COS 226 Lecture 21: Cryptology

Cryptology: Cryptography + Cryptanalysis

Protocols and algorithms
* for sending and receiving secret messages
* for reading someone else's secret messages
Traditional applications: military
Modern applications: commercial

Never can "prove" security: too much can go wrong

Role of efficient algorithms
* practical to encode messages?
* implement codebreaking attacks
* (new meaning to "easy", "hard")

Parallel universe (primarily secret until 1980s)
Old world:
    good guys and bad guys
    generated $$ for supercomputers
    machine code and machines
    codes and attacks
New world:
    everyone (Alice, Bob, and Eve)
    generates venture capital $$
    C++ and Java
    protocols

Unbreakable cryptosystem

ONE-TIME PAD

PROTOCOL

* Alice and Bob exchange keys by messenger

(time passes)

* Bob encrypts message with key (ciphertext)
* Bob sends ciphertext to Alice
* Alice decrypts ciphertext with key
* Bob and Alice destroy key

Eve can read ciphertext, but not message

Ex:

A T T A C K A T D A W N

encrypt with random key

A T T A C K A T D A W N	**message**
M A K G A U E B H M C V V M	**add key**
N S D H D E E C A M G W R	**ciphertext**

send this message

N S D H D E E C A M G W R

decrypt with same key (communicated previously)

N S D H D E E C A M G W R	**ciphertext**
M A K G A U E B H M C V V M	**subtract key**
A T T A C K A T D A W N	**message**

PROBLEMS

* need random key as long as message
* need secure protocol to distribute key

Linear feedback shift registers

Solution to key length problem
Small key, very large number of random bits

Register with bit values at time T+1
determined from values at time T

Ex: 11-bit register

T	9	8	7	...	1	0
T+1	10	9	8		2	10+3 (XOR)

0	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**initial value**
1	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	
1	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	
0	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	
1	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	
0	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	
0	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	
0	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	
0	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	
1	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	
0	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	

Bit 0 values comprise pseudo-random bitstream
N-bit initial value gives sequence of length 2^N
provided bit taps are "primitive"

Stream cipher example

To transmit the message "SEND FOOD"

ENCODE with S = 10011, E = 00101, N = 01110, etc.

. S E N D F O O D

. 10010001010111000100000000110011100111000100

ENCRYPT with key from LFBSR

message

. 10010001010111000100000000110011100111000100

key

. 00100110010000110101010000111101010011100101

ciphertext

. 10110111000110110001010000100100100100100001

TRANSMIT ciphertext

. V * M Q H I D I A

DECRYPT with keystream from same initial value

ciphertext

. 10110111000110110001010000100100100100100001

key

. 00100110010000110101010000111101010011100101

message

. 10010001010111000100000000110011100111000100

decode

. S E N D F O O D

Digression: random number generators

Can’t generate random numbers on a computer

von Neumann:

> "anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin"

Linear congruential generators

Additive congruential generators

True random bits hard to find

- clock
- time between keystrokes
- cosmic rays

Pseudo-random more convenient for keystream

Cheap source of random bits:

- some encrypted message?

Simple attack on LFBSR

If cryptanalyst knows part of the message,
the entire keystream amy be available

Ex: operator puts two spaces at the beginning

. S E N D F O

. 00000000001001000101011100010000000011001110

encrypts with key from LFBSR

. 00000000001001000101011100010000000011001110

. 00100110010000110101010000111101010011100101

. 00100110011001110000011001011010100000101011

transmits this ciphertext

. 00100110011001110000011001011010100000101011

. D Y S P F K J A K

First 10 bits of ciphertext are the key!

Any 11 bits of key (plus a machine)
gives ALL the key bits

Cryptanalyst

* knows from spies that machine is 11-bit LFBSR
* guesses that first two chars might be spaces
* tries both possibilities for 11th initial bit
* generates key
* reads message

WW-II ENIGMA

Typewriter-like machine
Symmetric code
    type A, it prints G
    type G, it prints A
Particular code depends on machine settings
KEY DISTRIBUTION: "codebook"

Sender types message, machine prints ciphertext
Ciphertext transmitted in the clear
Receiver types ciphertext, machine prints message

Typewriter-like technology + rotors and plugboards
Over 137629917937512000 (10^17) different settings
Thousands of machines, millions of messages

Good guys broke the code (and won the war) by
    * getting some machines
    * getting some rotors
    * getting a codebook
    * figuring out what some messages were
    * building a computer to try
        remaining possibilities

One of the good guys: Alan Turing (!)

