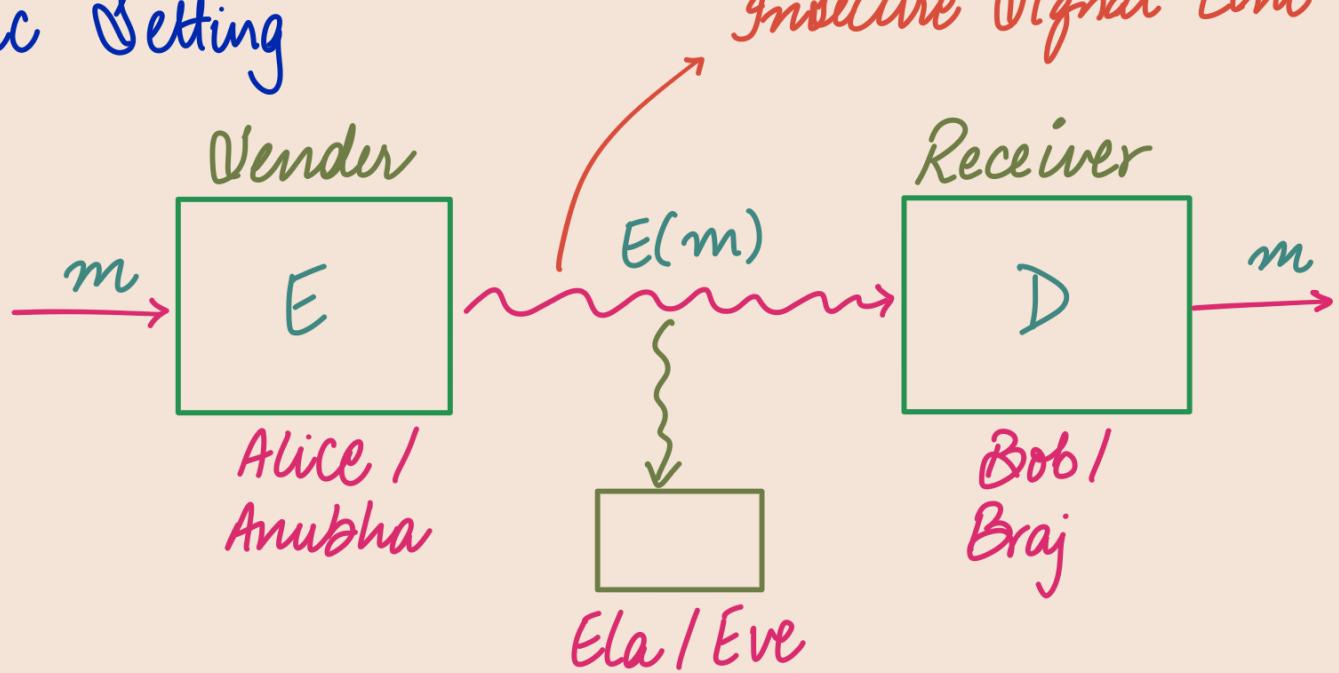


## Basic Setting



- Anubha's job is that of cryptography
- Ela's job is that of cryptanalysis + eavesdropping
- $E$  : Encryption
- $D$  : Decryption s.t.  $D(E(m)) = m$
- Ela can only get hold of  $E(m)$

Caesar's Cipher : Replace each letter in  $m$  by a letter 3 places (cyclically) to the right.

Ex: This → Wklv

# First Principle of Cryptography (+ Assumption)

Everything that is fixed will eventually be known to everyone.

↔ Everything fixed is known to everyone.

- ⇒ Need to have a variable element in  $D$ , say decryption key  $k_D$ . So that even if  $D$  is known,  $E(m)$  cannot be decrypted w/o key
- ⇒  $k_D$  needs to keep changing.  
But, if diff. keys can decrypt same message, then keys also fixed  
⇒  $E$  needs to have a variable element,  $k_E$  as well.
- Now,  $\because E \& D$ , the encryption methods, are fixed, assume that they are known to Ela.  
Though,  $k_E$  &  $k_D$  are still not known to Ela

## Properties required

- 1)  $D(E(m, k_E), k_D) = m$
  - 2) w/o knowledge of  $k_D$ , Ela **should not** be able to decrypt.
  - 3)  $D \& E$  should be efficient.
- 128 bits enough to prevent brute force attacks

## Substitution Cipher

- Generalization of Caesar's Cipher
- 1-1 mapping,  $k_E : [a-z] \rightarrow [a-z]$ ;  $k_D = k_E^{-1}$   
 $E(m) = E(a_1 a_2 \dots a_n)$   
 $= k_E(a_1) k_E(a_2) \dots k_E(a_n)$
- # of  $k_E = 26! \approx (26/e)^{26} \approx 10^{26}$

## Frequency Analysis

In the English Language, each of the 26 letters have typical frequencies. These frequencies can be used to 'guess' substitution key letter-by-letter.

Some other peculiarities of the language can also be used. For example, the letter  $q$  is almost always followed by the letter  $u$ .

- Having a large number of keys does not guarantee security. However, they are a necessity for security, just not a sufficiency.

## Permutation Cipher

Key is a permutation of  $[1, n]$  { $n$ : size of block}. To encrypt, break the plaintext into chunks of size  $n$ .

Ex:  $m = a_1 a_2 a_3 \dots a_n a_{n+1} a_{n+2} \dots$

$$\rightarrow E(m) = a_{\pi(1)} a_{\pi(2)} a_{\pi(3)} \dots a_{\pi(n)} a_{n+\pi(1)} a_{n+\pi(2)} \dots$$

$\pi$ : a permutation of  $[1, n]$   
 $n$ : fixed; assume it to be known.  
To find  $\pi$ , employ pairwise frequency analysis on blocks of size  $n$ .

## More Ciphers

- Substitution - Permutation
- Multiple substitutions

Types of Attacks : Need to formalize the ways Ela can try to break the cryptographic methods used by Anusha-Braj.

- 1) Ciphertext only attacks : (weakest form of attack)  
Ex: Frequency analysis, BruteForce attack.  
Ela only gets  $E(m)$
- 2) Known plaintext attack :  
Ela knows plaintexts corresponding to a few ciphertexts.
- 3) Chosen plaintext attack :  
Ela can get specific plaintexts encrypted.
- 4) Chosen ciphertext attack :  
Ela can get plaintext corresponding to some of the ciphertexts.
- 5) Chosen plaintext + ciphertext attack  
→ In the order of increasing strength of attacks.

