Block Ciphers

Data Encryption Standard (DES)

1973, National Bureau of Standard (NBS)
Later became Nat. Institute of Standards & Technology (NIST)

- Call for crypto algorithms to become a standard
- IBM: Submitted Lucifer, in 1974
- Sent to NSA for review, Modified Lucifer to produce DES


Criticism:
- Key too small (56 bits)
- Mis-trust of NSA - did they insert a hidden trap door giving them access?
- Security about design criteria, lack of public review.

1990: Biham & Shamir - Differential cryptanalysis faster than exhaustive search if only 15 “rounds” (DES uses 16 rounds), later IBM revealed design criteria based on resistance to Diff. Crypt.

1998: Deep Crack: Special purpose highly parallel machine
Cost: $250,000. Found DES key in 56 hours.
18,560 chips.

2000: NIST calls for new block cipher designs, led to AES
Feistel network:

$$L_0 \quad R_0$$

$$\begin{array}{c}
K \\
\downarrow \\
\leftarrow f \\
\downarrow \\
L_1 \\
\downarrow \\
R_1
\end{array}$$

One round

Repeat some number of rounds of this:

$$(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus f(R_{i-1}, K_i))$$

Decrypt:

Is it invertible (by someone who knows $K_i$?)

$$R_{i-1} = L_i$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) = L_{i-1} \oplus f(L_i, K_i)$$

So

$$L_{i-1} = R_i \oplus f(L_i, K_i)$$

i.e.

$$(L_{i-1}, R_{i-1}) = (R_i \oplus f(L_i, K_i), L_i)$$

or

$$(R_{i-1}, L_{i-1}) = (L_i, R_i \oplus f(L_i, K_i))$$

i.e. invert it by doing an encryption with $L_i, R_i$ reversed.

$f$ can be any function.
DES: Same idea, but bigger

64-bit message, 56-bit key

16 rounds.

S_1, ..., S_8 given by tables

See text for details.
IP = initial permutation, permutes bits according to a fixed table

E: expands 32 bits to 48 bits by duplicating some bits

S-boxes: Each is given by a table input b_0 b_1 b_2 b_3 by b_5
- Table has 4 rows, 46 columns, 4 bits per entry
- Use row b_0 b_1 b_2 b_3 by

FP = permutation given by a table

Key: given 64 bit key k_1 \ldots k_{64}, (includes 8 parity bits)
- output 16 48 bit keys K_i.

1. For (1 \leq i \leq 16) V_i = 1 if i \in \{1, 2, 9, 16\}, V_i = 2 else.

2. T \leftarrow PC_1(K). T = (T_0, T_1), 28 bits each
   (chooses 56 bits of K in a permuted order, discarding k_8, k_{10}, k_{24}, ..., k_{64})

3. For i = 1 to 16
   \begin{align*}
   C_i &= (C_{i-1} \llll V_i) \\
   D_i &= (D_{i-1} \llll V_i) \\
   K_i &= PC_2(C_i, D_i)
   \end{align*}
   PC2 selects 48 bits in a permuted order.

\[\Box\]
Modes of Operation: different ways to use block cipher for crypto:

1. Electronic Codebook (ECB)

- \( M = M_1, ..., M_k \) each \( M_i \) has length of block cipher message
- Decryption as usual

- Flaws: Alice can observe messages and possibly build up a codebook of encryptions.
  - Doesn't require key to be found

2. Cipher Block Chaining (CBC)

- \( C_j = E_K(M_j \oplus C_{j-1}) \). Decrypt: \( E_K^{-1}(c_j) \oplus C_{j-1} = M_j \)

- Both flaws go away: e.g. insertions corrupt all further cipher blocks, so are detected.

- Must choose an initial cipher block \( C_0 = IV \).

Options:
- 1. Use a constant for \( C_0 \).
  
  - Same message gives same cipher
- 2. Use random \( C_0 \) and send in clear.
- 3. \( C_0 \) is additional key, \( \rightarrow \) effects not clear
Output

3. Cipher Feedback (OFB)
Use the block cipher to generate a keystream
for a stream cipher.

Simple version: Pick $X_0 = IV$
for $i = 1, 2, \ldots$

$X_i = E_K(X_{i-1})$
$c_i = M_i \oplus X_i$

Decrypt:
for $i = 1, 2, \ldots$

$X_i = E_K(X_{i-1})$
$m_i = c_i \oplus X_i$

OFB: $M_i = 8$ brs.

Decrypt:
$Z_j = L_8(E_K(X_j))$
$X_{j+1} = R_{56}(X_j) \parallel Z_j$

$c_j = M_j \oplus Z_j$
$m_j = c_j \oplus Z_j$

- Key stream can be generated in advance (buffered)
- Errors only affect symbol modified
- All versions slower but hide more info (maybe).

