Expectations

I hope...

Good marks!

8 credits

SPAM

how to handle

DDOS

Avoiding
- SQL injection
- XSS attacks
- Denial of Service
- Port or Vulnerable Services

Authentification

Secure communication over insecure networks

Secure message exchange over insecure networks

learn something about information security, it is valuable in practice

details of SSL, TLS, VRR...

Firewall

SAP BW

Security on internet

Encryption

not this:

enjoy the nice building while it's still possible 😊
Internet

Technical part

- network of networks
- client-server/peer-to-peer
- types of data transmission:
  - duplex
  - point-to-point
  - multicast
  - broadcast
- media of transmission:
  - wireless
  - wired
- Routing (Software & Hardware)
  - static / dynamic
- Hardware involved:
  - network card (modem) (client)
  - switches, routers, card (client & provider)
- Software involved:
  - browser (client)
  - email-client/VPN-client...

Social Part

- Information (to get and to provide)
- communication (email, chatting)
- entertainment
- services:
  - internet marketing
  - educational
- open to everyone
- provide social groups (youtube, orkut)
- misuse of information, network
  - piracy
  - propaganda
  - anonymity
  - addiction

Virtual Reality

World Wide Web

protocol

standards
Security

Authentication
- Trust
- Certificates
  - Identity theft
  - Passwords

Authorization
- Access control
- Illegal use

Reliability

Integrity
- Hashing
- Signing
- Hash-codes

Privacy
- Eavesdrop
- Encryption
  - Sym
  - Asym

Patch Day

Firewall
Email

Goal
- send unreadable message, text-only
- fast
- reliable (msgs should arrive, at least in most cases), available
- cheap

+ multiple destinations
- asynchronous
- no acknowledgement

Format
- split into:
  - From <address> <date> ...
  - To: <address>
  - Subject: <subject>
  - Sender: <sender>
  - Date: <date>
  - To: <to>
  - From: <from>
  - Subject: <subject>
  - Sender: <sender>
- text only

Technicalities
- receive any mail (process)
- relay/forward mails via various servers
- Address information must be included and non-encrypted
- DNS servers necessary to provide information about the topology of the network
- SMTP (Send Mail Transfer Protocol)

Security objectives
- Only the intended addressee/recipient gets the mail
- Make sure that the sender is who he claims to be
- Make sure nobody uses my address as sender address
- Protect content from disclosure
- Protect content from modification
**Attack**
- Flooding
- Flooded mail servers
- Send emails to blocked domains
- DNS poisoning
- Redirect requests to hijacked mail server
- DDoS
- Skilled actors

**Defend**
- New servers
- Distributed infrastructure
- Block Faked Local
- Grey listing
- Public key infrastructure
- Peer-to-Peer
- Key servers on all machines
- Low certificates in locked room
Security objectives

- Protect content from disclosure
- from modification
- Identify sender.
- Protect receiving host
- from attacks by incoming messages
- Mail list handling (many recipients)

Basics

- Address (and more) in the message (header)
- Text only
- Accept from anywhere
- No acknowledgement

Attacks

- on server — exploit vulnerabilities

\[\text{Solution! No ultimate one.}\]

\[\rightarrow \text{layered software, e.g. spam}\]

\[\text{email} \rightarrow \text{SMTP} \rightarrow \text{SMTP}\]

- on clients — hoaxes
  - e.g. "Good Times"
Here is some important information. Beware of a file called Goodtimes.

Happy Chanukah everyone, and be careful out there. There is a virus on America Online being sent by E-Mail. If you get anything called "Good Times", DON'T read it or download it. It is a virus that will erase your hard drive. Forward this to all your friends. It may help them a lot.
Technology

1. Encryption
   → confidentiality: protect from disclosure

2. Signature
   → identification: identity sender
   → integrity: protect from modification
   → authentication, unideniability (non-repudiation):
     link document & sender/signer

3. Public Key Infrastructure
Encryption

Ceser
Replace every letter in the plain text with its third successor.

VHQL VHGL YLFL
**enc**
VENI VEDI VICI

We have an alphabet

$$\Sigma = \{ A, B, C, \ldots, Z \}$$

and the possible Ceser ciphers are:

$$C_i : \Sigma \rightarrow \Sigma,$$

$$a \mapsto (a+i) \mod 26,$$

remains the same.

To decrypt without knowing the key it suffices to try all 26 keys.

Better attack: find most frequent character. This must correspond to the most frequent character of the plain text's language.

Better: find affine codes:

$$Aa + i \mod 26.$$
we have to care that decryption is possible. \[\text{CORRECTNESS}\]

For \( a = 1 \), we get a generalized Caesar \( C_i \), which is simple to decrypt (by \( C_i^{-1} \)). But \( a = 0 \) is very bad, any character be mapped to the same and no decryption is possible.

With \( a = 2 \) we always have

\[
A_{2, i} \cdot (0) = A_{2, i} \cdot (13)
\]

\[
(2 \cdot 0 + i) \text{rem} \ 26 = (2 \cdot 13 + i) \text{rem} \ 26
\]

Thus we cannot decrypt.

The mathematical structure we need here is the ring of integers modulo 26.

This is an \( \mathbb{Z}_{26} \) - class consisting of

\( a \) set of legal values: \( \mathbb{Z}_{26} = \{0, 1, 2, \ldots, 25\}, \)

two operations:

\[
+: \mathbb{Z}_{26} \times \mathbb{Z}_{26} \rightarrow \mathbb{Z}_{26},
\]

\[
(a, b) \mapsto (a + b) \text{ rem } 26.
\]

\[
i: \mathbb{Z}_{26} \times \mathbb{Z}_{26} \rightarrow \mathbb{Z}_{26},
\]

\[
(a, b) \mapsto (a \cdot b) \text{ rem } 26.
\]
Properly defined: a set and the operations are well defined.

$P^+, P^-$

Associativity:

$A + (B + C) = (A + B) + C$

$a \cdot (B \cdot C) = (a \cdot B) \cdot C$

Neutral elements:

$N^+, N^-$

$a + 0 = a = 0 + a$

$a \cdot 1 = a = 1 \cdot a$

Inverse elements

$I^+$

$a \in F^+_b$, $a + b = 0 = b + a$

Commutativity

$a + b = b + a$

$C^+$

Distributivity:

$a \cdot (B + C) = a \cdot B + a \cdot C$

$a / b = a \cdot B - a \cdot C$

Sometimes (for us almost always) we further

Commutativity:

$a \cdot b = b \cdot a$

$a \neq 0 \exists b: a \cdot b = 1 = b \cdot a$

If we further have $I^-$ then we call it a \textit{field}.\textit{Field}
Examples

\( \mathbb{R} \): ring, comm. ring, field.
\( \mathbb{Z} \): ring, comm. ring, not a field.
\( \mathbb{Q} \): --- --- field.
\( \mathbb{Z}_p \) integers modulo a prime \( p \):

ring, comm. ring,
field? - We have to check \( \mathbb{Z}_p \)
whether any non-zero element
has a multiplicative inverse.

Actually, it is a field.
we see that \( \mathbb{Z}_p \) (comm.)

So we have the ring \( \mathbb{Z}_N \) of
integers modulo \( N \) defined similarly.

\[
\begin{align*}
\text{Googol} &= 100^{100} \\
\text{Googolplex} &= 100 \text{Googol}
\end{align*}
\]
back to ciphers:
we had Caesar: \( \mathbb{Z}_26 \to \mathbb{Z}_26, \ a \mapsto a + 5 \)

generalized Caesar: \( C_i: \mathbb{Z}_26 \to \mathbb{Z}_26, \ a \mapsto a + i \)

affine Codes: \( A_{a,i}: \mathbb{Z}_26 \to \mathbb{Z}_26, \ a \mapsto a + ti \)
for \( a \in \mathbb{Z}_26, \ i \in \mathbb{Z}_26. \)

Try to decrypt
\[ b = A_{a,i}^{-1}(a) = x a + x i \]

hence \( x a + x i = a \)?

Problem: The inverse \( a^{-1} \) of \( a \in \mathbb{Z}_{26} \) 
does not always exist 
even if we require \( a \neq 0 \).

Eq: \( 2 \cdot b = 1 \) in \( \mathbb{Z}_{26} \) has no solution.

Proof: Assume \( b \) exists.
\[ 2 \cdot b = (2 \cdot b) \pmod{26} \]
\[ \underbrace{\text{even}} \underbrace{\text{even}} \]
\[ \text{even} \text{ even} \]
\[ \text{even} \text{ even} \]
\[ \text{unmatched} \]

But \( 1 \) is not even!

Similarly, \( 13 \) has no inverse.

Actually, \( 2 \cdot 1 \) has an inverse \( \sim 5 \)!
had seen

cesar: \[ \mathbb{Z}_{26} \rightarrow \mathbb{Z}_{26}, \quad a \mapsto a + 3 \]

pen cesar ciphers:
\[ C_i : \mathbb{Z}_{26} \rightarrow \mathbb{Z}_{26}, \quad a \mapsto a + i \]

affine ciphers:
\[ A_{a,i} : \mathbb{Z}_{26} \rightarrow \mathbb{Z}_{26}, \quad a \mapsto a a + i \]
where \( a \in \mathbb{Z}_{26}^{	imes} \)

i.e. \( a \) shall be invertible w.r.t. multiplication in \( \mathbb{Z}_{26} \).

attacks: (a) brute force

Try all keys.
Feasible for all the above ciphers because they have only
\[ 1, 26, 12, 26 \]
different keys, resp., which
is quite small.

There are at most 12 numbers:
invertible in \( \mathbb{Z}_{26} \):

\[ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23 \]

The others cannot be invertible because they are either even or divisible by 13.
Fix any permutation \( \pi : \mathbb{Z}_{26} \rightarrow \mathbb{Z}_{26} \) and replace letters accordingly.

How many such maps are there?

A permutation is a map which is

- injective, i.e. (into)
  \[ \pi(a) = \pi(b) \implies a = b, \]
  or
  \[ a \neq b \implies \pi(a) \neq \pi(b), \]
  and
  - surjective, i.e. (onto)
  \[ \forall x \exists a : \pi(a) = x. \]

In our case, one of the properties implies the other because the sets \( \mathbb{Z}_{26} \) and \( \mathbb{Z}_{26} \) are both finite and of the same size. 

There are \( 26! \) such permutations of \( \mathbb{Z}_{26} \).

This is a very huge number.

\[ 26! > 10^{26}, \]

\[ \approx 10^{22}. \]

This much too for brute force attack.
Every letter is still mapped to the same character.
So we can analyze the frequencies in the ciphertext.

ASCII-Histogramm von <startbeispiel-de.txt> (869 Zeichen)
Häufigkeit (%)

So most frequent letters are identifiable.
All the ciphers so far are one substitution ciphers
Another class are permutation ciphers.

We permute positions of letters. Like the Spartans used their Skytale

\[ \approx 500 \text{ BC.} \]

If stick is not entirely used

\[ \to \text{ see group size} \]

- Brute force attack
  \[ \to \text{ try all stick sizes.} \]
- Consider pairs of letters to find probable `distances`, `group sizes`, ...
Belix ciphers?

Vigenère

THIS IS SECURITY
\rightarrow CARE CARE CARE

"VH"

Read each letter as a number in \( \mathbb{Z}_{26} \) and add the key.

Brute force attack: \( 26^k \) keys of length \( k \).

If \( k \) is large enough this is not feasible.

But: there is a way to determine the key length!

After that we can do frequency analysis (or brute force on the generalized Caesar keys).

Belix?

\rightarrow Use a key as long as the message.

But still: the key may have structure. \rightarrow This can be used.

\rightarrow Use a random key!
One-Time-Pad

Given a plaintext $p \in \{0,1\}^e$
and a key $k \in \{0,1\}^e$
the ciphertext is $c \in \{0,1\}^e$
given by $c_i = p_i \oplus k_i$

This is completely secure!

$$\text{Prob(plain text }= p \mid \text{ciphertext }= c)$$
$$= \text{Prob}(R=p = c \oplus k \mid C=c)$$
$$= \text{Prob}(k=k = p \oplus c, C=c)$$
$$= \frac{\text{Prob}(C=c)}{\text{Prob}(C=c)} = 2^{-e}$$

\text{as Theorem This is completely secure.}

Problem?
Bad usage: Using twice the same is bad:

\[ c_1 = p_1 \oplus k \] \quad \left\{ \begin{array}{c}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad c_1 \oplus c_2 = p_1 \oplus p_2 \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad c_2 = p_2 \oplus k \end{array} \right. \\

Even without redundancy in \( p_1, p_2 \),
this reveals half the information about them. (Only 2e bits are revealed.)

Even with good usage:

Problem? \\
- Hard to generate so much random data \\

Too long keys.
Cesar, ..., Enigma, ... One Time Pad:

\[ \text{Alice} \quad \longrightarrow \quad \text{Bob} \]

AES, too. But ...

Suppose you calculate \( \mathbb{Z}_{256} \).
Is this a field?

\[ Q: \text{If } x \cdot y = 0 \text{ in a field, is it possible that both } x \neq 0 \text{ and } y \neq 0? \]

Then of course \( y = x^{-1} \cdot x y = x^{-1} \cdot 0 = 0 \),\hspace{1cm} \text{OK, but } x \neq 0. \quad \text{NC!}

Fact: In any field, we have \( x y = 0 \Rightarrow x = 0 \lor y = 0. \)

In \( \mathbb{Z}_{256} \) we have \( 2 \cdot 128 = 0 \)
so this is \( \text{not} \) a field.
The field $\mathbb{F}_2$

$\mathbb{F}_2 \ni a = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7$, where $a_i \in \mathbb{F}_2 = \{0, 1\}$.

Representation: 8 bits for an element – 1 byte.

Addition: XOR, $(a + b)_i = a_i \oplus b_i$.

Multiplication: as for polynomials modulo $x^8 + x^4 + x^3 + x + 1$.

Example: $57 \cdot 83 = C1$:

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

Field: You can divide by every non-zero element.

The ShiftRows operation

The rows are shifted cyclically by zero, one, two, or three bytes.

Polynomials over the field $\mathbb{F}_2$

$R = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1) \ni a_0 + a_1 x + a_2 x^2 + a_3 x^3$, where $a_i \in \mathbb{F}_2$.

Addition: coefficient-wise $(a + b)_i = a_i \oplus b_i$, XOR.

Multiplication: as for polynomials modulo $x^8 + x^4 + x^3 + x + 1$. Another way to express

$d = a \cdot b$ is by the following matrix equation:

\[
\begin{pmatrix}
a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\
a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\
\end{pmatrix} \cdot \begin{pmatrix}
b_0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 \\
b_0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 \\
\end{pmatrix} = \begin{pmatrix}
d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\
d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\
\end{pmatrix}
\]

Not a field: $(x + 1)^8 = 0$.

The MixColumns operation

Each column is considered as a polynomial and multiplied by $c = 02 + 01 x + 01 x^2 + 03 x^3$.

Inverse: Multiply with $d = 0e + 06 x + 06 x^2 + 08 x^3$.

The S-box

$\mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$.

Highly nonlinear:

$y \rightarrow 02 x^{32} + 09 x^{37} + 0d x^{31} + 2b x^{25} + 06 x^{29} + 0d x^{35} + 05 x^{8} + 0f x^{3} + 03$.

Simple implementation using a 256 byte lookup table.

The SubBytes operation

Apply the S-box to every byte.

Nonlinear part of the key schedule

Due to the use of the S-box this map is non-linear.

The KeySchedule

The round keys are generated from the 128 to 256 bit key.

The AddRoundKey operation

Simple XOR with the round key.
The field $F_2$

$F_2 \ni a = a_0 + a_2 z + a_3 z^2 + a_4 z^3 + a_5 z^4 + a_6 z^5 + a_7 z^6$, where $a_i \in \mathbb{F}_2 = \{0, 1\}$.

Representation: 8 bits for an element – 1 byte.

Addition: XOR, $a_i \oplus b_i = a_i + b_i$.

Multiplication: as for polynomials modulo $z^8 + 1$.

Example $57 \cdot 83 = C1$:

$z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0 = z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0 + z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0 + z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0 + z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0 + z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0 + z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0 + z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0 + z^7 z^6 z^5 z^4 z^3 z^2 z^1 z^0$.

The rows are shifted cyclically by zero, one, two, or three bytes.

Field: You can divide by every non-zero element.

Polynomials over the field $F_2$

$R = F_2[z]/(z^8 + 1) \ni a_0 + a_1 z + a_2 z^2 + a_3 z^3$, where $a_i \in \mathbb{F}_2$.

Addition: coefficient-wise $[a + b]_\oplus = a_i + b_i$, XOR.

Multiplication: as for polynomials modulo $z^8 + 1$. Another way to express $d = a \cdot b$ is by the following matrix equation:

$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \end{bmatrix} \cdot \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} d_0 & d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 & d_7 \end{bmatrix}$

Not a field: $(z + 1)^6 = 0$.

The S-box

$F_2 \rightarrow F_2 \rightarrow F_2$.

Highly nonlinear:

$y \mapsto \begin{cases} 00 & y^3 \\ 05 & y^3 + y^2 + 1 \\ 09 & y^3 + y^2 \\ 25 & y^3 + y^2 + y + 1 \\ 28 & y^3 + y^2 + y + 1 \\ 36 & y^3 + y^2 + y + 1 \\ 40 & y^3 + y^2 + y + 1 \\ 41 & y^3 + y^2 + y + 1 \\ 45 & y^3 + y^2 + y + 1 \\ 47 & y^3 + y^2 + y + 1 \\ 50 & y^3 + y^2 + y + 1 \\ 52 & y^3 + y^2 + y + 1 \\ 54 & y^3 + y^2 + y + 1 \\ 56 & y^3 + y^2 + y + 1 \\ 59 & y^3 + y^2 + y + 1 \\ 5a & y^3 + y^2 + y + 1 \\ 5c & y^3 + y^2 + y + 1 \\ 5f & y^3 + y^2 + y + 1 \\ 63 & y^3 + y^2 + y + 1 \end{cases} + 63$.

Simple implementation using a 256 byte lookup table.

The MixColumns operation

Add a column of the state to the product of each column of the state and the fixed matrix:

$\begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ d_0 & d_1 & d_2 & d_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \end{bmatrix} \cdot \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \end{bmatrix}$

$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \end{bmatrix}$

Each column is considered as a polynomial and multiplied by $c = 02 + 01 z + 01 z^2 + 03 z^3$.

Inverse: Multiply with $d = 0e + 0e z + 0d z^2 + 0d z^3$.

