Cryptography

The science and study of secret writings

Cipher* – Is a secret method of writing that transforms *plaintext* into *ciphertext

The transformation is determined by a *key*

Cryptographic systems

- **One key**
- **Two key**
- **Public key**
- **Digital signatures**

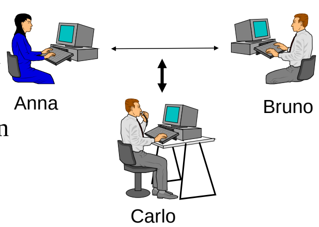
Cryptography

- Comes in two flavors: Symmetric and Asymmetric
- Best for protection of "online" communications
- Good for archival data
- So-so for electronic mail
- Not good for active databases

Communication Security

Secure communication should provide:

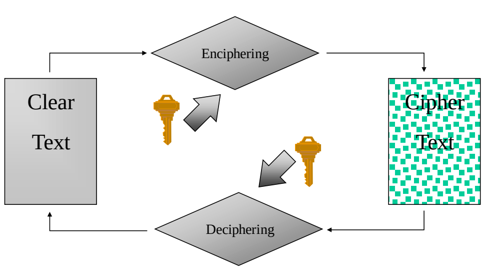
- Privacy
- Authentication
- Integrity
- Nonrepudiation



Terminology

- To lock (encipher): transforms into unintelligible form based on independent data element called a *key*
- To unlock (decipher): transforms back into intelligible form, again using a key
- Locked data is called *ciphertext* or *black*
- Unlocked data is called *plaintext*, *cleartext*, or *red*
- Keys are themselves data and can be locked and unlocked

Cryptography



General Observations

- Cryptography never solves a problem; it transforms a security problem into a key management problem
- It takes a secret to keep a secret

Cryptographic System (Cryptosystem)

- A plaintext message space M
- A ciphertext message space C
- A key space K
- A family of enciphering transformations

  Ek: M → C
- A family of deciphering transformations

  Dk: C → M

Crypto Systems Should Guarantee Both

- **Secrecy**
- **Authenticity**

Secrecy requirements

1. Should be computationally infeasible to systematically determine Dk from c, even if corresponding m is known
2. Should be computationally infeasible to determine m from intercepted c

Crypto Systems Should Guarantee Both

- **Secrecy**
- **Authenticity**

Authenticity requirements

1. Should be computationally infeasible to systematically determine Ek from c, even if corresponding m is known
2. Should be computationally infeasible to find c' such that Dk(c') is valid plaintext in the set M

Desirable Properties of Crypto Systems

- Enciphering and deciphering must be efficient for all keys
- System must be easy to use
- The security of the system should depend on the secrecy of the keys and not on the secrecy of the algorithms E or D

Cryptanalysis

- Cryptanalysis attempts to discover the key or the plaintext of an encrypted message
  - Assume analyst knows the algorithm but not the key
- Types of attack:
  - Ciphertext only
    - Given: C1 = Ek(M1), C2 = Ek(M2), ..., Ci = Ek(Mi)
    - Obtain: either M1,M2, ..., Mi or k
  - Known plaintext
    - Given: M1, C1 = Ek(M1), M2, C2 = Ek(M2), ..., Mi, Ci = Ek(Mi)
    - Obtain: either k or an algorithm to obtain Mi+1, from Ci+1 = Ek(Mi+1)

Cryptanalysis (continued)

- Types of attack (continued)
  - Chosen plaintext
    - Given: M1, C1 = Ek(M1), M2, C2 = Ek(M2), ..., Mi , Ci = Ek(Mi) where the attacker chooses M1, M2 , ..., Mi
    - Obtain: either k or an algorithm to obtain Mi+1, from Ci+1 = Ek(Mi+1)

Basis for Attacks

- Mathematical attacks
  - Based on analysis of underlying mathematics
- Statistical attacks
  - Make assumptions about the distribution of letters, pairs of letters (digrams), triplets of letters (trigrams), etc.
    - Called models of the language
  - Examine ciphertext, correlate properties with the assumptions.