### Assumption

Ela can carry out both chosen plaintext and ciphertext attacks.

### Linear Ciphers

- Break the message into blocks of size  $n$  and view the blocks as vectors in an  $n$ -dimensional space.
- View the key  $k_E$  as an  $n \times n$  invertible matrix  
$$E(v) = k_E \cdot v$$
$$k_D = k_E^{-1} \quad D(E(v)) = k_D \cdot E(v) = k_D \cdot k_E \cdot v = k_E^{-1} \cdot k_E \cdot v = v$$
- Qo, permutation cipher is a linear cipher.

- Similarly, substitution cipher is also a linear cipher. View the letters as a 26-dim. long one-hot vectors. Then, the key is a  $26 \times 26$  matrix.
- Breaking Linear Ciphers:**  
Known plain-text attack using  $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \text{ etc.}$   
OR  
In general, known plain-text attack using plaintexts with  $n$  linearly independent vectors.

- ★ Linear Ciphers can be broken using known plaintext methods.  $\therefore$  Our encryption methods cannot be linear ciphers.
- ★ Need to use an invertible non-linear transformation for encryption.  
But, a linear transformation on the entire block helps mixing the information well.  
 $\therefore$  A combination of the two is desirable.

Thm. Let  $u \in \mathbb{Z}^b$ ,  $u \neq 0$ , and  $K \in \mathbb{Z}^{b \times b}$ . Then, over random choices of  $K$ ,  $K \cdot u$  is a random vector in  $\mathbb{Z}^b$

- Given  $u \neq 0$  &  $c$ , let  $i^{\text{th}}$  entry of  $u$ ,  $u_i \neq 0$ .
- Probability that  $c = K \cdot u$  equals the probability that  $K_i = 1/u_i (c - \sum_{j \neq i} u_j K_j)$
- $\therefore K_i$  is a random vector, and  $c$  has a similar distribution, thus,  $c = K \cdot u$  is a random vector

But, we require  $K$  to be invertible.

∴ Encrypted text is required to be decrypted, all transformations done during encryption need to be invertible. Easy for LTs, but NLTs typically not invertible.

A generic way of doing it is given by Feistel.

### Feistel Structure

Let  $f$  be any NLT with  $f: \{0,1\}^n \rightarrow \{0,1\}^n$   
Define  $g: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n \times \{0,1\}^n$   
as :  $g(a, b) = (b, a \oplus f(b))$

Then,  $g$  is clearly an NLT  
But, it is also invertible :

$$g^{-1}(b', a') = (a' \oplus f(b'), b')$$

$$\Rightarrow g^{-1}(g(a, b)) = g^{-1}(b, a \oplus f(b)) = (a \oplus f(b) \oplus f(b), b) \\ = (a, b)$$

∴ Feistel structure can be used to ensure invertibility given any transformation  $f$

But, it only encrypts half of the input.  
A work-around is applying  $g$  twice to completely transform input.

Generally, greater number of rounds provide more security.

### DES (Data Encryption Standard)

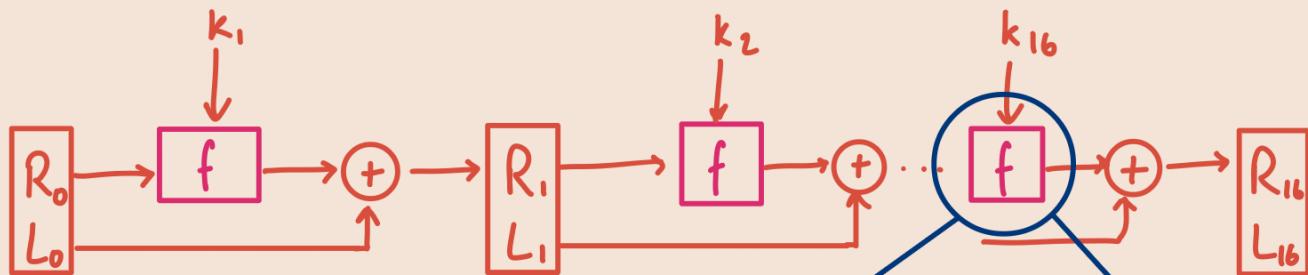
Designed in 1974, used until 2001.

Adopted by NBS (US) in 1976.

Parameters : Block Cipher w/ blocksize = 64 bits  
Key size = 56 bits, use 16 rounds of Feistel Structure.

- 1 byte = 8 bits: MSB stores parity of remaining 7 bits for robustness.

## Structure



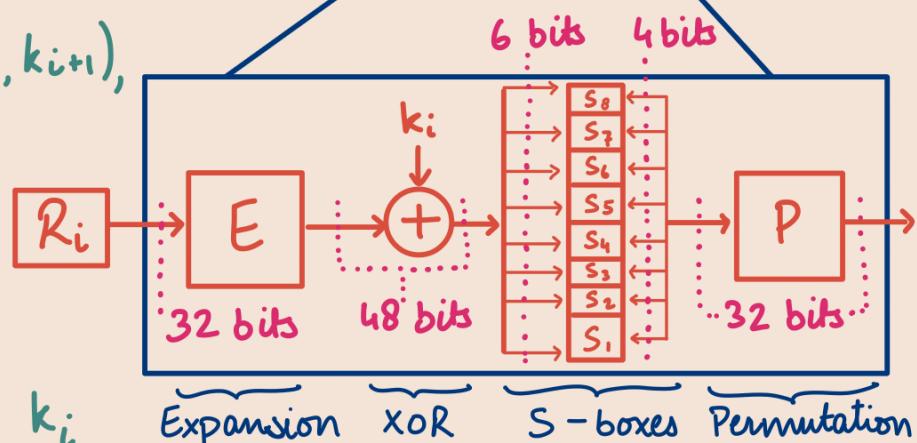
$$R_{i+1} = L_i \oplus f(R_i, k_{i+1}),$$

$$L_{i+1} = R_i,$$

$$\forall 0 \leq i < 16$$

Function  $f$  also depends

on round key  $k_i$



Round Keys: Key  $k_i$  is called round key for round  $i$ ,  $1 \leq i \leq 16$

- Each  $k_i$  is 48 bits long & is a fixed subset of 56 bits of the whole key.
- Plaintext block is  $L_0 R_0$ ,  $|L_0| = |R_0| = 32$  bits
- Input to round  $i+1$  is  $L_i R_i$  & its output is  $L_{i+1} R_{i+1}$ , with  $|L_{i+1}| = |R_{i+1}| = 32$  bits.