The Key Schedule

The round keys are generated from the 128 to 256 bit key.
\[ a = 1 + x^2 \]
\[ b = x + x^2 \]
\[ c = a \cdot b = (1 + x^2) \cdot (x + x^2) \]
\[ = x + x^2 + x^2 \cdot x + x^2 \cdot x^2 \]
\[ = x + x^2 + x^3 + x^4 \]
\[ = 0111 \ 1000 \]
\[ c \cdot c = x^2 + x^4 + x^6 + x^8 \]
\[ = 01011010 \]
\[ \leq 1111 \ 0010 \]

**Remark:** If we reduce modulo \( x^8 + 1 \), then we obtain not a field.

Because
\[ (x^4 + 1) \cdot (x^4 + 1) = 0 \]
\[ x^4 + x^4 = 0 \]
Not a field.

But I claim that

\[ p = x^4 + x^4 + x^3 + x + 1 \]

cannot be written as a product.

If we have \( p = p' \cdot q \)

and \( a = a' \cdot q \)

Then
\[ p' \cdot q = a' \cdot (p' \cdot q) = a' \cdot p = 0. \]
The field $\mathbb{F}_2$

$n \in \mathbb{N}$ \quad $\mathbb{F}_2 = \{0, 1\}$

where $a_i \in \mathbb{F}_2$

Representation: 8 bits for an element = 1 byte.

Addition: XOR, $(a_i + b_i) = a_i + b_i$, XOR.

Multiplication: as for polynomials modulo $x^4 + x + 1$.

Example 57 - 63 = $\mathbb{F}_2^3$

$\begin{align*}
&x^4 + x^3 + x^2 + x + 1
\end{align*}$

Field: You can divide by every non-zero element.

The ShiftRows operation

The rows are shifted cyclically by zero, one, two, or three bytes.

Polynomials over the field $\mathbb{F}_2$

$R = \mathbb{F}_2[\{x^4 + 1\}] \ni \{a_0 + a_1x + a_2x^2 + a_3x^3\}$

where $a_i \in \mathbb{F}_2$

Addition: coefficient-wise $(a_i + b_i) = a_i + b_i$, XOR.

Multiplication: as for polynomials modulo $x^4 + 1$. Another way to express $d = a \cdot b$ is by the following matrix equation:

$\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3
\end{bmatrix}
\cdot
\begin{bmatrix}
a_0 & a_1 & a_2 & a_3 \\
a_1 & a_0 & a_1 & a_2 \\
a_2 & a_1 & a_0 & a_1 \\
a_3 & a_2 & a_1 & a_0
\end{bmatrix}
\cdot
\begin{bmatrix}
b_0 \\
b_1 \\
b_2 \\
b_3
\end{bmatrix}
\cdot
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3
\end{bmatrix}
$\begin{align*}
\end{align*}$

Not a field: $(x + 1)^3 \not= 0$.

The MixColumns operation

Each column is considered as a polynomial and multiplied by $c = 02 + 01x + 01x^2 + 03x^3$

Inverse: Multiply with $d = GE + 09x + 0Dx^2 + 0Ex^3$. 

The S-box

$\begin{bmatrix}
\mathbb{F}_2^1 & \mathbb{F}_2^1 & \mathbb{F}_2^1
\end{bmatrix}
\longmapsto
\begin{bmatrix}
\mathbb{F}_2^1 & \mathbb{F}_2^1 & \mathbb{F}_2^1
\end{bmatrix}$

Highly nonlinear:

$y \rightarrow 05 \cdot y^9 + 09 \cdot y^9 + 0D \cdot y^9 + 28 \cdot y^9 + 25 \cdot y^9 + 26 \cdot y^9 + 23 \cdot y^9 + 35 \cdot y^9 + 38 \cdot y^9 + 63 \cdot y^9$

Simple implementation using a 256 byte lookup table.

The SubBytes operation

Apply the S-box to every byte.

Nonlinear part of the key schedule

Due to the use of the S-box this map is non-linear.

The Key Schedule

The round keys are generated from the 128 to 256 bit key.

The AddRoundKey operation

Simple XOR with the round key.
The field \( F_{2^8} \)

\[ F_{2^8} \ni a = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7, \]

where \( a_i \in F_2 = \{0, 1\} \).

Representation: 8 bits for an element – 1 byte.

Addition: XOR, \((a + b)_{F_{2^8}} = a_{F_{2^8}} \oplus b_{F_{2^8}}\).

Multiplication: as for polynomials modulo \( x^8 + 1 \).

Example 57: \( F_{2^8} \cdot F_{2^8} \):

\[ a \cdot b = a_7 x^7 + a_6 x^6 + a_5 x^5 + a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0, \]

Field: You can divide by every non-zero element.

The ShiftRows operation

The rows are shifted cyclically by zero, one, two, or three bytes.

Polynomials over the field \( F_{2^8} \)

\[ R = F_{2^8}[x]/(x^8 + 1) \ni a_0 + a_1 x + a_2 x^2 + a_3 x^3, \]

where \( a_i \in F_{2^8} \).

Addition: coefficient-wise \((a + b)_{F_{2^8}} = a_{F_{2^8}} \oplus b_{F_{2^8}} \), XOR.

Multiplication: as for polynomials modulo \( x^8 + 1 \). Another way to express \( d = a \cdot b \) is by the following matrix equation:

\[
\begin{bmatrix}
  a_0 & a_1 & a_2 & a_3 \\
  a_4 & a_5 & a_6 & a_7 \\
\end{bmatrix}
= 
\begin{bmatrix}
  a_0 & a_1 & a_2 & a_3 \\
  a_4 & a_5 & a_6 & a_7 \\
\end{bmatrix}
\cdot 
\begin{bmatrix}
  b_0 & b_1 & b_2 & b_3 \\
  b_4 & b_5 & b_6 & b_7 \\
\end{bmatrix}
\]

Not a field: \((x + 1)^8 = 0\).

The MixColumns operation

Each column is considered as a polynomial and multiplied by \( c = 02 + 01 x + 01 x^2 + 00 x^3 \).

Inverse: Multiply with \( d = 0E + 00 x + 00 x^2 + 0B x^3 \).

The S-box

\[ F_{2^8} \rightarrow F_{2^8} \rightarrow F_{2^8}, \]

Highly nonlinear:

\[ y \rightarrow 05 x^3 + 09 x^7 + 2D y^3 + 2B x^7 + 26 x^5 + 2F x^6 + 03 y^6 + 2D y^7 + 0B y^6 + 03. \]

Simple implementation using a 256 byte lookup table.

The SubBytes operation

Apply the S-box to every byte.

Nonlinear part of the key schedule

\[ (F_{2^8}^4) \rightarrow (F_{2^8}^4), \]

Due to the use of the S-box this map is non-linear.

The Key Schedule

The round keys are generated from the 128 to 256 bit key.

The AddRoundKey operation

Simple XOR with the round key.
The field $\mathbb{F}_2$

$\mathbb{F}_2 \ni a = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7$, where $a_i \in \mathbb{F}_2 = \{0, 1\}$.

Representation: 8 bits for an element – 1 byte.

Addition: XOR, $(a + b)_i = a_i \oplus b_i$.

Multiplication: as for polynomials modulo $x^8 + x^4 + x^3 + x + 1$.

**Example 57 - 83 = C1:**

$x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$

$z^7 + z^6 + z^5 + z^4 + z^3 + z^2 + z + 1$

$z^7 + z^6 + z^5 + z^4 + z^3 + z^2 + z + 1$

Field: You can divide by every non-zero element.

The ShiftRows operation

The rows are shifted cyclically by zero, one, two, or three bytes.

Polynomials over the field $\mathbb{F}_2$

$R = \mathbb{F}_2[x]/(x^8 + 1) \ni a_0 + a_1 x + a_2 x^2 + a_3 x^3$, where $a_i \in \mathbb{F}_2$.

Addition: coefficient-wise $(a + b)_i = a_i \oplus b_i$, XOR.

Multiplication: as for polynomials modulo $x^8 + 1$. Another way to express $d = a \cdot b$ is by the following matrix equation:

$$
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
a_0 & a_1 & a_2 & a_3 \\
a_3 & a_0 & a_1 & a_2 \\
a_2 & a_3 & a_0 & a_1 \\
a_1 & a_2 & a_3 & a_0 \\
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 \\
b_2 \\
b_3 \\
\end{bmatrix}
$$

Not a field: $(x + 1)^8 = 0$.

The MixColumns operation

Each column is considered as a polynomial and multiplied by $c = 02 + 01 x + 01 x^2 + 02 x^3$.

Inverse: Multiply with $d = GE + 09 x + 0D x^2 + 0B x^3$.

The S-box

$\mathbb{F}_2^8 \longrightarrow \mathbb{F}_2^8$:

Highly nonlinear:

$y \rightarrow 05 \cdot y^{x^3} + 09 \cdot y^{x^2} + 0D \cdot y^{x^2} + 28 \cdot y^x + 2F \cdot y^{x^3} + 07 \cdot y^{2x} + 2D$.

Simple implementation using a 256-byte lookup table.

The SubBytes operation

Apply the S-box to every byte.

Nonlinear part of the key schedule

Due to the use of the S-box this map is non-linear.

The Key Schedule

The round keys are generated from the 128 to 256 bit key.

The AddRoundKey operation

Simple XOR with the round key.
The field $\mathbb{F}_2$:

$$\mathbb{F}_2 \ni a = a_0 + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7,$$

where $a_i \in \mathbb{F}_2 = \{0, 1\}$.

Representation: 8 bits for an element = 1 byte.
Addition: XOR, $(a + b)_i = a_i \oplus b_i$.
Multiplication: as for polynomials modulo $x^8 + x^4 + x^3 + x^1 + 1$.

Example 57: $\mathbb{F}_2$: 83 = C1:

$$\begin{align*}
(x^7 + x^6 + x^3 + x^2 + 1) & \rightarrow x^7 + x^6 + x^3 + x^2 + 1 \\
(x^6 + x^5 + x^3 + x^2 + 1) & \rightarrow x^6 + x^5 + x^3 + x^2 + 1 \\
(x^5 + x^4 + x^3 + x^2 + 1) & \rightarrow x^5 + x^4 + x^3 + x^2 + 1 \\
(x^4 + x^3 + x^2 + x^1 + 1) & \rightarrow x^4 + x^3 + x^2 + x^1 + 1 \\
(x^3 + x^2 + x^1 + 1) & \rightarrow x^3 + x^2 + x^1 + 1 \\
(x^2 + x^1 + x^0) & \rightarrow x^2 + x^1 + x^0.
\end{align*}$$

Field: You can divide by every non-zero element.

The ShiftRows operation:

The rows are shifted cyclically by zero, one, two, or three bytes.

Polynomials over the field $\mathbb{F}_2$:

$$R = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x^1 + 1) \ni a_0 + a_2 x^2 + a_3 x^3 + a_4 x^4,$$

where $a_i \in \mathbb{F}_2$.
Addition: coefficient-wise $(a + b)_i = a_i + b_i$, XOR.
Multiplication: as for polynomials modulo $x^8 + x^4 + x^3 + x^1 + 1$. Another way to express $d = a \cdot b$ is by the following matrix equation:

$$\begin{bmatrix}
a_0 & a_1 & a_2 & a_3 \\
a_2 & a_3 & a_4 & a_5 \\
a_4 & a_5 & a_6 & a_7 \\
a_6 & a_7 & a_0 & a_1
\end{bmatrix} \cdot \begin{bmatrix}
b_0 \\
b_2 \\
b_4 \\
b_6
\end{bmatrix} = \begin{bmatrix}
d_0 \\
d_2 \\
d_4 \\
d_6
\end{bmatrix}.$$

Not a field: $(x + 1)^8 = 0$.

The MixColumns operation:

Each column is considered as a polynomial and multiplied by $c = 02 + 01 x + 01 x^2 + 03 x^3$.
Inverse: Multiply with $d = 02 + 03 x + 02 x^2 + 06 x^3$.

The S-box:

$$\begin{array}{c|cccccccc}
\mathbb{F}_2 & \rightarrow & \mathbb{F}_2 & \rightarrow & \mathbb{F}_2 \\
\hline
y & \rightarrow & y^{-1} & \rightarrow & (y^{-1})^{-1} \\
\hline
\end{array}$$

Highly nonlinear:

$y \rightarrow 05 \cdot y^{30} + 09 \cdot y^{31} + 09 \cdot y^{32} + 29 \cdot y^{33} + 29 \cdot y^{34} + 29 \cdot y^{35} + 29 \cdot y^{36} + 29 \cdot y^{37} + 33$.

Simple implementation using a 256 byte lookup table.

The SubBytes operation:

Apply the S-box to every byte.

Nonlinear part of the key schedule:

$$\begin{array}{c|cccccccc}
\mathbb{F}_2^8 & \rightarrow & \mathbb{F}_2^8 & \rightarrow & \mathbb{F}_2^8 \\
\hline
(y) & \rightarrow & (y) & \rightarrow & (y^{-1}) \\
\hline
\end{array}$$

Due to the use of the S-box this map is nonlinear.

The Key Schedule:

The round keys are generated from the 128 to 256 bit key.

The AddRoundKey operation:

Simple XOR with the round key.

©2005 Michael Nüsken, cosec, b-it, Bonn
Designed as Rijndael by Joan Daemen and Vincent Rijmen.

The field \( \mathbb{F}_2 \)

\( \mathbb{F}_2 \ni \alpha = a_0 + a_1 \cdot 2 + a_2 \cdot 2^2 + a_3 \cdot 2^3 + a_4 \cdot 2^4 + a_5 \cdot 2^5 + a_6 \cdot 2^6 + a_7 \cdot 2^7 \),
where \( a_i \in \mathbb{F}_2 = \{0, 1\} \).

Representation: 8 bits for an element – 1 byte.

Addition: XOR, \((a + b)_i = a_i + b_i\).

Multiplication: as for polynomials modulo \(x^8 + x^4 + x^3 + x + 1\).

**Example:** \( x^5 \cdot x^7 = x^{12} \), \( x^2 \cdot x^7 = x^9 \), \( x^6 \cdot x^7 = x^{13} \).

**The MixColumns operation**

Each column is considered as a polynomial and multiplied by 
\[ c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \] 

Inverse: Multiply with \( d = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \).

**The ShiftRows operation**

The rows are shifted cyclically by zero, one, two, or three bytes.

**Polynomials over the field \( \mathbb{F}_2 \)**

\[ R = \mathbb{F}_2[x]/(x^4 + 1) \ni \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3, \]
where \( \alpha_i \in \mathbb{F}_2 \).

Addition: coefficient-wise \((a + b)_i = a_i + b_i\), XOR.

Multiplication: as for polynomials modulo \(x^4 + 1\). Another way to express \( d = a \cdot b \) is by the following matrix equation:

\[ \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \]

Not a field: \((x + 1)^8 = 0\).

**The SubBytes operation**

Apply the S-box to every byte.

**Nonlinear part of the key schedule**

\[ (\mathbb{F}_2^8)^\bullet \longrightarrow (\mathbb{F}_2^8)^\bullet, \]

Due to the use of the S-box this map is non-linear.

**The Key Schedule**

The round keys are generated from the 128 to 256 bit key.

**The AddRoundKey operation**

Simple XOR with the round key.
The field \( \mathbb{F}_2 \)

\[ a = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7, \]
where \( a_i \in \mathbb{F}_2 = \{0, 1\} \).

Representation: 8 bits for an element — 1 byte.

Addition: \( (a + b)_i = a_i + b_i \).

Multiplication: as for polynomials modulo \( x^8 + x + 1 \),

\[ a \cdot b = \sum_{i=0}^{7} a_ib_{7-i} \mod (x^8 + x + 1), \]

\[ a \cdot b = \sum_{i=0}^{7} a_ib_{7-i} \mod (x^8 + x + 1). \]

Example: \( 57 \cdot 83 = 1 \) if:

\[ a = x_7 \cdots x_0, \quad a_i \in \{0, 1\}, \quad i = 0, 1, \ldots, 7. \]

Field: You can divide by every non-zero element.

The ShiftRows operation

The rows are shifted cyclically by zero, one, two, or three bytes.

Polynomials over the field \( \mathbb{F}_2 \)

\[ R = \mathbb{F}_2[x]/(x^8 + 1) \ni a_0 + a_1x + a_2x^2 + a_3x^3, \]
where \( a_i \in \mathbb{F}_2 \).

Addition: coefficient-wise \( (a + b)_i = a_i + b_i \), XOR.

Multiplication: as for polynomials modulo \( x^8 + 1 \). Another way to express \( d = a \cdot b \) is by the following matrix equation:

\[ \begin{bmatrix}
    d_0 \\
    d_1 \\
    d_2 \\
    d_3 \\
    d_4 \\
    d_5 \\
    d_6 \\
    d_7
\end{bmatrix}
= \begin{bmatrix}
    a_0 & a_1 & a_2 & a_3 \\
    a_1 & a_0 & a_2 & a_3 \\
    a_1 & a_2 & a_0 & a_3 \\
    a_1 & a_2 & a_3 & a_0 \\
    a_1 & a_3 & a_0 & a_2 \\
    a_1 & a_3 & a_2 & a_0 \\
    a_1 & a_3 & a_2 & a_1 \\
    a_0 & a_1 & a_2 & a_3
\end{bmatrix}
\begin{bmatrix}
    b_0 \\
    b_1 \\
    b_2 \\
    b_3 \\
    b_4 \\
    b_5 \\
    b_6 \\
    b_7
\end{bmatrix}.\]

Not a field: \( (x + 1)^8 = 0 \).

The MixColumns operation

Each column is considered as a polynomial and multiplied by \( c = 02 + 01x + 01x^2 + 03x^3 \).

Inverse: Multiply with \( d = 0E + 09x + 0Bx^2 + 03x^3 \).

The S-box

Highly nonlinear:

\[ y \rightarrow y^{13} + \begin{bmatrix}
    0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
    1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
    1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
    1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
    1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
    1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
    0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
    0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}. \]

Simple implementation using a 256 byte lookup table.

The SubBytes operation

Apply the S-box to every byte.

Nonlinear part of the key schedule

Due to the use of the S-box this map is non-linear.

The Key Schedule

The round keys are generated from the 128 to 256 bit key.

The AddRoundKey operation

Simple XOR with the round key.
The field $\mathbb{F}_2[x]$

$\mathbb{F}_2[x] = \mathbb{F}_2[x]/(x^4 + x + 1)$

$\forall a \in \mathbb{F}_2$, the map $\mathbb{F}_2 \ni a \mapsto a^2 \in \mathbb{F}_2$ is a field automorphism.

The rows are shifted cyclically by zero, one, two, or three bytes.