Transposition Cipher

Rearranges bits or characters in the data

- Simple transposition
- Rail-fence cipher
- Columnar transposition

Simple Transposition

- Ciphers simply break message into blocks and permute each block using some scheme
- Eg. Blocks of five with key (25413)
  - Consider
    CMPS IS FUN FOR ALL

CMPS_ IS FU N FOR _ALL_

becomes

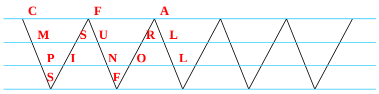
M SCP SUFI_ _RONF A L L

Rail Fence

- Transposition depends on a figure
- In this case the figure is a rail fence (or picket fence)

- Figure could be a scene, such as a landscape or city skyline

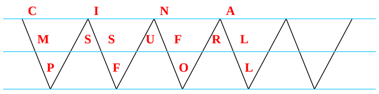
Rail Fence



If key is 2-4-3-1

MSURLSFPINOLCFA

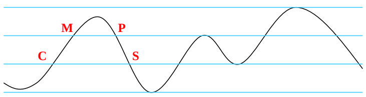
Rail Fence



If key is 1-2-3

CINAMSSUFRLPFOL

Mountain Scene



Columnar Transposition

- Uses a two dimensional array
- Text is placed in rows
- Columns are transposed
- Columns are read out as ciphered text

- Key is the transposition of the columns
  - e.g., for 4x4 matrix key could be 2-4-3-1

Columnar Transposition

Example (4x4 matrix and key = 2-4-3-1)

```
CMPS
ISFU
NFOR
ALLb
```

becomes

```
MSFLSURbPFOLCINA
```

What about (key = 1-2-3-4)?

Crypto Analysis

- Can detect transposition cipher by checking the character frequencies against the norm

a	0.080	h	0.060	n	0.070	t	0.090
b	0.015	i	0.065	o	0.080	u	0.030
c	0.030	j	0.005	p	0.020	v	0.010
d	0.040	k	0.005	q	0.002	w	0.015
e	0.130	l	0.035	r	0.065	x	0.005
f	0.020	m	0.030	s	0.060	y	0.020
g	0.015					z	0.002

Crypto Analysis

- Brute force by trying possible permutations and looking for readable text in the result
- Anagramming
  - If 1-gram frequencies match English frequencies, but other *n*-gram frequencies do not, probably transposition
  - Rearrange letters to form *n*-grams with highest frequencies

Substitution Ciphers

- Simple substitution
- Polyalphabetic
- Running key
- Vernam

Alphabet

0 – A	7 – H	14 – O	21 – V
1 – B	8 – I	15 – P	22 – W
2 – C	9 – J	16 – Q	23 – X
3 – D	10 – K	17 – R	24 – Y
4 – E	11 – L	18 – S	25 – Z
5 – F	12 – M	19 – T	
6 – G	13 – N	20 – U	

Simple Substitution

- Caesar cipher is most common example of simple substitution
  - Julius used shift of 4
  - Augustus used key of 3
- (letter value + key) mod 26
- Example (key = 3)

  CMPS IS FUN FOR ALL

  becomes

  FPSV LV IXQ IRU DOO

Attacking the Cipher

- Exhaustive search
  - If the key space is small enough, try all possible keys until you find the right one
  - Caesar cipher has 26 possible keys
- Statistical analysis
  - Compare to 1-gram model of English

Statistical Attack

- Compute frequency of each character in the ciphertext:

D .067	F .067	I .133	L .067
O .133	P .067	Q .067	R .067
S .067	U .067	V .133	X .067

- Apply 1-gram model of English
  - Frequency of characters (1-grams) in English is on next slide

Character Frequencies

a	0.080	h	0.060	n	0.070	t	0.090
b	0.015	i	0.065	o	0.080	u	0.030
c	0.030	j	0.005	p	0.020	v	0.010
d	0.040	k	0.005	q	0.002	w	0.015
e	0.130	l	0.035	r	0.065	x	0.005
f	0.020	m	0.030	s	0.060	y	0.020
g	0.015					z	0.002

