

15-499: Algorithms and Applications

Cryptography III

- Private Key Cryptosystems
- Public Key Cryptosystems

15-499

Page 1

Cryptography Outline

Introduction: terminology and background

Primitives: one-way hash functions, trapdoors, ...

Protocols: digital signatures, key exchange, ..

Number Theory: groups, fields, ...

➔ **Private-Key Algorithms:** Rijndael, DES, RC4

Cryptanalysis: Differential, Linear

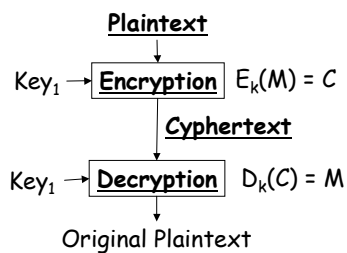
Public-Key Algorithms: Knapsack, RSA, El-Gamal,
Blum-Goldwasser

Case Studies: Kerberos, Digital Cash

15-499

Page 2

Private Key Algorithms



What granularity of the message does E_k encrypt?

15-499

Page 3

Private Key Algorithms

Block Ciphers: blocks of bits at a time

- DES (Data Encryption Standard)
Banks, linux passwords (almost), SSL, kerberos, ...
- Blowfish (SSL as option)
- IDEA (used in PGP, SSL as option)
- Rijndael (AES) - **the new standard**

Stream Ciphers: one bit (or a few bits) at a time

- RC4 (SSL as option)
- PKZip
- Sober, Leviathan, Panama, ...

15-499

Page 4

Private Key: Block Ciphers

Encrypt one block at a time (e.g. 64 bits)

$$c_i = f(k, m_i) \quad m_i = f^{-1}(k, c_i)$$

Keys and blocks are often about the same size.

Equal message blocks will encrypt to equal codeblocks

- Why is this a problem?

Various ways to avoid this:

- E.g. $c_i = f(k, c_{i-1} \oplus m_i)$

"**Cipher block chaining**" (CBC)

Why could this still be a problem?

Solution: attach random block to the front of the message

15-499

Page 5

Security of block ciphers

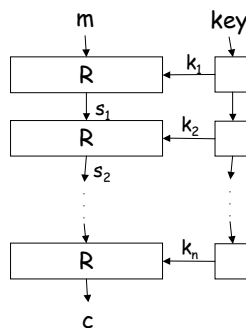
Ideal:

- k-bit \rightarrow k-bit key-dependent substitution (i.e. "random permutation")
- If keys and blocks are k-bits, can be implemented with 2^{2k} entry table.

15-499

Page 6

Iterated Block Ciphers



Consists of n **rounds**

R = the "round" function

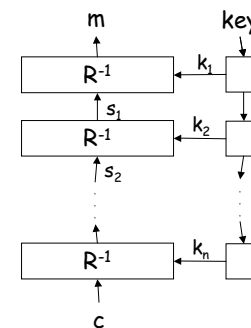
s_i = state after round i

k_i = the i^{th} round key

15-499

Page 7

Iterated Block Ciphers: Decryption



Run the rounds in reverse.

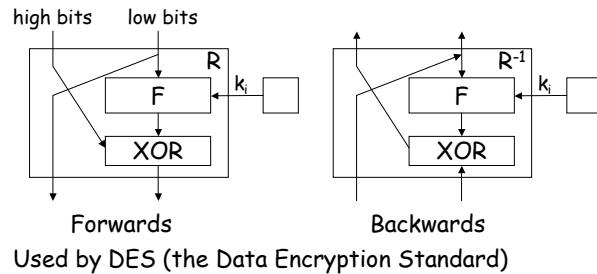
Requires that R has an inverse.

15-499

Page 8

Feistel Networks

If function is not invertible rounds can still be made invertible. Requires 2 rounds to mix all bits.



