

15-853: Algorithms in the Real World

Cryptography 1 and 2

15-853

Page 1

Cryptography Outline

Introduction: terminology, cryptanalysis, security

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

Protocols: digital signatures, key exchange, ..

Number Theory: groups, fields, ...

Private-Key Algorithms: Rijndael, DES

Public-Key Algorithms: Knapsack, RSA, El-Gamal, ...

Case Studies: Kerberos, Digital Cash

15-853

Page 2

Cryptography Outline

➔ **Introduction:**

- terminology
- cryptanalytic attacks
- security

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

Protocols: digital signatures, key exchange, ..

Number Theory: groups, fields, ...

Private-Key Algorithms: Rijndael, DES

Public-Key Algorithms: Knapsack, RSA, El-Gamal, ...

Case Studies: Kerberos, Digital Cash

15-853

Page 3

Some Terminology

Cryptography - the general term

Cryptology - the mathematics

Encryption - encoding but sometimes used as general term)

Cryptanalysis - breaking codes

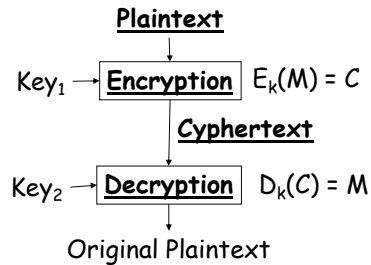
Steganography - hiding message

Cipher - a method or algorithm for encrypting or decrypting

15-853

Page 4

More Definitions



Private Key or **Symmetric**: $Key_1 = Key_2$

Public Key or **Asymmetric**: $Key_1 \neq Key_2$
Key₁ or Key₂ is public depending on the protocol

15-853

Page 5

Cryptanalytic Attacks

C = ciphertext messages

M = plaintext messages

Ciphertext Only: Attacker has multiple **Cs** but does not know the corresponding **Ms**

Known Plaintext: Attacker knows some number of **(C, M)** pairs.

Chosen Plaintext: Attacker gets to choose **M** and generate **C**.

Chosen Ciphertext: Attacker gets to choose **C** and generate **M**.

15-853

Page 6

What does it mean to be secure?

Unconditionally Secure: Encrypted message cannot be decoded without the key

Shannon showed in 1943 that key must be as long as the message to be unconditionally secure - this is based on information theory

A **one time pad** - xor a random key with a message (Used in 2nd world war)

Security based on computational cost: it is computationally "infeasible" to decode a message without the key.

No (probabilistic) polynomial time algorithm can decode the message.

15-853

Page 7

The Cast

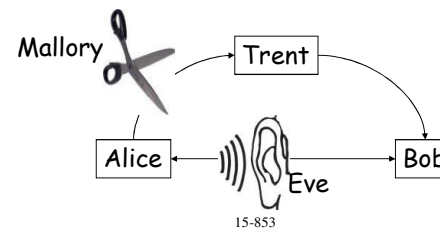
Alice - initiates a message or protocol

Bob - second participant

Trent - trusted middleman

Eve - eavesdropper

Mallory - malicious active attacker



15-853

Page 8

Cryptography Outline

Introduction: terminology, cryptanalysis, security

➔ **Primitives:**

- one-way functions
- one-way trapdoor functions
- one-way hash functions

Protocols: digital signatures, key exchange, ..

Number Theory: groups, fields, ...

Private-Key Algorithms: Rijndael, DES

Public-Key Algorithms: Knapsack, RSA, El-Gamal, ...

Case Studies: Kerberos, Digital Cash

15-853

Page 9

Primitives: One-Way Functions

(Informally): A function

$$Y = f(x)$$

*is **one-way** if it is easy to compute y from x but "hard" to compute x from y*

Building block of most cryptographic protocols

And, the security of most protocols rely on their existence.

Unfortunately, not known to exist. This is true even if we assume $P \neq NP$.

15-853

Page 10

One-way functions: possible definition

1. $F(x)$ is polynomial time
2. $F^{-1}(x)$ is NP-hard

What is wrong with this definition?

15-853

Page 11

One-way functions: better definition

For most y no single PPT (probabilistic polynomial time) algorithm can compute x

Roughly: at most a fraction $1/|x|^k$ instances x are easy for any k and as $|x| \rightarrow \infty$

This definition can be used to make the probability of hitting an easy instance arbitrarily small.