Statistical Analysis

- $f(c)$ frequency of character c in ciphertext
- $\varphi(i)$ correlation of frequency of letters in ciphertext with corresponding letters in English, assuming key is i
  - $\varphi(i) = \Sigma_{0 \le c \le 25} f(c)p(c - i)$

  $p(x)$ is frequency of character x in English

Example Analysis from Text

- Caesar cipher
  - Plaintext is `HELLO WORLD`
  - Key is 3
  - Ciphertext is `KHOOR ZRUOG`

- Frequency of each letter in ciphertext:

G	0.1	H	0.1	K	0.1	O	0.3
R	0.2	U	0.1	Z	0.1		

Statistical Analysis

$\varphi(i)$ correlation of frequency of letters in ciphertext with corresponding letters in English, assuming key is i

- $\varphi(i) = \Sigma_{0 \le c \le 25} f(c)p(c - i)$ so here,
  $\varphi(i) = 0.1p(6 - i) + 0.1p(7 - i) + 0.1p(10 - i) + 0.3p(14 - i) + 0.2p(17 - i) + 0.1p(20 - i) + 0.1p(25 - i)$
  - $f(x)$ is frequency of character c in ciphertext
  - $p(x)$ is frequency of character x in English

Correlation: $\varphi(i)$ for $0 \le i \le 25$

i	$\varphi(i)$	*i*	$\varphi(i)$	*i*	$\varphi(i)$	*i*	$\varphi(i)$
0	0.0482	7	0.0442	13	0.0520	19	0.0315
1	0.0364	8	0.0202	14	0.0535	20	0.0302
2	0.0410	9	0.0267	15	0.0226	21	0.0517
3	0.0575	10	0.0635	16	0.0322	22	0.0380
4	0.0252	11	0.0262	17	0.0392	23	0.0370
5	0.0190	12	0.0325	18	0.0299	24	0.0316
6	0.0660					25	0.0430

The Result

- Most probable keys, based on φ:
  - $i = 6$, $\varphi(i) = 0.0660$
    - plaintext `EBIIL TLOLA`
  - $i = 10$, $\varphi(i) = 0.0635$
    - plaintext `AXEEH PHKEW`
  - $i = 3$, $\varphi(i) = 0.0575$
    - plaintext `HELLO WORLD`
  - $i = 14$, $\varphi(i) = 0.0535$
    - plaintext `WTAAD LDGAS`
- Only English phrase is for $i = 3$
  - That's the key (3 or 'D')

Caesar's Problem

- Key is too short
  - Can be found by exhaustive search
  - Statistical frequencies not concealed well
    - They look too much like regular English letters
- So make it longer
  - Multiple letters in key
  - Idea is to smooth the statistical frequencies to make cryptanalysis harder

Polyalphabetic Ciphers

- Use multiple substitutions
- Most are periodic
  - These are essentially multiple Caesar ciphers
- Instead of adding the same key each time, each successive letter gets a different key added, but the keys repeat themselves
- When period is 1, this is equivalent to simple substitution

Polyalphabetic Ciphers

Example (key = SECUR)

CMPS IS FUN FOR ALL
SECU RS ECU RSE CUR

becomes

UQRN ZK ....

Attacking the Cipher

- Approach
  - Establish period; call it *n*
  - Break message into *n* parts, each part being enciphered using the same key letter
  - Solve each part
    - You can leverage one part from another

Establish Period

- Kasiski: *repetitions in the ciphertext occur when characters of the key appear over the same characters in the plaintext*
- Example:

key **VIGVIGVIGVIGVIGV**
plain **THEBOYHASTHEBALL**
cipher **OPKWWECIYOPKWIRG**

Note the key and plaintext line up over the repetitions (underlined). As distance between repetitions is 9, the period is a factor of 9 (that is, 1, 3, or 9)