15-499

Page 9

Product Ciphers

Each round has two components:

- **Substitution** on smaller blocks
Decorrelate input and output: "confusion"
- **Permutation** across the smaller blocks
Mix the bits: "diffusion"

Substitution-Permutation Product Cipher

Avalanch Effect: 1 bit of input should affect all output bits, ideally evenly, and for all settings of other in bits

15-499

Page 10

Rijndael

Selected by AES (Advanced Encryption Standard, part of NIST) as the new private-key encryption standard.

Based on an open "competition".

- Competition started Sept. 1997.
- Narrowed to 5 Sept. 1999
 - MARS by IBM, RC6 by RSA, Twofish by Counterplane, Serpent, and Rijndael
- Rijndael selected Oct. 2000.
- Official Oct. 2001? ([AES page on Rijndael](#))

Designed by Rijmen and Daemen (Dutch)

15-499

Page 11

Goals of Rijndael

Resistance against known attacks:

- Differential cryptanalysis
- Linear cryptanalysis
- Truncated differentials
- Square attacks
- Interpolation attacks
- Weak and related keys

Speed + Memory efficiency across platforms

- 32-bit processors
- 8-bit processors (e.g smart cards)
- Dedicated hardware

Design simplicity and clearly stated security goals

15-499

Page 12

High-level overview

An iterated block cipher with

- 10-14 rounds,
- 128-256 bit blocks, and
- 128-256 bit keys

Mathematically reasonably sophisticated

Blocks and Keys

The blocks and keys are organized as matrices of bytes. For the 128-bit case, it is a 4x4 matrix.

$$\begin{pmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{pmatrix} \quad \begin{pmatrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{pmatrix}$$

Data block Key

b_0, b_1, \dots, b_{15} is the order of the bytes in the stream.

Galois Fields in Rijdael

Uses $GF(2^8)$ over bytes.

The irreducible polynomial is:

$$M(x) = x^8 + x^4 + x^3 + x + 1 \text{ or } 100011011 \text{ or } 0x11B$$

Also uses degree 3 polynomials with coefficients from $GF(2^8)$.

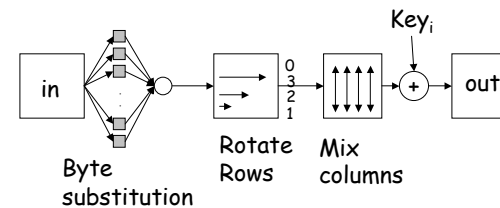
These are kept as 4 bytes (used for the columns)

The polynomial used as a modulus is:

$$M(x) = 00000001x^4 + 00000001 \text{ or } x^4 + 1$$

Not irreducible, but we only need to find inverses of polynomials that are relatively prime to it.

Each round



The inverse runs the steps and rounds backwards. Each step must be reversible!

Byte Substitution

Non linear: $y = b^{-1}$ (done over $GF(2^8)$)

Linear: $z = Ay + b$ (done over $GF(2)$, i.e., binary)

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ \vdots & & & & & & & & \end{pmatrix} \quad B = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

To invert the substitution:

$$y = A^{-1}(z - b) \quad (\text{the matrix } A \text{ is nonsingular})$$

$$b = y^{-1} \quad (\text{over } GF(2^8))$$

15-499

Page 17

Mix Columns

For each column:

$$\begin{pmatrix} a' \\ b' \\ c' \\ d' \end{pmatrix} = C(x) \cdot \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \quad C = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

This corresponds to 4 polynomial multiplies each of degree 4 using $GF(2^8)$ as the coefficients, and then reduced by $M(x) = (x^4+1)$

For example:

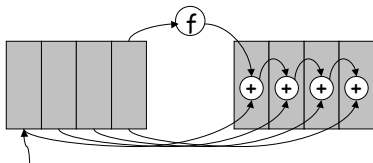
$$a' = (ax^3+bx^2+cx+d)(2x^3+3x^2+x+1) \bmod (x^4+1)$$

$M(x)$ is not irreducible, but the rows of C and $M(x)$ are coprime, so the transform can be inverted.