- $f$ : Non-linear function. Input is right half of round input (32 bits), round key (48 bits) and produces a 32-bit output.
- Further divided into 4 operations, 3 of which are linear and 1 is non-linear (S-boxes)

Expansion E : Produces 48 bit output by replicating 16 bits of input.

Input:  $b_0, b_1, \dots, b_{31}$

Output:  $b_{31}, b_0, b_1, b_2, b_3, b_4, b_3, b_4, b_5, b_6, b_7, b_8, b_7, b_8, \dots, b_{29}, b_{30}, b_{31}, b_0$

- Basic idea is to pair adjacent bits & repeat alternate pairs ( $b_0$  paired with  $b_{31}$ ); For example,  $b_1, b_2$  is not repeated but  $b_{31}, b_0, b_3, b_4$  are repeated.

Permutation P : Shuffles input bits as:

Input:  $b_0, b_1, b_2, \dots, b_{31}$

Output:  $b_{15}, b_7, b_{14}, b_{20}, b_{28}, b_{11}, b_{27}, b_{16}, \dots, b_{21}, b_{10}, b_3, b_{24}$

- Primary aim is to shuffle bits so that all 4 bits in a block move to different blocks, for each of the 8 blocks
- For example,  $b_0, b_1, b_2, b_3$  are not in the same 4-bit block in the output.

XOR ⊕ : The 48-bit expansion of the input half is bitwise XOR-ed with the round key  $k_i$

S-boxes : Only nonlinear operation in entire algorithm

- There are 8 S-boxes, each is a mapping from 6 bits to 4 bits
- Each of the 8 boxes are distinct transformations

S-1 : Columns indexed by middle 4 bits of input, and rows indexed by first & last bits of input.

- Numbers are btw 0 - 15 representing 4 bit outputs.
- Every row has all 16 numbers occurring once.
- Small S-boxes for better speed. Same reason

for other choices of operation.

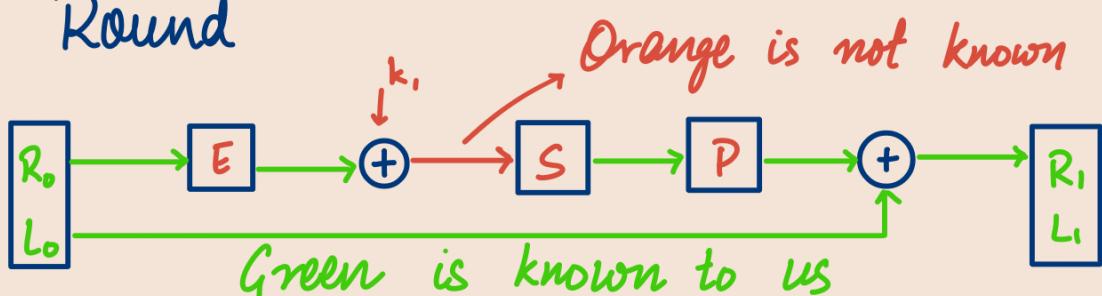
The box itself:

S-1	0000	0001	0010 = 2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	1110	0100	1101 = 13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	...	...											...	...	
10	4	...	...											...	...	
11	15	...	...											...	...	13

## Cryptanalysis of DES

- Brute-force attack to find out the key requires  $2^{56} \approx 10^{17}$  operations.
- Frequency analysis doesn't work :: variations in frequencies flattened out by multiple transformation
- Assume stronger forms of attacks: known-plaintext.
- Start w/ lesser number of rounds.

DES: One Round



- We know the output of S-boxes & E.
- Let  $E(R_0) = \alpha_1 \alpha_2 \dots \alpha_8$  w/  $|\alpha_i| = 6$
- $E(R_0)$  gets XOR-ed with key  $k_1 = k_{1,1} k_{1,2} \dots k_{1,8}$  w/  $|k_{1,i}| = 6$  &  $\beta_i = \alpha_i \oplus k_i$
- 6-bit string  $\beta_i$  is input to  $i^{\text{th}}$  S-box
- Let  $y_i = S_i(\beta_i)$  w/  $|y_i| = 4$
- Each  $y_i$  &  $\alpha_i$  is known
- $\therefore y_i$  is known, we can look up the table for  $S_i$  to find out which inputs can

produce  $y_i$  as output.

$S_i$  has exactly four occurrences of  $y_i$ .

Let  $X_i$  be the set of inputs to  $S_i$  that produce  $y_i$  as output.  $|X_i| = 4$ .  $\beta_i \in X_i$ .  $K_i := \{\alpha_i \oplus \beta | \beta \in X_i\}$

$k_{i,i} = \alpha_i \oplus \beta_i$ , we have  $k_{i,i} \in K_i$  &  $|K_i| = 4$

$\forall 1 \leq i \leq 8$ ,  $k_{i,i} \in K_i$ ,  $|K_i| = 4$

Concatenating strings from  $K_i$ , we get  $4^8 = 2^{16}$  strings, one of which is  $k_i$ . Significant improvement on brute-force attack already.

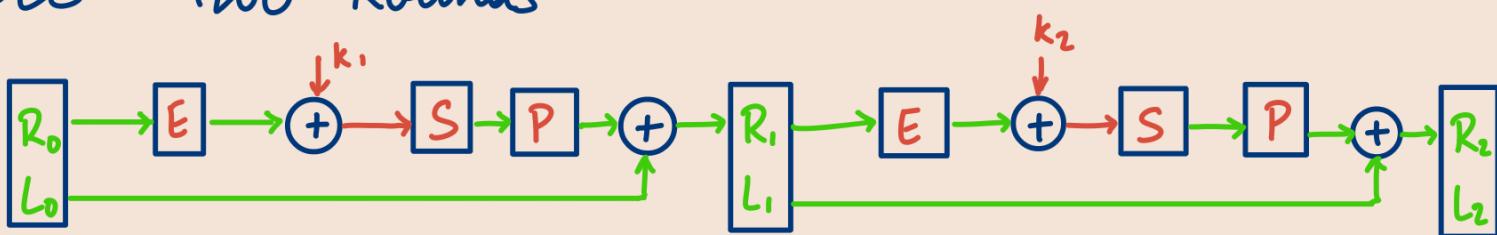
Take another pair of plaintext & corresponding ciphertext block.