Sample Cipher from Bishop

ADQYS MIUSB OXKKT MIBHK IZOOO
EQOOG IFBAG KAUMF VVTAA CIDTW
MOCIO EQOOG BMBFV ZGGWP CIEKQ
HSNEW VECNE DLAAV RWKXS VNSVP
HCEUT QOIOF MEGJS WTPCH AJMOC
HIUIX

Repetitions in Example

Letters	*Start*	*Repeats*	*Distance*	*Factors*
MI	5	15	10	2, 5
OO	22	27	5	5
OEQOOG	24	54	30	2, 3, 5
FV	39	63	24	2, 2, 2, 3
AA	43	87	44	2, 2, 11
MOC	50	122	72	2, 2, 2, 3, 3
QO	56	105	49	7, 7
PC	69	117	48	2, 2, 2, 2, 3
NE	77	83	6	2, 3
SV	94	97	3	3
CH	118	124	6	2, 3

Estimate of Period

- OEQOOG is probably not a coincidence
  - It’s too long for that
  - Period may be 1, 2, 3, 5, 6, 10, 15, or 30
- Most others (8/11) have 2 in their factors
- Almost as many (7/11) have 3 in their factors
- Six of eleven have 6 in their factors
- Begin with period of $2 \times 3 = 6$

Index of Coincidence (IC)

- Index of coincidence is probability that two randomly chosen letters from ciphertext will be the same

$IC = [n(n-1)]^{-1} \sum_{0 \le i \le 25} [F_i (F_i - 1)]$

- where n is length of ciphertext and F_i the number of times character i occurs in ciphertext

Compute IC

- Tabulated for different periods:

1	0.066	3	0.047	5	0.044
2	0.052	4	0.045	10	0.041
Large		0.038			

- For sample cipher IC = 0.043
  - Indicates a key of slightly more than 5
  - A statistical measure, so it can be in error, but it agrees with the previous estimate (which was 6)

Splitting Into Alphabets

alphabet 1: **AIKHOIATTOBGEEERNEOSAI**
alphabet 2: **DUKKEFUAWEMGKWDWSUFWJU**
alphabet 3: **QSTIQBMAMQBWQVLKVTMTMI**
alphabet 4: **YBMZOAFCOOFPHEAXPQEPOX**
alphabet 5: **SOIOOGVICOVCSVASHOGCC**
alphabet 6: **MXBOGKVDIGZINNVVCIJHH**

Use same approach as for monoalphabet on each of the six alphabets

Running Key Ciphers

- Cipher has key as long as the text
- Since security of substitution cipher increases with key length, this is more secure
- Uses nonrepeating text, such as a book
  - key specified by page and paragraph number

Consider Bishop Section 8.2.2.2 (p. 107)

Example (key = The one time pad is …)

CMPS IS FUN FOR ALL
THEO NE TIM EPA DIS

becomes

VTTG VW YCZ ….

Vernam Cipher

- Uses random characters as the key
- One time pads
  - Provably unbreakable
  - Why? Look at ciphertext DXQR. Equally likely to correspond to plaintext DOIT (key AJIY) and to plaintext DONT (key AJDY) and any other 4 letters
- Warning: keys *must* be random, or you can attack the cipher by trying to regenerate the key
  - Approximations, such as using pseudorandom number generators to generate keys, are *not* random

Product Ciphers

- Compose substitution and transposition ciphers
  - Lucifer
  - DES
  - AES

Conventional Cryptosystems

- One key
- Encipher and decipher with same key

Asymmetric Cryptosystems

- Two keys
- Encipher and decipher with different keys
- Computationally infeasible to determine one key from the other

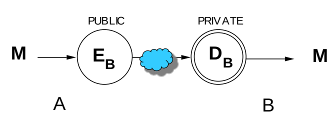
Public-key Cryptosystems

- Each user has both a public and a private key
- Two users can communicate knowing only each other's public key
- It must be computationally infeasible to determine a user's private key from their public key

