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

\[
\begin{align*}
\text{Plaintext} & \rightarrow \text{Encryption} & E_k(M) = C \\
\text{Cyphertext} & \rightarrow \text{Decryption} & D_k(C) = M
\end{align*}
\]

- Private Key or Symmetric: $K_1 = K_2$
- Public Key or Asymmetric: $K_1 \neq K_2$

Encryption ($K_1$, $M$) = $C$ (cyphertext)

Decryption ($K_2$, $C$) = $M$ (plaintext)

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

Diagram:

- Alice
- Eve
- Bob
- Trent
- Mallory
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

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 \( |x| \to \infty \)

This definition can be used to make the probability of hitting an easy instance arbitrarily small.
Some examples (conjectures)

Factoring:
\[ x = (u,v) \]
\[ y = f(u,v) = u^v \]
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., \( g^1, g^2, g^3, \ldots \) generates all values < \( p \)).

DES with fixed message: \( y = \text{DES}_k(m) \)
This would assume a family of DES functions of increasing key size

One-way functions in private-key protocols

\[ y = \text{ciphertext} \]
\[ m = \text{plaintext} \]
\[ x = \text{key} \]
\[ y = f(x) = E_k(m) \]

In a known-plaintext attack we know a \((y,m)\) pair.
The \( m \) along with \( E \) defines \( f(x) \)
\( f(x) \) needs to be easy
\( f^{-1}(y) \) should be hard
Otherwise we could extract the key \( x \).

One-way functions in public-key protocols

\[ y = \text{ciphertext} \]
\[ x = \text{plaintext} \]
\[ k = \text{public key} \]
\[ y = f(x) = E_k(x) \]
We know \( k \) and thus \( f(x) \)
\( f(x) \) needs to be easy
\( f^{-1}(y) \) should be hard
Otherwise we could decrypt \( y \).
But what about the intended recipient, who should be able to decrypt \( y \)?

Note the change of role of the key and plaintext from the previous example

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 \( ed = (p-1)(q-1) \text{ mod } n \)) can be used as trapdoors
In public-key algorithms
\( f(x) = \text{public key} \) (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

\[ E_{K_a}(m) \rightarrow Trent \rightarrow E_{K_b}(m + \text{sig}) \]

- \( K_a \) is a secret key shared by Alice and Trent
- \( K_b \) is a secret key shared by Bob and Trent
- \text{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

\[ Alice \rightarrow D_{K_b}(m) \rightarrow Bob \]

- \( K_1 = Alice's \) private key
- Bob decrypts it with her public key

More Efficiently

\[ Alice \rightarrow D_{K_b}(h(m)) + m \rightarrow Bob \]

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

Key Exchange

Private Key method

\[ E_{K_a}(k) \rightarrow Trent \rightarrow E_{K_b}(k) \]

\[ Alice \rightarrow Trent \rightarrow E_{K_b}(k) \]

Public Key method

\[ Alice \rightarrow E_{K_b}(k) \rightarrow Bob \]

- Alice generates \( k \)
- Bob generates \( E_{K_b}(k) \)
- \( k = Bob's \) 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 \((G, \cdot, I)\) is a set \(G\) with operator \(\cdot\) such that:
1. Closure. For all \(a, b \in G\), \(a \cdot b \in G\)
2. Associativity. For all \(a, b, c \in G\), \(a \cdot (b \cdot c) = (a \cdot b) \cdot c\)
3. Identity. There exists \(I \in G\), such that for all \(a \in G\), \(a \cdot I = I \cdot a = a\)
4. Inverse. For every \(a \in G\), there exist a unique element \(b \in G\), such that \(a \cdot b = b \cdot a = I\)

An Abelian or Commutative Group is a Group with the additional condition
5. Commutativity. For all \(a, b \in G\), \(a \cdot b = b \cdot a\)

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
\([0 \to 1, 1 \to 2, 2 \to 0]\) 0 \([0 \to 1, 1 \to 0, 2 \to 2] = [0 \to 0, 1 \to 2, 2 \to 1]\)

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

Key properties of finite groups
Notation: \(a^j = a \cdot a \cdot a \cdot \ldots \cdot j\) times