15-853

Page 12

Some examples (conjectures)

Factoring:

$$x = (u,v)$$
$$y = f(u,v) = u*v$$

If u and v are prime it is hard to generate them from y .

Discrete Log: $y = g^x \text{ mod } p$

where p is prime and g is a "generator" (i.e., g^1, g^2, g^3, \dots generates all values $< p$).

DES with fixed message: $y = \text{DES}_x(m)$

This would assume a family of DES functions of increasing key size (for asymptotics)

15-853

Page 13

One-way functions in public-key protocols

$$y = \text{ciphertext} \quad m = \text{plaintext} \quad k = \text{public key}$$

Consider: $y = E_k(m)$ (i.e., $f = E_k$)

We know k and thus f

$E_k(m)$ needs to be easy

$E_k^{-1}(y)$ should be hard

Otherwise we could decrypt y .

But what about the intended recipient, who should be able to decrypt y ?

15-853

Page 14

One-way functions in private-key protocols

$$y = \text{ciphertext} \quad m = \text{plaintext} \quad k = \text{key}$$

Is

$$y = E_k(m) \quad (\text{i.e. } f = E_k)$$

a one-way function with respect to y and m ?

What do one-way functions have to do with private-key protocols?

15-853

Page 15

One-way functions in private-key protocols

$$y = \text{ciphertext} \quad m = \text{plaintext} \quad k = \text{key}$$

How about

$$y = E_k(m) = E(k,m) = E_m(k) \quad (\text{i.e. } f = E_m)$$

should this be a one-way function?

In a **known-plaintext attack** we know a (y,m) pair.

The m along with E defines f

$E_m(k)$ needs to be easy

$E_m^{-1}(y)$ should be hard

Otherwise we could extract the key k .

15-853

Page 16

One-Way Trapdoor Functions

A **one-way** function with a "trapdoor"

The **trapdoor** is a key that makes it easy to invert the function $y = f(x)$

Example: **RSA** (conjecture)

$$y = x^e \text{ mod } n$$

Where $n = pq$ (p, q, e are prime)

p or q or d (where $ed = 1 \text{ mod } (p-1)(q-1)$) can be used as trapdoors

In public-key algorithms

$f(x)$ = public key (*e.g.*, e and n in RSA)

Trapdoor = private key (*e.g.*, d in RSA)

One-way Hash Functions

$Y = h(x)$ where

- y is a fixed length independent of the size of x . In general this means h is not invertible since it is many to one.
- Calculating y from x is easy
- Calculating any x such that $y = h(x)$ give y is hard

Used in digital signatures and other protocols.

Cryptography Outline

Introduction: terminology, cryptanalysis, security

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

Protocols:

- digital signatures
- key exchange

Number Theory: groups, fields, ...

Private-Key Algorithms: Rijndael, DES

Public-Key Algorithms: Knapsack, RSA, El-Gamal, ...

Case Studies: Kerberos, Digital Cash

Protocols

Other protocols:

- Authentication
- Secret sharing
- Timestamping services
- Zero-knowledge proofs
- Blind-signatures
- Key-escrow
- Secure elections
- Digital cash

Implementation of the protocol is often the weakest point in a security system.

Protocols: Digital Signatures

Goals:

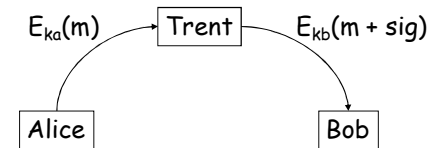
1. Convince recipient that message was actually sent by a trusted source
2. Do not allow repudiation, *i.e.*, that's not my signature.
3. Do not allow tampering with the message without invalidating the signature

Item 2 turns out to be really hard to do

15-853

Page 21

Using private keys

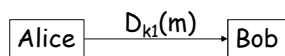


- **ka** is a secret key shared by Alice and Trent
 - **kb** is a secret key shared by Bob and Trent
- sig** is a note from Trent saying that Alice "signed" it.
To prevent repudiation Trent needs to keep m or at least $h(m)$ in a database