Secrecy

Assume Public Key for User K = Ek

Assume Private Key for User K = Dk



$D_B(E_B(M))=M$

Digital Signature

A property private to a user that is used for signing messages

Digital Signature

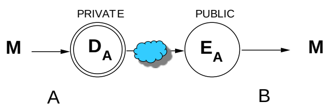
For A to sign a message sent to B the following properties must be satisfied by A's signature:

- B must be able to validate A's signature on the message
- It must be impossible for anyone, including B, to forge A's signature
- It must be possible for a judge or third party to settle a dispute between A and B

Authentication

Assume Public Key for User K = Ek

Assume Private Key for User K = Dk

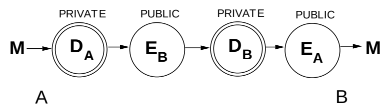


$E_A(D_A(M))=M$

Secrecy and Authentication

Assume Public Key for User K = Ek

Assume Private Key for User K = Dk



$E_A(D_B(E_B(D_A(M))))=M$

Public Key Encryption

Based on problems that are known to be hard to solve

Merkle-Hellman Knapsack

RSA

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:
  $n = p_1^{e_1} \times \ldots \times p_k^{e_k}$
- 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:
  $\gcd(18, 30) = 6$ $\quad\quad$ $\gcd(0, 20) = 20$
  $\gcd(-21, 49) = 7$
- 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
  $a = r + kn$
- Example:
  $29 \bmod 13 = 3$ $\quad$ $13 \bmod 13 = 0$ $\quad$ $-1 \bmod 13 = 12$
  $29 = 3 + 2{\times}13$ $\quad$ $13 = 0 + 1{\times}13$ $\quad$ $12 = -1 + 1{\times}13$
- 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 mod 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

Multiplicative Inverses (1)

- The residues modulo a positive integer n are the set

  $Z_n = \{0, 1, 2, \ldots, (n-1)\}$
- Let x and y be two elements of Z_n such that

  $xy \bmod n = 1$

  We say that y is the multiplicative inverse of x in 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 Z_n has a multiplicative inverse if and only if x and n are relatively prime

- Example
  - The elements of Z_{10} with a multiplicative inverse are 1, 3, 7, 9

Corollary

If is p is prime, every nonzero residue in Z_p has a multiplicative inverse

Theorem

A variation of Euclid's GCD algorithm computes the multiplicative inverse of an element x of 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

Merkle-Hellman Knapsack

- Superincreasing sequence is a sequence of positive integers where each element is greater than the sum of the previous elements
- Merkle-Hellman uses two knapsacks
  - Easy knapsack - superincreasing sequence
  - Hard knapsack - derived by modifying elements of the easy one
- Modification is such that any solution of one knapsack is a solution of the other

Merkle-Hellman Key Selection

- Choose superincreasing sequence S of m integers
- Choose a modulus n greater than the sum of the elements of S
- Choose multiplier w that is relatively prime to n
- Construct H by replacing each integer in S by Hi = w*Si mod n

- Encryption

  C = H*M
- Decryption

  w^{-1}*C = w^{-1}*H*M

  = w^{-1}*w*S*M

  = S*M

Example

S = [1, 2, 4, 9]

n = 17

w = 15

H = [15, 13, 9, 16]

Another Example

S = [47, 52, 112, 216, 436, 868, 1732, 3470, 6937,13876]

n = 27749

w = 113

H = [5311, 5876, 12656, 24408, 21519, 14837, 1473, 3624, 6909, 14044]

M = 0110110101

C = (0*5311 + 1*5876 + … +0*6909 +1*14044)
 = 72556

w^{-1} = 7367

(w^{-1}*C) mod 27749 = (7367*72556) mod 27749
 = 18814

S = [47, 52, 112, 216, 436, 868, 1732, 3470, 6937,13876]