4. Cipher Feedback (CFB)
Simple: Pick $X_0 = IV$
for $i = 1, 2, \ldots$

$X_i = E_K(c_{i-1})$
$c_i = X_i \oplus m_i$

Decrypt:
for $i = 1, 2, \ldots$

$X_i = E_K(c_{i-1})$
$m_i = X_i \oplus c_i$

$Z_j = L_8(E_K(X_j))$
$c_j = M_j \oplus Z_j$
$X_{j+1} = R_{56}(X_j) \parallel c_j$
S. CTR: \( x_0 = IV \).

\[
\text{For } i = \{ 0, 1, \ldots \}:
\]
\[
x_i = x_{i-1} + 1 \quad (\text{mod } 2^{64})
\]
\[
z_j = L_8 \left( E_k(x_j) \right)
\]
\[
c_j = M_j \oplus z_j.
\]

Again, its a stream cipher, but calculation of many key stream symbols (the \( z_j \)) can be done in parallel.

---

Attacks:

1977: Estimated a 1 day attack would cost \$20 million (Diffie-Hellman).

1987: Recertified for lack of replacement.

1992: 

2. Custom architecture.
3. Programmable logic arrays.

1997: RSA challenge to find a key to decrypt a message. $10k prize after 5 months later: distributed attack succeeds.

1998: EFF: Deep Crack. Spent $250k to build special purpose DES cracker. Ciphertext only attack looking for keys that decrypt after testing interesting plaintext. 1856 chips, 96 hours to find a key.

Each chip: 24 search units.

Input 128 bit ciphertext block \( \rightarrow 2 \times 64 \) bits. \( a, b \).

For key \( k \):
- decrypt \( a \) if it's interesting, decrypt \( b \).
- If it's interesting, pass to controller.

Interesting = letters, #s, punctuation.

\[
\text{Prob (byte is interesting)} \approx 1/4, \quad \text{so} \quad \text{Prob(1-8 byte blocks)} \approx \frac{1}{2^{32}}.
\]
Improvements:

1. Double DES: encrypt twice w/ different keys.

$$E_{k_1, k_2}(m) = E_{k_1}(E_{k_2}(m))$$ - Vulnerable to meet in middle.

Recall we saw: composition of two affine ciphers is an affine cipher.

Fact: $$E_{k_1, k_2}(m)$$ is not just another DES encryption.

Meet in the middle attack:

Given a known message/ciphertext, we want to find $$k_1, k_2$$.
To do brute force, we need to examine $$2^{112}$$ pairs $$(k_1, k_2)$$.
so $$2^{112} \times 2$$ DES encryptions.

Better: Compute a list $$T_2 = \{(E(M), L_2)\}$$.
Compute a list $$T_{a_1} = \{(D_{L_2}(c), L_1)\}$$.
Look for $$(X, L_1) \in T_{a_1}, (X, L_2) \in T_2$$. (How?)

Actual $$(k_1, k_2)$$ is among $$\{(L_1, L_2) : \exists X: (X, L_1) \in T_{a_1}, (X, L_2) \in T_2\}$$.

Since $$E_{k_1, k_2}(m) = c$$ implies $$E_{k_2}(m) = D_{k_1}(c)$$.

$$2 \times 2 = 2^{57}$$ DES encryptions + $$57 \times 57$$ comparisons for sort.

$$\frac{2^{63}}{2^{57}}$$
2. Triple DES: First try: $E_{K_1} \circ E_{K_2} \circ E_{K_3}(m)$.

Can still do meet in the middle, so wasted key.

$(2^{256} + 2^{112}$ encrypt times for a 168 bit key)

2nd try: $E_{K_1} \circ E_{K_2} \circ E_{K_1}(m)$

Now meet in the middle fails - you can't separate $K_1, K_2$. So you get $2^{112}$ bit security w/ 112 bit key.

Slight improvement: use $E_{K_1} \circ D_{K_2} \circ E_{K_1}(m)$

Same security (since $D_{K_2} = E_{K_2}$)

Advantage: if $K_1 = K_2$, then it reduces to single DES

So one device can be used for single + triple DES.
A simple version: \( R_i, L_i \) have 6 bits
message \( \rightarrow \) 12 bits

\[ f: \]
\[ \begin{array}{c}
R \\
\downarrow^B \\
E(R) \\
\downarrow^E \\
K \\
\downarrow \\
A \\
\downarrow \\
S_1 \\
\downarrow \\
S_2 \\
\downarrow \\
f(R,K)
\end{array} \]

Expander: 6 bits \( \rightarrow \) 8 bits

Round
8 bit Key.