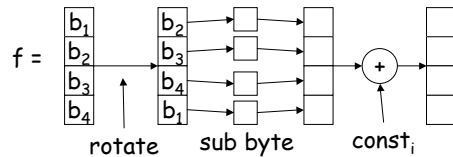
15-499

Page 18

Generating the round keys



Words corresponding to columns of the key

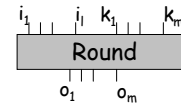


15-499

Page 19

Linear Cryptanalysis

A known plaintext attack used to extract the key



Consider a linear equality involving i , o , and k

$$\text{e.g.: } k_1 \oplus k_6 = i_2 \oplus i_4 \oplus i_5 \oplus o_4$$

To be secure this should be true with $p = .5$

(probability over all inputs and keys)

If true with $p = 1$, then linear and easy to break

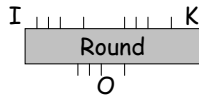
If true with $p = .5 + \epsilon$ then you might be able to use this to help break the system

15-499

Page 20

Differential Cryptanalysis

A chosen plaintext attack used to extract the key



Considers fixed "differences" between inputs, $\Delta_I = I_1 - I_2$, and sees how they propagate into differences in the outputs, $\Delta_O = O_1 - O_2$.

"difference" is often exclusive OR

Assigns probabilities to different keys based on these differences. With enough and appropriate samples (I_1, I_2, O_1, O_2) , the probability of a particular key will converge to 1.

15-499

Page 21

Cryptography Outline

Introduction: terminology and background

Primitives: one-way hash functions, trapdoors, ...

Protocols: digital signatures, key exchange, ..

Number Theory: groups, fields, ...

Private-Key Algorithms: Rijndael, DES, RC4

Cryptanalysis: Differential, Linear

➔ **Public-Key Algorithms:** Knapsack, RSA, El-Gamal, Blum-Goldwasser

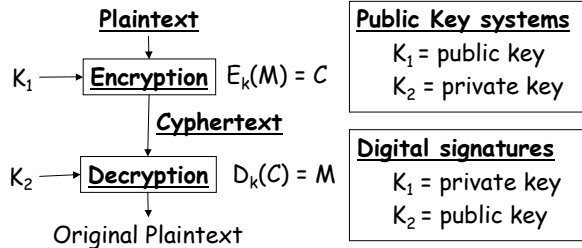
Case Studies: Kerberos, Digital Cash

15-499

Page 22

Public Key Cryptosystems

Introduced by Diffie and Hellman in 1976.



Typically used as part of a more complicated protocol.

15-499

Page 23

One-way trapdoor functions

Both Public-Key and Digital signatures make use of one-way trapdoor functions.

Public Key:

- Encode: $c = f(m)$
- Decode: $m = f^{-1}(c)$ using trapdoor

Digital Signatures:

- Sign: $c = f^{-1}(m)$ using trapdoor
- Verify: $m = f(c)$

15-499

Page 24

Example of SSL

SSL (**Secure Socket Layer**) is the standard for the web (**https**).

Protocol (somewhat simplified):

B->A: hello www.amazon.com I would like a secure connection
 A->B: Hi, I'm amazon.com, |amazon.com|_{verisign}
 B->A: Prove it
 A->B: Bob this is amazon, <[Bob this is amazon]|_{amazon's private key}
 B->A: ok Amazon, here is a shared-key, {key}_{amazon's public key}
 A->B: (message,[message])_{key}

|h|_{issuer} = Certificate
 = Issuer, <h,h's public key, time stamp>_{issuer's private key}

<...>_{private key} = Digital signature {...}_{public key} = Public encryption
 [..] = Secure Hash (...)_{key} = Private encryption