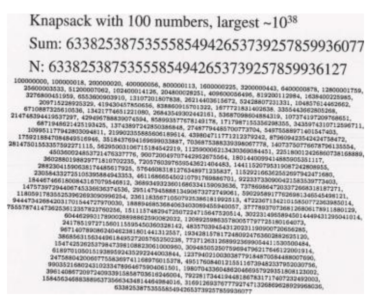
(w^{-1}*C) mod 27749 = M*S = 18814

M =

Conventional Wisdom Says

Real knapsacks should contain at least 250 items

Each item in the knapsack should have on the order of 200 bits in its binary representation



100000000, 100000018, 200000020, 400000056, 800000113, 1600000225, 3200000443, 6400000878, 12800001759,
25600003533, 51200007062, 102400014126, 204800028251, 409600056496, 819200112984, 1638400225985,
3276800451959, 6553600903910, 13107201807838, 26214403615672, 52428807231331, 104857614462662,
209715228925329, 419430457850656, 838860915701322, 1677721831402638, 3355443662805268,
6710887325610536, 13421774651221080, 26843549302442161, 53687098604884319, 107374197209768651,
214748394419537297, 429496788839074594, 858993577678149178, 1717987155356298355, 3435974310712596711,
6871948621425193425, 13743897242850386848, 27487794485700773704, 54975588971401547403,
109951177942803094811, 219902355885606189614, 439804711771212379242, 879609423542424758472,
1759218847084849516946, 3518437694169699033887, 7036875388339398067778, 14073750776678796135554,
28147501553357592271115, 56295003106715184542219, 112590006213430369084451, 225180012426860738168889,
450360024853721476337776, 900720049707442952675564, 1801440099414885905351117,
3602880198829771810702235, 7205760397659543621404483, 14411520795319087242808955,
28823041590638174485617925, 57646083181276348971235837, 115292166362552697942471680,
230584332725105395884943353, 461168665450210791769886701, 922337330900421583539773403,
1844674661800843167079546812, 3689349323601686334159093636, 7378698647203372668318187271,
14757397294406745336636374536, 29514794588813490673272749061, 59029589177626981346545498121,
118059178355253962693090996254, 236118356710507925386181992513, 472236713421015850772363985014,
944473426842031701544727970030, 1888946853684063403089455940057, 3777893707368126806178911880129,
7555787414736253612357823760256, 15111574829472507224715647520514, 30223149658945014449431295041014,
60446299317890028898862590082032, 120892598635780057797725180164073,
241785197271560115595450360328142, 483570394543120231190900720656285,
967140789086240462381801441312557, 1934281578172480924763602882625120,
3868563156344961849527205765250238, 7737126312689923699054411530500484,
15474252625379847398108823061000960, 30948505250759694796217646122001914,
61897010501519389592435292244003844, 123794021003038779184870584488007690,
247588042006077558369741168976015378, 495176084012155116739482337952030756,
990352168024310233478964675904061501, 1980704336048620466957929351808123002,
3961408672097240933915858703616246004, 7922817344194481867831717407232492003,
15845634688388963735663434814464984016, 31691269376777927471326869628929968036,
63382538753555854942653739257859936077

Knapsack with 100 numbers, largest ~10^{38}

Sum: 63382538753555854942653739257859936077

N: 63382538753555854942653739257859936127

Fermat's Little Theorem

Theorem

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

- Example ($p = 5$):

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

$3^4 \bmod 5 = 81 \bmod 5 = 1$ $\quad$ $4^4 \bmod 5 = 256 \bmod 5 = 1$

Corollary

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

Proof

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

Totient Function

- 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

$Z^*_{10} = \{ 1, 3, 7, 9 \}$

$\phi(10) = 4$

$Z^*_{21} = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 \}$

$\phi(21) = 12$

- If p is prime, we have

$Z^*_p = \{1, 2, \ldots, (p-1)\}$ $\qquad$ $\phi(p) = p - 1$

Euler's Theorem

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 Algorithm

- Uses multiplication of large primes to produce keys
- Relies on the difficulty of factoring large numbers for secrecy