Repeat the same analysis as above to get sets  $K'_i |_{i=1}^{i=8}$ ,  $|K'_i| = 4$ ,  $k_{i,i} \in K'_i \Rightarrow k_{i,i} \in K'_i \cap K_i$

Likely that  $|K'_i \cap K_i| = 1$ , uniquely identifying  $k_{i,i}$ . If it is not unique, repeat.

Once all sets have size 1, the entire key  $k$  is uniquely identified.

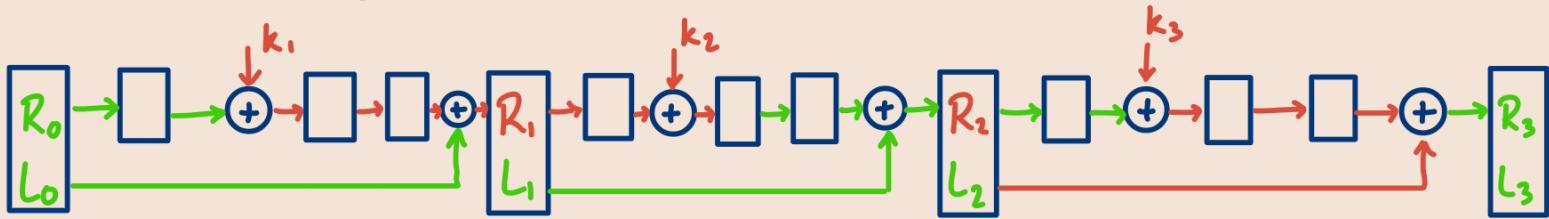
DES: Two Rounds



We know the output of S-boxes as well as output of E for both rounds.

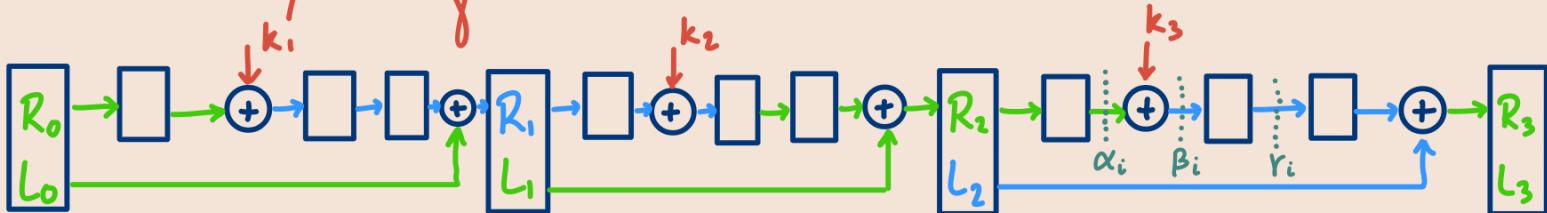
Using the same strategy as before, we can extract  $k_1$  as well as  $k_2$  easily.

## DES : Three Rounds



- $R_i = L_2$  is not known.
- There is no round for which outputs of both E & S-boxes are known.
- Cannot use earlier strategy for this.
- Idea: Use two inputs & see how their XOR moves.
- Let  $L_0, R_0$  &  $L'_0, R'_0$  be two inputs. Then  

$$L_0 \oplus L'_0 = R_0 \oplus R'_0 = L_2 \oplus L'_2$$
- Let  $\underline{\quad}$  denote values whose XOR's we know for two inputs of the above kind. Then:



- Let  $E(R_2) = \alpha_1, \alpha_2, \dots, \alpha_8$ ,  $|\alpha_i| = 6$ . Similarly,  $E(R'_2)$ .
- Let  $\beta_i = \alpha_i \oplus k_{3,i}$ ,  $|k_{3,i}| = 6$ . Similarly,  $\beta'_i$ .
- Let  $y_i = S_i(\beta_i)$ . Similarly,  $y'_i$ . We know  $y_i \oplus y'_i$ .

Q. How many pairs  $(\delta_i, \delta'_i)$  exist s.t.  $S_i(\delta_i) \oplus S_i(\delta'_i) = y_i \oplus y'_i$   
 $X_i := \{(\delta_i, \delta'_i) \mid S_i(\delta_i) \oplus S_i(\delta'_i) = y_i \oplus y'_i\}$

Lemma:  $|X_i| = 256$

Proof:  $\exists 16$  pairs  $(\epsilon_i, \epsilon'_i)$  st.  $\epsilon_i \oplus \epsilon'_i = y_i \oplus y'_i$   
For each  $\epsilon_i \exists 4$  values of  $\delta_i$  s.t.  $S_i(\delta_i) = \epsilon_i$   
Similarly, for each  $\epsilon'_i$ .  $\therefore |X_i| = 16 \cdot 4 \cdot 4 = 256$

Q. How many XOR values are there for pairs in  $X_i$ ?  
Trivial upper bound = 64  $\because |\delta_i| = 6 = |\delta'_i| \Rightarrow 2^6 = 64$

But, all pairs st.  $\delta_i \oplus \delta'_i \neq \beta_i \oplus \beta'_i$  can be removed from  $X_i$ . Known to us as well

Q. What is the size of  $X_i$  now?

If  $|X_i| = 64$ , then:

$$\delta_i \oplus \delta'_i = \beta_i \oplus \beta'_i \Rightarrow \delta'_i = \delta_i \oplus \beta_i \oplus \beta'_i$$

$$S_i(\delta_i) \oplus S_i(\delta_i \oplus \beta_i \oplus \beta'_i) = Y_i \oplus Y'_i$$

$$\Rightarrow S_i(\delta_i \oplus \beta_i \oplus \beta'_i) = S_i(\delta_i) \oplus Y_i \oplus Y'_i + \delta_i$$

$\oplus \sim +$ ,  $\delta_i \sim x$ ,  $\beta_i \oplus \beta'_i \sim c$ ,  $Y_i \oplus Y'_i \sim d$ ,  $S_i \sim f$

$$\rightarrow f(x+c) = f(x) + d \xrightarrow{\text{Similar to a linear function}}$$

This would make  $S_i$  vulnerable to linear attacks.