4 bits, 4 bits

S-boxes.

\[ \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
S_1: & 100 & 000 & 001 & 110 & 010 & 100 & 111 & 000 \\
& 001 & 100 & 110 & 010 & 000 & 111 & 101 & 011 \\
S_2: & 100 & 000 & 110 & 101 & 111 & 001 & 011 & 010 \\
& 101 & 011 & 000 & 111 & 110 & 010 & 000 & 100 \\
\end{array} \]

IF \( A = (a_0, a_1, a_2, a_3) \) \( a_0 \rightarrow \) row \( S_0 \)

\((a_1, a_2, a_3) \rightarrow \) column

\[ \begin{array}{c}
The A = (0, 0, 1, 1) \rightarrow \text{row 0, column 6 output = 111.} \quad \text{(from S1)}
\end{array} \]

\[ E(x_1, \ldots, x_6) = (x_1 \times 2 \times x_4 \times 3 \times 4 \times 3 \times 5 \times 6). \]

\[ K_{0/3} = 5 \times 7 \text{ bits} \]
Key = K = 9 bits
K_i = 8 bits starting w/i-th bit

\( \text{eg. } K = 110101110, K_3 = 01011101 \)

Differential Cryptanalysis:

Use pairs of inputs with carefully chosen XOR
Exploit nonuniformity in distribution of output XOR.

ie, \((L_0, R_0) \oplus (L_0^*, R_0^*) = \delta\) pick \(\delta\) so
dist. of \(\text{DES}(L_0, R_0) \oplus \text{DES}(L_0^*, R_0^*)\) is not uniform

First analyze Rounds 2, 3, 4. So input = \(L_1, R_1\)

Chosen plaintext attack. Structure known, Key unknown.
Given several pairs \(L_i, R_i; L_i, R_i\)

\( L_3 = R_2 = L_1 \oplus F(R_1, K_2) \)

\( L_3^* = R_2^* = L_1^* \oplus F(R_1^*, K_2) \)

\( R_3 = R_2 \oplus F(R_3, K_4) = L_1 \oplus F(R_1, K_2) \oplus F(R_3, K_4) \)

\( L_1^* \) is another input pair with \( R_i^* = R_1^* \)

\( \omega \cdot L_1' = L_1 \oplus L_1^* \), \( R_1' = R_1 \oplus R_1^* \), \( L_1' \oplus R_1' = i-th \text{ stage output} \)

\( F(R_1, K_2) = F(R_1^*, K_2) \) so

\( R_4' = R_4 \oplus R_4^* = L_1' \oplus F(R_3, K_4) \oplus F(R_3^*, K_4) \)

\( \text{so } R_4' \oplus L_1' = F(L_4, K_4) \oplus F(L_4^*, K_4) \)

So if input XOR outputs are known, then \( F(L_4, K_4) \oplus F(L_4^*, K_4) \) known
Next: find inputs to S-boxes at round 4 to compute \( F(L_y, K_y) \):

\[ L_y \rightarrow E(L_y) \rightarrow E(L_y) \oplus K_y \rightarrow \text{input to } S_1, S_2. \]

\[ L_y^* \rightarrow E(L_y^*) \rightarrow E(L_y^*) \oplus K_y \rightarrow \text{input to } S_1, S_2. \]

So input X-OR to Sboxes is \( E(L_y) \oplus E(L_y^*) = E(L_y \oplus L_y^*) = E(L_y) \)

So we know: 1. X-OR of inputs to \( S_1 \) and \( S_2 \) at round 4

2. X-OR of the outputs from \( S_1, S_2 \) at round 4

A. Make a table of inputs pairs to \( S_1 \) that give the known input X-OR.

E.g. if input X-OR = 1011, the input pairs are 16.5

\[ \{ (0000, 1011), (1000, 0011), (0100, 1111) \} \ldots \]

B. Then see which ones give a given output X-OR 100

\[ \Rightarrow (1010, 0001) \text{ or } (0001, 1010). \]

B. \( L_y, L_y^* \) known:

E.g. \( L_y = 10101011 \), \( L_y^* = 01101000 \)

\[ E(L_y) = 11001001 \]

\[ E(L_y^*) = 01010110 \]

So \( E(L_y) \oplus E(L_y^*) = 10110111 \)

E.g. \( L_y = (01110, L_y^* = 000010, E(L_y) = 10110110, E(L_y^*) = 00000010 \)

Inputs to \( S_1 \) are 1011 \( \oplus K_y^L \), \( 0000 \oplus K_y^L = K_y^L \).

Output X-OR is 100
So $(1011 \oplus K^L, K^L_L)$ is in the table.