15-853

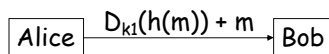
Page 22

Using Public Keys



K_1 = Alice's private key
Bob decrypts it with her public key

More Efficiently



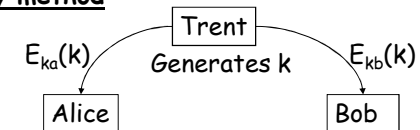
$h(m)$ is a one-way hash of m

15-853

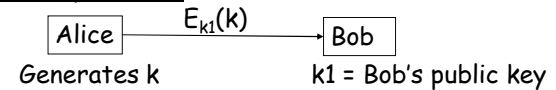
Page 23

Key Exchange

Private Key method



Public Key method



Or we can use a direct protocol, such as Diffie-Hellman (discussed later)

15-853

Page 24

Cryptography Outline

Introduction: terminology, cryptanalysis, security

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

Protocols: digital signatures, key exchange, ..

➔ **Number Theory Review:**

- Groups
- Fields
- Polynomials and Galois fields

Private-Key Algorithms: Rijndael, DES

Public-Key Algorithms: Knapsack, RSA, El-Gamal, ...

Case Studies: Kerberos, Digital Cash

15-853

Page 25

Number Theory Outline

Groups

- Definitions, Examples, Properties
- Multiplicative group modulo n
- The Euler-phi function

Fields

- Definition, Examples
- Polynomials
- Galois Fields

Why does number theory play such an important role?

It is **the** mathematics of finite sets of values.

15-853

Page 26

Groups

A **Group** $(G, *, I)$ is a set G with operator $*$ such that:

1. **Closure.** For all $a, b \in G$, $a * b \in G$
2. **Associativity.** For all $a, b, c \in G$, $a * (b * c) = (a * b) * c$
3. **Identity.** There exists $I \in G$, such that for all $a \in G$, $a * I = I * a = a$
4. **Inverse.** For every $a \in G$, there exist a unique element $b \in G$, such that $a * b = b * a = I$

An **Abelian or Commutative Group** is a Group with the additional condition

5. **Commutativity.** For all $a, b \in G$, $a * b = b * a$

15-853

Page 27

Examples of groups

- Integers, Reals or Rationals with Addition
- The nonzero Reals or Rationals with Multiplication
- Non-singular $n \times n$ real matrices with Matrix Multiplication
- Permutations over n elements with composition
[0→1, 1→2, 2→0] o [0→1, 1→0, 2→2] = [0→0, 1→2, 2→1]

We will only be concerned with **finite groups**, I.e., ones with a finite number of elements.

15-853

Page 28

Key properties of finite groups

Notation: $a^j \equiv a * a * a * \dots * a$... j times

Theorem (Fermat's little): for any finite group $(G, *, I)$ and $g \in G$, $g^{|G|} = I$

Definition: the **order** of $g \in G$ is the smallest positive integer m such that $g^m = I$

Definition: a group G is **cyclic** if there is a $g \in G$ such that $\text{order}(g) = |G|$

Definition: an element $g \in G$ of order $|G|$ is called a **generator** or **primitive element** of G .

15-853

Page 29

Groups based on modular arithmetic

The group of positive integers modulo a prime p

$$\mathbb{Z}_p^* \equiv \{1, 2, 3, \dots, p-1\}$$

$*$ _p \equiv multiplication modulo p

Denoted as: $(\mathbb{Z}_p^*, *_{p})$

Required properties

1. Closure. Yes.
2. Associativity. Yes.
3. Identity. 1.
4. Inverse. Yes.

Example: $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$

$$1^{-1} = 1, 2^{-1} = 4, 3^{-1} = 5, 6^{-1} = 6$$

15-853

Page 30

Other properties

$$|\mathbb{Z}_p^*| = (p-1)$$

By Fermat's little theorem: $a^{(p-1)} = 1 \pmod{p}$

Example of \mathbb{Z}_7^*

| x | x^2 | x^3 | x^4 | x^5 | x^6 |
|----------|-------|-------|-------|-------|-------|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 1 | 2 | 4 | 1 |
| <u>3</u> | 2 | 6 | 4 | 5 | 1 |
| 4 | 2 | 1 | 4 | 2 | 1 |
| <u>5</u> | 4 | 6 | 2 | 3 | 1 |
| 6 | 1 | 6 | 1 | 6 | 1 |

Generators $\left\{ \begin{array}{l} 3 \\ 5 \end{array} \right.$

For all p the group is cyclic.

