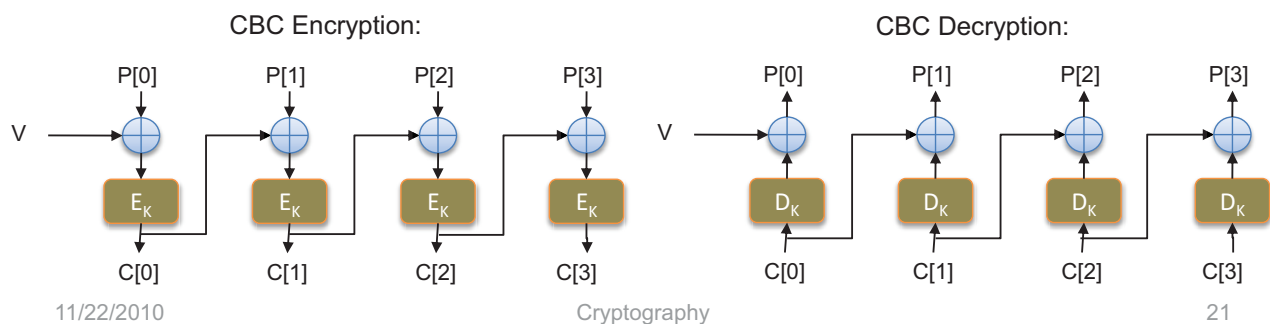
Figure 8.6: How ECB mode can leave identifiable patterns in a sequence of blocks: (a) An image of Tux the penguin, the Linux mascot. (b) An encryption of the Tux image using ECB mode. (The image in (a) is by Larry Ewing, lewing@isc.tamu.edu, using The Gimp; the image in (b) is by Dr. Juzam. Both are used with permission via attribution.)

11/22/2010

20

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)

Java AES Encryption Example

- Source

<http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html>

- Generate an AES key

```
KeyGenerator keygen = KeyGenerator.getInstance("AES");  
SecretKey aesKey = keygen.generateKey();
```

- Create a cipher object for AES in ECB mode and PKCS5 padding

```
Cipher aesCipher;  
aesCipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

- Encrypt

```
aesCipher.init(Cipher.ENCRYPT_MODE, aesKey);  
byte[] plaintext = "My secret message".getBytes();  
byte[] ciphertext = aesCipher.doFinal(plaintext);
```

- Decrypt