But, in reality, due to the non-linearity of  $S_i$ , we often have  $|X_i| \leq 16$

$$\text{Let } K_i := \{\alpha_i \oplus \delta_i \mid (\delta_i, \delta'_i) \in X_i\} \quad \& \quad |X_i| \leq 16$$

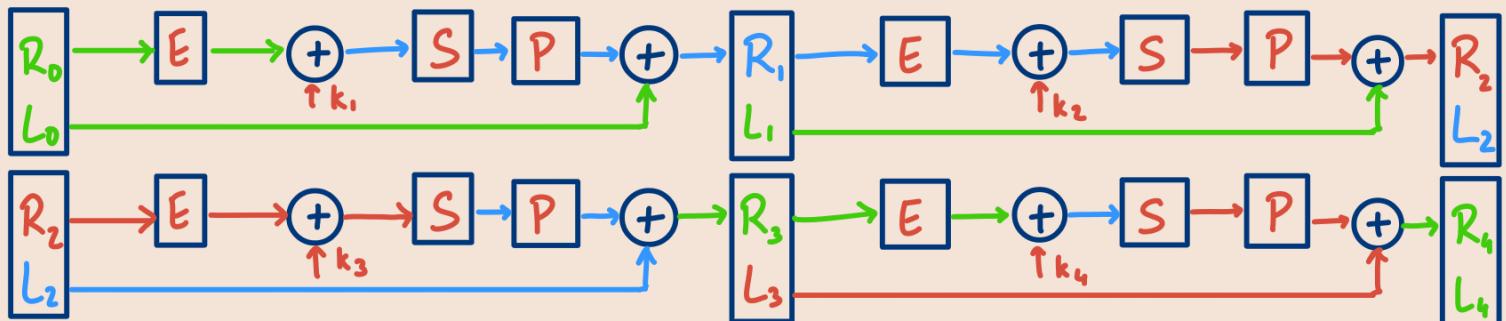
$$\Rightarrow |K_i| \leq 16 \quad \text{w/ } k_{3,i} \in K_i$$

A brute-force attack requires  $16^8 = 2^{32}$  trials.  
Instead, repeat this analysis for different input pairs and keep taking intersections until we have a single candidate for  $k_3$ .

Once we know  $k_3$ , we can work to know  $L_2 = R_1$ , which reduces the system to DES w/ 2 rounds which we know how to break.

\* Chosen plaintext attack was required to break DES: 3 rounds.

DES: Four Rounds



just because we know the XOR of the inputs to an S-box does not mean that we can know the XOR of the output due to the design of S-boxes.

Can work with finding the key upto some probability, ie, guessing the output XOR.

For example,

S1-Box : Input XOR = 001100  $\Rightarrow$  Output XOR = 1110

with probability  $14/64$   
 $\{$  64 pairs of input  $(2^6)$  w/ given input XOR:  
14 are s.t. output XOR = 1110. One would expect 4  
but S-boxes do not work like that

3 pairs w/ probability higher than  $14/64$  but this will suffice.

we would like for input XOR of S1-box of round to be 001100. Can feed 000000 to other S-boxes so that we know the output XOR (guaranteed to be 0000).

From here on, represent numbers in hexadecimal

Let  $X_i = \{(s_i, s'_i) : s_i \oplus s'_i = \beta_i \oplus \beta'_i\}$ ,

$$S_i(s_i) \oplus S_i(s'_i) = y_i \oplus y'_i \},$$

$|X_i| \leq 16$ . Let  $K_i = \{\alpha_i \oplus s_i : (s_i, s'_i) \in X_i\}$

$$\Rightarrow |K_i| = |X_i| \leq 16$$

We have, with probability  $\gg 14/64$ ,  $k_{4,i} \in K_i$  can repeat the experiment multiple times & take the most frequently occurring element amongst these sets and let this element be our guess for  $k_{4,i}$ . Then, we would like to know

$$P[k_{4,i} \neq m \in K_i]. |k_{4,i}| = 6 \Rightarrow 64 \text{ candidates}$$

for  $m$  and  $|K_i| = 16$

$$\therefore P[k_{4,i} \neq m \in K_i] = 16/64$$

Compute  $l$  sets  $K_i^1, K_i^2, \dots, K_i^l$ . Expected no. of sets in which  $m \neq k_{4,i}$  occurs:  $\sum_{j=1}^l \frac{|K_i^j|}{64} \leq \frac{16l}{64}$  } Assumption

Similarly, expected no. of sets  $k$  occurs in

$$\geq \frac{14l}{64} + \sum_j |K_i^j| \cdot \frac{50}{64}$$

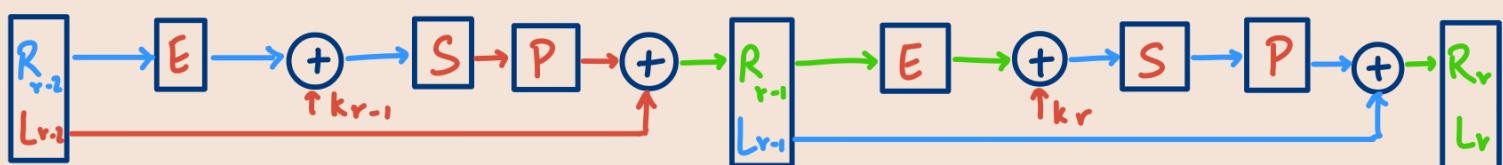
output XOR is the  
one we wanted

output XOR is wrong  
in  $j^{th}$  group

- When  $|K_i^j| = 16 \rightarrow \frac{14l}{64} + \frac{50l}{64} \cdot \frac{1}{4} = \frac{26.5l}{64}$

- Taking  $l=64$  gives a substantial difference of 10 frequencies.