RSA Key Selection

– Select two large primes *p* and *q*
– Compute *n* = p*q
– Compute ϕ(n) = (p-1)*(q-1)
– Choose an integer *e* between *3* and ϕ(n) that has no common factor with ϕ(n)
– Select an integer *d* such that d**e* mod ϕ(n) = 1
– *e*,*n* are made public
– *p*,*q*,*d*, ϕ*(n)* are kept secret

- **Encryption**

$$C = M^e \mod n$$

- **Decryption**

$$M = C^d \mod n$$

Example

p = 5
q = 7
n = p*q = 5*7 = 35
∅(n) = (p-1)(q-1) = 4*6 = 24
e = 11
d = 11

Example

p = 53
q = 61
n = p*q = 53*61 = 3233
∅(n) = (p-1)(q-1) = 52*60 = 3120
e = 71
d = 791

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

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***)

Product Ciphers

- Compose substitution and transposition ciphers
  - Lucifer
  - DES
  - AES

Lucifer

- Developed by IBM in 1974
- S Boxes - Nonlinear Substitution Boxes
  - 4
- P Boxes - Permutation Boxes
  - 128
- 128 bit key

DES - Data Encryption Standard

- Enciphers 64-bit blocks
- Outputs 64 bits of ciphertext
- Uses 56-bit key
- Adapted by NBS(NIST) for unclassified US government applications
- Initial and final permutation
- 16 rounds (iterations)
  - S boxes and P boxes

Controversy

- Considered too weak
  - Diffie, Hellman said in a few years technology would allow DES to be broken in days
    - Design using 1999 technology published
  - Design decisions not public
    - S-boxes may have backdoors

Strength of DES

- Undesirable properties
- Special purpose machine attacks
- Double DES
- Triple DES

Undesirable Properties

- 4 weak keys
  - They are their own inverses
- 12 semi-weak keys
  - Each has another semi-weak key as inverse
- S-boxes exhibit irregular properties
  - Distribution of odd, even numbers non-random
  - Outputs of fourth box depends on input to third box

Electronic Frontier Foundation

- Built a special purpose machine
- Cost budget $210,000
  - $80,000 design
  - $130,000 material
- Crack DES key in 4.5 days
- Design and algorithms published in scannable form

Double DES

- Encrypt (k1) Encrypt(k2)
- Susceptible to “meet-in-the-middle” attack
  - Plaintext attack
  - Reduces the number of keys to check from

$\mathbf{2^{112}}$ to $\mathbf{2^{57}}$

Triple DES

- Encrypt(k1) Decrypt(k2) Encrypt(k3)
- By using same key for all three it is identical to DES
- Having all three keys unique is referred to as “triple key triple DES”

Advanced Encryption Standard (AES)

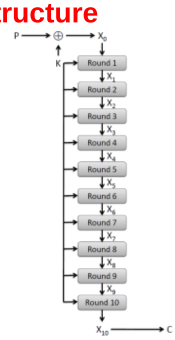
- NIST initiated a competition for AES in 1999
- Rijndael was selected in October 2000
  - Vincent Rijmen and Joan Daemen
- Became a Federal Information Processing standard (PUB 197) in November 2001
- NSA approved for classified information in June 2003

AES

- Encrypts 128 word blocks
- Various key lengths
  - 128 uses 10 rounds
  - 192 uses 12 rounds
  - 256 uses 14 rounds
- Single S box - one byte in one byte out
- P box based on square of bytes
- 16 bytes of key per round

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 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}.