15-853

Page 31

What if n is not a prime?

The group of positive integers modulo a non-prime n

$$\mathbb{Z}_n \equiv \{1, 2, 3, \dots, n-1\}, n \text{ not prime}$$

$*$ _p \equiv multiplication modulo n

Required properties?

1. Closure. ?
2. Associativity. ?
3. Identity. ?
4. Inverse. ?

How do we fix this?

15-853

Page 32

Groups based on modular arithmetic

The **multiplicative group modulo n**

$$Z_n^* \equiv \{m : 1 \leq m < n, \gcd(n,m) = 1\}$$

* \equiv multiplication modulo n

Denoted as $(Z_n^*, *_n)$

Required properties:

- Closure. Yes.
- Associativity. Yes.
- Identity. 1.
- Inverse. Yes.

Example: $Z_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$

$$1^{-1} = 1, 2^{-1} = 8, 4^{-1} = 4, 7^{-1} = 13, 11^{-1} = 11, 14^{-1} = 14$$

15-853

Page 33

The Euler Phi Function

(Also called the totient function)

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

If n is a product of two primes p and q, then

$$\phi(n) = pq(1 - 1/p)(1 - 1/q) = (p-1)(q-1)$$

Note that by Fermat's Little Theorem:

$$a^{\phi(n)} = 1 \pmod{n} \text{ for } a \in Z_n^*$$

Or for $n = pq$

$$a^{(p-1)(q-1)} = 1 \pmod{n} \text{ for } a \in Z_{pq}^*$$

This will be very important in RSA!

15-853

Page 34

Generators

Example of Z_{10}^* : $\{1, 3, 7, 9\}$

| x | x^2 | x^3 | x^4 |
|----------|-------|-------|-------|
| 1 | 1 | 1 | 1 |
| <u>3</u> | 9 | 7 | 1 |
| <u>7</u> | 9 | 3 | 1 |
| 9 | 1 | 9 | 1 |

Generators \rightarrow (3, 7)

For $n = (2, 4, p^e, 2p^e)$, p an odd prime, Z_n is cyclic

15-853

Page 35

Operations we will need

Multiplication: $a^*b \pmod{n}$

- Can be done in $O(\log^2 n)$ bit operations, or better

Power: $a^k \pmod{n}$

- The power method $O(\log n)$ steps, $O(\log^3 n)$ bit ops

```

fun pow(a, k) =
  if (k = 0) then 1
  else if (k mod 2 = 1)
  then a * (pow(a, k/2))2
  else (pow(a, k/2))2

```

Inverse: $a^{-1} \pmod{n}$

- Euclids algorithm $O(\log n)$ steps, $O(\log^3 n)$ bit ops

15-853

Page 36

Euclid's Algorithm

Euclid's Algorithm:

$\text{gcd}(a,b) = \text{gcd}(b, a \bmod b)$
 $\text{gcd}(a,0) = a$

Note: $\text{gcd}(a,b)$ is the smallest positive number of the form $ax + by$ for integers x,y

"Extended" Euclid's algorithm:

- Find x and y such that $ax + by = \text{gcd}(a,b)$
- Can be calculated as a side-effect of Euclid's algorithm.
- Note that x and y can be zero or negative.

This allows us to find $a^{-1} \bmod n$, for $a \in \mathbb{Z}_n^*$

In particular return x in $ax + ny = 1$.

15-853

Page 37

Euclid's Algorithm

```
fun euclid(a,b) =
  if (b = 0) then a
  else euclid(b, a mod b)

fun ext_euclid(a,b) =
  if (b = 0) then (a, 1, 0)
  else
    let (d, x, y) = ext_euclid(b, a mod b)
    in (d, y, x - (a/b) y)
    end
```

Diagram: Blue arrows point from the labels 'gcd' and 'y' to the 'gcd' parameter in the recursive call of `ext_euclid`. Another blue arrow points from the label 'x' to the 'x' parameter in the recursive call.

The code is in the form of an inductive proof.

Exercise: prove the inductive step

15-853

Page 38

Discrete Logarithms

If g is a generator of \mathbb{Z}_n^* , then for all y there is a unique $x \pmod{\phi(n)}$ such that

$$y = g^x \pmod n$$

This is called the **discrete logarithm** of y and we use the notation

$$x = \log_g(y)$$

In general finding the discrete logarithm is conjectured to be hard...as hard as factoring.