DES:  $r$  Rounds



**Characteristic:** An  $s$ -round characteristic is a tuple  $(x_0, y_0, p_1, x_1, y_1, p_2, x_2, y_2, \dots, p_s, x_s, y_s)$  where  $|x_i| = |y_i| = 32$ ,  $0 \leq p_i \leq 1$ , and when input XOR of round  $i+1$  is  $x_i y_i$ , its output XOR is  $x_{i+1} y_{i+1}$  with probability  $p_{i+1}$ .

**Observation:** To break  $r$ -round DES, we need an  $(r-2)$ -round characteristic.

- A 2-round characteristic was used to break 4-round DES:  $(6\bar{0}, \bar{0}, 1, \bar{0}, 6\bar{0}, 14/64, 6\bar{0}, 00828000)$

- The probability of output XOR being  $x_{i+1} y_{i+1}$  given that the input XOR was  $x_i y_i$  is  $\prod_p p$ , which will keep decreasing with increasing  $i$ .
- If probability is  $p$ , then the key will be present in  $\approx p \cdot l + (1-p) \cdot l/4$  sets, while other values will be present in  $\approx l/4$  sets.

- So, it is present in  $\approx 3p/4$  more sets. We would need  $l \approx 20/p$  to be one of a candidate.
- This technique of breaking DES using XOR values is known as Differential Cryptanalysis. Proposed by Biham & Shamir in 1990.
- 2-round characteristic:  $(\bar{0}, 196\bar{0}, 1'234, 196\bar{0}, \bar{0}, 1, \bar{0}, 196\bar{0})$  can be concatenated  $r$  times to create a  $2r$ -round characteristic.
- For 16 rounds, need a 14-round characteristic  $p = (1'234)^7 \approx 1'2^{55} \Rightarrow l \approx 2^{59}$   
 $\therefore 16$  is the minimum number of rounds that makes DES fully resistant against differential cryptanalysis.
- Linear Cryptanalysis: Proposed by Matsui in 1994.

- Property of S5: Input =  $b_0 b_1 b_2 b_3 b_4 b_5$ , Output =  $c_0 c_1 c_2 c_3$ .  
 Then  $b_0 \oplus c_0 \oplus c_1 \oplus c_2 \oplus c_3 = 0$   
 with probability  $12/64 \xrightarrow{\text{Linear Nature}}$
- Using linearity, consider round  $i$ ; Input =  $L_{i-1} R_{i-1}$   
 then we have,  $\underbrace{\oplus \text{ these four bits of } L_{i-1} \text{ together}}$
- $R_{i-1}[15] \oplus k_i[22] \oplus L_{i-1}[7, 18, 24, 29] \oplus R_i[7, 18, 24, 29] = 0$   
 with probability  $12/64$   
 holds with probability  $12/64$   
 15<sup>th</sup> bit of  $R_{i-1}$
- 3 round DES:  
 $R_0[15] \oplus k_1[22] \oplus L_0[7, 18, 24, 29] \oplus R_1[7, 18, 24, 29]$   
 $= L_2[7, 18, 24, 29] = R_3[7, 18, 24, 29] \oplus \underbrace{f(R_2, k_3)[7, 18, 24, 29]}_{\text{Output of 3rd round after P}}$
- Guess 6 bits of  $k_3$ . If guess is wrong, then eqn

holds for roughly half of the times. If guess is right, it only holds  $\frac{12}{64}$  of the times.

This way, we can get  $6+1$  bits of the key.

6 bits from  $k_3$  & 1 bit from  $k_1$ .  
Doing the same for 3<sup>rd</sup> round would lead to 7 more bits, 1 from  $k_3$  & 6 bits from  $k_1$ .

Can form an equation for 16-round DES too which holds w/ probability  $\frac{1}{2^{22}}$ .

Then  $\approx 2^{47}$  plaintext attacks required for 14 bits of the key.

The rest 42 bits can be found from Brute Force attacks. So, overall  $\approx 2^{47}$  attacks required.

However, this can be improved by drawing useful inferences (through frequency analysis) about the way English messages are encoded using ASCII codes (for example, the MSB is a 0 for alphanumeric characters).

Doing so, could allow us to break DES using ciphertext-only attacks.

## Finite Rings and Fields

- Rings: Collection of numbers with  $+$ ,  $-$ ,  $*$   
Ex: Integers w/ known operations. excepting division by 1 zero
- Fields: Collection of numbers with  $+$ ,  $-$ ,  $*$ ,  $/$   
Ex: Rationals, Reals, Complex Numbers
- Groups: A set of elements with one operation

Let  $(G, \cdot)$  be a group. Then:

- 1)  $a \cdot b \in G \quad \forall a, b \in G \rightarrow$  Closure
- 2)  $a \cdot (b \cdot c) = (a \cdot b) \cdot c \quad \forall a, b, c \in G \rightarrow$  Associativity
- 3)  $\exists e \in G : a \cdot e = e \cdot a = a \quad \forall a \in G$   
Identity element  $\rightarrow$  Existence of 'e'
- 4)  $\forall a \in G \quad \exists! b \in G$  s.t.  $a \cdot b = e$   
Existence of: Inverse of a

Examples:  $(\mathbb{Z}, +)$ ,  $(\mathbb{Q}, +)$ ,  $(\mathbb{Q} - \{0\}, *)$ ,  $(\mathbb{R}^{n \times n}, +)$   
 $(n \times n$  invertible matrices,  $*$ ),  $(S_n, \circ)$

Permutation of  $[1, n]$

\* Commutative Groups: A Group with the additional property:  $a \cdot b = b \cdot a \quad \forall a, b \in G$

## Rings

A set of elements  $R$  with operators such that:

- 1)  $(R, +)$  is a commutative group
  - 2)  $(R, *)$  satisfies properties 1-3 of groups (i.e., existence of inverse is not guaranteed)
  - 3)  $a * (b + c) = a * b + a * c \quad \forall a, b, c \in R$
- Ex:  $(\mathbb{Z}, +, *)$ ,  $(\mathbb{Q}, +, *)$ ,  $(\mathbb{R}^{n \times n}, +, *)$ ,  
 $(\mathbb{Q}[x], +, *)$

Commutative Rings: Ring  $(R, +, *)$  with additional property:  $a * b = b * a \quad \forall a, b \in R$

## Fields

A set of elements with operations such that:

- 1)  $(F, +, *)$  is a commutative ring
- 2)  $(F - \{0\}, *)$  is a commutative group

identity of addition

Ex:  $(\mathbb{Q}, +, \times)$  or replace  $\mathbb{Q}$  with  $\mathbb{R}, \mathbb{C}$ .  
 •  $F - \{0\} =: F^*$

## Finite Groups / Rings / Fields

Finite number of elements.