Let $K^L = 0001$ or $1010$.

Do this with a few more input/output pairs and we will know $K^L$.

Similar for $S_2$. This gives 8 of 9 key bins.

Try both possibilities.

Extend to 4 rounds:
Weakness in $S_1$: of 16 input pairs with $\text{xor} = 0011$, 12 have output $\text{xor} = 011$.

(on average, 2 pairs give each output $\text{xor}$).

Similar for $S_2$: input 1100 $\rightarrow$ output $\text{xor} 010$ for 8 pairs.

So take random $R_0$, $R_0^L = R_0 \oplus 00011100$

So $E(001100) = 00111100$

$P_{01b} (\text{output} = 011010) = \frac{12}{16} \cdot \frac{8}{16} = \frac{3}{8}$.

Choose random inputs with $\text{xor} = 01110100001100$

Look at round 4 outputs. Assume round 1 output = 001100000000

Do 3 round analysis. Correct key should appear at least $3/8$ of the time, any other key occurs randomly.
Password security

Login: enter user name, enter password.

Naive approach: computer keeps a list of pairs, (user name, password), look up entered pair to validate.

But then anyone who gets access to this file can access accounts.

Solution: encrypt passwords.

Use \( F(x) \), 1-way function.

Should be efficient to compute \( y = F(x) \) but hard to find \( x \) given \( y \).

When user logs in, host computes \( F(x) \), compares to stored value \( y \).

One approach: use Block Cipher \( E_k(m) \) store \( E_k(2v) = y \).

Dictionary attack: Spot the file of encrypted pass words is known.

Attacker: compute \( F(x) \) for every \( x \) in a big dictionary (plus variations, eg reversal).

Search the password file for values of \( F(x) \) with \( x \) in the dictionary.
Forcing dictionary attacks:

1. Speed off: for responsiveness, we want f fast, but to prevent dictionary attacks, we want f slow.

   So choose f so it takes ~1 sec on typical host.

   So 60 million word dictionary takes ~2 years (N all ≤ 5 letter words using letters & digits)

   (but maybe attacker uses a faster machine). Trades off attack time/host time

2. Salt: randomly add k bits of "salt".

   Setup: when user sets up password.
   send password p to host
   host chooses random s, k bits
   store: (A, f(p,s), s).

   Login: send (A,p) to host
   host finds (A, x, s) in table.
   compares: x = f(p,s).

   What can dictionary attacker do?
   For each value d ∈ Dictionary & each k-bit s,
   compute f(d,s).

   Increases attack time by factor 2^k.

   Eg., if dictionary has ~ 2^25 words, f(d,s) takes 1 sec.
   on host, if k = 12, then 2^37 seconds on host
   For dict. attack - 2^37 = 4000 years.
   Even if attacker is 1000 times as fast as host, it takes 4 years

   But doesn't prevent an attack on one user.
Choosing $f$:

one optimal base it on DES.

First 8 bytes of password $\rightarrow$ 7 bits ASCII $= 56$ bits.
Use this as a DES key. (Pad w/ 0's if needed). $\rightarrow$ K

Encrypt (0,0,...,0) using $K$ as key for 25 rounds of DES.

Add salt to DES: Pick random 12-bit salt. $= S_1, ..., S_{12}$.
Modify DES: $E$ Expand w/ $E$: 32 bits $\rightarrow$ 48 bits $E(R) = E_{1...E_{48}}$
if $S_1 = 1$ swap $E_1, E_{25}$
if $S_2 = 1$ $E_2, E_{26}$

$\vdots$
if $S_{12} = 1$ $E_{12}, E_{36}$.

Prevents attacks w/ special purpose DES chips.
A5S

1997: NIST call for candidates to replace DES
- must allow key size 128, 192, or 256.
- Blocks of 128 message bits
- adaptable to various hardware
  (8 bit smart card, 32 bit general purpose...)
  (e.g. no 2^32 arithmetic)

1998 - 15 candidates; 5 finalists chosen
  after conferences and various papers
  attacking candidates
  (MARS/IBM, RC6/RSA labs, Rijndael (Netherlands);
  Serpent (Andersson/Bihan, Knudse); Twofish (US))

2000 - Rijndael chosen.

Key: 128, 192, 256 bits. 128 rounds
Rounds: 10, 12, 14  
round: \( F_2^{128} \rightarrow F_2^{128} \)
\( \{0, 1\}^{128} \)

Not a Feistel net.