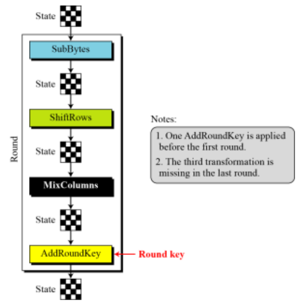
Goodrich + Tamassia

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

Structure of Each Round

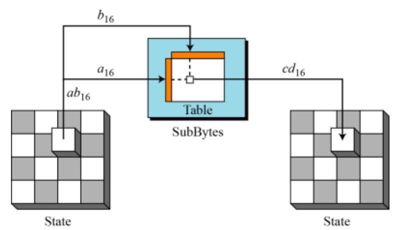


Changing Text to State

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \text{State}$$

SubBytes Substitution Step

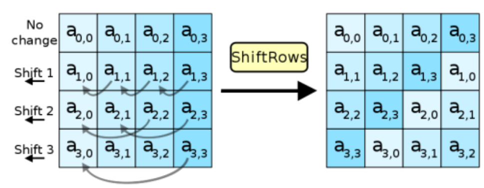


Shift Rows Step

ShiftRows step: a permutation step

- Row 1 no shift
- Row 2 left shift 1
- Row 3 left shift 2
- Row 4 left shift 3

Shift Rows Step



MixColumns Step

MixColumns step: a matrix multiplication step

Each column multiplied by known matrix

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

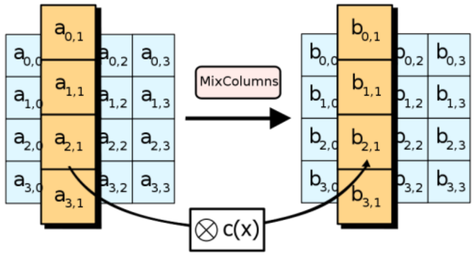
where

1 means no change

2 means shifting to the left

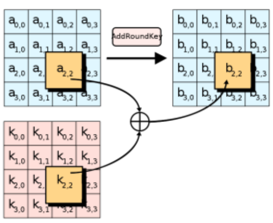
3 means shifting to the left and then peforming XOR with initial unshifted value

MixColumns Step



AddRoundKey Step

- Each byte of the state is combined with a byte of the round subkey using the XOR operation



Round Subkeys

- Each round key subkey is derived from the main key using Rinjdael's key schedule
- Each subkey is the same size as the state
- Subkey for each round plus one more

Block Ciphers

- Break message M into successive blocks M1, M2, ...
- Encipher each Mi with the same key k

Ek(M) = Ek(M1)Ek(M2) ...

Block Ciphers

- Advantages
  - Only one execution of the encryption algorithm per n characters
  - Errors in one ciphertext block have no effect on other blocks
- Disadvantages
  - More susceptible to cryptanalysis
    - Identical blocks of plaintext yield identical blocks of ciphertext
    - Vulnerable to ciphertext searching
    - More susceptible to replay

Block Chaining

- Inserts some bits of the previous ciphertext block into unused portions of the current plaintext block before encrypting it
- Reduces the number of available message bits per block

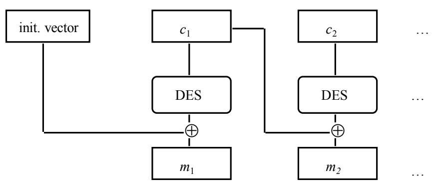
Cipher Block Chaining

- Exclusive ORs previous ciphertext block with the current plaintext block then encrypts the result

  $C_i = E_k(M_i \oplus C_{i-1})$

- Ci is functionally dependent on all previous blocks
- Useful for checksumming and digital signatures

CBC Mode Decryption



Cipher Feedback

- Part of previous ciphertext is shifted into a shift register
- Shift register is encrypted with the user's key and the result is XOR'd with the plaintext block

One-way Hash Function

- Takes a variable length input and produces a fixed length output
  - input is called the *preimage*
  - output is called the *hash value* or *message digest*
- Transformation is irreversible
- Called digest function, cryptographic checksum, message integrity check

Where to Encrypt and Decrypt

Link encryption

End-to-End encryption

Link Encryption

- Enciphers and deciphers a message M at each node between the source and destination
  - Each host need only know the keys for its immediate neighbors
  - Data is exposed at each intermediate node

End-to-End Encryption

- Encipher the message at the source and decipher it at the destination
  - User needs a separate key for each correspondent
  - More susceptible to traffic flow analysis because the destination is always exposed

Two Approaches Can Be Combined

Source sends a message that is the encrypted version of the original message over link encrypted communication system