Polynomials over the field $\mathbb{F}_2$

$R = \mathbb{F}_2[x]/(x^4 + 1) \ni a_0 + a_1 x + a_2 x^2 + a_3 x^3$.

Addition: $a_0 + a_2 = a_3$ or $a_2 + a_3 = a_0$ XOR.

Multiplication: as for polynomials modulo $x^4 + 1$. Another way to express $d = a \cdot b$ is by the following matrix equation:

$$
\begin{bmatrix}
\alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 \\
\beta_0 & \beta_1 & \beta_2 & \beta_3
\end{bmatrix} = 
\begin{bmatrix}
\alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 \\
\beta_0 & \beta_1 & \beta_2 & \beta_3
\end{bmatrix} 
\begin{bmatrix}
\alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 \\
\beta_0 & \beta_1 & \beta_2 & \beta_3
\end{bmatrix}
$$

Not a field: $(x + 1)^4 = 0$.

The MixColumns operation

Each column is considered as a polynomial and multiplied by $c = 02 + 012 = 02 x + 012 x^3$.

Inverse: Multiply with $d = GE + 09z + 01z x^2 + 08z x^3$.

The S-box

$\mathbb{F}_2 \xrightarrow{S} \mathbb{F}_2^8$.

Highly nonlinear:

$y = 01y^2 + 00y^3 + 29y^4 + 25y^5 + 06y^6 + 07y^7 + 10y^8 + 03y^9 + 02y^{10}$.

Simple implementation using a 256 byte lookup table.

The SubBytes operation

Apply the S-box to every byte.

Nonlinear part of the key schedule

Due to the use of the S-box this map is non-linear.

The Key Schedule

The round keys are generated from the 128 to 256 bit key.

The AddRoundKey operation

Simple XOR with the round key.
Security of One-Time Pad

What happens?

plaintext: \( p \in \{0, 1\}^n \) string of \( n \) bits

How are they distributed?

Somehow! So: for any \( p \in \{0, 1\}^n \)

\[
\text{prob}(w|P = p) = \prod_{p \in \{0, 1\}^n} \text{random variable message}
\]
is given such that \( \sum_{p} \prod_{p} = 1 \)

key:

\( k \in \{0, 1\}^n \) string of \( n \) bits

How are they distributed?

For any \( k \in \{0, 1\}^n \) we have:

\[
\text{prob}(w|K = k) = 2^{-n}
\]
random variable

And

\[
\text{prob}(P = p \land K = k) = \text{prob}(P = p) \cdot \text{prob}(K = k)
\]
in other words:

the r.v. \( P \) and \( K \) are

independent.

Lastly spoken: we choose the key

independently of the plaintext.

ciphertext: \( c \in \{0, 1\}^n \) bit string of length \( n \).

let \( C(w) = P(w) \oplus K(w) \)
What is $\text{prob}(P = p \mid C = c) = ?$

Example: $\text{prob}(P = 0 \ldots 0) = 1$
then Eve can easly guess the correct plain text.
But does the cipher text help her guess? No.

Theorem: For any plain text $p \in \{0,1\}^n$, and any cipher text $c \in \{0,1\}^n$, we have
$$\text{prob}(P = p \mid C = c) = \text{prob}(P = p).$$
In other words: the cipher text does not help Eve at all.

Proof:
$$\text{prob}(P = p \mid C = c) = \frac{\text{prob}(P = p \land C = c)}{\text{prob}(C = c)}$$
$$= \frac{\text{prob}(P = p \land K = c \oplus p)}{\text{prob}(C = c)}$$
$$= \frac{\text{prob}(P = p) \cdot \text{prob}(K = c \oplus p)}{\text{prob}(C = c)}$$

Now, $\text{prob}(C = c) = \text{prob}(P \oplus K = c)$
$$= \sum_{p \in \{0,1\}^n} \text{prob}(P = p \land P \oplus K = c)$$
$$= \sum_{p} \text{prob}(P = p \land K = c \oplus p)$$
\[ \sum_{p} \text{prob}(P=p) \cdot \text{prob}(K = c \oplus p) = 2^{-n} = 1 \]

\[ = 2^{-n} \cdot \text{prob}(K = c \oplus p) \]

So,
\[
\text{prob}(P = p \mid C = c) = \text{prob}(P = p) \cdot \frac{\text{prob}(K = c \oplus p)}{\text{prob}(C = c)}
\]

\[ = \text{prob}(P = p), \quad 1 \]

That's best of all we can hope for:

Eve does not learn anything from the cipher text.
Calculating and deciding inverses

First, let's summarize where we need this:

Suppose \( N \in \mathbb{N}^+ \).

\( \mathbb{Z}_N \) ring of integers modulo \( N \):

- Elements: \( \{ 0, 1, 2, \ldots, N-1 \} \)
- Operations:
  - \( +: (a, b) \rightarrow (a+b) \mod N \)
  - \( \cdot: (a, b) \rightarrow (a \cdot b) \mod N \)
  - \( -: a \rightarrow \begin{cases} 0 & \text{if } a = 0, \\ N-a & \text{if } a > 0 \\ = (-a) \mod N \end{cases} \)

\textbf{ToDo:} \( \cdot: a \rightarrow \begin{cases} 1/a & \text{if exist} \\ \text{FAIL otherwise} \end{cases} \)

\textbf{Theorem:} \( \text{DON'T PANIC PANIC (C)} \)

Maybe: \( \text{I'\text{\textregistered}} \)

Suppose \( N = p \) is prime. Then (as is to be proved)

\( \mathbb{Z}_p \) is a field, which we call \( \mathbb{F}_p \).

Consider polynomials with coefficients in \( \mathbb{F}_p \).

(Think of \( p = 2 \)) Suppose \( m \) is a polynomial of degree \( n \geq 1 \).

\( \mathbb{F}_p [X]/\langle m \rangle \) ring of polynomials modulo \( m \) with coefficients in \( \mathbb{F}_p \)

- Elements: \( a_0 + a_1 X + a_2 X^2 + \ldots + a_n X^n \), with \( a_0, a_1, \ldots, a_n \in \mathbb{F}_p \)
- Operations:
  - \( +: (a, b) \rightarrow (a+b) \mod m \)
  - \( -: (a, b) \rightarrow (a+b) \mod m \)
  - \( -: a \rightarrow -a \mod m = -a \)
Note that \( \mathbb{F}_2^8 \cong \mathbb{F}_{256} \cong \mathbb{F}_2[x] / (x^8 + x^4 + x^3 + x + 1) \) in AES, and this has no non-trivial factors.

Let's start with a better known situation: where mod \( N \).

We are given \( a \in \mathbb{Z}_N \).

Find \( b \in \mathbb{Z}_N \) such that \( b \cdot a = 1 \) in \( \mathbb{Z}_N \).

i.e., \( b \cdot a \equiv 1 \pmod{N} \) in \( \mathbb{Z} \).

i.e., Find \( b \in \mathbb{Z} \) such that

\[ b \cdot a + t \cdot N = 1 \quad \text{in } \mathbb{Z} \]

Find \( b, t \in \mathbb{Z} \) such that

\[ b \cdot a + t \cdot N = 1 \quad \text{in } \mathbb{Z} \]

Let's try some examples:

\( N = 42, \quad a = 5 \).

Our aim is to find such that \( b \cdot a + t \cdot N \) is as small as possible (but positive).

Trivially: \( b = 1, \quad t = 0 \) : \( b \cdot a + t \cdot N = a = 5 \).

\( b = 0, \quad t = 1 \) : \( b \cdot a + t \cdot N = N = 42 \).

And new equation: \( b \cdot a + t \cdot N = 42 - 5 = 37 \).

And again: \( b \cdot a + t \cdot N = 42 - 5 = 37 \).

Or all this at once:
### Math Problem

#### Text

Given the equation:

\[ \sqrt{c} + 1 = 2 \]

Solve for \( c \).

#### Solution

1. Subtract 1 from both sides:
   \[ \sqrt{c} = 1 \]
2. Square both sides:
   \[ c = 1 \]

### Table

<table>
<thead>
<tr>
<th>Value</th>
<th>1</th>
</tr>
</thead>
</table>

### Diagram

- Graph showing the relationship between the variables.
- Arrows indicating the direction of the solution process.
- Key points marked for clarity.

### Comment

- Note on the solution process.
- Additional notes on graph interpretation.

---

### Additional Notes

- Explanation of graphing techniques.
- Steps for solving similar equations.
Theorem

The above Extended Euclidean Algorithm needs $O(n^3)$ operations.
Even $O(n^2)$ is true.

Another example

\[
\begin{array}{c|c|c|c|c}
0 & 25 & 1 & 1 & 1 \\
1 & 3 & 0 & -1 & 0 \\
2 & 20 & 1 & -3 & -4 \\
3 & 5 & 4 & 5 & -18 \\
4 & 0 & 5 & 5 & 5 \\
\end{array}
\]

$(25 = 1 \cdot 3 + 0.6)$
$(3 = 0 \cdot 25 + 1.6)$

Stop indicator

To use last a cross check: $0 = 5 \cdot 95 - 18 \cdot 25$

This line is always easy to check but most easily if the last non-zero $r_i$ equals 1.

Lemma

The EEA computes the greatest common divisor $g$
and $x, t$ such that
$g = s \cdot a + t \cdot b$

Indeed, if $e$ is the number of the line with last non-zero $r_i$ then $g = e$.
Actually, in the algorithm we choose a quotient $q_i$ suitably and then

$$r_{i+1} = r_{i-1} - q_i \cdot r_i$$

We do that until $r_{i+1} = 0$. Then

$$\gcd(r_i, r_{i+1}) = \gcd(r_i, 0) = r_i$$

Reminder: The greatest common divisor $g$ of two elements $a, b$ is an element $g$ such that

1. $g \mid a$ and $g \mid b$
2. If $d \mid a$ and $d \mid b$ then $d \mid g$
3. $\forall h : h \mid a \iff h \mid b$
   $$\Rightarrow h \mid g$$
   $$~(h \leq g)$$

Now, we can show

$$\gcd(r_i, r_{i+1}) = \gcd(r_i, r_{i-1})$$

Let $h$ be a common divisor of $r_i$ and $r_{i-1}$. Then

$$r_{i+1} = r_{i-1} - q_i \cdot r_i = r_{i-1} - q_i \cdot r_i = (r_{i-1} - q_i \cdot r_i) \cdot h.$$

Thus $h$ divides $r_{i+1}$, so $h$ divides $r_i$ and $r_{i+1}$.
Thus, the other way around, say $k$ is a common divisor of $r_{i+1}$ and $r_i$.

Then $r_{i-1} = r_{i+1} - q_i r_i$

is a multiple of $b$.

Thus $k$ is a common divisor of $r_i$ and $r_{i-1}$.

By induction we have

$\gcd(a, b) = \gcd(r_0, r_1) = \ldots = \gcd(r_e, 0) = r_e$.

Further, for any $i$ we have

$r_i = s_i a + t_i b$

This is trivially true for $i = 0$ and $i = 1$.

And for $i > 1$ we have

$r_{i-1} = s_{i-1} a + t_{i-1} b$

$(-q_{i-1}) r_{i-1} = s_{i-1} a + t_{i-1} b$

$r_i = r_{i-1} - s_{i-1} r_{i-1} = (s_{i-1} - q_{i-1} s_{i-1}) a + (t_{i-1} - q_{i-2} t_{i-1}) b$

In particular,

$g = r_e = e a + e b$

$\frac{e}{k}$ is claimed values.
Claim: \( \ell \leq 2 \cdot n \cdot 2^\alpha \cdot b^1 / a^\alpha \cdot b^\beta \).

We choose \( q_i \) such that

\[
\begin{align*}
\lambda_i &= \lambda_{i-1} - q_i \cdot \gamma_i, \\
\lambda_{i-1} &= q_i \cdot \gamma_i + \beta_i - 1, \\
|\lambda_i| &< 1 \cdot n_1. \quad \text{(for integers)}.
\end{align*}
\]

It is easy to see that

\[
|\lambda_{i+1}| < \frac{1}{2} \cdot |\lambda_{i-1}|
\]

Thus

\[
|\lambda_0| < \frac{1}{2} \cdot \max(\lambda_{-1}, 1) \cdot e_{12}
\]

that implies that

\[
\ell \leq \frac{\log \max(1, 1, 1)}{\max(1, 1, 1)}
\]

\[
\ell \leq 2 \cdot n.
\]

Thus the number of lines is at most twice the number of bits in \( x, b \).

And each step costs at most \( O(n^2) \).

In total we have \( O(n^2) \) bit operations at most.

Actually, the bound is bad, one can prove that we need \( O(n^2) \) bit operations.
Recall that for the EEU
one only need

- a ring, i.e. a set for equality
- a division with remainder,
i.e. for any \( a, b \) with \( b \neq 0 \)
there exists \( q, r \) such that

\[ a = q \cdot b + r \]

and \( r(+) \leq r(b) \) or \( r(+) = 0 \).

for some suitable measure \( r(+) \).

Example.
\[ \mathbb{Z} : \quad r(a) = |a| . \]

\[ \mathbb{F}[X] \text{ with } F \text{ a field} : \]
\[ \text{ring of polynomials} \]
\[ r(a) = \deg a \]
\[ r(0) = -\infty \]

Division in \( \mathbb{F}[X] \), say \( F = \mathbb{F}_2 \).

\[ a = x^2 + x^3 + x + 1 \]

\[ b = x^4 + x + 1 \]

\[ x^7 + x^5 + x^4 + x^3 + x^2 + 1 = (x^3 + 1) \cdot 5 \]

\[ \overbrace{- (x^7 + x^4 + x^3)} \]

\[ \overbrace{\phantom{\frac{x^7}{x^4}} + x^4 + x^3 + x^2 + 1} \]

\[ \overbrace{\phantom{\frac{x^7}{x^4}} + (x^2 + x)} \]

\[ \overbrace{\phantom{\frac{x^7}{x^4}} - (x^4 + x + 1)} \]

\[ \overbrace{\phantom{\frac{x^7}{x^4}} - x^2 - x} \]

\[ r(+) < r(b) : \text{DONE!} \]
Let's do an EEA for

\[ a = x^8 + x^4 + x^3 \]

\[ b = x - (x - 1) = x^2 + x \]

Predict \( \gcd(a, b) = ? \)

<table>
<thead>
<tr>
<th>( i )</th>
<th>( r_i )</th>
<th>( q_i )</th>
<th>( s_i )</th>
<th>( t_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( x^8 + x^4 + x^3 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>( x^2 + x )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( x )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Checks: true!

\( \gcd: x \)

\( \text{repr:} \)

\[
\begin{align*}
x &= A \cdot (x^8 + x^4 + x^3) \\
&+ (x^6 + x^5 + x^3 + x + 1) \cdot (x^2 + x)
\end{align*}
\]

Efficient? Yes: each step reduces the degree by 1. Thus after degree many steps (+2) we are done.
We have been talking about the rings $\mathbb{Z}_N$ (integers modulo $N$) (operation: modulo $N$)

[poly] polynomials over $\mathbb{Q}_m$ and modulo $m$ (operations like for polynomials but taking remainders modulo $m$)

What about these rings?

In $\mathbb{Z}_N$ we had translated the task to find $b$ such that $a b \equiv 1 \pmod{N}$ to the task of finding $b, t$ such that

$$b \cdot a + t \cdot N = 1 \pmod{N}.$$  

(*) 

Such a solution can be found using the EEA, if it exists...?

We know: if $\gcd(a, N) = 1$ then the EEA finds $b, t$ such that

$$b a + t N = 1 \pmod{N}.$$  

Otherwise, if $\gcd(a, N) \neq 1$?

Then $a = a' g$, $N = N' g$ 

whence $b a + t N = (b a' + t N') g$ 

Assuming $\gcd(a', N') = 1$ we would have $g \mid 1$, hence $g$ is trivial.

So $g = 1$. By $(*)$, so $\{\}$ has no solution.

So $ab \not\equiv 1 \pmod{N}$ has no solution, i.e. it has no inverse.
The EEA decides whether $a \in \mathbb{Z}_N$ has an inverse and if so, it finds the inverse.

Actually, $a$ has an inverse if $a \in \mathbb{Z}_N^*$

$$\iff \gcd(a, N) = 1.$$ 

Thus, 

$$\mathbb{Z}_N^* = \{ a \mid \gcd(a, N) = 1 \}$$

Same for polynomials:

$$\left( \frac{\mathbb{Z}[X]}{\langle m \rangle} \right)^*$$

$$= \{ a \mid \gcd(a, m) = 1 \}$$

and the EEA computes the inverse if it exists:

$$b \cdot a + \ell \cdot m = 1 \quad \Rightarrow \quad b = a^{-1} \mod m$$
Example \( \mathbb{Z}_6[x] \) 

b. \((x-1) = 1?\)

\[
\begin{array}{c|c|c|c}
   x^2 & x & \text{?} \\
   \hline
   0 & X^2 + X + 1 & \text{\textcolor{red}{?}} \\
   1 & X - 1 & X + 2 \\
   0 & 3 & \text{\checkmark} \\
\end{array}
\]

\(0 \text{ mod 6}\)!

What happened?

???

Answer: \( \mathbb{Z}_6 \) is not a field.

Thus, there is no division with remainder for polynomials over \( \mathbb{Z}_6 \).

Thus, EEA needs not work.

Check the conditions!
When is \( \mathbb{Z} \) a field?

If \( \mathbb{Z} \) is not irreducible.

Two definitions:

\( \text{p is irreducible } \iff \text{ whenever we write } p = a \cdot b \)

\( \text{then } a \cdot b \text{ is multiplicatively } \underline{\text{incredible}}. \)

\( \text{in other words, } p \text{ cannot be written as a proper product.} \)

\( \forall a, b : p = a \cdot b \Rightarrow a \mid c \text{ v } b \mid c. \)

\( \text{p is prime } \iff \text{ whenever } p \text{ divides a product } a \cdot b \)

\( \text{then } p \text{ divides one of the factors } a, b. \)

\( \forall a, b : p \mid a \cdot b \Rightarrow p \mid a \text{ v } p \mid b \)

Remark: \( p \) prime \( \Rightarrow p \) irr.

Example where \( \subseteq \) does not hold:

\[ \mathbb{Z}[\sqrt{-5}] = \{ a + b\sqrt{-5} \mid a, b \in \mathbb{Z} \} \]

\[ (a + b\sqrt{-5})(a' + b'\sqrt{-5}) = aa' + (ba' + ab')\sqrt{-5} + bb'(-5) = (aa' - 5bb') + (ba' + ab')\sqrt{-5}. \]

New:

\[ 6 = 2 \cdot 3 = (1 + \sqrt{-5})(1 - \sqrt{-5}) \]

Actually, 2, 3, \( 1 \pm \sqrt{-5} \) are all irreducible.