Each round has 4 layers
2. ShiftRow Transform (SR): linear mixing for high diffusion
3. MixColumn (MC): more linear mixing
4. Addround key (ARK): XOR the round key w/ output from (3)

1 round: BS \rightarrow SR \rightarrow MC \rightarrow ARK.
1. ARK using $K_0$
2. 9 rounds of BS, SR, MC, ARK using $K_1,\ldots,K_9$
3. 1 round of BS, SR, ARK using $K_{10}$

Input = 128 bits = 16 by 8 $a_{ij}$, $0 \leq i, j \leq 3$

\[
\begin{pmatrix}
 a_{00} & a_{01} & a_{02} & a_{03} \\
 a_{10} & \cdots & a_{13} \\
 a_{20} & \cdots & a_{23} \\
 a_{30} & \cdots & a_{33}
\end{pmatrix}
\]
Algebraic group: $G$ as set, o an operator: $G \times G \rightarrow G$, e, eG so

1. $\forall a,b,c \in G: (a \cdot b)c = a \cdot (b \cdot c)$
2. $\forall a \in G: ae = ea = a$
3. $\forall a, b \in G: ab \cdot b^{-1}a^{-1} = e$ (i.e., $b = a^{-1}$)

If also:
4. $\forall a, b, c \in G: ab = ba$

then $G$ is Abelian

E.g., $(\mathbb{Z}, +, 0), (\mathbb{Q} \setminus \{0\}, \cdot, 1)$ (Abelian)

$(\mathbb{Z}_m, +, 0), (\mathbb{Z}_m^*, \cdot, 1)$ (Abelian)

$\left(M_2(\mathbb{R}), +, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, 0 \right)$ Abelian.

$\left(GL_2(\mathbb{R}), \cdot, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}\right)$ not Abelian

$M_k(\mathbb{Z}), GL_k(\mathbb{Z}), M_k(\mathbb{Z}_m), GL_k(\mathbb{Z}_m)$

Lemma: $G$ a group, $a \in G, \Rightarrow \exists! b: ab = e$

Proof: $a \cdot b \cdot a = a \cdot b = e$, then $a^{-1}(a \cdot b) = a^{-1}(a \cdot e) \Rightarrow (a^{-1}a)b = (a^{-1}a)e \Rightarrow eob = e \cdot e \Rightarrow b = e$ \hfill \Box

Lemma: $G$ a finite Abelian group. $\forall a \in G$, $a^{16} = e$

Proof: Let $\pi G = \pi G$. Note that $f(a) = \pi G$ is a permutation of $\pi G$

\[ f^{-1}(a) = a^{-1}b \]

$\Rightarrow Z = \pi G a \cdot a \Rightarrow a^{16} \Rightarrow Z \Rightarrow e = a^{16}$ \hfill \Box

Generalizes Euler
Subgroups, quotients

\((G, \circ, e)\) an Abelian group.

\(H \leq G\) is a subgroup of \(G\) if \(a, b \in H \implies a \circ b \in H, a^{-1} \in H\)

e.g. \(G = \mathbb{Z}, +\), \(m \in \mathbb{Z}\), \(H_m = m\mathbb{Z} = \{ma : a \in \mathbb{Z}\}\)

Lemma: every subgroup of \(\mathbb{Z}\) is an \(H_m\).

\(H \subseteq \mathbb{Z}\) a subgroup. Let \(m = \min\{\text{positive } a \in H\}\) \((\text{sup } H \neq \emptyset)\)

let \(c \in H\), \(\text{let } c = q_m + r, 0 \leq r < m\). Then \(r = c - q_m \in H_m\), so \(r = 0\) \(\square\)

Abelian

Let \(G\) be a group, \(H \leq G\) a subgroup. Then \(a \equiv b \iff a - b \in H\)

\(H\) is an equivalence relation.

Let \(G/H = \{\equiv \text{equivalence classes}\} = \{a + H, a \in G\}\)

Then \(G/H\) is a group: \((a + H) + (b + H) = a + b + H\)

\(a \equiv b \iff b \in H_m\), \(\Rightarrow G/H = \mathbb{Z}_m\)
A ring is a set \( R \) with two operations \( +, \cdot \) and elements \( 0, 1 \in R \) so that:

1. \((R, +, 0)\) is an Abelian group
2. \(\forall a, b, c \in R: a \cdot (b \cdot c) = (a \cdot b) \cdot c\)
3. \(\forall a \in R: a \cdot 1 = 1 \cdot a = a\)
4. \(\forall a, b, c \in R: a \cdot (b + c) = a \cdot b + a \cdot c \)

If also
5. \(\forall a, b \in R: a \cdot b = b \cdot a\), then \( R \) is commutative.

For example, \((\mathbb{Z}, +, 0, 1)\), \((\mathbb{Q}, +, 0, 1)\), \((\mathbb{R}, +, 0, 1)\) are commutative.

Any ring: \((M_n(R), +, \cdot, 0, I)\) is noncommutative.