Public Key History

Some algorithms

- **Merkle-Hellman**, 1978, based on "knapsack problem"
- **McEliece**, 1978, based on algebraic coding theory
- **RSA**, 1978, based on factoring
- **Rabin**, 1979, security can be reduced to factoring
- **ElGamal**, 1985, based on Discrete logs
- **Blum-Goldwasser**, 1985, based on quadratic residues
- **Elliptic curves**, 1985, discrete logs over Elliptic curves
- **Chor-Rivest**, 1988, based on knapsack problem
- **NTRU**, 1996, based on Lattices
- **XTR**, 2000, based on discrete logs of a particular field

Diffie-Hellman Key Exchange

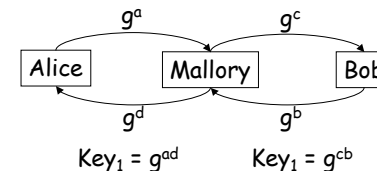
A group $(G,*)$ and a primitive element (generator) g is made public.

- Alice picks a , and sends g^a to Bob
- Bob picks b and sends g^b to Alice
- The shared key is g^{ab}

Note this is easy for Alice or Bob to compute, but assuming discrete logs are hard is hard for anyone else to compute.

Can someone see a problem with this protocol?

Person-in-the-middle attack



Mallory gets to listen to everything.

Merkle-Hellman

Gets "security" from the **Subset Sum** (also called **knapsack**) which is NP-hard to solve in general.

Subset Sum (Knapsack): Given a sequence $W = \{w_0, w_1, \dots, w_{n-1}\}$, $w_i \in \mathbb{Z}$ of weights and a sum S , calculate a boolean vector B , such that:

$$\sum_{i=0}^{i < n} B_i W_i = S$$

Even deciding if there is a solution is NP-hard.

15-499

Page 29

Merkle-Hellman

W is **superincreasing** if: $w_i \geq \sum_{j=0}^{j=i-1} w_j$

It is easy to solve the subset-sum problem for superincreasing W in $O(n)$ time.

Main idea:

- Hide the easy case by multiplying each w_i by a constant a modulo a prime p

$$w'_i = a * w_i \text{ mod } p$$

- **Public key:** W' $E_{W'}(M) = \sum_{i=0}^{i < n} w'_i m_i$

- **Private key:** a, p : $D_w(S) = \text{solve for } B_i$

15-499

Page 30

Merkle Hellman: Problem

Was broken by Shamir in 1984.

Shamir showed how to use integer programming to solve the particular class of Subset Sum problems in polynomial time.

Lesson: don't leave your trapdoor loose.

15-499

Page 31

RSA

Invented by Rivest, Shamir and Adleman in 1978

Based on **difficulty of factoring**.

Used to **hide** the size of a group Z_n^* since:

$$|Z_n^*| = \phi(n) = n \prod_{p|n} (1 - 1/p)$$

Factoring has not been reduced to RSA

- an algorithm that generates m from c does not give an efficient algorithm for factoring

On the other hand, factoring has been reduced to finding the private-key.

- there is an efficient algorithm for factoring given one that can find the private key.

15-499

Page 32

RSA Public-key Cryptosystem

What we need:

- p and q, primes of approximately the same size
- $n = pq$
 $\phi(n) = (p-1)(q-1)$
- $e \in \mathbb{Z}_{\phi(n)}^*$
- $d = e^{-1} \bmod \phi(n)$

Public Key: (e,n)

Private Key: d

Encode:

$m \in \mathbb{Z}_n$
 $E(m) = m^e \bmod n$

Decode:

$D(c) = c^d \bmod n$

15-499

Page 33

RSA continued

Why it works:

$$\begin{aligned} D(c) &= c^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{1+k(p-1)(q-1)} \bmod n \\ &= m^{1+k\phi(n)} \bmod n \\ &= m(m^{\phi(n)})^k \bmod n \\ &= m \end{aligned}$$

Why is this argument not quite sound?

What if $m \notin \mathbb{Z}_n^*$ then $m^{\phi(n)} \neq 1 \bmod n$

Answer 1: Not hard to show that it still works.