Theorem (Fermat's little): for any finite group \((G, \cdot, I)\) 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 $p$
$Z_p^* = \{1, 2, 3, ..., p-1\}$
$*_p$ = multiplication modulo $p$
Denoted as: $(Z_p^*, *_p)$

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

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

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>$x$</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>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</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 $n$
$Z_n^* = \{1, 2, 3, ..., n-1\}$, $n$ not prime
$*_{n}$ = 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 $n$
$Z_n^* = \{m : 1 \leq m < n, \gcd(n,m) = 1\}$
$*_{n}$ = multiplication modulo $n$
Denoted as $(Z_n^*, *_{n})$

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

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

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

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

Or for \( n = pq \)
\[ a^{(p-1)(q-1)} \equiv 1 \pmod{n} \quad \text{for} \quad a \in \mathbb{Z}_{pq} \]

This will be very important in RSA!

---

Generators

Example of \( \mathbb{Z}_{10}^* \): \{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</td>
<td>9</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>9</td>
<td>1</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:
\[ \gcd(a,b) = \gcd(b,a \mod b) \]
\[ \gcd(a,0) = a \]

"Extended" Euclid's algorithm:
- Find \( x \) and \( y \) such that \( ax + by = \gcd(a,b) \)
- 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 \( a^{-1} \pmod{n} \), for \( a \in \mathbb{Z}_n^* \)
In particular return \( x \) in \( ax + ny = 1 \).
Euclid's Algorithm

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

fun ext_euclid(a, b) =
  gcd
  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 \( Z_n^* \), 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.

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.

Finite Fields

\( Z_p \) (p prime) with + and \( * \pmod{p} \) 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 \( 2^n \) elements)?
Polynomials over $\mathbb{Z}_p$

$\mathbb{Z}_p[x]$ = polynomials on $x$ with coefficients in $\mathbb{Z}_p$.
- Example of $\mathbb{Z}_3[x]$: $f(x) = 3x^4 + 1x^3 + 4x^2 + 3$
- $\deg(f(x)) = 4$ (the degree of the polynomial)
Operations: (examples over $\mathbb{Z}_3[x]$)
  - Addition: $(x^3 + 4x^2 + 3) + (3x^2 + 1) = (x^3 + 2x^2 + 4)$
  - Multiplication: $(x^3 + 3) \cdot (3x^2 + 1) = 3x^5 + x^3 + 4x^2 + 3$
  - $0_3 = 0, 1_3 = 1$
  - $+ \text{ and } \cdot \text{ are associative and commutative}$
  - Multiplication distributes and $0$ cancels
Do these polynomials form a field?

Division and Modulus

Long division on polynomials ($\mathbb{Z}_3[x]$):

\[
\begin{array}{cc}
\text{Dividend: } x^3 + 4x^2 + 0x + 3 \\
\text{Divisor: } x^2 + 1 \\
\text{Quotient: } 1x + 4 \\
\end{array}
\]

\[
\begin{array}{cccccc}
& x^2 + 0x^1 + 1x^0 & \\
\times & x^2 + 4x^1 + 3x^0 & \\
\hline
x^4 & 4x^3 & + & 0x^2 & + & 4x^1 & + & 3x^0 & \\
\hline
\end{array}
\]

\[(x^3 + 4x^2 + 3)/(x^2 + 1) = (x + 4)\]

\[(x^3 + 4x^2 + 3) \mod (x^2 + 1) = (4x + 4)\]

\[(x^2 + 1)(x + 4) + (4x + 4) = (x^3 + 4x^2 + 3)\]

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).

\[e.g. \, Z_3[x] \mod (x^2+2x+1)\]

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.

\[e.g. \, (x^2 + 2) \text{ for } Z_3[x]\]

Analogous to a prime number.