Ex:  $(\mathbb{Z}_n := \{0, 1, \dots, n-1\} \equiv \text{Set of integers modulo } n,$   
 $+ (\text{mod } n), * (\text{mod } n)) \leftarrow \text{Finite ring}$   
 $(\mathbb{F}_p := \{0, 1, \dots, p-1\}, + (\text{mod } p), * (\text{mod } p))$   
 ↙ Same as  $\mathbb{Z}_{n=p}$  except  $p$  is a prime  
 Finite Field. Consider  $0 \neq a \in \mathbb{F}_p$   
 $\Rightarrow \exists k, l \text{ s.t. } ka + lp = 1 \Rightarrow ka = 1 \pmod{p}$   
 $\Rightarrow k \pmod{p} \in \mathbb{F}_p \text{ & is inverse of } a.$

- Closure of arithmetic operations over finite fields guarantees that numbers do not grow arbitrarily large irrespective of the number of operations

## Function Fields

- Let  $F[x]$  be the set of all polynomials in  $x$  with coefficients from field  $F$ .
- Then,  $(F[x], +, *)$  is a commutative ring where arithmetic is over polynomials.
- Let  $F(x)$  be the set of rational functions in  $x$ , that is:  

$$F(x) = \{ f(x)/g(x) \mid f(x), g(x) \in F[x], g(x) \neq 0\}$$
- Then,  $(F(x), +, *)$  is a field:
  - Multiplicative inverse of  $f/g$ ,  $f \neq 0$ , is  $g/f$

## Prime Extension Fields

- Let  $f(x) \in F_p[x]$  be an irreducible polynomial over  $F_p$

→  $f(x)$  cannot be factored as  $f_1(x)f_2(x)$  with  $f_1, f_2 \in F_p[x]$ , both of degree  $> 0$ .

Let degree of  $f$  be  $d$

Define  $F_{p^d}$  to be the set of all polynomials of degree  $< d$  in  $F_p[x]$

Then,  $(F_{p^d}, +, *)$  is a field with arithmetic modulo  $p$  and  $f(x)$ :

- All co-efficients are reduced mod  $p$  and all powers of  $x$  of degree  $\geq d$  are reduced mod  $f(x)$
- Mult. inv. of  $g \in F_{p^d}^*$  is  $h \in F_{p^d}$  such that  $gh + rf = 1 \pmod{p}$

A finite field has size  $p^d$  where  $p$  is a prime and  $d > 0$

**Order:** Let  $(G, \cdot)$  be a finite commutative group and  $a \in G$ . Order of  $a$  is the smallest non-zero number  $k$  such that  $a^k = e$

Order is well defined for each  $a \in G$ ,  $G$  finite

- Let  $A := \{a^i \mid i > 0\} \subseteq G$ .  $G$  finite  $\Rightarrow A$  finite
- Let  $a^k$  be the largest power of  $a$  in  $A$

Then  $a^{k+i} = a^i$  for some  $i \leq k \Rightarrow a^{k+i-i} = e$

**Lemma:** If order of  $a$  equals  $k$ , then for every  $l$  such that  $a^l = e$ :  $k \mid l$

- Let  $m = (l, k) \Rightarrow uk + lv \Rightarrow a^m = a^{uk} \cdot a^{lv} = e$
- $\therefore k = \text{order} \therefore k \leq m$

$\therefore m \mid k$ ,  $k \leq m \Rightarrow k = m \Rightarrow k \mid l$

Theorem: Let  $(G, \cdot)$  be a finite commutative group. Then  $\forall a \in G$ ,  $a^{|G|} = e$

Hint: Use  $A := \{ab \mid b \in G\}$ , then look at  $\prod_{i \in A} i$  &  $\prod_{i \in G} i$

Corollary: Order of each element divides  $|G|$

### Cyclic Groups

Let  $(G, \cdot)$  be a commutative group.  $G$  is cyclic if  $\exists a \in G$  such that  $G = \{a^i \mid i \in \mathbb{Z}\}$

Element  $a$  is called generator of the group. If  $G$  is finite, then order of  $a$  equals  $|G|$

- Eg:  $(\mathbb{Z}, +)$ ,  $(F_p, +)$  are cyclic groups w/ generator  $1$ .
- $(\mathbb{Q}, +)$ ,  $(F_{pd}, +) \mid_{d>1}$  are not cyclic groups.

Theorem: For a finite field  $F$ ,  $(F^*, *)$  is a cyclic group.

Proof: Let  $\prod_i p_i^{r_i} = m = |F^*|$  and  $S_i = \{a \mid a \in F^*, \text{ order of } a \text{ divides } p_i^{r_i}\}$  is a group.

- $S_i$  is a cyclic group
  - Let  $a_i \in S_i$  be an element w/ max. order  $p_i^{s_i}$ , for some  $s_i \leq r_i$
  - Order of each element of  $S_i$  divides  $p_i^{s_i}$
  - $\therefore \underbrace{y^{p_i^{s_i}}}_{\text{At most } p_i^{s_i} \text{ roots}} = 1 \quad \forall y \in S_i$
  - But,  $a_i$  has exactly  $p_i^{s_i}$  distinct powers, all in  $S_i$ ,  $\therefore a_i$  is generator of  $S_i$
  - By Structure Theorem of Finite Fields,

each  $b \in F^*$  can be uniquely written as a product of one element of  $S_i$  for each  $i$

- $\therefore m = \prod_i p_i^{s_i} \Rightarrow s_i = r_i \ \forall i$
- $a := \prod_i a_i^{r_i}$
- $a$  is a generator of  $F^*$ :
- Let  $a^{m'} = 1 \Rightarrow 1 = \prod_i a_i^{m'} \Rightarrow a_i^{m'} = 1$
- $\therefore p^{r_i} | m' \Rightarrow m | m' \Rightarrow m' = m$

Theorem: Let  $F$  be any finite field. Then polynomial  $q(y) \in F[y]$  of degree  $d$  has at most  $d$  roots in  $F$ .