Answer 2: jackpot - you've factored n

15-499

Page 34

RSA computations

To **generate the keys**, we need to

- **Find two primes p and q.** Generate candidates and use primality testing to filter them.
- **Find $e^{-1} \bmod (p-1)(q-1)$.** Use Euclid's algorithm. Takes time $\log^2(n)$

To **encode and decode**

- **Take m^e or c^d .** Use the power method. Takes time $\log(e) \log^2(n)$ and $\log(d) \log^2(n)$.

In practice e is selected to be small so that encoding is fast.

15-499

Page 35

Security of RSA

Warning:

- Do not use this or any other algorithm naively!

Possible security holes:

- Need to use "safe" primes p and q. In particular p-1 and q-1 should have large prime factors p_1 and q_1 , and p_1 and q_1 should have large factors, ...
- p and q should not have the same number of digits. Can use a middle attack starting at $\sqrt{\log(n)}$.
- E cannot be too small
- Don't use same n for different e's.
- You should always "pad"

15-499

Page 36

Algorithm to factor given d and e

If an attacker has an algorithm that generates d from e, then he/she can factor n in PPT. Variant of the Rabin-Miller primality test.

Function TryFactor(e, d, n)

1. write $ed - 1$ as $2^r \cdot r$, r odd
2. choose w at random $< n$
3. $v = w^r \bmod n$
4. if $v = 1$ then return(fail)
5. while $v \neq 1 \bmod n$
6. $v_0 = v$
7. $v = v^2 \bmod n$
8. if $v_0 = n - 1$ then return(fail)
9. return(pass, gcd($v_0 + 1$, n))

Las Vegas algorithm
Probability of pass
is $> .5$.
Will return p or q
if it passes.
Try until you pass.

15-499

Page 37

RSA Performance

Performance: (600Mhz PIII) (from: ssh toolkit):

| Algorithm | Bits/key | | Mbits/sec |
|--------------|----------|-------------|-----------|
| RSA Keygen | 1024 | .35sec/key | |
| | 2048 | 2.83sec/key | |
| RSA Encrypt | 1024 | 1786/sec | 3.5 |
| | 2048 | 672/sec | 1.2 |
| RSA Decrypt | 1024 | 74/sec | .074 |
| | 2048 | 12/sec | .024 |
| ElGamal Enc. | 1024 | 31/sec | .031 |
| ElGamal Dec. | 1024 | 61/sec | .061 |
| DES-cbc | 56 | | 95 |
| twofish-cbc | 128 | | 140 |
| Rijdael | 128 | | 180 |

15-499

Page 38

RSA in the "Real World"

Part of many standards: PKCS, ITU X.509, ANSI X9.31, IEEE P1363

Used by: SSL, PEM, PGP, Entrust, ...

The standards specify many details on the implementation, e.g.

- e should be selected to be small, but not too small
- "multi prime" versions make use of $n = pqr...$ this makes it cheaper to decode especially in parallel (uses Chinese remainder theorem).

15-499

Page 39

Factoring in the Real World

Quadratic Sieve (QS):

$$T(n) = e^{(1+o(n))(\ln n)^{1/2} (\ln(\ln n))^{1/2}}$$

- Used in 1994 to factor a 129 digit (428-bit) number. 1600 Machines, 8 months.

Number field Sieve (NFS):

$$T(n) = e^{(1.923+o(1))(\ln n)^{1/3} (\ln(\ln n))^{2/3}}$$

- Used in 1999 to factor 155 digit (512-bit) number. 35 CPU years. At least 4x faster than QS

15-499

Page 40

ElGamal

Based on the difficulty of the discrete log problem.
Invented in 1985

Digital signature and Key-exchange variants

- Digital signature is AES standard
- Public Key used by TRW (avoided RSA patent)

Works over various groups

- Z_p ,
- Multiplicative group $GF(p^n)$,
- Elliptic Curves

15-499

Page 41

ElGamal Public-key Cryptosystem

$(G, *)$ is a group

- α a generator for G
- $a \in Z_{|G|}$
- $\beta = \alpha^a$

G is selected so that it is hard to solve the discrete log problem.