Galois Fields

The polynomials $\mathbb{Z}_p[x] \mod p(x)$
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 $\mathbb{GF}(p^n)$.
The special case $n = 1$ reduces to the fields $\mathbb{Z}_p$.
The multiplicative group of $\mathbb{GF}(p^n)/(0)$ is cyclic (this will be important later).
**GF($2^n$)**

Hugely practical!
The coefficients are bits $\{0,1\}$.
For example, the elements of GF($2^n$) 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

---

**Multiplication over GF($2^n$)**

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 $\mathbb{GF}(p^n)$

Recall that $\mathbb{GF}(p^n)$ were defined in terms of coefficients that were themselves fields (i.e., $\mathbb{Z}_p$). We can apply this recursively and define:

$\mathbb{GF}(p^n)[x] = \text{polynomials on } x \text{ with coefficients in } \mathbb{GF}(p^n)$.
- Example of $\mathbb{GF}(2^3)[x]$: $f(x) = 001x^2 + 101x + 010$ Where 101 is shorthand for $x^2 + 1$.

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

Plaintext
\[
\text{Key}_1 \rightarrow \boxed{\text{Encryption}} \quad E_k(M) = C
\]
\[
\text{Ciphertext}
\]
\[
\text{Key}_1 \rightarrow \boxed{\text{Decryption}} \quad D_k(C) = M
\]
Original Plaintext
What granularity of the message does $E_k$ encrypt?

Polynomials with coefficients in $\mathbb{GF}(p^n)$

We can make a finite field by using an irreducible polynomial $M(x)$ selected from $\mathbb{GF}(p^n)[x]$.
For an order $m$ polynomial and by abuse of notation we write: $\mathbb{GF}(\mathbb{GF}(p^n)^m)$, which has $p^m$ 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 $\mathbb{GF}(p^n)$, so this is really just another representation of $\mathbb{GF}(2^{25})$.
This representation, however, has practical advantages.
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, ...

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

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
- \( R \) = the "round" function
- \( s_i \) = state after round \( i \)
- \( k_i \) = the \( i^{th} \) round key
**Iterated Block Ciphers: Decryption**

Run the rounds in reverse. Requires that R has an inverse.

\[ m \rightarrow R^{-1} \rightarrow k_1 \rightarrow s_1 \rightarrow k_2 \rightarrow s_2 \rightarrow k_n \rightarrow c \]

**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

**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_{10} & k_{14} \\
  k_3 & k_7 & k_{11} & k_{15}
\end{pmatrix}
\]

Data block Key

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

Galois Fields in Rijndael

Uses \( GF(2^8) \) over bytes.
The irreducible polynomial is:
\( M(x) = x^8 + x^4 + x^3 + x + 1 \) or 100011011 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:
\( M(x) = 0000001x^4 + 00000001 \) or \( x^4 + 1 \)
Not irreducible, but we only need to find inverses of polynomials that are relatively prime to it.
Each round

![Diagram](image)

Key

Byte substitution

Rotate Rows

Mix columns

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)

\[
A = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
\end{pmatrix}
\]

\[
B = \begin{pmatrix}
1 \\
0 \\
0 \\
1 \\
\end{pmatrix}
\]

To invert the substitution:

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

\( b = y^1 \) (over \( GF(2^8) \))

Mix Columns

For each column \( a \) in data block

\( a_0 \)

\( a_1 \)

\( a_2 \)

\( a_3 \)

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

\( b_0 \)

\( b_1 \)

\( b_2 \)

\( b_3 \)

where \( b(x) = b_3x^3 + b_2x^2 + b_1x + b_0 \)

Implementation

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

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

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

\( C = \)

\[
\begin{pmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2 \\
\end{pmatrix}
\]

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

\[ f = \begin{array}{c}
\begin{array}{c}
\text{rotate} \\
\text{sub byte} \\
\text{const}_i
\end{array}
\end{array}\]

Performance

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

<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>95</td>
</tr>
<tr>
<td>twofish-cbc</td>
<td>128</td>
<td>140</td>
</tr>
<tr>
<td>Rijndael</td>
<td>128</td>
<td>180</td>
</tr>
</tbody>
</table>

Hardware implementations go up to 2.5 Gbits/sec

Linear Cryptanalysis

A known plaintext attack used to extract the key

Consider a linear equality involving i, o, and k
- e.g.: \[ k_i \oplus k_b = i_2 \oplus i_4 \oplus i_6 \oplus o_4 \]
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 \( p = .5 + \varepsilon \) then you might be able to use
this to help break the system

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.