15-853

Page 39

Fields

A **Field** is a set of elements F with binary operators $*$ and $+$ such that

1. $(F, +)$ is an **abelian group**
2. $(F \setminus \{0\}, *)$ is an **abelian group** the "multiplicative group"
3. **Distribution:** $a*(b+c) = a*b + a*c$
4. **Cancellation:** $a*I_+ = I_+$

The **order** of a field is the number of elements.

A field of finite order is a **finite field**.

The reals and rationals with $+$ and $*$ are fields.

15-853

Page 40

Finite Fields

Z_p (p prime) with $+$ and $*$ mod p , is a **finite** field.

1. $(Z_p, +)$ is an **abelian group** (0 is identity)
2. $(Z_p \setminus 0, *)$ is an **abelian group** (1 is identity)
3. **Distribution:** $a*(b+c) = a*b + a*c$
4. **Cancellation:** $a*0 = 0$

Are there other finite fields?

What about ones that fit nicely into bits, bytes and words (i.e with 2^k elements)?

Polynomials over Z_p

$Z_p[x]$ = polynomials on x with coefficients in Z_p .

- Example of $Z_5[x]$: $f(x) = 3x^4 + 1x^3 + 4x^2 + 3$
- $\deg(f(x)) = 4$ (the **degree** of the polynomial)

Operations: (examples over $Z_5[x]$)

- Addition: $(x^3 + 4x^2 + 3) + (3x^2 + 1) = (x^3 + 2x^2 + 4)$
- Multiplication: $(x^3 + 3) * (3x^2 + 1) = 3x^5 + x^3 + 4x^2 + 3$
- $I_+ = 0, I_* = 1$
- $+$ and $*$ are associative and commutative
- Multiplication distributes and 0 cancels

Do these polynomials form a field?

Division and Modulus

Long division on polynomials ($Z_5[x]$):

$$\begin{array}{r}
 \boxed{1x+4} \\
 x^2+1 \overline{) x^3+4x^2+0x+3} \\
 \underline{x^3+0x^2+1x+0} \\
 4x^2+4x+3 \\
 \underline{4x^2+0x+4} \\
 \hline
 \end{array}$$

$(x^3 + 4x^2 + 3)/(x^2 + 1) = (x + 4)$

$(x^3 + 4x^2 + 3) \bmod (x^2 + 1) = (4x + 4)$

$(x^2 + 1)(x + 4) + (4x + 4) = (x^3 + 4x^2 + 3)$

Polynomials modulo Polynomials

How about making a field of polynomials modulo another polynomial? This is analogous to Z_p (i.e., integers modulo another integer).

e.g. $Z_5[x] \bmod (x^2+2x+1)$

Does this work?

Does $(x + 1)$ have an inverse?

Definition: An **irreducible polynomial** is one that is not a product of two other polynomials both of degree greater than 0.

e.g. $(x^2 + 2)$ for $Z_5[x]$

Analogous to a prime number.

Galois Fields

The polynomials

$$\mathbb{Z}_p[x] \bmod p(x)$$

where

$$p(x) \in \mathbb{Z}_p[x],$$

$p(x)$ is irreducible,

and $\deg(p(x)) = n$ (i.e. $n+1$ coefficients)

form a finite field. Such a field has p^n elements.

These fields are called **Galois Fields** or **GF(p^n)**.

The special case $n = 1$ reduces to the fields \mathbb{Z}_p

The multiplicative group of $GF(p^n)/\{0\}$ is cyclic (this will be important later).

15-853

Page 45

GF(2^n)

Hugely practical!

The coefficients are **bits** $\{0,1\}$.

For example, the elements of $GF(2^8)$ can be represented as a **byte**, one bit for each term, and $GF(2^{64})$ as a **64-bit word**.

- e.g., $x^6 + x^4 + x + 1 = 01010011$

How do we do addition?

Addition over \mathbb{Z}_2 corresponds to xor.

- Just take the xor of the bit-strings (bytes or words in practice). This is dirt cheap

15-853

Page 46

Multiplication over GF(2^n)

If n is small enough can use a table of all combinations.