So: \( 2 \mid (1 + \sqrt{-5})(1 - \sqrt{-5}) \) but \( 2 \nmid (1 + \sqrt{-5}) \)
When is \( \mathbb{Z}_N \) a field?

If \( N \) is not irreducible

then \( \mathbb{Z}_N \) is not a field.

**Pf**  

\( N = N_1 \cdot N_2 \) with \( N_1, N_2 \) both non-trivial, i.e. \( N_1 \neq \pm 1, \ N_2 \neq \pm 1 \).

Then \( (N_1 \mod N) \cdot (N_2 \mod N) = 0 \mod N \)

and

\( N_1 \mod N \neq 0 \), (otherwise \( N_1 = 0 \mod N \), i.e. \( N_2 = \pm 1 \) & c.

\)

**Example:**

\( N = 4 \):

\[ 2 \cdot 2 = 0 \mod 4 \]

\( N = 6 \):

\[ 2 \cdot 2 = 0 \mod 6 \]

So these are no fields!

If \( N \) is irreducible

then \( \mathbb{Z}_N \) is a field.

**Pf**  

we need to show that all elements but 0 have a multiplicative inverse.

We know \( \mathbb{Z}_p^x = \{ a \in \mathbb{Z}_p | \gcd(a, p) = 1 \} \)

Now the divisors of \( p \) are \( \pm 1, \pm p \)

Consider \( a \in \mathbb{Z}_p \), i.e. \( a \in \{ 0, 1, \ldots, p-1 \} \).

Then \( \gcd(a, p) \neq \pm 1 \), if and only if \( a = 0 \).

Thus \( \mathbb{Z}_p^x = \{ 1, 2, \ldots, p-1 \} \) (iff \( a \) is a multiple of \( p \)).
Theorem \( \mathbb{F}_q[X]/<m> \) is a field

iff \( m \) is irreducible.

\[ \text{Proof:} \]

If \( m \) is reducible,

the unit \( m = m_1 \cdot m_2 \) as a proper product.

\[ m \cdot (m_1 \cdot \text{mod } m_2) \cdot (m_2 \cdot \text{mod } m_1) = 0 \text{ in } \mathbb{F}_q[X]/<m>. \]

If \( m \) is irreducible,

the \( \left( \mathbb{F}_q[X]/<m> \right)^* = \{ a \in \mathbb{F}_q[X]/<m> \mid \gcd(a,m) = 1 \} \)

now, if \( m \) has no proper factors then \( a \neq 0 \)

is enough to ensure that \( \gcd = 1 \), so any
element and \( 0 \) has an inverse. \( \Rightarrow \) it's a field.
Ex: \( \mathbb{Z}_2 \) is ill.

\[ a \text{ field } \implies \overline{\mathbb{F}_2} \]

\[ \overline{\mathbb{F}_2}[X] : \quad X^2 + X + 1 \text{ is ir.} \]

\[ \overline{\mathbb{F}_2}[X] / \langle X^2 + X + 1 \rangle \text{ a field: } \overline{\mathbb{F}_4}. \]

\[ \{ a_0 + a_1 X \mid a_0, a_1 \in \overline{\mathbb{F}_2} \} = \{ 0, 1, X, X+1 \} \]

\( \overline{\mathbb{F}_4}[Y] : \quad Y^2 + Y + 1 \text{ is ir.} \)

\[ \overline{\mathbb{F}_4}[Y] / \langle Y^2 + Y + 1 \rangle \text{ a field: } \overline{\mathbb{F}_{4^2}} = \overline{\mathbb{F}_{16}}. \]

\[ a, b_0 + b_1 Y + b_2 Y^2 \mid b_0, b_1, b_2 \in \overline{\mathbb{F}_4} \]

\( \mathbb{F}_{256} = \overline{\mathbb{F}_2}^8. \)

So: wonderful deal.

\[ \mathbb{F}_2 \Rightarrow \mathbb{Z}_2 \]

\[ \mathbb{F}_4 \]

\( \mathbb{F}_{64} \)

It turns out that \( \mathbb{F}_{64} \cong \mathbb{F}_{64} \)

from above. \( \Box \)
There exists a field with \( q \) elements if \( q \) is a prime power and essentially one such, a power of a prime, eg. 74.

Set of invertible numbers:

\[ \mathbb{Z}_n^x \]

\[ (\prod_{q \in \mathbb{P}} \mathbb{F}_q[x] / \langle m \rangle)^x \]

\[ \mathbb{Z}_6^x = \{ 1, 5 \} \]

\[ \left( \mathbb{Z}_2[x] / \langle x^2+1 \rangle \right)^x = \{ 1, x, x^3 \} \]

Addition stays well defined:

in \( \mathbb{Z}_6 \): \( 1 + 1 = 2 \)

\[ \begin{align*}
    0^x & \rightarrow 0^x \\
    1^x & \rightarrow 1^x \\
    2^x & \rightarrow 2^x \\
    3^x & \rightarrow 3^x \\
    4^x & \rightarrow 4^x \\
    5^x & \rightarrow 5^x
\end{align*} \]

Multiplication? Works! Panic!
Whenever $R$ is a ring, commutative, then the set of $R^*$ of invertible elements is a commutative group w.r.t. to multiplication!

\[ \mathbb{Z}_{15}^* = \{ \pm 1, \pm 2, \pm 4, \pm 7 \} \] is a comm. group!

\[ \mathbb{Z}_{256}^* \bigg/ \left( \mathbb{Z}_4^* \right) \] is a comm. group.

**Def:** A comm. group is a set with one operation such that the axioms PABIC hold.
How to exchange a key without a pre shared secret?
How to bulk secretly even if Eve listens to everything including the description of the scheme?

Diffie & Hellman (1976) Key exchange:

Setup: a group: \( \mathbb{Z}_p^x \), \( p \) prime
\( q \) prime
\( g \in \mathbb{Z}_p^x \) with good properties (related to \( q \! \))

Example: \( \mathbb{Z}_{23}^x \)
\( q = 23, \ p = 47 \).

It is a group of 23 elements.

\( g = 2 \) : \( G = \langle g \rangle \)

Group generated by \( g \)
\( \{ 1, g, g^2, \ldots, g^{22}, g^{23} \} \)

Alice (Cesar)
\( x \in \mathbb{Z} \)
\( h_A = g^x \)

Bob (Cleopatra)
\( y \in \mathbb{Z} \)
\( h_B = g^y \)

\( k_A = h_B^x \)

\( k_B = h_A^y \)

\( k_A = k_B \)

\( g^{xy} \)
\[ k_A = (g^y)^x = g^{yx} = g^{x'y} = (g^x)^y = k_B, \]
so Alice and Bob have a shared secret now. They can use it to encrypt further messages.

Correctness? This is \( k_A = k_B \).

Efficiency? \( O(n^2) \) field operations per multiplication.

Auction: We sell 2^{26}. Who does it cheapest?

- First bid: 27 mult.
- Dennis: 15 mult: \( \text{calc } 2^{26}, \) square.
- Tillman: 10 mult: \( \text{calc } 2^{26}, \) square, square.
- Til: 8 mult: ---

(2, 2^2, 2^3, 2^6, 2^7, 2^4, 2^8)

\( 2 \times 3 \times 5 \times 6 \)
6 multiplications!
we sell $2^{35}$ !

summit: 12 $2^2$: scale $2^3$, then raise this to the fifth power.

Titman 7 $2^5$:

$2, 2^4, 2^8, 2^{16}, 2^{32}, 2^{64}, 2^{128}$

Titman 6 $2^5$:

$2, 2^4, 2^8, 2^{16}, 2^{32}, 2^{64}$

summit 6 $2^5$

$2, 2^4, 2^8, 2^{16}, 2^{32}, 2^{64}$

square & multiply (Repeated squaring)

Note: $35 = 10001_2$

New compo:

$2^{10001_2}$

square

$2^{10001_2}$

square

$2^{10001_2}$

square & multi with 2

$2^{10001_2}$

square & multi with 2

$2^{10001_2}$

square & multi with 2

$2^{10001_2}$

Try: $2^{782}$

Square & multi $\rightarrow 14$ multi

same thing $\rightarrow 12$ multi

optimum: 11 multi.
Theorem: Given a group $G$ and an element $g \in G$, we can compute the map

$$
\mathbb{Z} \rightarrow G \\
g \rightarrow g^e
$$

with

$$
2 \left( \left\lceil \log_2 e + 1 \right\rceil - 1 \right)
$$

#bits for $e$

$s-1 \leq \log_2 e < s$

group operations

to calculate.

Poor proof? Implementation? $ightarrow$ Ex

SECURITY?

What does $EVE$ see?

Setup: group $G$, generator $g$

Communication: $h_A = g^x$, $h_B = g^y$

Wants: common key: $g^{xy}$

DH (Diffie-Hellman problem)

\[
\left( g, g^x, g^y \right) \rightarrow g^{xy}.
\]

For example with $G = \mathbb{Z}_7^*$, $g = 2$ we might ask:

$g = 2$, $x = 3$, $y = 5$. What is $g^{xy}$?
It is enough to find \( x \) or \( y! \)

because then, say we found \( x \), we can compute \( y = (g^y)^x = 5^x \).

Consider the

DLP (Discrete Logarithm Problem)

\[
\left( g, g^x \right) \rightarrow \mathbb{Z}
\]

What we have seen is:

If we can solve the DLP then solve the DHP.

So we must choose the setup, group \( G \) and the generator \( g \), such that

at least the DLP is difficult.

\[ \text{Necessary for the security:} \]

DLP is difficult

(in \( G = \langle g \rangle \)).

Good examples:

use \( g \in \mathbb{Z}_p^\times \) such that \( g^q = 1, g + 1 \)

where \( q \) is a large prime (\& \( q | p - 1 \)).
Other groups, with particularly difficult DLP:

**Elliptic curves**

Given an equation

\[ y^2 = x^3 + ax + b \]

with \( a, b \in \mathbb{F}_9, \quad q^2 + 4q, \quad 3q^2 \).

Over \( \mathbb{R} \) the picture is like this:

We require

\[ P + Q + R = 0 \]

so we should define

\[ P + Q = -R = S \]

This defines a group.

If we add one point: \( O = \text{ zero element } \).

Define: \( P + Q = S \) as above if

\[
\begin{align*}
&\text{if } P + Q = O, \\
&\text{if } P + Q \neq O \\
&\text{if } Q = -P
\end{align*}
\]

Group? \( P \)

N by construction of \( O \).

Mirror also \( x \)-axis \( \checkmark \)

\( C \) obvious \( \checkmark \)

(\text{difficult to see. But true!!})
For these groups the DLP is supposedly 'more' difficult. Thus we can use smaller versions (measured by $q$, say) to get same security.

E.g., using $\mathbb{Z}_p^*$ with 2024-bit $p$ corresponds to $E$ over $\mathbb{F}_p$ with 160-bit $p$.

In total, $E$ might be cheaper at same security.

and under luckier
You're to find, say \( p, q, \) and \( g \) such that 
\[ g \in \mathbb{Z}_p^\times, \quad g^q = 1, \quad g \neq 1. \]

Need to know more about exponentiation, powering.

Say we are given \( g \in G, \) \( G \) some group.

(Think \( G = \mathbb{Z}_p^\times \) for example.)

Consider \( g, g^2, g^3, g^4, g^5, g^6, \ldots \)

\[ \infty \]

\[ \exists p = 1+1, \quad g = 2. \]

| \( e \in \mathbb{Z} \) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| \( 2^e \mathbb{Z}_n \) | 1 | 2 | 4 | 3 | 5 | -1 | -2 | -4 | 3 | -5 |

\[ \rightarrow 1 \quad 2 \ldots \]

repeat this part!
Correctness

Efficiency ($\text{square & multiply}$)

Security

Eve has to solve the DHP:

\[(g, g^x, g^y) \rightarrow g^{xy}\]

at least with some probability.

There, "amplification" is possible!

From a solution for $\left(g, g^x, g^y, (g^y)^x\right)$

we can derive $g^{xy}$, so try various $x\neq y$.

If Eve can solve the DLP:

\[(g, g^x) \rightarrow x\]

with some probability

then she can solve the DHP.

"Amplification" possible.
Beware of Eve becoming active: Mallory.

(Written in the middle attack):

Alice

Bob

Mallory (Wilma)

\[
\begin{align*}
(g^x)^y &= (g^y)^x' = g^{xy'} = (g^{x'})^y' = g^{x'y}
\end{align*}
\]

\[
E_{g^x} (m) \rightarrow \text{decrypt} \& \text{ reencrypt} \rightarrow E_{g^y} (m)
\]

Mallory can read everything.

Somehow Alice should whom she is talking to!

ALWAYS: be aware of your model of security. Which type of attacks do you consider?
Turning in circles.

We work in some group $G$ and there is an element $g \in G$.

Question: when does $g, g^2, g^3, \ldots$ start repeating?

Then (Lagrange), write:

Given $g \in G$, $G$ a group

then $x \cdot g = 1$.

In other words, the picture of $1, x, x^2, x^3, \ldots$
looks not only like

and the length of the circle divides $\#G$ for any $x$.

If $G$ commutative,

Take a list of all group elements:

$g_1, g_2, g_3, \ldots, g_{\#G}$

and multiply each element with $x$:

$xg_1, xg_2, xg_3, \ldots, xg_{\#G}$.

Up to order, this is also a list of all group elements!

(a) If $xg_i = xg_j$ then $g_i = g_j$ or $i = j$. [Simply multiply $xg_i = xg_j$ with $x^{-1}$ on the right.]
Take an arbitrary element of $G$, say $g_i$. Add it to the new list! We look in with $xg_j = g_i$. So take $j$, $g_j = x^{-1}g_i \in \mathbb{G}$.

Then $xg_j = x \cdot x^{-1}g_i = g_i$.

Thus up to order both lists are equal. Multiply all elements on each list:

$g_1 \cdot g_2 \cdot \ldots \cdot g_{\#G} = xg_1 \cdot xg_2 \cdot \ldots \cdot xg_{\#G}$

The commutative & lists are equal up to order.

Divide and obtain:

$L = x^{-\#G}$

Another example:

$p = 23$, $\pi = 5$: $G = \mathbb{Z}_{23}^\times$, $\#G = 22$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$0$</th>
<th>$1$</th>
<th>$2$</th>
<th>$3$</th>
<th>$4$</th>
<th>$5$</th>
<th>$6$</th>
<th>$7$</th>
<th>$8$</th>
<th>$9$</th>
<th>$10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^i$</td>
<td>$1$</td>
<td>$5$</td>
<td>(2)</td>
<td>$10$</td>
<td>$4$</td>
<td>$-3$</td>
<td>$8$</td>
<td>$-6$</td>
<td>$-7$</td>
<td>$-12$</td>
<td>$9$</td>
</tr>
</tbody>
</table>

$\#G = 11 \ 12 \ 13 \ 14 \ 15 \ 16 \ 17 \ 18 \ 19 \ 20 \ 21 \ 22$
\[ p = 23, \ x = 2: \ G = \mathbb{Z}_{23}^x, \ \#G = 22. \]

\[ \begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
1 & 2 & 4 & 8 & -7 & -9 & -5 & -10 & 3 & 6 \\
\end{array} \rightarrow x^{22} = 1 \]

\[ p = 23, \ x = -1: \ G = \mathbb{Z}_{23}^x, \ \#G = 22 \]

\[ \begin{array}{cccc}
1 & 0 & 2 \\
1 & -1 & 1 \\
\end{array} \rightarrow x^{22} = 1 \]

**Corollary (Euler)**

Suppose \( a \in \mathbb{Z} \), a coprime to \( N \) i.e. \( \gcd(a, N) = 1 \).

Then \( a^{\varphi(N)} = 1 \) in \( \mathbb{Z}_N^x \)

where \( \varphi(N) := \# \mathbb{Z}_N^x \).

**Proof**

This is simply the Theorem of Coprime applied to \( G = \mathbb{Z}_N^x \).

**Corollary (Little Fermat Theorem)**

Suppose \( \varphi \) prime, \( 1 < a < p \).

Then \( a^{p-1} = 1 \) in \( \mathbb{Z}_p^x \).

**Proof**

Apply the previous to \( N = \text{prime} \) and compute \( \varphi(p) = p - 1 \). 

\( \mathbb{Z}_p \) is a field.
First consequence:
Square & multiply an exponentially may first reduce the exponent by the group size (if known).

Example

\[ 5324 \cdot 427 \quad 2 \quad 2 \quad \in \mathbb{Z}^x_{11} \]

\[ \quad \in \mathbb{Z}^x_{11} \quad \text{10 elements!} \]

Consequence for Diffie–Hellman key exchange:

Technical: choose \( x \in \mathbb{Z} \)
in the interval \( 0 \leq x < \#G \).

Security: \( g^k = 1 \) should not happen for a small \( k \).

Definition: \( \text{ord}_G g = \min \{ k \in \mathbb{N}_{\geq 0} \mid g^k = 1 \} \)
is called the order of \( g \) in \( G \).
Using this notion we can reformulate the theorem of Lagrange:

**Corollary**

Give a group $G$ and $x \in G$.

Then, $\text{ord } G \cdot x$ divides $\# G$.

**Proof**

Suppose $x^k = 1$ and $k$ is minimal.

By Lagrange we have $x^\# G = 1$.

Say, $\# G = 10$.

And $x^3 = 1$.

So we have to show that $3$ is not minimal!

By EEA we obtain $s, t$ such that $s \cdot k + t \cdot \# G = g$

and $g = \gcd(k, \# G)$.

Then $x^g = \left(\frac{x^k}{x^1}\right)^s \left(\frac{x^1}{x^1}\right)^t = 1^s 1^t = 1$.

Since $k$ is minimal we have $g \geq k$.

But $g \mid k$, thus $g = k$.

And of course $g \mid \# G$, thus $k \mid \# G$. 

$\therefore$
**RSA (1978)**

Rivest
Shamir
Adleman

**Purpose:** setup parameters and then send encrypted messages.

**Setup**
Choose two primes \( p, q \) (large, say 512 bits, each)

Let \( N = p \cdot q \).

Let \( L = (q-1)(p-1) \).

(Actually, \( \# \mathbb{Z}_N^* = L \)!

**Throw away** \( p, q, L \).

**Store:** Private key \((N, d)\).

**Publish:** Public key \((N, e)\).

Bob \(\rightarrow\) Alice, Encrypt a message \( x \in \mathbb{Z}_N^{(0)} \).

\[ y \leftarrow x^e \text{ in } \mathbb{Z}_N. \]

Send \( y \)

Alice, Decrypt this:

\[ z \leftarrow y^d \text{ in } \mathbb{Z}_N. \]

**Claim Always:**

\[ z = x. \]
Easy: If \( x \in \mathbb{Z}_N^* \) then
\[
z = y^d = (x^e)^d = x^{ed} = x^{1 + \ell t}
\]
for some \( t \).