\(R[x] = \{ \sum a_n x^n : a_n \in R \}\) usual polynomial ring.

\((\sum a_i x^i)(\sum b_j x^j) = \sum_{i+j} (\sum a_i b_j) x^n\)

If \( R \) is commutative if \( R \) is.

Ideals: A comm. ring. An ideal is a set \( I \subseteq R \) so that:

1. \(\forall a \in I, b \in R: a \cdot b \in I\)
2. \(\forall a, b \in I: a + b \in I\)

For example:
- \(R = \mathbb{Z}, I = \{ ma: a \in \mathbb{Z} \}, m \mathbb{Z} \text{Fixed} = (m)\)
- \(R = (AX^2) \times I = \{ f(x) \cdot g(x): g(x) \in R \}, f \in R \text{Fixed} = (f)\)
- \(R = (TX^2), I = \{ \sum a_i x^i: a_i \in R \}, F \in R^2 = (2, x^2)\)

In general: \(\forall R \text{ ring}, a_1, \ldots, a_n \in R, (a_1, \ldots, a_n) = \{ \sum_{i=1}^n b_i a_i: b_i \in R \}\)

is an ideal.
Ring Quotients

If \( R \) is a commutative ring and \( I \) is a proper ideal, then \( R/I \) is a group under:

\[
(a+I)(b+I) = ab+I
\]

This works since \( aI, bI, I^2 \subseteq I \).

Exercise 1: \( R=\mathbb{Z}, I=(m)=H_m, R/I=\mathbb{Z}/m \).

\( R = \mathbb{Z}[x] = \{ \sum_{i=0}^{d} a_i x^i : a_i \in \{0,1\} \} \)

\( I = (x^2+x+1), \quad R/I = \{ a+I : a \in \{0,1\} \} \)

\( (a_0+a_1 x)(b_0+b_1 x) = (a_0 b_0 + a_0 b_1 + a_1 b_0) \mod (x^2+x+1) \)

Homomorphisms:

\( f: R \to S \) is a homomorphism if for all \( a,b \in R \):

\[
f(a+b) = f(a) + f(b) \quad f(ab) = f(a) f(b) \quad f(-a) = -f(a) \quad f(a^{-1}) = f(a)^{-1}
\]

Example: \( f: \mathbb{Z} \to \mathbb{Z}[x] \) by \( a \to a \mod (x^2+x+1) \).

Lemma: \( R \) a commutative ring \( \Rightarrow \exists f: \mathbb{Z} \to R \) defined by \( f(1) = 1 \).

If \( f: R \to S \) is a ring homomorphism, then \( \ker(f) = \{ a : f(a) = 0 \} \subseteq R \).

Then \( \ker(f) \) is an ideal in \( R \) and \( R/\ker(f) \approx \text{Im}(f) \).

If \( f: \mathbb{Z} \to R \Rightarrow \ker(f) = (m) \). \( m = \text{characteristic of } R \).
Finite Fields:

Field = set \( F \) with ops. +, \(*\), special elements 0, 1.

- So that 1. \( a(0) = (a \cdot 0) + a = 0 \cdot a + a = a \)
  2. \( a + 0 = a \)
  3. \( a + b = b + a \)
  4. \( a \cdot b \cdot c = (a \cdot b) \cdot c = a \cdot (b \cdot c) \)
  5. \(0 \cdot a = 0 \)
  6. \( a \cdot 1 = a \)
  7. \( a \cdot 0 = 0 \cdot a = 0 \)
  8. \( a \cdot 1 = a \)
  9. \((a + b) \cdot (c + d) = a \cdot c + a \cdot d + b \cdot c + b \cdot d \)

- i.e. a field is a ring where every non-zero element has a mult. inverse.

Examples:
- \( \mathbb{Q} \), \( \mathbb{R} \), \( \mathbb{F}_2 \), \( \mathbb{F}_p \).

If \( F \) is a field, then \( F[x] \) has \(+, \cdot\) as usual.

Examples:
- If \( F = \mathbb{F}_2 \), then \( F[x] \) is irreducible.

Construction:
- \( \mathbb{F}_2 \langle x \rangle \) mod \( x^2 + x + 1 \): 
  - \( f(x) \equiv g(x) \mod x^2 + x + 1 \) if \( x^2 + x + 1 \mid f(x) - g(x) \).

So \( \mathbb{F}_2 \langle x \rangle \) has \( 2^2 = 4 \) elements:
- \( \{0, 1, x, x+1\} \)
- \( (ax) + (bx) = (a+b)x \)
- \( (ax) \cdot (bx) = abx^2 \)

Thus: if \( f(x) \) is irreducible, then \( \mathbb{F}_2 \langle x \rangle / \langle f(x) \rangle \) is a field.