The size will be $2^n \times 2^n$ (e.g. 64K for $GF(2^8)$).

Otherwise, use standard shift and add (xor)

Note: dividing through by the irreducible polynomial on an overflow by 1 term is simply a test and an xor.

e.g. $0111 / 1001 = 0111$

$1011 / 1001 = 1011 \text{ xor } 1001 = 0010$

^ just look at this bit for $GF(2^3)$

15-853

Page 47

Multiplication over GF(2^8)

```
typedef unsigned char uc;
```

```
uc mult(uc a, uc b) {  
    int p = a;  
    uc r = 0;  
    while(b) {  
        if (b & 1) r = r ^ p;  
        b = b >> 1;  
        p = p << 1;  
        if (p & 0x100) p = p ^ 0x11B;  
    }  
    return r;  
}
```

15-853

Page 48

Finding inverses over $GF(2^n)$

Again, if n is small just store in a table.

- Table size is just 2^n .

For larger n , use Euclid's algorithm.

- This is again easy to do with shift and xors.

Polynomials with coefficients in $GF(p^n)$

Recall that $GF(p^n)$ were defined in terms of coefficients that were themselves fields (i.e., Z_p).

We can apply this **recursively** and define:

$GF(p^n)[x]$ = polynomials on x with coefficients in $GF(p^n)$.

- Example of $GF(2^3)[x]$: $f(x) = 001x^2 + 101x + 010$
Where 101 is shorthand for x^2+1 .

Polynomials with coefficients in $GF(p^n)$

We can make a finite field by using an irreducible polynomial $M(x)$ selected from $GF(p^n)[x]$.

For an order m polynomial and by abuse of notation we write: $GF(GF(p^n)^m)$, which has p^{nm} elements.

Used in **Reed-Solomon codes** and **Rijndael**.

- In Rijndael $p=2$, $n=8$, $m=4$, i.e. each coefficient is a byte, and each element is a 4 byte word (32 bits).

Note: all finite fields are isomorphic to $GF(p^n)$, so this is really just another representation of $GF(2^{32})$.

This representation, however, has practical advantages.

Cryptography Outline

Introduction: terminology, cryptanalysis, security

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

Protocols: digital signatures, key exchange, ...

Number Theory: groups, fields, ...

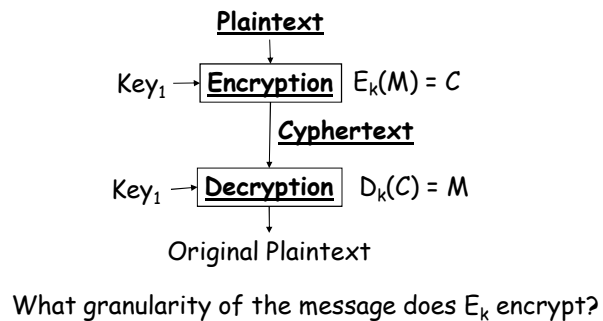
➔ **Private-Key Algorithms:**

- Block ciphers and product ciphers
- Rijndael, DES
- Cryptanalysis

Public-Key Algorithms: Knapsack, RSA, El-Gamal, ...

Case Studies: Kerberos, Digital Cash

Private Key Algorithms



15-853

Page 53

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, ...

15-853

Page 54

Private Key: Block Ciphers

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

$$c_i = f(k, m_i) \quad m_i = f'(k, c_i)$$

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

15-853

Page 55

Security of block ciphers

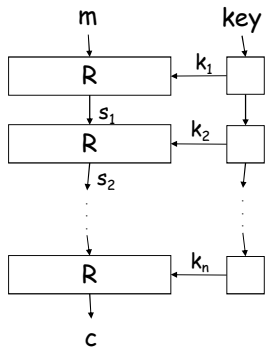
Ideal:

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

15-853

Page 56

Iterated Block Ciphers



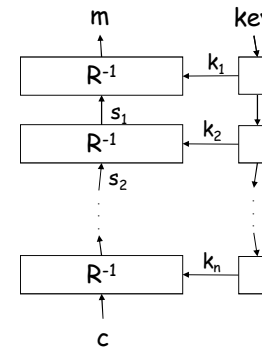
Consists of n **rounds**

R = the "round" function
 s_i = state after round i
 k_i = the i^{th} round key

15-853

Page 57

Iterated Block Ciphers: Decryption



Run the rounds in reverse.