Public Key: (α, β) and some description of G

Private Key: a

Encode:

Pick random $k \in Z_{|G|}$

$$E(m) = (y_1, y_2) \\ = (\alpha^k, m * \beta^k)$$

Decode:

$$D(y) = y_2 * (y_1^a)^{-1} \\ = (m * \beta^k) * (\alpha^{ka})^{-1} \\ = m * \beta^k * (\beta^k)^{-1} \\ = m$$

You need to know a to easily decode y !

15-499

Page 42

ElGamal Example

$N = 587$

$G = 5$

15-499

Page 43

Probabilistic Encryption

For RSA one message goes to one cipher word. This means we might gain information by running $E_{\text{public}}(M)$.

Probabilistic encryption maps every M to many C randomly. Cryptanalyst can't tell whether $C = E_{\text{public}}(M)$.

ElGamal is an example (based on the random k), but it doubles the size of message.

15-499

Page 44

BBS "secure" random bits

BBS (Blum, Blum and Shub, 1984)

- Based on difficulty of factoring, or finding square roots modulo $n = pq$.

Fixed

- p and q are primes such that $p = q = 3 \pmod{4}$
- $n = pq$ (is called a Blum integer)

For a particular bit seq.

- **Seed:** random x relatively prime to n .
- **Initial state:** $x_0 = x^2$
- **i^{th} state:** $x_i = (x_{i-1})^2$
- **i^{th} bit:** lsb of x_i

Note that: $x_0 = x_i^{-2^i \pmod{\phi(n)}} \pmod{n}$
Therefore knowing p and q allows us to find x_0 from x_i

15-499

Page 45

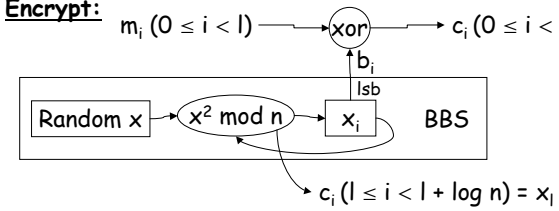
Blum-Goldwasser: A stream cypher

Public key: $n (= pq)$

Private key: p or q

Encrypt:

$m_i (0 \leq i < l) \xrightarrow{\text{xor}} c_i (0 \leq i < l)$



Decrypt:

Using p and q , find $x_0 = x_i^{-2^i \pmod{(p-1)(q-1)}} \pmod{n}$

Use this to regenerate the b_i and hence m_i

15-499

Page 46

Quantum Cryptography

In quantum mechanics, there is no way to take a measurement without potentially changing the state. E.g.

- Measuring position, spreads out the momentum
- Measuring spin horizontally, "spreads out" the spin probability vertically

Related to Heisenberg's uncertainty principal

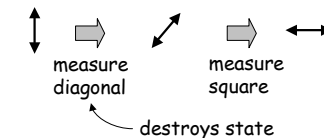
15-499

Page 47

Using photon polarization

$\updownarrow = \nearrow$ or \searrow ? (equal probability)

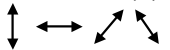
$\nearrow = \updownarrow$ or \leftrightarrow ? (equal probability)



15-499

Page 48

Quantum Key Exchange

1. Alice sends bob photon stream randomly polarized in one of 4 polarizations: 
2. Bob measures photons in random orientations
e.g.: X + + X X X + X (orientations used)
 \ | - \ / / - \ (measured polarizations)
and tells Alice in the open what orientations he used, but not what he measured.
3. Alice tells Bob in the open which are correct
4. Bob and Alice keep the correct values
Susceptible to a **man-in-the-middle** attack

15-499

Page 49

In the "real world"

Not yet used in practice, but experiments have verified that it works.
IBM has working system over 30cm at 10bits/sec.
More recently, up to 10km of fiber.



15-499

Page 50