```
aesCipher.init(Cipher.DECRYPT_MODE, aesKey);  
byte[] plaintext1 = aesCipher.doFinal(ciphertext);
```

Stream Cipher

- Key stream
 - Pseudo-random sequence of bits $S = S[0], S[1], S[2], \dots$
 - 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
 - Works for unreliable channels if plaintext has packets with sequence numbers
- Self-synchronizing stream cipher
 - Key stream obtained from the secret key and q previous ciphertexts
 - Lost packets cause a delay of q steps before decryption resumes

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 a key and t (counter) is a b -bit value
 - The key stream is the concatenation of ciphertexts
$$E_K(t), E_K(t + 1), E_K(t + 2), \dots$$
 - Can use a shorter counter concatenated with a random value
 - Synchronous stream cipher

11/22/2010

Cryptography

25

Attacks on Stream Ciphers

- Repetition attack
 - if key stream reused, attacker obtains XOR of two plaintexts
- Insertion attack [Bayer Metzger, TODS 1976]
 - retransmission of the plaintext with
 - a chosen byte inserted by attacker
 - using the same key stream
 - e.g., email message resent with new message number

Original	P	P[i]	P[i+1]	P[i+2]	P[i+3]
	S	S[i]	S[i+1]	S[i+2]	S[i+3]
	C	C[i]	C[i+1]	C[i+2]	C[i+3]
Retransmission	P	P[i]	X	P[i+1]	P[i+2]
	S	S[i]	S[i+1]	S[i+2]	S[i+3]
	C	C[i]	C'[i+1]	C'[i+2]	C'[i+3]

11/22/2010

Cryptography

26

Public Key Encryption

Facts About Numbers

- Prime number p :
 - p is an integer
 - $p \geq 2$
 - The only divisors of p are 1 and p
- Examples
 - 2, 7, 19 are primes
 - -3, 0, 1, 6 are not primes
- Prime decomposition of a positive integer n :
$$n = p_1^{e_1} \times \dots \times p_k^{e_k}$$
- 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 a and b , denoted $\gcd(a, b)$, is the largest positive integer that divides both a and b

- The above definition is extended to arbitrary integers

- Examples:

$$\gcd(18, 30) = 6$$

$$\gcd(0, 20) = 20$$

$$\gcd(-21, 49) = 7$$

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

$$\gcd(a, b) = 1$$

- 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:

$$29 \bmod 13 = 3$$

$$13 \bmod 13 = 0$$

$$-1 \bmod 13 = 12$$

$$29 = 3 + 2 \times 13$$

$$13 = 0 + 1 \times 13$$

$$12 = -1 + 1 \times 13$$

- Modulo and GCD:

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

- Example:

$$\gcd(21, 12) = 3$$

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

Euclid's GCD Algorithm

- Euclid's algorithm for computing the GCD repeatedly applies the formula
 $\text{gcd}(a, b) = \text{gcd}(b, a \bmod b)$
- Example
 $-\text{gcd}(412, 260) = 4$

Algorithm *EuclidGCD*(a, b)
Input integers a and b
Output $\text{gcd}(a, b)$

if $b = 0$
 return a
else
 return *EuclidGCD*($b, a \bmod b$)

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

Analysis

- Let a_i and b_i be the arguments of the i -th recursive call of algorithm *EuclidGCD*
- We have

$$a_{i+2} = b_{i+1} = a_i \bmod a_{i+1} < a_{i+1}$$
- Sequence a_1, a_2, \dots, a_n decreases exponentially, namely

$$a_{i+2} \leq \frac{1}{2} a_i \text{ for } i > 1$$
 - Case 1 $a_{i+1} \leq \frac{1}{2} a_i$ $a_{i+2} < a_{i+1} \leq \frac{1}{2} a_i$
 - Case 2 $a_{i+1} > \frac{1}{2} a_i$ $a_{i+2} = a_i \bmod a_{i+1} = a_i - a_{i+1} \leq \frac{1}{2} a_i$
- Thus, the maximum number of recursive calls of algorithm *EuclidGCD*(a, b) is

$$1 + 2 \log \max(a, b)$$
- Algorithm *EuclidGCD*(a, b) executes $O(\log \max(a, b))$ arithmetic operations
- The running time can also be expressed as $O(\log \min(a, b))$

Multiplicative Inverses (1)

- The **residues** modulo a positive integer n are the set

$$\mathbb{Z}_n = \{0, 1, 2, \dots, (n - 1)\}$$

- Let x and y be two elements of \mathbb{Z}_n such that

$$xy \bmod n = 1$$

We say that y is the **multiplicative inverse** of x in \mathbb{Z}_n and we write $y = x^{-1}$

- Example:
 - Multiplicative inverses of the residues modulo 11

x	0	1	2	3	4	5	6	7	8	9	10
x^{-1}		1	6	4	3	9	2	8	7	5	10

Multiplicative Inverses (2)

Theorem

An element x of \mathbb{Z}_n has a multiplicative inverse if and only if x and n are relatively prime

- Example
 - The elements of \mathbb{Z}_{10} with a multiplicative inverse are 1, 3, 7, 9

Corollary

If p is prime, every nonzero residue in \mathbb{Z}_p has a multiplicative inverse

Theorem

A variation of Euclid's GCD algorithm computes the multiplicative inverse of an element x of \mathbb{Z}_n or determines that it does not exist

x	0	1	2	3	4	5	6	7	8	9
x^{-1}		1		7				3		9

Example: Measuring Lengths

- Consider a stick of length a and a stick of length b such that a and b are relatively prime
- Given two integers i and j , we can measure length
$$n = ia + jb$$
- We show that any integer n can be written as $n = ia + jb$ for some integers i and j
 - Let s be the inverse of a in \mathbb{Z}_b . We have $sa \bmod b = 1$
 - There exists integer t such that $sa + tb = 1$
 - Pick $i = ns$ and $j = nt$
- Thus, given two sticks of relatively prime integer lengths, we can measure any integer length
- Example, measure length 2 with sticks of length 3 and 7



11/22/2010

Cryptography

35

Example: Double Hashing

- Consider a hash table whose size n is a prime
- In open addressing with double hashing, an operation on key x probes the following locations modulo n
$$i, i + d, i + 2d, i + 3d, \dots, i + (n - 1)d$$
where $i = h_1(x)$ and $d = h_2(x)$
- We show that each table location is probed by this sequence once
 - Suppose $(i + jd) \bmod n = (i + kd) \bmod n$ for some integers j and k in the range $[0, n - 1]$
 - We have $(j - k)d \bmod n = 0$
 - Since n is prime, we have that n and d are relatively prime
 - Thus, d has an inverse d^{-1} in \mathbb{Z}_n
 - Multiplying each side by d^{-1} , we obtain $(j - k) \bmod n = 0$
 - We conclude that $j = k$

11/22/2010

Cryptography

36

Powers

- Let p be a prime
- The sequences of successive powers of the elements of \mathbb{Z}_p exhibit repeating subsequences
- The sizes of the repeating subsequences and the number of their repetitions are the divisors of $p - 1$
- Example ($p = 7$)

x	x^2	x^3	x^4	x^5	x^6
1	1	1	1	1	1
2	4	1	2	4	1
3	2	6	4	5	1
4	2	1	4	2	1
5	4	6	2	3	1
6	1	6	1	6	1

11/22/2010

Cryptography

37

Fermat's Little Theorem

Theorem

Let p be a prime. For each nonzero residue x of \mathbb{Z}_p , we have $x^{p-1} \bmod p = 1$

- Example ($p = 5$):

$$1^4 \bmod 5 = 1$$

$$2^4 \bmod 5 = 16 \bmod 5 = 1$$

$$3^4 \bmod 5 = 81 \bmod 5 = 1$$

$$4^4 \bmod 5 = 256 \bmod 5 = 1$$

Corollary

Let p be a prime. For each nonzero residue x of \mathbb{Z}_p , the multiplicative inverse of x is $x^{p-2} \bmod p$

Proof

$$x(x^{p-2} \bmod p) \bmod p = xx^{p-2} \bmod p = x^{p-1} \bmod p = 1$$

11/22/2010

Cryptography

38

Euler's Theorem

- The multiplicative group for Z_n , denoted with Z_n^* , is the subset of elements of Z_n relatively prime with n
- The totient function of n , denoted with $\phi(n)$, is the size of Z_n^*
- Example
$$Z_{10}^* = \{ 1, 3, 7, 9 \} \quad \phi(10) = 4$$
- If p is prime, we have
$$Z_p^* = \{ 1, 2, \dots, (p - 1) \} \quad \phi(p) = p - 1$$

Euler's Theorem

For each element x of Z_n^* , we have $x^{\phi(n)} \bmod n = 1$

- Example ($n = 10$)
$$3^{\phi(10)} \bmod 10 = 3^4 \bmod 10 = 81 \bmod 10 = 1$$
$$7^{\phi(10)} \bmod 10 = 7^4 \bmod 10 = 2401 \bmod 10 = 1$$
$$9^{\phi(10)} \bmod 10 = 9^4 \bmod 10 = 6561 \bmod 10 = 1$$

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 $Z_{\phi(n)}$
- 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:
 - ♦ $M = 66^{77} \bmod 119 = 19$

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

- $C = M^3 \pmod{55}$

- Decryption

- $M = C^{27} \pmod{55}$

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