Ref: "Alan Turing, the Enigma" A. Hodges

Modern applications

Widespread application of computing
creates opportunities
New research in cryptography
provides necessary technology

* ecommerce
* communication among citizens
* better military systems

Challenges:
(social and political issues)
key distribution
efficient encode/decode

One-way trapdoor functions

Concept that opened the door to
modern cryptography (Diffie-Hellman, 1975)

One-way functions
- easy to compute
- difficult to compute inverse

Ex: write message on mirror, smash mirror

One-way trapdoor functions
- easy to compute
- difficult to compute inverse
- easy to compute inverse with key

Ex: put message in locked mailbox
person with key can get message

Key to cryptography is to find
- * good one-way functions
- * trapdoors

FACTORING
- p*q = N: easy
- find p given N: difficult
- find p given q, N: easy

DISCRETE LOG
- x^t mod p = M: easy
- find t given x, p, M: difficult

Diffie-Hellman key exchange

Publicly available keys:

large integers N and g (100 digits)

Alice chooses random x and computes

X = g^x mod N

Bob chooses random y and computes

Y = g^y mod N

Alice and Bob exchange X and Y

(but keep x and y secret)

Alice computes

Y^x = g^(xy) mod N = K

Bob computes

X^y = g^(yx) mod N = K

Computations all "easy"

hundreds of 100-digit multiplications

Both Alice and Bob have K

Eve needs to solve discrete logarithm

on 100 digit numbers to get K

Public-key cryptosystem

Every registered user is issued

P: public (encryption) key

S: secret (decryption) key

Public keys are published (phone book)

User responsible for keeping private key secret

PROTOCOL

* Alice computes C = P(M) using Bob's public key
* Alice transmits ciphertext to Bob
* Bob computes M = S(C) using his private key

Eve can read ciphertext, but not message

(without Bob's private key)

Works if all (S, P) pairs satisfy

1. S(P(M)) = M for every message M
2. All (S, P) pairs are distinct
3. Deriving S from P is as hard as reading M
4. Both S and P are easy to compute

1. and 2. easily arranged

3. and 4. hard to design

RSA encryption/decryption

Every registered user gets

encryption key P: two integers N and p

decryption key S: N and a third integer s

Assume all numbers 100s of digits

Encode the message as a number

Break into pieces smaller than N

(less than lgN bits)

Alice computes C = P(M) using Bob's public key

by computing M^p mod N

Bob computes M = S(C) using his private key

by computing C^s mod N

N, p, and s chosen to make the system work as follows:

* generate three large primes s, x, y
* take N = x*y
* choose p so that

$$ps \bmod (x-1)(y-1) = 1$$

RSA example

Suppose x = 47 y = 79 s = 97

Then compute N = 3713 and p = 37

A T T A C K A T D A W N

ENCODE: A = 01, T = 20, C = 03, K = 11, etc.

. A T T A C K A T D A W N

. 01202001031100012000040123 14

ENCRYPT using public key 37

0120^37 = 1040, 2001^37 = 2932, (mod 3713) ...

. 0120 2001 0311 0001 2000 0401 2314

. 1404 2932 3536 0001 3284 2280 2235

TRANSMIT ciphertext

. 1404293235360001328422802235

. N D * * * * A * * V * V *

DECRYPT using secret key 97

1404^97 = 1040, 2001^97 = 2932, (mod 3713) ...

. 1404 2932 3536 0001 3284 2280 2235

. 0120 2001 0311 0001 2000 0401 2314

DECODE

. 01202001031100012000040123 14

. A T T A C K A T D A W N

RSA algorithms

ENCRYPT, DECRYPT

- Use successive squaring (see Lecture 20)
  - THM: Less than 2 lg N multiplications are required to compute x^N mod M

FIND RANDOM 100-DIGIT PRIME

- Hard way: generate random 100-digit number, try to factor it.
- Fast way: probabilistic method

FIND p GIVEN s, x, and y:

- Extended Euclidean algorithm

RSA applicability rests on fast multiplication (!)

- present: use hybrid method (ex: PGP)
- future: unnoticed special-purpose chips

Bottom line:

- practical algorithm
- widely applicable
- depends on fundamental research results in theoretical CS and number theory

(RSA inc. went public last week for $250 million)