Because \( ed = 1 + \ell t \) for some \( t \),
so \( ed = 1 + \ell t \) for some \( t \).

\[
x = x \cdot (x^e)^d = 1 \quad \text{by Theorem (Euler)}
\]

Thus \( \ell = \# \mathbb{Z}_N^* \).

(Exercise 3.2):
\[
\# \mathbb{Z}_N^* = \# \mathbb{Z}_p^* = (p-1)(q-1) = \ell.
\]

\[
x = x \cdot 1 = x \quad \text{Well!}
\]

What if \( x \notin \mathbb{Z}_N^* \)?

First: The probability is very small:
\[
\text{prob}(x \notin \mathbb{Z}_N^* \mid x \in \mathbb{Z}_N)
\]
\[
= \frac{p(q-1)}{p^2} \approx \frac{2}{511}
\]
\[
\approx 10^{-153}
\]

Second: Any such 'bad' message \( x \) reveals \( p \) and \( q \) by computing \( \gcd(x, N) \).
Third: \( x^{ed} = x \)

even in the 'bad' cases.

First proof for this: use ad hoc \( x = x^p \)

or \( x = x^q \) and see what happens. (Ex)

Second proof: new tool:

**Chinese Remainder Theorem**

\[
\begin{align*}
\text{If } \gcd(m,n) &= 1 \text{ is not one, then} \\
\text{let } \frac{\text{gcd}(m,n)}{g} &= m \quad \text{such that } \frac{n}{g} \cdot \text{gcd}(m,n) = 0 \quad \text{in } \mathbb{Z}_m \\
\text{and } \frac{n}{g} \cdot \text{gcd}(m,n) &= 0 \quad \text{in } \mathbb{Z}_n \\
\text{so the cell } (0,0) \text{ gets } 0 \text{ and } \frac{\text{gcd}(m,n)}{g}, \\
\text{and thus } \mathbb{Z}_m \times \mathbb{Z}_n \text{ cannot fill the table.} \\
\text{But if } \gcd(m,n) = 1 \text{ then the table gets filled and any cell gets exactly one element.}
\end{align*}
\]
CRT (naive) formulation.
Suppose \( m, n \) are coprime integers.

Given \( x \in \mathbb{Z}_m, y \in \mathbb{Z}_n \), find a number \( z \in \mathbb{Z}_{mn} \) such that
\[
z = x \pmod{m},
\]
\[
z = y \pmod{n}.
\]

Given \( x \in \mathbb{Z}, 0 \leq x < m, y \in \mathbb{Z}, 0 \leq y < n \), find a number \( z \in \mathbb{Z} \) such that
\[
z \equiv m \cdot x, \quad z \equiv n \cdot y.
\]

Actually, we have a map:
\[
\mathbb{Z}_m \times \mathbb{Z}_n \rightarrow \mathbb{Z}_{mn},
\]
\[
\hat{z} : \hat{x} \mod m \mapsto \hat{x} \mod mn.
\]

This is a nice map; for \( x, y \) we have
\[
\pi(x+y) = \pi(x) + \pi(y)
\]
and
\[
\pi(x \cdot y) = \pi(x) \cdot \pi(y).
\]

This is somehow obvious if \( \pi \) is defined by choosing \( x \in \mathbb{Z} \) such \( x = \hat{x} \mod mn \).

Then \( z(x+y) = \hat{x+y} \mod mn \)
\[
\hat{z}(\hat{x} + \hat{y}) \mod mn = \hat{x} \mod mn + \hat{y} \mod mn = \pi(x) + \pi(y). \quad \cdots \]
Consider this:

**CRT'**

Suppose \( m, n \) are coprime.

Then the map

\[
\mathbb{Z}_{mn} \to \mathbb{Z}_m \times \mathbb{Z}_n
\]

is a bijective \textit{ring} morphism, i.e., a \textit{ring iso} morphism.

In particular, we obtain a \textit{group iso} morphism

\[
\mathbb{Z}_{mn}^x \to \mathbb{Z}_m^x \times \mathbb{Z}_n^x.
\]

So we obtain the corollary:

\[
\# \mathbb{Z}_{mn}^x = \# \mathbb{Z}_m^x \cdot \# \mathbb{Z}_n^x
\]

\[
\varphi(m,n) = \varphi(m) \cdot \varphi(n)
\]

provided \( m, n \) are coprime.

[Note that \( \varphi(4) = 2 \neq \varphi(2) \cdot \varphi(2) \).]

**Proof (CRT')**

Assume the naive version. It says that the map \( \mathbb{Z} \to \mathbb{Z}_m \times \mathbb{Z}_n \) is surjective and thus \( \mathbb{Z}_{mn} \to \mathbb{Z}_m \times \mathbb{Z}_n \) is surjective.

Now, since \( \# \mathbb{Z}_{mn} = m \cdot n = \# \mathbb{Z}_m \cdot \# \mathbb{Z}_n = \# (\mathbb{Z}_m \times \mathbb{Z}_n) \)
the map must also be injective.
Proof (CET)

So given \( x \in \mathbb{Z}_m, \ y \in \mathbb{Z}_n \)

Let \( z \in \mathbb{Z} \) such:

\[ z = x \cdot 1 \in \mathbb{Z}_m, \]
\[ z = y \cdot 1 \in \mathbb{Z}_n. \]

Consider \((x, y) = (1, 0) \implies z_1 \)

and \((x, y) = (0, 1) \implies z_2 \)

so

\[ z_1 = 1 \cdot 1 \in \mathbb{Z}_m, \]
\[ z_2 = 0 \cdot 1 \in \mathbb{Z}_n. \]

Claim: If we can find \( z_1 \) and \( z_2 \) then

\[ z = x \cdot z_1 + y \cdot z_2 \]

solves the original problem.

\[ z = m \cdot z_1 + y \cdot z_2 = x \cdot 1 + y \cdot 0 = x \cdot 1 \in \mathbb{Z}_m \]

\[ z = n \cdot z_1 + y \cdot z_2 = 0 \cdot 1 + y \cdot 1 = y \cdot 1 \in \mathbb{Z}_n. \]

By symmetry it suffices to find \( z_1 \):

so we look for

\[ z_1 = 1 = a \cdot m \]

for some \( a \)

and

\[ z_1 = 0 + b \cdot n \]

for some \( b \).

That is:

\[ a = a \cdot m + b \cdot n \]

for some \( a, b \).

And also

\[ z_2. \]
We factored \( a, b \) by \( \text{EEA} \)

since \( \gcd(a, n) \) are coprime.

Then  
\[
2a = b \cdot u 
\]
gives  
\[
z_1 = 1 \cdot an = 1 \cdot z_2
\]
and  
\[
2a = b \cdot u
\]
gives  
\[
z_1 = b \cdot u = 0 \cdot z_2
\]

So we are done.

### C2A

\[ \text{ln: } x, y \in \mathbb{Z}_m, \; y \in \mathbb{Z}_n \]

\[ \text{of: } z \in \mathbb{Z}_{mn}. \]

\[ \text{Compute } t = an + bn, \]

then  
\[
z = (x \cdot bn + y \cdot an) \mod mn.
\]

### RSA Example:

\[ p = 5, \; q = 7, \]

\[ N = 35 \]

\[ \phi(N) = 24 \]

\[ d = 17. \]

By coincidence  
24 is very special:
any number \( x \in \mathbb{Z}_N^* \)
has square \( x^2 \).

\[
\begin{array}{c|c|c|c}
24 & 10 & 1 \\
17 & 1 & -1 \\
7 & 2 & 3 \\
3 & 2 & -2 \\
1 & 3 & -3 \\
0 & -17 & 24
\end{array}
\]
Wow, CRT is fun!

Alice has to calculate

\[ y^d \mod \phi_n \]

Why not do this in \( \mathbb{Z}_p \times \mathbb{Z}_q \)?

Compute

\[ z_p = (y \mod p)^d \]

and

\[ z_q = (y \mod q)^d \]

and then use CRT to find

\[ z = z_p \mod \mathbb{Z}_p, \quad z = z_q \mod \mathbb{Z}_q. \]

Say Alice' job is to return this value \( z \).

And further say Alice is a smart card
and we can disturb Alice so she
makes an error in exactly one place.

So we get

\[ z' = z_p \mod \mathbb{Z}_p \]

\[ z' \neq z_q \mod \mathbb{Z}_q. \]

But we may have prepared \( y = x^e \mod n \) then \( x = y^d \mod n \)

and

\[ z' - x = 0 \mod \mathbb{Z}_p \]

\[ z' - x = 0 \mod \mathbb{Z}_q \]

Thus \( \gcd (z' - x, n) = p \).
Remind:

**CRT** given \( m, n \) coprime

then

\[ \mathbb{Z}_{mn} \cong \mathbb{Z}_m \times \mathbb{Z}_n \]

Example of its use:

Say \( m, n \) are both prime.

How many \( x \in \mathbb{Z}_{mn} \) are there

with \( x^2 = 1 \) in \( \mathbb{Z}_{mn} \)?

Answer: look for solutions in \( \mathbb{Z}_m \)

and find \( \pm 1 \in \mathbb{Z}_m \) there.

Since \( m \) is prime, \( \mathbb{Z}_m \) is a field

and there can be no other solutions.

If \( p \) is a polynomial over some field

and \( p(x) = 0 \) the

\[ p(T) = q(T) \cdot (T-x) + r \]

Hence! \[ \text{with } \deg(r) < \deg(T-x) \]

\[ \begin{cases} 0 \text{ or } -\infty \end{cases} \]

Now,

\[ p(x) = q(x) - (x-x) + r \]

\[ \frac{r}{0} = 0 \]

so \( r = 0 \). Thus \( p(T) = q(T) \cdot (T-x) \).

Thus there can be at most \( \deg(p) \) zeros.

Also in \( \mathbb{Z}_n \) we find \( \pm 1 \) as only solutions.

Thus in \( \mathbb{Z}_{mn} \times \mathbb{Z}_n \) we have 4 solutions:

\[ (+1, +1), (-1, -1), (+1, -1), (-1, +1) \]
RSA is correct!

$x^d = x \text{ in all cases!}$

\textbf{Proof:}

We want this equation in $\mathbb{Z}_{pq}$.

By CRT

$\mathbb{Z}_{pq} \cong \mathbb{Z}_p \times \mathbb{Z}_q$.

So is $x^d = x \mod \mathbb{Z}_p$?

Now, we know that

$x^{p-1} = 1 \mod \mathbb{Z}_p$

by Fermat's Little Theorem, provided $x \neq 0$.

Thus

$x^p = x \mod \mathbb{Z}_p$

for $x \neq 0$. But this is true for $x = 0$ as well! Inductively, this gives

$x^{k+1} = x^k \cdot x^{p-1}$

for $k \geq 0$ in $\mathbb{Z}_p$ for any $x \neq 0$.

Now, $ed = 1 + \lambda \cdot (p-1) \cdot (q-1)$;

so

$ed = 1 + \frac{(p-1) \cdot (q-1)}{k}$

in $\mathbb{Z}_p$.

Similarly, $x^{ed} = x \mod \mathbb{Z}_p$.

So

$x^{ed} = x \mod \mathbb{Z}_p \times \mathbb{Z}_q$

so

$x^{ed} = x \mod \mathbb{Z}_{pq}$.
RSA is efficient

Tasks:

Setup:

- generate primes
  - generate a random number (pseudo random number generator)
  - test whether it is prime
    - good probabilistic test available $O(n^3)$
- multiply $O(n^2)$
- select e, d:
  - generate a random number $O(n^3)$
- EEA $O(n^2)$

Encryption: one exponentiation $O(n^3)$

Decryption: same

So everything is polynomial time.

In practice:

Setup for 1024 or 2048 bits takes, say a minute.

Encrypt/decrypt takes a few milliseconds.
Is RSA secure?

What would be a total break?

1. Eve knows \((N, e)\) and some \(y\) and lots of pairs \((x, x^e)\) [and maybe she can get some pairs \((y, y^d)\)...].

2. Eve can find the primes \(p, q\) s.t. \(N = pq\).

3. Eve can find the rep. length \(L\).

4. Eve can derive \(d\).

5. Eve finds \(x\) such that \(x^e = y\).

Obvious: \((i) \iff (ii)\)

\(\Gamma \Leftarrow\): Consider \((T-p)(T-q)\)
\[= T^2 - (p+q)T + pq\]
\[\Rightarrow L = (p-1)(q-1)\]
\[= pq - p - q + 1\]
\[= N + 1 - (p+q)\]
\[\Rightarrow p + q = N + 1 - L.\]

So Eve knows this polynomial.

And thus can compute its zeroes.

( midnight fun:)
Compute

\[ t = \frac{gcd(e_1, e_2 - 1)}{N} \]

and try \( t, t+1, \ldots \).

This gives \( L \) then...

\( \square \Rightarrow \square \vee \)

**OPEN PROBLEM:** Does \( \square \) imply \( \square \)?

"The security of RSA is based on the difficulty of factoring."

Would \( \square \Leftrightarrow \square \) be enough?

(assuming that factoring is difficult)

Suppose an attacker can given \( y \)
compute \( \text{Bit}_0(y) \).

Is that a problem? YES!

say \( \text{Bit}_0(y) = 0 \) then \( x = 2^{x'} \)

Hence \( y = 2^e x'^e \)

\( y' = y / 2^e = x'^e \)

Now \( \text{Bit}_0(x') = \text{Bit}_1(y) \)!

... This gives \( x \)!
Definition of Security is a very intricate problem!

Best to date:

There should not be a probabilistic polynomial time Turing machine that can decide a yes-no question on $x$ with non-negligible advantage.

Signatures

12.8.07

- identifies the signer
- makes sure the document is not modified
- binds Joachim to his yesterday's statement
El Gamal signatures (1978)

Setup: \( G \) group, strictly: \( G = \mathbb{Z}_p^* \)

\[ p \text{ a large prime} \]

\[ \text{so that the discrete log problem is difficult.} \]

\[ \text{(In particular: } p-1 = 4k \text{ should not be a product of small primes.}) \]

\[ l := \text{order}(g) \text{ is large, actually it should contain a large prime factor.} \]

Further, let \( \alpha : G \rightarrow \mathbb{Z}_e \) be some very very simple map (like polynomial image mod \( p \) as steps mod \( e \))

Personal setup: \( \alpha \in \mathbb{Z}_e \) as a private key

\( \alpha \) computes

\[ \alpha = g^\alpha \text{ as a public key.} \]

Signature

Verify: If \( \ast \)

\[ a^b \cdot F = g \text{ msg, } r \in \mathbb{Z}_e, b \in G \]

then \( (b, g) \) is a valid signature for the message \( \text{msg} \).

Verifier knows \( g \) from setup, \( a \) from public key,

\( \alpha \) from \( \text{signed document} \)

Necessarily, \( g \in \mathbb{Z}_e, b \in G \).
Total break

(i) Find \((b, g)\) so that it is a signature to \(msg\).

\[d = a^b \mod\]

\[d \cdot b^y = g^{msg}\]

and the brute force...

Plan: Choose \(d\), then find \(b^k\)

by taking a log.

Choose \(b\) such that gives the found \(b^k\).

find another log to get \(y\):

\[b^y = g^{msg} \div d\]

Need 2 Dls to solve the \('ElGamal Problem\')

(ii) Choose \(b\) then compute a log:

\[b = g^{msg} \div a\]

Need 1 DL to solve the \('EP\')

(iii) Other plan: choose \(g\) and try to find \(b\)...

\[a^{b^k} \div b = sh\]

Seems to be even more difficult...
The signer can use the secret key $x$ so she has to solve
\[ g^x \cdot y = msg \]
so Alice chooses $b \in \mathbb{Z}_p^*$.
She chooses $\beta \in \mathbb{Z}_e^*$ and computes $b := g^\beta$.
Now she has to solve
\[ g \cdot x \cdot \beta \cdot \beta = g \]
which solving
\[ g^x \cdot \beta^y = msg \text{ in } \mathbb{Z}_e \]
gives a solution.
(We make sure that $\beta$ is invertible: $\beta \in \mathbb{Z}_e^*$)
then
\[ X = \beta^{-1}(\text{msg} - ab^x) \in \mathbb{Z}_e. \]

\[ \text{sign(msg)} \]
\[ \begin{array}{c}
\beta \in \mathbb{Z}_e^* \\
\rho := g^\beta \\
y \in \mathbb{Z}_e \text{ solves } ab^x + \beta \rho = \text{msg}. \\
\text{return } (b, y)
\end{array} \]

Technical problem: the signed document is 5x as large as the unsigned one.
We use a hash function value instead of the message itself.

Let \( h : \{0, 1\}^* \rightarrow \mathbb{Z}_e \) be a hash function, easily computable.

It should be one-way. Given \( y \in \mathbb{Z}_e \), it should be difficult to find a message \( \text{msg} \in \{0, 1\}^* \) with hash value \( h(\text{msg}) = y \).

It should be collision-resistant.

It should be second preimage resistant. Given a message \( \text{msg}_1 \in \{0, 1\}^* \) to find another message \( \text{msg}_2 \in \{0, 1\}^* \) such that \( h(\text{msg}_1) = h(\text{msg}_2) \).
It should be collision resistant:

It should be difficult to find two messages \( \text{msg}_1, \text{msg}_2 \) that are different \( \text{msg}_1 \neq \text{msg}_2 \) with same hash value \( h(\text{msg}_1) = h(\text{msg}_2) \)

Definition Security of a signature

An attacker that can

- given signatures for any number of chosen documents (which may depend on each other)
- can forge a new document with a valid signature

with a non-negligible probability in polynomial time

breaks the scheme.

A signature scheme is considered secure if there is no such attacker.

Details \( \rightarrow \) ‘Provable security’ or Reduktionist security
Let's apply this to the ElGamal scheme:

**Setup**

Choose a group \( G \), say \( G = \mathbb{Z}_p^* \), choose an element \( g \in G \) of large order \( e = \text{ord}_g(g) \), prime

(say \( e > 768 \text{ bit}, \) and \( p > 2048 \text{ bit} \).)

To some security against known attacks on DL in general, in groups \( \mathbb{Z}_p^* \).

\[ x \in \mathbb{Z}_e, \quad \alpha \in \mathbb{Z}_e \quad \text{second signing key} \]

\[ a := g^x \in G \quad \text{public signing key} \]

**Signature generation**

Given a message \( m \).

