15-853 Algorithms in the Real World

Cryptography 1 and 2

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

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
More Definitions

Private Key or Symmetric: Key₁ = Key₂
Public Key or Asymmetric: Key₁ ≠ Key₂
Key₁ or Key₂ is public depending on the protocol

More Definitions

Plaintext
Key₁ → Encryption → Encryption
Cyphertext
Key₂ → Decryption → Original Plaintext

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.

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.

The Cast

Alice - initiates a message or protocol
Bob - second participant
Trent - trusted middleman
Eve - eavesdropper
Mallory - malicious active attacker
**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

---

**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 \( P \neq NP \).

---

**One-way functions: possible definition**

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

What is wrong with this definition?

---

**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.
Some examples (conjectures)

**Factoring:**
\[ x = (u,v) \]
\[ y = f(u,v) = u \cdot v \]
If \( u \) and \( v \) are prime it is hard to generate them from \( y \).

**Discrete Log:**
\[ y = g^x \mod p \]
where \( p \) is prime and \( g \) is a "generator" (i.e., \( g^1, g^2, g^3, \ldots \) 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).

---

**One-way functions in public-key protocols**

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

Consider:
\[ y = E_k(m) \quad (i.e., f = E_k) \]
We know \( k \) and thus \( f \)
\[ E_k(m) \text{ needs to be easy} \]
\[ E_k^{-1}(y) \text{ should be hard} \]
Otherwise we could decrypt \( y \).
But what about the intended recipient, who should be able to decrypt \( y \)?

---

**One-way functions in private-key protocols**

\[ y = \text{ciphertext} \quad m = \text{plaintext} \quad k = \text{key} \]
Is
\[ y = E_k(m) \quad (i.e. f = E_k) \]
a one-way function with respect to \( y \) and \( m \)?

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

---

**One-way functions in private-key protocols**

\[ y = \text{ciphertext} \quad m = \text{plaintext} \quad k = \text{key} \]
How about
\[ y = E_k(m) = E(k,m) = E_m(k) \quad (i.e. f = E_m) \]
should this be a one-way function?

In a known-plaintext attack we know a \((y,m)\) pair.
The \( m \) along with \( E \) defines \( f \)
\[ E_m(k) \text{ needs to be easy} \]
\[ E_m^{-1}(y) \text{ should be hard} \]
Otherwise we could extract the key \( k \).
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)
\[ y = x^e \mod n \]
Where \( n = pq \) (p, q, e are prime)
p or q or d (where \( ed = 1 \mod (p-1)(q-1) \)) 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.

---

**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.

---

**Using Public Keys**

\[ D_{k1}(m) \rightarrow \text{Bob} \]

- K1 = Alice’s private key.
- Bob decrypts it with her public key.

More Efficiently:

\[ D_{k1}(h(m)) + m \rightarrow \text{Bob} \]

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

---

**Key Exchange**

**Private Key method**

\[ E_{k1}(k) \rightarrow \text{Bob} \]

Trent generates k.

**Public Key method**

\[ E_{k1}(k) \rightarrow \text{Bob} \]

Trent generates k.

k1 = Bob’s public key.

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

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.

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 \(a, b \in G\), \(a * b = b * a\)

Examples of groups

- Integers, Reals or Rationals with Addition
- The nonzero Reals or Rationals with Multiplication
- Non-singular n x n real matrices with Matrix Multiplication
- Permutations over n elements with composition \([0 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 0] \circ [0 \rightarrow 1, 1 \rightarrow 0, 2 \rightarrow 2] = [0 \rightarrow 0, 1 \rightarrow 2, 2 \rightarrow 1]\)

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

**Notation:** \( a^j \equiv a \ast a \ast a \ast \ldots \ast \) \( j \) times

**Theorem (Fermat's little):** for any finite group \((G, \ast, I)\) 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 \( G \).

---

Groups based on modular arithmetic

The group of positive integers modulo a prime \( p \)

\( Z_p^* = \{1, 2, 3, \ldots, p-1\} \)

\( \ast_p = \text{multiplication modulo } p \)

Denoted as: \((Z_p^*, \ast_p)\)

**Required properties**
3. Identity. 1.
4. Inverse. Yes.

**Example:** \( Z_7^* = \{1,2,3,4,5,6\} \)

\( 1^1 = 1, 2^1 = 4, 3^1 = 5, 6^1 = 6 \)

---

Other properties

\(|Z_p^*| = (p-1)| \)

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

**Example of \( Z_7^* \)**

<table>
<thead>
<tr>
<th></th>
<th>( x )</th>
<th>( x^2 )</th>
<th>( x^3 )</th>
<th>( x^4 )</th>
<th>( x^5 )</th>
<th>( x^6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

**Generators**

For all \( p \) the group is cyclic.