Rijndael uses: \( \mathbb{F}_2 \langle x \rangle / \langle x^8 + x^4 + x^3 + x + 1 \rangle \).

Addition:
- \( \sum a_i x^i + \sum b_i x^i = \sum (a_i + b_i) x^i \) in \( \mathbb{F}_2 \langle x \rangle \).
\[ \left( \sum a_i x^i \right) \left( \sum b_j x^j \right) = \sum_{k=0}^{14} \left( \sum a_i b_j \mod 2 \right) x^k. \]

But reduce mod \( x^8 + x^4 + x^3 + x + 1 \).

ie replace \( x^4 \) by \( x^6 \cdot (x^4 + x^3 + 1) \)
\( x^{13} \) by \( x^5 \cdot (x^4 + \ldots) \)
\( \vdots \)
\( x^8 \) by \( (x^4 + \ldots) \)

**Fact:** This is a field. (All fields w/ finite # elements can be constructed this way.)

\[ x^5 = x^4 + x^3 + x^{11}; \quad x^9 = x^8 + x^4 + x^3 + x^2; \quad x^{10} = x^9 + x^5 + x^4 + x^3; \quad x^{12} = x^7 + x^5 + x^3 + x^2; \quad x^{13} = x^9 + x^7 + x^5 + x^3; \quad x^{14} = x^6 + x^5 + x^3 + x^2 + 1. \]

We can write
\[ \left( \sum a_i x^i \right) \left( \sum b_j x^j \right) = \sum_{n=0}^{14} m_n(a, b) x^n + (a_0, \ldots, a_7) \]

where \( m_n \) is of form \( m_n(a, b) = \sum_{i,j} a_i b_j \).

**Residue details:** Input = 128 bytes = 16 bytes, \( a_{ij} 0 \leq j \leq 3 \)

Arrange input as a matrix of bytes \( [a_{ij}] \)

\[
\begin{bmatrix}
a_{00} & a_{01} & a_{02} & a_{03} \\
a_{10} & \ddots & \ddots & \ddots \\
a_{20} & \ddots & \ddots & \ddots \\
a_{30} & \ddots & \ddots & \ddots \\
\end{bmatrix}
\]
Classification of finite fields

- If $F$ is a finite field, then $\exists$ prime $p$ such that $\frac{p}{a+b} = 0 \forall a \neq b$

$\iff \exists p = \text{char}(F)$

- $|F| = p^d$ some.

- $\forall$ prime $p$, $d \geq 2$, $\exists$ finite field $\text{IF} = p^d$
  
  $F$ is unique up to isomorphism $= \text{IF}_{p^d}$. $\text{IF}_p = \mathbb{Z}_p$

- $\text{IF}_{p^n}$ can be written as $\text{IF}_p[x]/(g(x))$, $g(x)$ irreducible.

- $\forall a \in \text{IF}_p$, $a \in \text{IF}_p$ iff $r = p \land d | r$

- $\exists g \in \text{IF}_p^d$: $\text{IF}_p = \{0, 1, g, g^2, \ldots, g^{p^d-2}\}$, $g$ is primitive.

---

Division theorem for polynomials / field $F$. Define $\text{lc}(f(x)) = \text{leading coeff}.$

The: $F$ a field, $a(x), b(x) \in F[x]$. Then $\exists q(x), r(x) \in F[x]$

$\text{so } a(x) = q(x)b(x) + r(x) \text{, } \deg(r) < \deg(b)$.

**Proof:** Let $s = \{a(x) - u(x)b(x), u(x) \in F[x]\}, s \neq \emptyset$

Let $q(x)$ be so $\deg(a(x) - q(x)b(x))$ is minimal

$\exists \deg(r) > \deg(b)$, then $a(x) - (q(x)+r(x) x^{\deg(r)-\deg(b)})b(x) = s(x)$

has degree $< \deg(r)$.
1. ByteSub: Uses a table to implement an S-box

S: 8 bits → 8 bits, see next page for details.

applied to each byte of the 4×4 byte input matrix
input A → output B

2. ShiftRows: Apply a left circular shift to each row:
   
   - row 0: 0 bytes
   - row 1: 1 byte
   - row 2: 2 bytes
   - row 3: 3 bytes

   input B → output C.

3. MixColumns: Treat each byte as an element of $\mathbb{F}_{2^8}$.
   
   So output from SR is a 4×4 matrix with entries in $\mathbb{F}_{2^8}$.