Choose \( \beta \in \mathbb{Z}_e \), compute \( b = g^\beta \in G \), and solve \( \alpha b^x \beta^y = h(m) \)

where

\[ * : G \rightarrow \mathbb{Z}_e \text{ is some simple (almost in verible) function} \]

and

\[ h : \{0,1\}^* \rightarrow \mathbb{Z}_e \text{ is a hash function.} \]

Output : \( (b, \beta) \) as a signature.
Suppose \( h \) is not 2\(^{nd}\) preimage resistant, i.e., there is an algorithm \( \text{TWO} \) which computes given \( \text{msge} \), another \( \text{msge} \), \( \text{msg}_1 \) with \( h(\text{msg}_1) = h(\text{msge}) \) in \( \text{poly-time} \) with non-negligible probability.

Then \( \forall \) (Choose \( \text{msge} \) arbitrarily, ask the signer for a signature \( b^* \) on \( \text{msge} \rightarrow (b, b^*) \) with \( a b = g \).

Ask \( \text{TWO} \) for \( \text{msg} \neq \text{msg}_1 \) with \( h(\text{msg}) = h(\text{msge}) \).

Output \( (\text{msg}, (b, b^*)) \).

Clearly, \( \forall \) runs in same time as \( \text{TWO} \), and it succeeds if \( \text{TWO} \) succeeds.

So if \( \text{TWO} \) is too good then \( \forall \) is too good and hence the scheme is insecure.
Together:

If the signature scheme is secure
then the hash must be 2nd preimage resistant.

Suppose \( h \) is not collision resistant,

i.e., there is an algorithm \texttt{Collision}

which outputs \( \text{msg}_1 \neq \text{msg}_2 \)

with \( h(\text{msg}_1) = h(\text{msg}_2) \)

in polynomial time with non-negligible probability.

Thus:

\texttt{Call \texttt{Collision} to get}

\( \text{msg}_1 \neq \text{msg}_2 \) with \( h(\text{msg}_1) = h(\text{msg}_2) \).

Ask the signer for a signature \( (b, s) \) on \( \text{msg}_2 \).

Output: \( (\text{msg}_2, (b, s)) \).

Thus we get \( a^b \cdot b^x = g \quad h(\text{msg}_1) = h(\text{msg}_2) \).

Again: if \texttt{Collision} is too good then it is too good.

Theorem: If the scheme is secure

then the hash function must be collision-resistant.

Similarly:

If the scheme is secure

then the DL with bases \( g \in G \) must be difficult.
Three properties for hash functions:

1. Is one-way
2. Is 2nd preimage resistant
3. Is collision resistant.

If 2nd attacks one-wayness
then two’s: stripped
 msg
 ask for a preimage msg of h(msg).
 000
 output msg.

is a slightly worse attack on 2nd preimage resistance. Small gap!

If 2nd attacks 2nd preimage resistance
then collision’s: choose msg, randomly.

Call Two with msg,
to obtain msg + msg,
with h(msg) = h(msg).

Output: (msg, msg)
is a polynomial attacker with some success prob. as Two.
Brake force on these three properties: $\rho$.
Say $h: \mathbb{S^*} \rightarrow \mathbb{Z}$, with $\epsilon$ an $n$-bit number.

One way: $E[\frac{1}{\#Z_{\epsilon}}]$

so we expect $\#Z_{\epsilon} = \epsilon$ trials until we find $\text{msg}$ with $h(\text{msg}) = k$.

2nd preimage: $\epsilon \approx 2^n$

We expect $\epsilon$ trials.

Collision: $\sqrt{\epsilon} \approx 2^{n/2}$

Theacher: Repeat

1. Choose a new $\text{msg}_i$.
2. Until $h(\text{msg}_i) \notin \{h(\text{msg}_1), ..., h(\text{msg}_{i-1})\}$
3. Output the two colliding $\text{msg}_i, \text{msg}_j$.

Run time: $O(\sqrt{\epsilon}) = O(2^{n/2})$
Trailing on Signatures

We have seen El Gamal signatures

\[ a^b \cdot b^c = \text{h(m)} \]

- Need to work in a group with difficult DLP
- Need a collision-resistant hash function.

For variants of this scheme reductions to these two necessary conditions are available.

Problem: Hash crisis!

MD4 \( \rightarrow \) 128-bit \( \rightarrow \) BROKEN
(64 rounds) need only 2 or 3 hash computations
\( \rightarrow \) seconds for a new collision

MD5 \( \rightarrow \) 128-bit \( \rightarrow \) BROKEN
(96 rounds) \( \rightarrow \) about 15 minutes for a new collision

SHA-1 \( \rightarrow \) 160-bit \( \rightarrow \) BROKEN
(80 rounds)

SHA-2 \( \rightarrow \) 160-bit \( \rightarrow \) BROKEN
(80 rounds)

RIPEMD \( \rightarrow \) 160-bit \( \rightarrow \) BROKEN
(80 rounds)

(Instead of, say, a year for \( 2^{64} \) hash computations)

Similar design!
One further family: SHA-2

SHA-256 $\rightarrow$ 256 bits: similar design
($>80$ rounds)
probably secure in practice
because of its
dimensions.

No tested replacement, yet.

Practical security

No better attack than "generic" ones.

For your information, the following
slides show the definition of MD4, MD5
and SHA-1.
Algorithm. MD4.
Input: A message $x \in \{0, 1\}^*$.
Output: A hash value $H \in \{0, 1\}^{128}$.

Constants and round functions:
1. $h \leftarrow (67452301, \text{EFCDBB89}, \text{98BADCFE}, 10325476)$.
   
   $K_j \leftarrow \begin{cases} 
   00000000, & 0 \leq j < 16, \\
   5A827999, & 16 \leq j < 32, \\
   6ED9EBA1, & 32 \leq j < 47, \\
   j, & 0 \leq j < 16, \\
   j_1j_0j_3j_2, & 16 \leq j < 32, \\
   j_0j_1j_2j_3, & 32 \leq j < 48, 
   \end{cases}$
   (32 bits of $\sqrt{2}$)
   (32 bits of $\sqrt{3}$)

   $z[j] = \begin{cases} 
   j_1j_0j_3j_2, & 16 \leq j < 32, \\
   j_0j_1j_2j_3, & 32 \leq j < 48, 
   \end{cases}$

   where $j_i$ denotes bit $i$ of the binary representation of $j$.

   $s[0..15] = [3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19],$
   $s[16..31] = [3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13],$
   $s[32..47] = [3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15].$

   $f_j(B, C, D) = \begin{cases} 
   (B \land C) \lor (\overline{D} \land D), & 0 \leq j < 16, \\
   (B \land C) \lor (C \land D) \lor (D \land B), & 32 \leq j < 48. 
   \end{cases}$

Precalculations:
2. Padding: $\bar{x} \leftarrow x1|0^d|\langle |x| \rangle_{64}$ with $0 \leq d < 512$ such
   that $|\bar{x}|$ is a multiple of $512 = 16 \cdot 32$.
3. Cut $\bar{x}$ into 32-bit words: $\bar{x} = x_0x_1x_2 \ldots x_{16m-1}$.
4. Initialize: $(H_1, H_2, H_3, H_4) \leftarrow h$.

Main calculations:
5. For $i = 0..m-1$ do 6–10
6. $(A, B, C, D) \leftarrow (H_1, H_2, H_3, H_4)$.
7. For $j = 0..47$ do 8–9
8. $t \leftarrow (A + f_j(B, C, D) + x_{2[j]} + K_j) \otimes s[j]$.
9. $(A, B, C, D) \leftarrow (D, t, B, C)$.
10. $(H_1, H_2, H_3, H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$.
11. Return $H_1|H_2|H_3|H_4$. 

Michael Nüksen
Electronic passports and biometrics, December 7, 2006
Algorithm. MD5.
Input: A message $x \in \{0, 1\}^*$. 
Output: A hash value $H \in \{0, 1\}^{128}$.

Constants and round functions:
1. $h \leftarrow (67452301, \text{EFCADB89}, \text{98BADCFE}, \text{10325476})$.
   
   $K_j \leftarrow \text{32 Bits von } |\sin(j + 1)|$.
   
   $z[j] = \begin{cases} j, & 0 \leq j < 16, \\
   j_1j_0j_3j_2, & 16 \leq j < 32, \\
   j_0j_1j_2j_3, & 32 \leq j < 48, 
   \end{cases}$
   
   where $j_i$ denotes bit $i$ of the binary representation of $j$.

   $s[0..15] = [7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22]$, 
   
   $s[16..31] = [5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20]$, 
   
   $s[32..47] = [4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23]$, 
   
   $s[48..63] = [6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21]$.

   $f_j(B, C, D) = \begin{cases} (B \land C) \lor (B \land D), & 0 \leq j < 16, \\
   (B \land D) \lor (C \land D), & 16 \leq j < 32, \\
   B \oplus C \oplus D, & 32 \leq j < 48, \\
   C \oplus (B \lor D), & 48 \leq j < 64. \end{cases}$

Precalculation:
2. Padding: $\tilde{x} \leftarrow x|1|0^d|\lfloor |x|/64 \rfloor$ with $0 \leq d < 512$ such that $|\tilde{x}|$ is a multiple of $512 = 16 \cdot 32$.
3. Cut $\tilde{x}$ into 32-bit words: $\tilde{x} = x_0x_1x_2 \ldots x_{16m-1}$.
4. Initialize: $(H_1, H_2, H_3, H_4) \leftarrow h$.

Main calculation:
5. For $i = 0..m - 1$ do 6–10
6. $(A, B, C, D) \leftarrow (H_1, H_2, H_3, H_4)$.
7. For $j = 0..63$ do 8–9
8. $t \leftarrow (A + f_j(B, C, D) + x_{z[j]} + K_j) \odot s[j]$.
9. $(A, B, C, D) \leftarrow (D, B + t, B, C)$.
10. $(H_1, H_2, H_3, H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$.
11. Return $H_1 | H_2 | H_3 | H_4$.
Algorithm. SHA-1.
Input: A message $x \in \{0, 1\}^*$. 
Output: A hash value $H \in \{0, 1\}^{160}$.

Constants and round functions:
1. $h \leftarrow (67452301, \text{EFCDA89}, \text{98BADCFE}, 10325476, \text{C3D2E1F0})$.
2. $K_j \leftarrow \begin{cases} 
5A827999, & 0 \leq j < 20, 
6ED9EBA1, & 20 \leq j < 40, 
8F1BCDC, & 40 \leq j < 60, 
\text{CA62C1D6}, & 60 \leq j < 80.
\end{cases}$

$f_j(B, C, D) = \begin{cases} 
(B \land C) \lor (\overline{B} \land \overline{D}), & 0 \leq j < 20, 
B \oplus C \oplus D, & 20 \leq j < 40, 
(B \land C) \lor (C \land D) \lor (D \land B), & 60 \leq j < 80.
\end{cases}$

Precalculations:
2. Padding: $\tilde{x} \leftarrow x | 1 | 0^d | \langle |x| \rangle_6$ mit $0 \leq d < 512$ so, that $|\tilde{x}|$ is a multiple of $512 = 16 \cdot 32$.
3. Cut $\tilde{x}$ in 32-bit words: $\tilde{x} = x_0 x_1 x_2 \ldots x_{16m-1}$.
4. Initialize: $(H_1, H_2, H_3, H_4, H_5) \leftarrow h$.

Main calculation:
5. For $i = 0..m-1$ do 6-13
6. For $j = 0..15$ do $W_j \leftarrow x_{16i+j}$.
7. For $j = 16..79$ do
8. $W_j \leftarrow (W_{j-3} \oplus W_{j-8} \oplus W_{j-14} \oplus W_{j-16}) \otimes 1$.
9. $(A, B, C, D, E) \leftarrow (H_1, H_2, H_3, H_4, H_5)$.
10. For $j = 0..79$ do 11-12
11. $t \leftarrow A \otimes 5 + f_j(B, C, D) + E + W_j + K_j$.
12. $(A, B, C, D, E) \leftarrow (t, A, B \otimes 30, C, D)$.
13. $(H_1, H_2, H_3, H_4, H_5) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D, H_5 + E)$.
14. Return $H_1 | H_2 | H_3 | H_4 | H_5$. 

Michael Nüsken

Electronic passports and biometrics, December 7, 2006
### OSI Model

<table>
<thead>
<tr>
<th>Level</th>
<th>OSI</th>
<th>TCP/IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
<td>Application</td>
</tr>
<tr>
<td>6</td>
<td>Presentation</td>
<td>Not present in the model</td>
</tr>
<tr>
<td>5</td>
<td>Session</td>
<td>Transport</td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
<td>Internet</td>
</tr>
<tr>
<td>3</td>
<td>Network</td>
<td>Host-to-network</td>
</tr>
<tr>
<td>2</td>
<td>Data link</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1-21. The TCP/IP reference model.

---

![Protocol Layers Diagram](image)

Figure 1-22. Protocols and networks in the TCP/IP model initially.
AH - authentication header
ESP - encapsulating security protocol

<table>
<thead>
<tr>
<th>Task</th>
<th>AH</th>
<th>ESP except</th>
<th>ESP both</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access control</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Connectionless integrity</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Data origin authentication</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Rejection of replayed packets</td>
<td>+</td>
<td>(++)</td>
<td>+</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Limited traffic flow confidentiality</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

SA - security association
4 - SPI (security parameter index) (32 bit #)
contains:
- IP destination address
- security protocol identifier
- sequence number counter (32 bit) 
- sequence counter overflow
- anti-replay
- AH info: authentication algorithm, keys, key life time...
ESP info: Encryption algorithm, key sizes, life times, initial values...

- life time of SA (usually 8 hours)
- IPSec protocol mode: tunnel, transport, wildcard
- MTU: max. packet size & aging variables

SPD - security policy database
SAP -> entries for each SA
SPD -> allowed IPs

Authentication header

<table>
<thead>
<tr>
<th># octets</th>
<th>next header</th>
<th>payload length</th>
<th>unused</th>
<th>SPI (Security Parameter Index)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>sequence number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>authentication data</td>
</tr>
</tbody>
</table>

data

authentication data
= signature
= essentially this is a secure hash value keyed.
Encapsulating Security Protocol header

<table>
<thead>
<tr>
<th># octets</th>
<th>( \text{SPI (Security Parameters Index)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>sequence number</td>
</tr>
<tr>
<td>variable</td>
<td>IV (Initialization vector)</td>
</tr>
<tr>
<td>variable</td>
<td>data</td>
</tr>
<tr>
<td>variable</td>
<td>padding</td>
</tr>
<tr>
<td>1</td>
<td>padding length (in units of octets)</td>
</tr>
<tr>
<td></td>
<td>next header/protocol type</td>
</tr>
<tr>
<td>variable</td>
<td>authentication data</td>
</tr>
</tbody>
</table>

SA, Security Association
 simplex protected connection

SAD = database of all inbound & outbound SAs

SPD = database of rules: which packets to < DISCARD < BYPASS < PROTECT

AH/ESP = security envelopes
Tunnel and transport mode

Situation:

- 131.59.7.3
- 131.55.11.7
- 131.20.4.7
- 131.40.7.4
Transport mode is best suited for station-to-station connections.

Tunnel mode also allows to connect two subnets.

Side remark:

AH protects IP header. It is unclear why this is necessary; but even if so ESP in tunnel mode would provide that!

NAT — network address translation

AH protects destination IP address, so exchanging it destroys the signature.
IPv4 / IPv6

IPv4

<table>
<thead>
<tr>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bits</td>
</tr>
<tr>
<td>4 bits</td>
</tr>
<tr>
<td>1 octet</td>
</tr>
<tr>
<td>2 octets</td>
</tr>
<tr>
<td>2 octets</td>
</tr>
<tr>
<td>3 bits</td>
</tr>
<tr>
<td>13 bits</td>
</tr>
<tr>
<td>1 octet</td>
</tr>
<tr>
<td>2 octets</td>
</tr>
<tr>
<td>4 octets</td>
</tr>
<tr>
<td>4 octets</td>
</tr>
<tr>
<td>variable</td>
</tr>
</tbody>
</table>

version
header length (in 4-octet units)
type of service
length of header plus data in this fragment
packet identification
flags (don't fragment, and last fragment)
fragment offset
hops remaining, known as TTL (time to live)
protocol (next header)
header checksum
source address
destination address
options

50 = ESP
51 = AH

IPv6

<table>
<thead>
<tr>
<th># octets</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>variable</td>
</tr>
</tbody>
</table>

version (4 bits)
type of service
flow label
payload length
next header
hops remaining
source address
destination address

IPv6

<table>
<thead>
<tr>
<th># octets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>variable</td>
</tr>
</tbody>
</table>

next header
length of this header
data for this header
authenticates all immutable fields - IP and the data.

IPv4

- immutable: type of service
- immutable: fragment offset
- immutable: fragment offset
- immutable: fragm offset
- immutable: fragm offset
- fragment offset
- fragment offset
- fragment length
- fragment length
- always 0,
- but similar to payload length

IPv6

- type of each option indicates whether it's immutable or not:
- type of service: immutable

Other things:
- destination address
- immutable but predictable
- to use predicted value for signature

ESP

can do encryption and optionally authentication.

It does not include any IP header info in the signature!

→ You can use 'null-encryption' if you don't want to encrypt.
IPsec: more details

Say we look again at tunnel mode:

Before:

```
| IP hdr |
```

New IP hdr | IPsec | IP | data

Specifically, ESP: ESP hdr.

```
| Security Parameters Index (SPI) |
| Sequence Number |
| Payload Data* (variable) |
| Padding (0-255 bytes) |
| 32-bit counter |
| Payload Length |
| Next Header |
| Integrity Check Value-ICV (variable) |
```

Figure 1. Top-Level Format of an ESP Packet

* If included in the Payload field, cryptographic synchronization data, e.g., an Initialization Vector (IV, see Section 2.3), usually is not encrypted per se, although it often is referred to as being part of the ciphertext.
Table 1. Separate Encryption and Integrity Algorithms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI</td>
<td>4</td>
<td>M</td>
<td>Y</td>
<td>Y</td>
<td>plain</td>
</tr>
<tr>
<td>Seq# (low-order bits)</td>
<td>4</td>
<td>M</td>
<td>Y</td>
<td>Y</td>
<td>plain</td>
</tr>
<tr>
<td>IV</td>
<td>variable</td>
<td>O</td>
<td>Y</td>
<td>Y</td>
<td>plain</td>
</tr>
<tr>
<td>Padding</td>
<td>0-255</td>
<td>M</td>
<td>Y</td>
<td>Y</td>
<td>cipher[3]</td>
</tr>
<tr>
<td>Pad Length</td>
<td>1</td>
<td>M</td>
<td>Y</td>
<td>Y</td>
<td>cipher[3]</td>
</tr>
<tr>
<td>Next Header</td>
<td>1</td>
<td>M</td>
<td>Y</td>
<td>Y</td>
<td>cipher[3]</td>
</tr>
<tr>
<td>Seq# (high-order bits)</td>
<td>4</td>
<td>if ESN [5]</td>
<td>Y</td>
<td>Y</td>
<td>not xmted</td>
</tr>
<tr>
<td>ICV Padding</td>
<td>variable if need</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>not xmted</td>
</tr>
<tr>
<td>ICV</td>
<td>variable M [6]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>plain</td>
</tr>
</tbody>
</table>

[1] M = mandatory; O = optional; D = dummy
[2] If tunnel mode -> IP datagram
   If transport mode -> next header and data
[3] ciphertext if encryption has been selected
[4] Can be used only if payload specifies its "real" length
[5] See section 2.2.1
[6] mandatory if a separate integrity algorithm is used

---

How to apply this encr. & auth?