---

What if \( n \) is not a prime?

The group of positive integers modulo a non-prime \( n \)

\( Z_n = \{1, 2, 3, \ldots, n-1\} \), \( n \) not prime

\( \ast_p = \text{multiplication modulo } n \)

**Required properties?**
1. Closure. ?
2. Associativity. ?
3. Identity. ?
4. Inverse. ?

How do we fix this?
Groups based on modular arithmetic

The multiplicative group modulo n
\[ \mathbb{Z}_n^* = \{ m : 1 \leq m < n, \gcd(n, m) = 1 \} \]
* is multiplication modulo n
Denoted as \((\mathbb{Z}_n^*, \cdot)\)

Required properties:
- Closure. Yes.
- Associativity. Yes.
- Identity. 1.
- Inverse. Yes.

Example: \(\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}\)
\[ 1^1 = 1, 2^1 = 2, 4^1 = 4, 7^1 = 7, 11^1 = 11, 13^1 = 13, 14^1 = 14 \]

The Euler Phi Function
(Also called the totient function)

\[ \phi(n) = |\mathbb{Z}_n^*| = n \prod_{p \mid n} (1 - 1/p) \]

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:
\[ a^{\phi(n)} = 1 \pmod{n} \text{ for } a \in \mathbb{Z}_n^* \]

Or for \(n = pq\)
\[ a^{(p-1)(q-1)} = 1 \pmod{n} \text{ for } a \in \mathbb{Z}_{pq}^* \]
This will be very important in RSA!

Generators

Example of \(\mathbb{Z}_{15}^*\): \(\{1, 3, 7, 9\}\)

<table>
<thead>
<tr>
<th>(x)</th>
<th>(x^2)</th>
<th>(x^3)</th>
<th>(x^4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3/2</td>
<td>9/4</td>
<td>7/4</td>
<td>1/4</td>
</tr>
<tr>
<td>7/2</td>
<td>9/4</td>
<td>3/4</td>
<td>1/4</td>
</tr>
<tr>
<td>9/2</td>
<td>1/4</td>
<td>9/4</td>
<td>1/4</td>
</tr>
</tbody>
</table>

For \(n = (2, 4, p^e, 2p^e), p\) an odd prime, \(\mathbb{Z}_n\) is cyclic

Operations we will need

Multiplication: \(a \cdot b \pmod{n}\)
- 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: \(a^{-1} \pmod{n}\)
- Euclid’s algorithm \(O(\log n)\) steps, \(O(\log^3 n)\) bit ops
Euclid's Algorithm

Euclid's Algorithm:
\[ \text{gcd}(a, b) = \text{gcd}(b, a \mod b) \]
\[ \text{gcd}(a, 0) = a \]

"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 \( a^{-1} \mod n \), for \( a \in \mathbb{Z}_n^* \)
In particular return \( x \) in \( ax + ny = 1 \).

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

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

The code is in the form of an inductive proof.

Exercise: prove the inductive step

Discrete Logarithms

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

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.

Fields

A Field is a set of elements \( F \) with binary operators \( \times \) and + such that
1. \((F, +)\) is an abelian group
2. \((F \setminus I, \times)\) 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.
Finite Fields

$\mathbb{Z}_p$ (p prime) with $+$ and $\ast$ mod $p$, is a finite field.

1. $(\mathbb{Z}_p, +)$ is an abelian group (0 is identity)
2. $(\mathbb{Z}_p \setminus 0, \ast)$ is an abelian group (1 is identity)
3. Distribution: $a \ast (b+c) = a \ast b + a \ast c$
4. Cancellation: $a \ast 0 = 0$

Are there other finite fields?

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

Polynomials over $\mathbb{Z}_p$

$\mathbb{Z}_p[x] = \text{polynomials on } x \text{ with coefficients in } \mathbb{Z}_p$.

- Example of $\mathbb{Z}_5[x]$: $f(x) = 3x^4 + 1x^3 + 4x^2 + 3$
- $\deg(f(x)) = 4$ (the degree of the polynomial)

Operations: (examples over $\mathbb{Z}_5[x]$)

- Addition: $(x^3 + 4x^2 + 3) + (3x^2 + 1) = (x^3 + 2x^2 + 4)$
- Multiplication: $(x^3 + 3) \ast (3x^2 + 1) = 3x^5 + x^3 + 4x^2 + 3$

$1, 0 = 0, 1 = 1$

$+$ and $\ast$ are associative and commutative

$\ast$ distributes and $0$ cancels
Do these polynomials form a field?