Correctness

- We show the correctness of the RSA cryptosystem for the case when the plaintext M does not divide n

- Namely, we show that

$$(M^e)^d \bmod n = M$$

- Since $ed \bmod \phi(n) = 1$, there is an integer k such that

$$ed = k\phi(n) + 1$$

- Since M does not divide n , by Euler's theorem we have

$$M^{\phi(n)} \bmod n = 1$$

- Thus, we obtain

$$(M^e)^d \bmod n =$$

$$M^{ed} \bmod n =$$

$$M^{k\phi(n) + 1} \bmod n =$$

$$MM^{k\phi(n)} \bmod n =$$

$$M (M^{\phi(n)})^k \bmod n =$$

$$M (M^{\phi(n)} \bmod n)^k \bmod n =$$

$$M (1)^k \bmod n =$$

$$M \bmod n =$$

$$M$$

- Proof of correctness can be extended to the case when the plaintext M divides n

Algorithmic Issues

- The implementation of the RSA cryptosystem requires various algorithms

- Overall

- Representation of integers of arbitrarily large size and arithmetic operations on them

- Encryption

- Modular power

- Decryption

- Modular power

- Setup

- Generation of **random numbers** with a given number of bits (to generate candidates p and q)

- Primality testing** (to check that candidates p and q are prime)

- Computation of the **GCD** (to verify that e and $\phi(n)$ are relatively prime)

- Computation of the **multiplicative inverse** (to compute d from e)

Modular Power

- The repeated squaring algorithm speeds up the computation of a modular power $a^p \bmod n$
- Write the exponent p in binary

$$p = p_{b-1}p_{b-2} \cdots p_1p_0$$
- Start with

$$Q_1 = a^{p_{b-1}} \bmod n$$
- Repeatedly compute

$$Q_i = ((Q_{i-1})^2 \bmod n) a^{p_{b-i}} \bmod n$$
- We obtain

$$Q_b = a^p \bmod n$$
- The repeated squaring algorithm performs $O(\log p)$ arithmetic operations

- Example

$$\begin{aligned}
 &-3^{18} \bmod 19 \quad (18 = 10010) \\
 &-Q_1 = 3^1 \bmod 19 = 3 \\
 &-Q_2 = (3^2 \bmod 19) 3^0 \bmod 19 = 9 \\
 &-Q_3 = (9^2 \bmod 19) 3^0 \bmod 19 = 81 \bmod 19 = 5 \\
 &-Q_4 = (5^2 \bmod 19) 3^1 \bmod 19 = (25 \bmod 19) 3 \bmod 19 = 18 \bmod 19 = 18 \\
 &-Q_5 = (18^2 \bmod 19) 3^0 \bmod 19 = (324 \bmod 19) \bmod 19 = 17 \cdot 19 + 1 \bmod 19 = 1
 \end{aligned}$$

p_{5-i}	1	0	0	1	0
$2^{p_{5-i}}$	3	1	1	3	1
Q_i	3	9	5	18	1

11/22/2010

Cryptography

45

Modular Inverse

Theorem

Given positive integers a and b , let d be the smallest positive integer such that

$$d = ia + jb$$

for some integers i and j .

We have

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

- Example

- $a = 21$
- $b = 15$
- $d = 3$
- $i = 3, j = -4$
- $3 = 3 \cdot 21 + (-4) \cdot 15 = 63 - 60 = 3$

- Given positive integers a and b , the extended Euclid's algorithm computes a triplet (d, i, j) such that
 - $d = \gcd(a, b)$
 - $d = ia + jb$
 - To test the existence of and compute the inverse of $x \in \mathbb{Z}_n$, we execute the extended Euclid's algorithm on the input pair (x, n)
 - Let (d, i, j) be the triplet returned
 - $d = ix + jn$
- Case 1: $d = 1$
 i is the inverse of x in \mathbb{Z}_n
- Case 2: $d > 1$
 x has no inverse in \mathbb{Z}_n

11/22/2010

Cryptography

46

Pseudoprimality Testing

- The number of primes less than or equal to n is about $n / \ln n$
- Thus, we expect to find a prime among $O(b)$ randomly generated numbers with b bits each
- Testing whether a number is prime (**primality testing**) is a difficult problem, though polynomial-time algorithms exist
- An integer $n \geq 2$ is said to be a base- x **pseudoprime** if
 - $x^{n-1} \bmod n = 1$ (Fermat's little theorem)
- Composite base- x pseudoprimes are rare:
 - A random 100-bit integer is a composite base-2 pseudoprime with probability less than 10^{-13}
 - The smallest composite base-2 pseudoprime is 341
- Base- x pseudoprimality testing for an integer n :
 - Check whether $x^{n-1} \bmod n = 1$
 - Can be performed efficiently with the repeated squaring algorithm

11/22/2010

Cryptography

47