1. Encapsulate the transport/tunnel mode as payload.
2. Add padding (TFC and error padding, if needed) as needed/wanted.
3. Encrypt as specified by SA and IV.
4. Sign (authenticate) the (encrypted) packet including ICV padding, ESN, but excluding ICV.

So here:

First encrypt
Then authenticate/sign
A signature must always protect the plaintext.

One solution: first authenticate, then encrypt.

One advantage of the other order: we can check integrity first and save decryption if it fails.

Note: encrypted text + encryption key also fixes/identifies the plain text uniquely.

Second solution: first encrypt, then authenticate this + the keys.

ESP does that in a weak sense: the authenticated part includes the SPI, yet not the keys itself.
Encryption and authentication algorithms

**RFC 4305**

**Encryption algorithms:**

**MUST**

- NULL

**MUST**

- Triple DES - CBC (RFC 2456)

**SHOULD**

- AES - CBC with 128 bit key (RFC 3602)

**SHOULD**

- AES - CTR

**SHOULD NOT**

- DES - CBC (RFC 2405)

**Authentication algorithms**

**MUST**

- HMAC - SHA-1 - 96 (RFC 2404)

**MUST**

- NULL

**SHOULD**

- AES - XCBC - MAC - 96

**MAY**

- HMAC - MD5 - 96

**RFC 3602**

AES-CBC encryption

```
N → P1 → P2 → P3 → P4
   |   |   |   |
   |   |   |   |
   |   |   |   |+
   |   |   |   |
   |   |   |   | AES
   |   |   |   |
   |   |   |   | C1
   |   |   +   +
   |   |     |+
   |   |     | |
   |   |     | AES
   |   |     | C2
   |   |     +   +
   |   |       |+
   |   |       | |
   |   |       | AES
   |   |       | C3
   |   |       +   +
   |   |         |+
   |   |         | |
   |   |         | AES
   |   |         | C4
```
RFC 3686  AES - CTR mode encryption

\[ P_i = \text{AES} \]

\[ C_i \]

\[ \text{advantage: much easier to reproduce} \]

\[ \text{you'd need to decrypt in order (sufficient to know the position i)} \]

RFC 3566  AES-XCBC-MAC-96 authentication

\[ M_1 \]

\[ M_2 \]

\[ 0 = E_0 \]

\[ K_1 = \text{AES}_k (0x0101 \ldots 01) \]

\[ K_2 = \text{AES}_k (0x0202 \ldots 02) \]

\[ K_3 = \text{AES}_k (0x0303 \ldots 03) \]

where \( K_2 \) is used when there is no padding and \( K_3 \) for other modes.
Ask yourself: what would happen if somebody tries to change the message? Can the attacker get the same signature (ICV)?

Last key specifically vulnerable.

RFC 2404 / 2104 HMAC-SHA1-96

\[
\text{SHA1}(K \oplus \text{ipad}, \text{SHA1}(K \oplus \text{ipad}, \text{msg}))
\]

\[
\begin{align*}
&\text{0x56} \\
&\text{repeated} \\
&\text{0x5c} \\
&\text{repeated}
\end{align*}
\]

Take the first 96 bits of this.

These lecture notes contain a description of SHA1 on an earlier page → see there.
<table>
<thead>
<tr>
<th><strong>Address Information</strong></th>
<th><strong>Connection Information</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Client: 131.220.240.254</td>
<td>Entry: VPN@BIT-cosec</td>
</tr>
<tr>
<td>Server: 131.220.240.255</td>
<td>Time: 0 day(s), 02:30:41</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Bytes</strong></th>
<th><strong>Crypto</strong></th>
<th><strong>Transport</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Received: 28469355</td>
<td>Encryption: 256-bit AES</td>
<td>Transparent Tunneling: Active on TCP port 10000</td>
</tr>
<tr>
<td>Sent: 62937820</td>
<td>Authentication: HMAC-SHA1</td>
<td>Local LAN: Enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Packets</strong></th>
<th><strong>Compression</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypted: 80253</td>
<td>None</td>
</tr>
<tr>
<td>Decrypted: 62291</td>
<td></td>
</tr>
<tr>
<td>Discarded: 95</td>
<td></td>
</tr>
<tr>
<td>Bypassed: 147</td>
<td></td>
</tr>
</tbody>
</table>
Internet Key Exchange version 2

Initial exchange comprises 4 messages, only the first two are not encrypted.

Initiator

\[ \text{Hdr, SA_i, UE_i, Ni} \rightarrow g^q \rightarrow \]

Responder

Initiator's

\( \text{SA proposals} \)

Initiator's

\( \text{Nounce} \)

DH group

\[ \mathbb{Z}_p \times \mathbb{Z}_q \]

Group 1: 768-bit MODP

\[ p = 2^364 - 2^95 + 2 \times q \quad (L_2 \pi + 179666) \]

\( g = 2 \)

(Too small in practice, only for DES-CBC)

Group 2: 1024-bit MODP

\[ p = 2^{1064} - 2^60 + 2^6 \times q \quad (L_2 \pi + 129095) \]

\( g = 2 \)
This establishes IKE-SA.

All further messages are protected by keys derived from this (and possibly further DH key exchanges; REKEY-1gs).

Rekeying possible at any time by any partner.

Any further exchange consists of a request and a response.

CREATE_CHILD_SA

Hdr, SKh' [N, 1] SA, Nc, [IKEi, ] [TSi, TSr]

opt. REKEY-SA

identifying the SA being rekeyed

(proposed after) Nounce in preferred traffic selectors

opt. proposed traffic selectors

Hdr, SKh SA, Mr, [IKEi, ] [TSi, TSr]
If the predicted group is not the chosen one an informational msg with the chosen group is send back and the initiator has to retry - with same proposals.

INFORMATIONAL exchanges

- for notifications (error msg),
  delete,
  configuration.

→ always msg & response.
  so empty msg is interpreted as "are you still there?"

→ always protected under IKE-SA

Eg. if a connection should be closed:

ESP SA, AH SA exist in pairs
→ both have to be closed.

close incoming ESP-SA DELETE close outgoing SA

close outgoing SA DELETE close incoming SA
Node crash or similar

- incoming SPI unknown
- if another IKE SA exists with that sender
  - may send informational msg using that
  - else may send unprotected notification.

The other node MUST NOT trust this kind of answers. Instead sock half closed SAs
are considered anomalous, and the other node should retry some times.

The other node sends empty into msg.
If the node responds it STILL ALIVE
If the node does not respond in a dozen (or so) attempts: assume SA is dead and close it.

Never delete an SA because of unprotected information.
IPSEC & IKE

MICHAEL NÜSKEN

25 June 2007

Before all: we are talking about a collection of protocols. Each partner of the exchange has to keep some information on the connection. This is in our context called the security association (SA). It contains specification about the algorithms that should be used for encryption and authentication, it contains keys for these, it may contain traffic selectors (filtering rules), and more. Each SA manages a simplex connection for one type of service. In each direction there will be an SA for the key exchange (IKE_SA) and one for the encapsulating security payload or for the authentication header. So each partner has to maintain at least four SAs. Such an SA is selected by an identifier, the so-called security parameter index (SPI). It is chosen randomly but so that it is unique.

1. IPsec

The secure internet protocol modifies the internet protocol slightly. We have the choice between transport and tunnel mode. In tunnel mode, an IP packet

<table>
<thead>
<tr>
<th>IP header</th>
<th>IP payload</th>
</tr>
</thead>
</table>

is wrapped in with a new IP header and an IPsec header to

<table>
<thead>
<tr>
<th>new IP header</th>
<th>IPsec header</th>
<th>IP header</th>
<th>IP payload</th>
</tr>
</thead>
</table>

In transport mode, only the IPsec header is added:

<table>
<thead>
<tr>
<th>IP header</th>
<th>IPsec header</th>
<th>IP payload</th>
</tr>
</thead>
</table>

There are two types of IPsec headers: the encapsulating security payload (ESP) and the authentication header (AH).
1.1. **IPsec encapsulating security payload.** The ESP specifies that and how its payload is encrypted and (optionally) authenticated. Actually, this ‘header’ is split into a part before and one after the data:

<table>
<thead>
<tr>
<th>Security Parameter Index (SPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
</tr>
<tr>
<td>IV (optional)</td>
</tr>
<tr>
<td>Payload data [variable]</td>
</tr>
<tr>
<td>TFC padding [optional, variable]</td>
</tr>
<tr>
<td>Padding (0-255 octets)</td>
</tr>
<tr>
<td>Padding length</td>
</tr>
<tr>
<td>Integrity Check Value (ICV) [variable]</td>
</tr>
</tbody>
</table>

The security parameter index identifies the SA and thus all necessary algorithms and key material. To create the secured packet from the original one, it is first padded. Padding is used to enlarge the data length to a multiple of a block size that might be associated with the encryption. Traffic flow confidentiality (TFC) padding can be used to disguise the real size of the packet. Then the data is encrypted; in tunnel mode including the old IP header. To be precise, all the information from Payload data to Next header is encrypted. Next, a message authentication code is calculated for this encrypted text and security parameter index, sequence number, initialization vector (IV) and possibly further padding; actually the message authentication code covers the entire packet but the header and the integrity check value plus the extended sequence number and integrity check padding if any.

1.2. **IPsec authentication header.** The AH authenticates its payload and also parts of the IP header. (Yes, this does violate the hierarchy.)
2. Internet key exchange (version 2)

Any message in the internet key exchange starts with a header of the form

<table>
<thead>
<tr>
<th>Next payload</th>
<th>Major version</th>
<th>Minor version</th>
<th>Exchange type</th>
<th>X</th>
<th>I</th>
<th>V</th>
<th>R</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Message ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Length</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Clearly, the version is 2.0 with the present drafts (major version: 2, minor version: 0). The flags X are reserved, the I(nitiator) bit is set whenever the message comes from the initiator of the SA, the V(ersion) bit is set if the transmitter can support a higher major version, the R(esponse) bit is set if this message is a response to a message with this Message ID. The header is usually followed by some payloads like

<table>
<thead>
<tr>
<th>Next payload</th>
<th>Critical</th>
<th>Reserved(0)</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The C(ritical) bit indicates that the payload is critical. In case the recipient does not support a critical payload it must reject the entire message. A non-critical payload can be simply skipped. All the payloads defined in RFC4306 are to handled as critical ones whatever the C bit says.
2.1. Initial exchange.

Protocol 2.1. IKE-SA-INIT.

1. Prepare SAi1, the four lists of supported cryptographic algorithms for Diffie-Hellman key exchange (groups), for the pseudo random function used to derive keys, for encryption, and for authentication. Guess the group for Diffie-Hellman and compute KEi = g^a.
   Choose a nonce Ni.
2. Choose SAr1 from SAi1 unless no variant is supported.
Compute $\text{KE}_{\text{r}} = g^b$ if the group was guessed correctly. (Otherwise send:

$$\text{Hdr, N(INVALID}_\text{ KE PAYLOAD, group)}$$

.)

Choose a nonce $\text{Nr}$.

3. Both parties now derive the session keys. We assume that $\text{prf}$ is the selected pseudo random function which gets a key and a bit string as input.

$$\text{SKEYSEED} = \text{prf}(N_i | \text{Nr}, g^{ab})$$

$$\text{SK_d} | \text{SK}_\text{ai} | \text{SK}_\text{ar} | \text{SK}_\text{ei} | \text{SK}_\text{er} | \text{SK}_\text{pi} | \text{SK}_\text{pr} = \text{prf}+(\text{SKEYSEED, N}_i | \text{Nr} | \text{SPI} | \text{SPIr})$$

where $\text{prf}+(K, S) = T_1 | T_2 | T_3 | \ldots$, and $T_1 = \text{prf}(K, S | 0x01)$, $T_i = \text{prf}(K, T_{i-1} | S | i)$ for $i > 1$.

$\text{SK_d}$ is used for the derivation of keys in a child SA. $\text{SK}_\text{ai}$ and $\text{SK}_\text{ei}$ are used for authenticating and encrypting messages sent by the initiator, $\text{SK}_\text{ar}$ and $\text{SK}_\text{er}$ for messages sent by the responder.

4. The initiator sends its identity $\text{ID}_i$, optionally one or more certificates $\text{CERT}$, a certificate request $\text{CERTREQ}$ (possibly including a list of trusted CAs), and optionally the responders identity $\text{ID}_r$ (it may be that the responder serves multiple identities ‘behind’ it).

Further she computes an authentication $\text{AUTH}$ (using the key from the first $\text{CERT}$ payload) for the entire first message concatenated with the responders nonce $\text{Nr}$ and the value $\text{prf}$(\text{SK}_\text{pi}, \text{ID}_i)$. The authentication method can be RSA digital signature (1), shard key message integrity code (2), or DSS digital signature (3).

The initiator starts to negotiate a child SA in $\text{SA}_2$ with proposed traffic selectors $\text{TS}_i$, $\text{TS}_r$. 

$$\text{Hdr, SK} \left\{ \text{ID}_i, [\text{CERT, }] \right\}$$

$$\text{[CERTREQ, ]}$$

$$\left\{ [\text{ID}_r,] \right\}$$

$$\text{AUTH, } \text{SA}_2, \text{TS}_i, \text{TS}_r$$
5. The responder sends its identity IDr, certificate(s). He computes an authentication AUTH for the entire second message concatenated with the initiator’s nonce Ni and the value prf(SK_pr, IDr). Further he supplies the answer SAr 2 to the child SA creation and sends the accepted traffic selectors TSi, TSr.

\[ \text{KEYMAT} = \text{prf+}(SK_d, Ni | Nr) \]

2.2. Creating additional child SAs. Further children can be created under this IKE_SA using a CREATE_CHILD_SA exchange:

In case a CHILD_SA shall be rekeyed the notification payload N of type REKEY_SA specifies which SA is rekeyed. This can be used to established additional SAs as well as to rekey ages ones. Create new ones and afterwards delete the old ones. Also the IKE_SA can be rekeyed similarly.

In a CREATE_CHILD_SA exchange including an optional Diffie-Hellman exchange new keying material uses also the new Diffie-Hellman key \( g^{ir} \), it is concatenated left to the nonces. (Though the Diffie-Hellman key exchange is optional, it is recommended to either used it or at least to limit the number of uses of the original key.)

2.3. Denial of Service. If the server has a lot of half open connections (ie. the first message arrived, the second was sent but the third message is pending) it may choose to send a cookie first. (In order to defeat a denial of service attack.) It is suggested to use a stateless cookie consisting of a version identifier and a hash value of the initiator’s nonce Ni, her IP IPi, her security parameter index SPIi and some secret:

\[ \text{Cookie} = \text{verID} | \text{hash(Ni, IPi, SPIi, secret_{verID})} \]
This way the secret can be exchanged periodically, say every second, and the server only needs to store the last few (randomly) generated secrets.

The authentication AUTH then refers to the second version of the corresponding message, so the one including the cookie or responding to that, respectively. So the protocol becomes:

2.4. Extended authentication protocols. The initiator may leave out AUTH and thereby tell the responder that she wants to perform an extensible authentication which is then carried out immediately.

2.5. IP compression. The parties can negotiate IP compression.

2.6. ID payload. The ID payload

<table>
<thead>
<tr>
<th>Next payload</th>
<th>Reserved(0)</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID type</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Identification data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

can be an IP address (ID type 1), a fully-qualified domain name string (2), a fully-qualified RFC822 email address string (3), an IPv6 address (5), an ASN.1 X.500 Distinguished Name [X.501] (9), an ASN.1 X.500 general name [X.509] (10), a vendor specific information (11).

2.7. CERT payload. The CERT payload

<table>
<thead>
<tr>
<th>Next payload</th>
<th>Reserved(0)</th>
<th>Payload length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cert encoding</td>
<td>Certificate data</td>
<td>Certificate data</td>
</tr>
</tbody>
</table>
can be encoded in various widely used formats. Note that it can also carry revocation lists.

3. IKE version 1

The version 1 of the internet key exchange distinguishes between a main mode and an aggressive mode. Further it allows four variants in each mode depending on the desired type of authentication. Authentication can be based on

- public signature keys,
- public encryption keys, original protocol,
- public encryption keys, revised protocol, or
- a pre-shared secret.

We only give the bare protocol summaries here, using notation similar to the one used for version 1. (They are not based on RFC240x but on the book Kaufmann et al. 2002.)