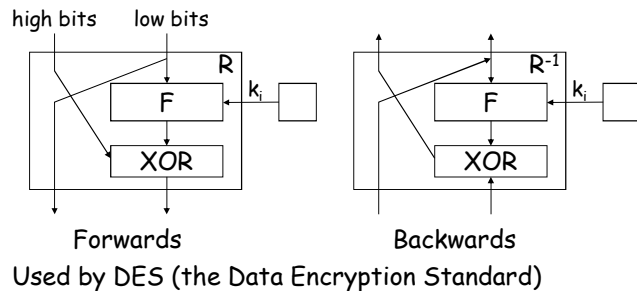
Requires that R has an inverse.

15-853

Page 58

Feistel Networks

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



15-853

Page 59

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

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

15-853

Page 60

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 Counterpane, Serpent, and Rijndael
- Rijndael selected Oct. 2000.
- Official Oct. 2001? ([AES page on Rijndael](#))

Designed by Rijmen and Daemen (Dutch)

15-853

Page 61

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

15-853

Page 62

High-level overview

An iterated block cipher with

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

Mathematically reasonably sophisticated

15-853

Page 63

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.

15-853

Page 64

Galois Fields in Rijndael

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 $GF(2^8)$.

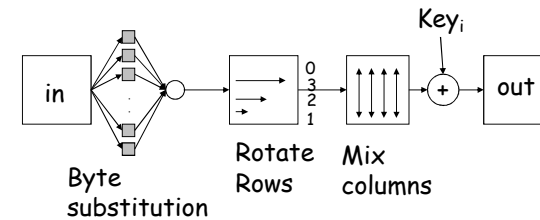
These are kept as 4 bytes (used for the columns)

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)

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

To invert the substitution:

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

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

Mix Columns

For each column a in data block

$$\begin{matrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{matrix}$$

$$\text{compute } b(x) = (a_3x^3 + a_2x^2 + a_1x + a_0)(3x^3 + x^2 + x + 2) \text{ mod } x^4 + 1$$

where coefficients are taken over $GF(2^8)$.

$$\text{New column } b \text{ is } \begin{matrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{matrix} \quad \text{where } b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

Implementation

Using $x^j \bmod (x^4 + 1) = x^{(j \bmod 4)}$

$$(a_3x^3 + a_2x^2 + a_1x + a_0)(3x^3 + x^2 + x + 2) \bmod x^4 + 1$$

$$= (2a_0 + 3a_1 + a_2 + a_3) + (a_0 + 2a_1 + 3a_2 + a_3)x + (a_0 + a_1 + 2a_2 + 3a_3)x^2 + (3a_0 + a_1 + a_2 + 2a_3)x^3$$

Therefore, $b = C \cdot a$

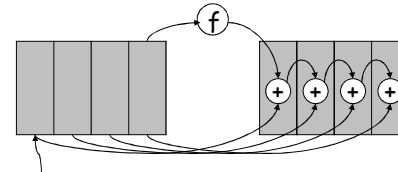
$$C = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

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

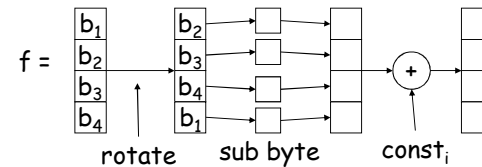
15-853

Page 69

Generating the round keys



Words corresponding to columns of the key



15-853

Page 70

Performance

Performance: (64-bit AMD Athlon 2.2Ghz, 2005, Open SSL):

| Algorithm | Bits/key | Mbits/sec |
|--------------|----------|-----------|
| DES-cbc | 56 | 399 |
| Blowfish-cbc | 128 | 703 |
| Rijndael-cbc | 128 | 917 |

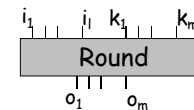
Hardware implementations go up to 32 Gbits/sec

15-853

Page 71

Linear Cryptanalysis

A known plaintext attack used to extract the key



Consider a linear equality involving i , o , and k

- 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

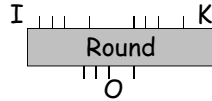
If true with $p = .5 + \epsilon$ then you might be able to use this to help break the system

15-853

Page 72

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.