Division and Modulus

Long division on polynomials ($\mathbb{Z}_5[x]$):

\[
\begin{array}{c|ccccc}
  & x^3 + 4x^2 + 0x + 3 \\
\hline
x^2 + 1 & x^3 + 4x^2 + 0x + 3 \\
  & x^3 + 0x^2 + 1x + 0 \\
  & 4x^2 + 4x + 3 \\
  & 4x^2 + 0x + 4 \\
\end{array}
\]

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

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

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

Polynomials modulo Polynomials

How about making a field of polynomials modulo another polynomial? This is analogous to $\mathbb{Z}_p$ (i.e., integers modulo another integer).

e.g. $\mathbb{Z}_5[x] \bmod (x^2+2x+1)$

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 $\mathbb{Z}_5[x]$

Analogous to a prime number.
Galois Fields

The polynomials
\[ Z_p[x] \mod p(x) \]
where
- \( p(x) \in Z_p[x] \),
- \( p(x) \) 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 \( Z_p \).
The multiplicative group of \( GF(p^n) / \{0\} \) is cyclic (this will be important later).

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 \( Z_2 \) corresponds to xor.
- Just take the xor of the bit-strings (bytes or words in practice). This is dirt cheap

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. \( 0111 / 1001 = 0111 \)
  \( 1011 / 1001 = 1011 \ xor \ 1001 = 0010 \)
  ^ just look at this bit for \( GF(2^3) \)

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;
}
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:

$GF(p^n)[x] = \text{polynomials on } x \text{ with coefficients in } GF(p^n)$.
- 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**

<table>
<thead>
<tr>
<th>Plaintext</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key₁ → Encryption</td>
</tr>
<tr>
<td>$E_k(M) = C$</td>
</tr>
<tr>
<td>CipherText</td>
</tr>
<tr>
<td>Key₁ → Decryption</td>
</tr>
<tr>
<td>$D_k(C) = M$</td>
</tr>
<tr>
<td>Original Plaintext</td>
</tr>
</tbody>
</table>

What granularity of the message does $E_k$ encrypt?

**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

**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, ...

**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.
Iterated Block Ciphers

Consists of \( n \) rounds

\[ R = \text{the "round" function} \]

\[ s_i = \text{state after round } i \]

\[ k_i = \text{the } i\text{th round key} \]

Iterated Block Ciphers: Decryption

Run the rounds in reverse.

Requires that \( R \) has an inverse.

Feistel Networks

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

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

Used by DES (the Data Encryption Standard)
Rijndael

Selected by AES (Advanced Encryption Standard, part of NIST) as the new private-key encryption standard.
Based on an open "competition".
- Narrowed to 5 Sept. 1999
  • MARS by IBM, RC6 by RSA, Twofish by Counterplane, Serpent, and Rijndael
- Official Oct. 2001? (AES page on Rijndael)
Designed by Rijmen and Daemen (Dutch)

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

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.

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

Data block   Key

\(b_0, b_1, \ldots, b_{15}\) is the order of the bytes in the stream.
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)
$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$
$$B = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

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
$$a_0 \quad a_1 \quad a_2 \quad a_3$$
compute $b(x) = (a_3x^3 + a_2x^2 + a_1x + a_0)(3x^3 + x^2 + x + 2) \mod x^4 + 1$
where coefficients are taken over $GF(2^8)$.
New column $b$ is
$$b_0 \quad b_1 \quad b_2 \quad b_3$$
where $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$
Implementation

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

$(a_3 x^3 + a_2 x^2 + a_1 x + a_0) (3x^3 + x^2 + x + 2) \mod x^4 + 1$

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

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

Therefore, $b = C \cdot a$

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

Generating the round keys

Words corresponding to columns of the key

$b_1$ $b_2$ $b_3$ $b_4$

rotate sub byte const

Performance

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Bits/key</th>
<th>Mbits/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES-cbc</td>
<td>56</td>
<td>399</td>
</tr>
<tr>
<td>Blowfish-cbc</td>
<td>128</td>
<td>703</td>
</tr>
<tr>
<td>Rijndael-cbc</td>
<td>128</td>
<td>917</td>
</tr>
</tbody>
</table>

Hardware implementations go up to 32 Gbits/sec

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
Differential Cryptanalysis

A chosen plaintext attack used to extract the key

\[ \Delta I = I_1 - I_2 \]

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.