Randomized Primality Testing

- Compositeness witness function **witness(x, n)** with error probability q for a random variable x
 - Case 1: n is prime
 - witness(x, n) = false** always
 - Case 2: n is composite
 - witness(x, n) = true** in most cases, **false** with small probability $q < 1$
- Algorithm **RandPrimeTest** tests whether n is prime by repeatedly evaluating **witness(x, n)**
- A variation of base- x pseudoprimality provides a suitable compositeness witness function for randomized primality testing (Rabin-Miller algorithm)

Algorithm **RandPrimeTest(n, k)**

Input integer n , confidence parameter k and composite witness function **witness(x, n)** with error probability q

Output an indication of whether n is composite or prime with probability 2^{-k}

$t \leftarrow k / \log_2(1/q)$

for $i \leftarrow 1$ **to** t

$x \leftarrow \text{random}()$

if **witness(x, n) = true**

return " n is composite"

return " n is prime"

11/22/2010

Cryptography

48

Cryptographic Hash Functions

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
 - 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
- Hash table
 - Search data structure based on storing items in locations associated with their hash value
 - Chaining or open addressing deal with collisions
 - Domain of hash values proportional to the expected number of items to be stored
 - The hash function should spread plaintexts uniformly over the possible hash values to achieve constant expected search time

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
- A **random oracle** is a theoretical model for a cryptographic hash function from a finite input domain \mathcal{P} to a finite output domain \mathcal{X}
 - Pick randomly and uniformly a function $h: \mathcal{P} \rightarrow \mathcal{X}$ over all possible such functions
 - Provide only oracle access to h : one can obtain hash values for given plaintexts, but no other information about the function h itself

11/22/2010

Cryptography

51

Birthday Attack

- The brute-force **birthday attack** aims at finding a collision for a hash function h
 - Randomly generate a sequence of plaintexts X_1, X_2, X_3, \dots
 - For each X_i compute $y_i = h(X_i)$ and test whether $y_i = y_j$ for some $j < i$
 - Stop as soon as a collision has been found
- If there are m possible hash values, the probability that the i -th plaintext does not collide with any of the previous $i - 1$ plaintexts is $1 - (i - 1)/m$
- The probability F_k that the attack fails (no collisions) after k plaintexts is
$$F_k = (1 - 1/m) (1 - 2/m) (1 - 3/m) \dots (1 - (k - 1)/m)$$
- Using the standard approximation $1 - x \approx e^{-x}$
$$F_k \approx e^{-(1/m + 2/m + 3/m + \dots + (k-1)/m)} = e^{-k(k-1)/2m}$$
- The attack succeeds/fails with probability $1/2$ when $F_k = 1/2$, that is,
$$e^{-k(k-1)/2m} = 1/2$$
$$k \approx 1.17 m^{1/2}$$
- We conclude that a hash function with b -bit values provides about $b/2$ bits of security

11/22/2010

Cryptography

52

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

Secure Hash Algorithm (SHA)

- Developed by NSA and approved as a federal standard by NIST
- SHA-0 and SHA-1 (1993)
 - 160-bits
 - Considered insecure
 - Still found in legacy applications
 - Vulnerabilities less severe than those of MD5
- SHA-2 family (2002)
 - 256 bits (SHA-256) or 512 bits (SHA-512)
 - Still considered secure despite published attack techniques
- Public competition for SHA-3 announced in 2007

Iterated Hash Function

- A **compression function** works on input values of fixed length
- An **iterated hash function** extends a compression function to inputs of arbitrary length
 - padding, initialization vector, and chain of compression functions
 - inherits collision resistance of compression function
- MD5 and SHA are iterated hash functions