3.1. Main mode, public signature keys.

3.2. Aggressive mode, public signature keys.
3.3. Main mode, public encryption keys, original protocol.

\[ S_{Ai} \rightarrow S_{Ar} \]

\[ K_{Ei}, \{N_i\}_B, \{ID_i\}_B \]

\[ K_{Er}, \{N_r\}_A, \{ID_r\}_A \]

\[ SK = f(g^{ab}, N_i, N_r) \]

\[ SK \{AUTH, [CERT]\} \]

3.4. Aggressive mode, public encryption keys, original protocol.

\[ S_{Ai}, K_{Ei}, \{N_i\}_B, \{ID_i\}_B \rightarrow S_{Ar}, K_{Er}, \{N_r\}_A, \{ID_r\}_A, AUTH \]

\[ AUTH \]

3.5. Main mode, public encryption keys, revised protocol.

\[ S_{Ai} \rightarrow S_{Ar} \]

\[ K_A = \text{hash}(N_i, \text{cookiei}) \]

\[ \{N_i\}_B, K_A \{KEi\}, K_A \{ID_i\}, K_A \{CERT\} \]

\[ K_B = \text{hash}(N_r, \text{cookier}) \]

\[ \{N_r\}_A, K_B \{KEr\}, K_B \{IDr\} \]

\[ SK = f(g^{ab}, N_i, N_r, \text{cookiei}, \text{cookier}) \]

\[ SK \{AUTH\} \]
3.6. Aggressive mode, public encryption keys, original protocol.

\[ K_A = \text{hash}(N_i, \text{cookiei}) \]
\[ S_{Ai}, \{N_i\}_{Bob}, K_A \{KEi\}, K_A \{IDi\}, K_A \{CERT\} \]

\[ K_B = \text{hash}(N_r, \text{cookiei}) \]
\[ S_{Ar}, \{N_r\}_{Alice}, K_B \{KEr\}, K_B \{IDr\}, \text{AUTH} \]

\[ SK = f(g^{ab}, N_i, N_r, \text{cookiei}, \text{cookiei}) \]
\[ SK \{\text{AUTH}\} \]

3.7. Main mode, pre-shared secret.

\[ SA_{Ai} \]
\[ SA_{Ar} \]

\[ KEi, N_i \]
\[ KEr, N_r \]

\[ SK = f(\text{secret}, g^{ab}, N_i, N_r, \text{cookiei}, \text{cookiei}) \]
\[ SK \{IDi, \text{AUTH}\} \]

\[ SK \{IDr, \text{AUTH}\} \]

3.8. Aggressive mode, pre-shared secret.

\[ SA_{Ai}, KEi, N_i, IDi \]
\[ SA_{Ar}, KEr, N_r, IDr, \text{AUTH} \]

\[ \text{AUTH} \]

\[ SK = f(\text{secret}, g^{ab}, N_i, N_r, \text{cookiei}, \text{cookiei}) \]

References


Michael Nüsken
b-it, Bonn, Germany
History of IKE:

- NSA proposal: ISAKMP
  - only framework
  - ruled out both candidates
  - IETF could take up the development.
    - OAKLEY, SKEME... (new drafts)
    - IKE puts into ISAKMP.

- Problem:
  - no clear design
  - too many variants
    - > 150 pages, >3 RFCs
    - partially very unclear
    - difficult to read.
Hey, Sri... Cerf...

Of. Hn. Ayr... Cerfega...

Guess

as odyssey or odyssey

rambles & sherry bars

a functionally all his new

next varying

"phrase 2"

create古典主义 = 2 ms15. 4 ms6.5, "phrase 5"

initial exchange: 1 way,

any request gets a response

Jesus, simple rules

11762; 7568
Fact-finding committees

1. IKEv1 aggressive mode

2. IKEv1 main mode

3. IKEv2

Look at: Pros & Cons!

Specific questions:

1. SECURITY, SECURITY, SECURITY, SECURITY.

2. Session key agreement
   - How long? Random?
   - Do both parties contribute to it?
   - Man in the middle

3. Perfect forward security
   - Can an attacker given the long-term secrets and all messages decrypt?
   - Escrow obilege
   - Is the conversation secret even if the long-term secrets are known to the attacker in advance?
(3) Denial of Service

(4) End point identifiers hiding

   - Does an eavesdropper get info about identities?
   - Does an active attacker get identification information from Initiator (client) or responder (server)?

(5) Live partner reassurance

   → Replay?

(6) Plausible denialability

   Does the protocol log prove that:
   - Alice talked?
   - Bob talked?
   - Alice talked to Bob?
   - Bob talked to Alice?

(7) Stream protection

   How is a logical data stream protected? → confidential & authentic & integrity

(8) Negotiating crypto parameters

   → Pros
   → Cons
**Main Mode**

- **Pros**
  - Nonces: prevent replay attacks (life partner recons)
  - Active attacks (i.e., man-in-the-middle) would modify message 5 or 6 → detectable
  - Certificates → plausible deniability

- **Cons**
  - No stream protection (no sequence numbers)
  - Active attacks: could reveal certificate content → confidentiality partially compromised (no endpoint identifiers hiding, esp. for client)
  - Crypto parameter exchange: CP could be modified to enforce usage of a weak algorithm
  - If attacker knows a and b, no perfect forward security, all messages can be decrypted (same for escrow foliage)

**Pro-Shared Secret**

- Man-in-the-middle/active attacks do not work anymore: key depends on preshared secret
- Certificates are no longer necessary → simpler infrastructure

**Pre-Shared Secret**

- Shared secret needs to be exchanged in a secure way (i.e., different channels)
- DOS still possible
## IKEv1 aggressive mode

<table>
<thead>
<tr>
<th><strong>PROs</strong></th>
<th></th>
<th><strong>CONS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>less #messages =&gt; faster</td>
<td></td>
<td>authentication without using a session key</td>
</tr>
<tr>
<td>authentication with a pre-shared key allows for a wide range of identifiers (not only IP addresses)</td>
<td></td>
<td>not all modes hide the identities(?)</td>
</tr>
<tr>
<td>reply not possible, because we use nonces</td>
<td></td>
<td>original protocol (pke): does not use cookies</td>
</tr>
<tr>
<td>revised protocol (public key encryption): cookies for counter measuring DOS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IKEv2

0) see below ↓

1) key exchange: two Diffie-Hellman groups / size of group min 768 bits.
   Randomness: Pseudo-Random Function
   * IKEv2 has one single four-message exchange
   * no entity-in-the-middle with Certificate.

3) * it does, but better than Version(1) in terms of DoS attack.
   * InSecure Because

4) * An outside Attacker can't get any info when listening to a conversation
   An active attacker can REQ first & get the certificate

5) Due to randomness of b the chance to be able to reuse an old conversation is minute

2) Since a & b are short-term secrets: No
SSL  Secure Socket Layer
TLS  Transport Layer Security

First steps: 1994 (?) Netscape

Decision:

Reasons:  
- Wanted fast, easily embeddable solution.
- Should link application (Browser) to application (webserver) rather than station to station
- Encryption maybe, but definitely authentication of server and optionally of client needed.

IPsec was not there yet.
Some shape

initial handshake (\(\equiv\) IKE SA init)

Protocol 19.1. (simplified) SSLv3/TLS

\[ K = f(S, R_{Alice}, R_{Bob}) \]

\[ S = \text{pre master key} \quad (\text{bit}) \]

\[ R_{Alice}/R_{Bob} = \text{random numbers} \quad (\text{bit}) \]

\[ K = \text{master key} \quad (\text{384 bits}) \]

\[ \text{hash}('\text{CLIENT}', K, \text{msg182}) \]

\[ \text{hash}('\text{SERVER}', K, \text{msg182}) \]

From \(K\) we derive:
- 2 encryption keys
- 2 authentication/ integrity keys
- 2 IV (for CBC mode...)

\[ \text{msg1} \]
\[ \text{msg2} \]
\[ \text{msg3} \]
\[ \text{msg4} \]
If a session-id was fixed, one TCP session may use the same keys using the "Session resumption.

![Diagram of session resumption protocol]

Protocol 19-3. Session resumption if both sides remember session-id

Further purpose: This allows to upgrade to higher security ciphers.

[Background: US export restriction on any cryptography using more than 40-bit keys in the symmetric scenario or more than 512-bit RSA...]

That restriction has been dropped in the meantime...

SSL fulfilled this restriction by offering modes that publish 88 of 128 bits secret key.

Another reason may be that Bob's policies have changed...

Why? How does Alice know?
SSL/TLS does not have to care about fragmentation, reassembly, ...

Note: shape of the protected record is:

\[ \text{Hdr}, \; \text{ENC}_{K_e}(m) \; \text{MAC}_{K_a}(m) \; \text{pad} \]
possible ciphers

<table>
<thead>
<tr>
<th>CipherSuite</th>
<th>Key Exchange</th>
<th>Cipher</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_NULL_WITH_NULL_NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>TLS_RSA_WITH_NULL_MD5</td>
<td>RSA</td>
<td>NULL</td>
<td>MD5</td>
</tr>
<tr>
<td>TLS_RSA_WITH_NULL_SHA</td>
<td>RSA</td>
<td>NULL</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_RSA_WITH_RC4_128_MD5</td>
<td>RSA</td>
<td>RC4_128</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_RSA_WITH_RC4_128_SHA</td>
<td>RSA</td>
<td>RC4_128</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_RSA_WITH_DEA_CBC_SHA</td>
<td>RSA</td>
<td>DEA_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>RSA</td>
<td>3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_DSS_WITH_DES_CBC_SHA</td>
<td>DH_DSS</td>
<td>DES_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA</td>
<td>DH_DSS</td>
<td>3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_RSA_WITH_DES_CBC_SHA</td>
<td>DH_RSA</td>
<td>DES_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>DH_RSA</td>
<td>3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_DSS_WITH_DES_CBC_SHA</td>
<td>DHE_DSS</td>
<td>DES_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>DHE_RSA</td>
<td>3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_RSA_WITH_DES_CBC_SHA</td>
<td>DHE_RSA</td>
<td>DES_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>DHE_RSA</td>
<td>3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_anon_WITH_RC4_128_MD5</td>
<td>DH_anon</td>
<td>RC4_128</td>
<td>MD5</td>
</tr>
<tr>
<td>TLS_DH_anon_WITH_DES_CBC_SHA</td>
<td>DH_anon</td>
<td>DES_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>TLS_DH_anon_WITH_3DES_EDE_CBC_SHA</td>
<td>DH_anon</td>
<td>3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
</tbody>
</table>

Key Exchange
Algorithm Description
DHE_DSS Ephemeral DH with DSS signatures
DHE_RSA Ephemeral DH with RSA signatures
DH_anon Anonymous DH, no signatures
DH_DSS DH with DSS-based certificates
DH_RSA DH with RSA-based certificates
NULL No key exchange
RSA RSA key exchange

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Type</th>
<th>Material Key</th>
<th>Expanded Material</th>
<th>IV Size</th>
<th>Block Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>Stream</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>IDEA_CBC</td>
<td>Block</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>RC2_CBC_40</td>
<td>Block</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>RC4_40</td>
<td>Stream</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>RC4_128</td>
<td>Stream</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>DES_40_CBC</td>
<td>Block</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>DES_CBC</td>
<td>Block</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>3DES_EDE_CBC</td>
<td>Block</td>
<td>24</td>
<td>24</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Our questions?

- Session key agreement:
  - Need PKI to verify server identity
  - Use HTTPS or email over SSL/TLS
  - Sponsors' and email client are usually delivered with built-in root certificates, so that we can easily verify certificates going to one of those.

And it's there.
- Perfect forward security / Escrow attack
  SSL seems to be vulnerable
to this attack
If S is used as in this top-level
view, we simply decrypt SSL bob
and derive all further keys as
necessary...

- Denial of Service
  - No extra protection
  - and this is not necessary
    because lowres lay eos
    will care for this.

- Endpoint identifier hiding
  - Server id is not hidden.
  - Client id is hidden as
    long as it closely inspects
    and verifies the server's
certicate.

- Live partner reassurance
  - Message ids and random
    numbers (used as nonces)
    protect from that

- Deniability?
  - Alice cannot prove that Bob hacked her.
  - Other way round: with login/password: NO:
Stream protection

- The keys and their use guarantee that all records belong to the same session.

- Negotiate crypto parameters:
  - Yes, they are.
  - Downgrade? In the first place: yes, but have to forge msg 4.

Certificates may contain upgrade information allowing the client to resume the session with better ciphers.

- Use of version #5.

Apart from SSLv2, we have control here.
SSH

SSL → SSH
    +---- Appl. ---+  → TCP
         |              |
         +---- Data/IP +  → Phys/Link

1995 Tata Ylönen
started as a secure replacement of
remote terminals (telnet, rsh, ...)

1996 ssh 2

1993 openSSH → ssh daemon

Now:
  - sftp, scp: files transfer
  - forward X11
  - tunnel TCP/IP

Identification:
  - RSA certificate
  (rather than X.509 certificate or similar)

Key exchange:
  - DH

Encryption:
  - AES 128, but many others possible

Authentication:
  - HMAC SHA1 but others possible
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes128-ctr</td>
<td>RECOMMENDED</td>
<td>AES (Rijndael) in SDCTR mode, with 128-bit key</td>
</tr>
<tr>
<td>aes192-ctr</td>
<td>RECOMMENDED</td>
<td>AES with 192-bit key</td>
</tr>
<tr>
<td>aes256-ctr</td>
<td>RECOMMENDED</td>
<td>AES with 256-bit key</td>
</tr>
<tr>
<td>3des-ctr</td>
<td>RECOMMENDED</td>
<td>Three-key 3DES in SDCTR mode</td>
</tr>
<tr>
<td>twofish-ctr</td>
<td>OPTIONAL</td>
<td>Blowfish in SDCTR mode</td>
</tr>
<tr>
<td>twofish128-ctr</td>
<td>OPTIONAL</td>
<td>Twofish in SDCTR mode, with 128-bit key</td>
</tr>
<tr>
<td>serpent-ctr</td>
<td>OPTIONAL</td>
<td>Serpent in SDCTR mode, with 128-bit key</td>
</tr>
<tr>
<td>serpent192-ctr</td>
<td>OPTIONAL</td>
<td>Serpent with 192-bit key</td>
</tr>
<tr>
<td>serpent256-ctr</td>
<td>OPTIONAL</td>
<td>Serpent with 256-bit key</td>
</tr>
<tr>
<td>cast128-ctr</td>
<td>OPTIONAL</td>
<td>CAST-128 in SDCTR mode</td>
</tr>
<tr>
<td>3des-cbc</td>
<td>REQUIRED</td>
<td>Three-key 3DES in CBC mode</td>
</tr>
<tr>
<td>blowfish-cbc</td>
<td>OPTIONAL</td>
<td>Blowfish in CBC mode</td>
</tr>
<tr>
<td>twofish256-cbc</td>
<td>OPTIONAL</td>
<td>Twofish in CBC mode, with a 256-bit key</td>
</tr>
<tr>
<td>twofish-cbc</td>
<td>OPTIONAL</td>
<td>Alias for &quot;twofish256-cbc&quot;  (this is being retained for historical reasons)</td>
</tr>
<tr>
<td>twofish192-cbc</td>
<td>OPTIONAL</td>
<td>Twofish with a 192-bit key</td>
</tr>
<tr>
<td>twofish128-cbc</td>
<td>OPTIONAL</td>
<td>Twofish with a 128-bit key</td>
</tr>
<tr>
<td>aes125-cbc</td>
<td>OPTIONAL</td>
<td>AES in CBC mode, with a 256-bit key</td>
</tr>
<tr>
<td>aes192-cbc</td>
<td>OPTIONAL</td>
<td>AES with a 192-bit key</td>
</tr>
<tr>
<td>aes128-cbc</td>
<td>RECOMMENDED</td>
<td>AES with a 128-bit key</td>
</tr>
<tr>
<td>serpent256-cbc</td>
<td>OPTIONAL</td>
<td>Serpent in CBC mode, with a 256-bit key</td>
</tr>
<tr>
<td>serpent192-cbc</td>
<td>OPTIONAL</td>
<td>Serpent with a 192-bit key</td>
</tr>
<tr>
<td>serpent256-cbc</td>
<td>OPTIONAL</td>
<td>Serpent with a 128-bit key</td>
</tr>
<tr>
<td>arcfour</td>
<td>OPTIONAL</td>
<td>the ARFCOUR stream cipher with a 128-bit key</td>
</tr>
<tr>
<td>idea-cbc</td>
<td>OPTIONAL</td>
<td>IDEA in CBC mode</td>
</tr>
<tr>
<td>cast128-cbc</td>
<td>OPTIONAL</td>
<td>CAST-128 in CBC mode</td>
</tr>
<tr>
<td>none</td>
<td>OPTIONAL</td>
<td>No encryption; NOT RECOMMENDED</td>
</tr>
<tr>
<td>hmac-sha1</td>
<td>REQUIRED</td>
<td>HMAC-SHA1 (digest length = key length = 20)</td>
</tr>
<tr>
<td>hmac-sha1-96</td>
<td>RECOMMENDED</td>
<td>First 96 bits of HMAC-SHA1 (digest length = 12, key length = 20)</td>
</tr>
<tr>
<td>hmac-md5</td>
<td>OPTIONAL</td>
<td>HMAC-MD5 (digest length = key length = 16)</td>
</tr>
<tr>
<td>hmac-md5-96</td>
<td>OPTIONAL</td>
<td>First 96 bits of HMAC-MD5 (digest length = 12, key length = 16)</td>
</tr>
<tr>
<td>none</td>
<td>OPTIONAL</td>
<td>No MAC; NOT RECOMMENDED</td>
</tr>
</tbody>
</table>

**diffie-hellman-group1-sha1**  MUST

Oakley Group 2 [RFC2409] (1024-bit MODP Group)

**diffie-hellman-group14-sha1**  MUST

Oakley Group 14 [RFC3526] (2048-bit MODP Group)
Public Key Infrastructure

- Manage certificates \( \rightarrow \) Trust
- Distribute certificates \( \rightarrow \) Availability

Trust models

- Anarchy model, web of trust
  users sign others' keys
  manage 'key rings'

![Diagram](image)

(as in PGP)

- Monopoly model
  One world CA
  Pros: Mathematically appealing
  - Simple
  - CA's public key certificate \( \rightarrow \) ease of use
  Cons: High load on CA: identity users?
  - Very critical, only one point to break
  - High cost
  - High concentration of power
    (selecting users to certify)
High danger of teachers' sabotage...

Monopoly model + registration authorities (RA)

→ solves bottleneck
   but still high risk

Oligarchy

\[ CA_1 \quad CA_2 \quad CA_3 \quad \ldots \]

→ even less secure because even one compromised CA is a problem
   (need several certificates to resolve it)

→ CAs trusted by vendor of your software

→ might be easy to introduce a bogus CA
   in such a list

→ In practice checking all these root CAs
   is difficult to impossible.

→ Users do not understand!

PSYCHOLOGY

CRYPTO
Warning. This was signed by an unknown CA. Would you like to accept the certificate anyway?
[OK]

Would you like to accept this certificate without being asked in the future?
[OK]

Would you like to always accept certificates from the CA that issued that certificate?
[OK]

Would you like to always accept certificates from any CA?
[OK]
(User thinks: Grrrr.... isn’t it enough by now?)

Since you’re willing to trust anyone for anything, would you like me to make random edits to the files on your hard drive without bothering you with a pop-up box?
[OK]
(User thinks: Gosh, another box.... No more pop-ups? YES!)

---

Notes added in proof

. **P2P web of trust reveals social network:**
  Who knows me? Who do I know?
  → Ask the key server.

. Ask: "Who generates the private key?"
  P2P: Your own computer.
  Thawte (CA run by AOL): The CA does!
Organization

1) Go to the public course this afternoon on intrusion detection
   limits of

or alternatively inform yourself on the topic

2) Next monday we discuss that

3) No course next wednesday