   Multiply on left by a fixed matrix:

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

   \[
   C = M \cdot C.
   \]

   \[
   \text{det} = x^4 + x^3 + 1 ≠ 0.
   \]

4. AddRoundKey: XOR with \( K = \begin{pmatrix} k_{00} & \cdots & k_{07} \\ k_{10} & \cdots & k_{17} \end{pmatrix} \) → E.
S-boxes: Again, use $\mathbb{F}_2^8$: let $a = a_0 \ldots a_7$

$S(a) = \text{inverse of } a_0 a_1 x + \ldots + a_7 x^7$ as an element of $\mathbb{F}_2^8$.

(S(0...0) = 0...0). Then \( \text{perm}^i(y) \rightarrow M_{iy} y + v \)

Advantage: there's no mystery.

\[ \text{It is a permutation.} \]

Key schedule: Key = 128 bits in a matrix of bytes.

Columns = $w(0), \ldots, w(3)$, \( (e^{r(i)} x^4 \text{ if } 4 \mid i) \in \mathbb{F}_2^8 \).

Recursively construct: $w(i) = w(i-4) \oplus w(i-1)$ if $4 \mid i$.

$w(i) = w(i-4) \oplus T(w(i-1))$ if $4 \mid i$.

where \( T: \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \rightarrow \begin{pmatrix} a \\ b \\ c \\ a \end{pmatrix} \rightarrow \begin{pmatrix} S(b) \\ S(c) \\ S(d) \\ g(a) \end{pmatrix} \rightarrow \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix} \rightarrow \begin{pmatrix} e \oplus r(i) \\ f \\ g \\ h \end{pmatrix} \)

Round key for round $j$ is

\( \begin{pmatrix} W(4j) \\ W(4j+1) \\ W(4j+2) \\ W(4j+3) \end{pmatrix} \)
Decryption: just invert each step:

1. Byte Sub: use another lookup table, Inv Byte Sub.
   \[
   \tilde{z} \rightarrow N^{-1}(\tilde{z} + \tilde{v}) \rightarrow \text{treat as } a \in \mathbb{F}_{2^8} \text{ and invert}
   \]
   \[
   (\text{BS: } \tilde{z} \rightarrow Nz + \tilde{v})
   \]

2. Shift row: shift rows by 0, 3, 2, 1.

3. Mix column: multiply by the inverse matrix
   (inverse as \( \text{mtx} / \mathbb{F}_{2^8} \))

4. Add round key = its own inverse

Mix round K0, K1, K2, K3...

---

Encryption: ARK
BS, SR, MC, ARK
: : :
BS, SR, MC, ARK
BS, SR, ARK

decrypt: ARK, ISR, IBS
ARK, IMC, ISR, IBS
: : : :
ARK, IMC, ISR, IBS
ARK

but IBS \& ISR cannot

MC, ARK = C \rightarrow MC \rightarrow MC \oplus K = E

Inverse: \( V \rightarrow E \oplus K \rightarrow M^{-1}(E \oplus K) = M^{-1}E \oplus M^{-1}K = M^{-1}E \oplus K \).
Let IARK = \( X \cdot \text{inv} \cdot K \)

so \( \text{inverse}(\text{MC}, \text{ARK}) = \text{MC}, \text{IARK} \).

so decrypts is

\[
\begin{align*}
\text{ARK} & \quad \text{IBS}, \text{ISR} \\
\text{MC}, \text{IARK} & \quad \text{IBS}, \text{ISR} \\
\text{MC} & \quad \text{IARK}, \text{IBS}, \text{ISR} \\
\text{ARK} & \quad \text{IARK}, \text{IBS}, \text{ISR} \\
\text{I} & \quad \text{ARK} \\
\text{IBS}, \text{ISR}, \text{MC}, \text{IARK} & \\
\text{IBS}, \text{ISR}, \text{MC}, \text{ARK} & \\
\text{IBS}, \text{ISR}, \text{ARK} & \\
\text{IBS}, \text{ISR}, \text{ARK} & \\
\text{IBS}, \text{ISR}, \text{ARK} & \\
\text{I} & \quad \text{ARK} \\
\text{IBS}, \text{ISR}, \text{MC}, \text{IARK} & \\
\text{IBS}, \text{ISR}, \text{MC}, \text{ARK} & \\
\text{IBS}, \text{ISR}, \text{ARK} & \\
\text{IBS}, \text{ISR}, \text{ARK} & \\
\text{I} & \quad \text{ARK}
\end{align*}
\]

which is the same structure as AES.

**Analysis:**

- resists all known attacks
- Finite Field inversion in S-box yields linear approximation & difference distinguishables that are non-uniform. (resistant to linear & differential crypt).
- Mix columns gives high degree of mixing so attacks cannot isolate S-boxes
- Key schedule mixing foil's attacks based on knowing part of the key.