### Advanced Encryption Scheme (AES)

- Designed in ~1997, adopted in 2001, originally called Rijndael.
- Developed by Vincent Rijmen and Joan Daemen.

Parameters: Blocksize = 128 bits

Keysize = 128, 192, or 256 bits

Rounds = 10, 12 or 14

Each round has four operations in a sequence:  
ByteSub, ShiftRow, MixColumn, AddRoundKey.

### Viewing a Block

- A block is 128 bits, or 16 bytes long
- It is viewed as an element of  $F_{256}^{4 \times 4}$

## ByteSub

$$b_{ij} = \begin{cases} Y_{aij}, & \text{if } a_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

Non-linear part

$$C_{ij} = T \cdot b_{ij} + c \quad \rightarrow \text{linear part}$$

- $T$  is a fixed  $8 \times 8$  invertible matrix over  $F_2$  and  $c$  is a fixed column vector over  $F_2$
- In  $F_{256}^*$ ,  $Y_{aij} = a_{ij}^{254} \therefore a_{ij}^{255} = 1$
- The only non-linear operation. Easily seen to be invertible.

## ShiftRow

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{11} & a_{12} & a_{13} & a_{10} \\ a_{22} & a_{23} & a_{20} & a_{21} \\ a_{33} & a_{30} & a_{31} & a_{32} \end{bmatrix}$$

- A left rotation of  $i$  columns is applied on  $i^{th}$  row
- Each column in new matrix consists of one element from every column of old matrix

## MixColumn

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \xrightarrow{\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

- A fix matrix is multiplied to block matrix
- Viewing each column as a degree 3 polynomial in  $F_{256}[x]$ , the operation is same as multiplying column polynomial by fixed polynomial  $3x^3 + x^2 + x + 2 \bmod x^4 + 1$

## AddRoundKey

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} + \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{bmatrix}$$

- Round key matrix is added to block matrix
- Addition is bitwise XOR in  $F_{256}$
- Each round key matrix is derived from key using a fixed algorithm.

Rounds: 10, 12 or 14 depending on key size

- One additional ARK performed before first round.
- Last round does not have MC operation

Observations:

- MC only operation that 'mixes' elements,
- Together with SR, in two rounds, all elements mixed up.
- BS is the only non-linear operation — linear part chosen so that at least one operation not in  $F_{256}$ .

Decryption

- The rounds and ordering of round operations get reversed.
- ARK remains same, except use of round keys is reversed.
- SR and BS can be exchanged.
- ARK and MC can be exchanged.
- $M \cdot (B + K) = M \cdot B + M \cdot K = (M \cdot B) + \underline{(M \cdot K)}$
- $\therefore$  MC not used in last round, some other key added  
MC & ARK of 2<sup>nd</sup> last round can be exchanged and viewed as operations in 1<sup>st</sup> encryption round.

AES: 1 round

- Sequence of operations: ARK → BS → SR → ARK
- No mixing  $\Rightarrow$  byte-by-byte encryption
- Can be broken w/ a 256 byte chosen-plaintext attack.

AES: 2 rounds

- ARK → BS → SR → MC → ARK → BS → SR → ARK
- Mixing only on four byte groups.
- Can be broken w/ a  $2^{32} \approx 4\text{ GB}$  chosen plaintext attacks

AES: 3 rounds

- Differential cryptanalysis does not work:  
$$\frac{x}{x} + \frac{1}{x+d_0} = \frac{d_0}{x(x+d_0)}$$
 — two plaintexts w/ fixed diff  
BS — Takes some specific value w/ prob.  $\leq 2^{-28}$
- After MC, difference propagated to four bytes  
$$\frac{d_0+08}{x(x+d_0)} = d_1 \Rightarrow \begin{cases} 2\text{-byte} \\ q^n \text{ in } x \\ \text{At most 2 roots} \end{cases}$$
 256
- After 2 rounds, prob. of diff. being a fixed value is  $\leq 2^{-28}$
- Linear Cryptanalysis does not work due to similar reasons.

- \* A modified form of differential cryptanalysis, called square attack, does break three round AES.
- In this, we consider a set of 256 plaintext blocks being encrypted simultaneously & trace patterns

## AES: 3-rounds

Input block:  $4 \times 4$  matrix of elements from  $F_{256}$

Patterns of 256 values from  $F_{256}$ :

P : all 256 values are distinct

C : all 256 values are same

Z : sum of 256 values is 0

X : everything

P is also Z; C is also Z

Add RoundKey: Same one byte added (XOR-ed) to each byte.

$$P \xrightarrow{\text{ARK}} P$$

$$C \xrightarrow{\text{ARK}} C$$

$$Z \xrightarrow{\text{ARK}} Z$$

$$X \xrightarrow{\text{ARK}} X$$

ByteSub: Non-linear operation, but one-to-one.

$$P \xrightarrow{\text{BS}} P$$

$$C \xrightarrow{\text{BS}} C$$

$$Z \xrightarrow{\text{BS}} X$$

$$X \xrightarrow{\text{BS}} X$$

ShiftRow: Relocates the byte to some other location, but the bit-pattern of a byte remains the same.

$$P \xrightarrow{\text{SR}} P$$

$$C \xrightarrow{\text{SR}} C$$

$$Z \xrightarrow{\text{SR}} Z$$

$$X \xrightarrow{\text{SR}} X$$

## Mix Column:

$$\begin{bmatrix} C \\ C \\ C \\ C \end{bmatrix} \rightarrow \begin{bmatrix} C \\ C \\ C \\ C \end{bmatrix}, \quad \begin{bmatrix} P \\ C \\ C \\ C \end{bmatrix} \rightarrow M \begin{bmatrix} \alpha_{i,1} \\ \alpha_{i,2} \\ \alpha_{i,3} \\ \alpha_{i,4} \end{bmatrix} \rightarrow \begin{bmatrix} \sum_k m_{1k} \alpha_{ik} \\ \sum_k m_{2k} \alpha_{ik} \\ \sum_k m_{3k} \alpha_{ik} \\ \sum_k m_{4k} \alpha_{ik} \end{bmatrix} \xrightarrow{\alpha_{ik} \neq 0 \Rightarrow P} \begin{bmatrix} P \\ P \\ P \\ P \end{bmatrix}$$

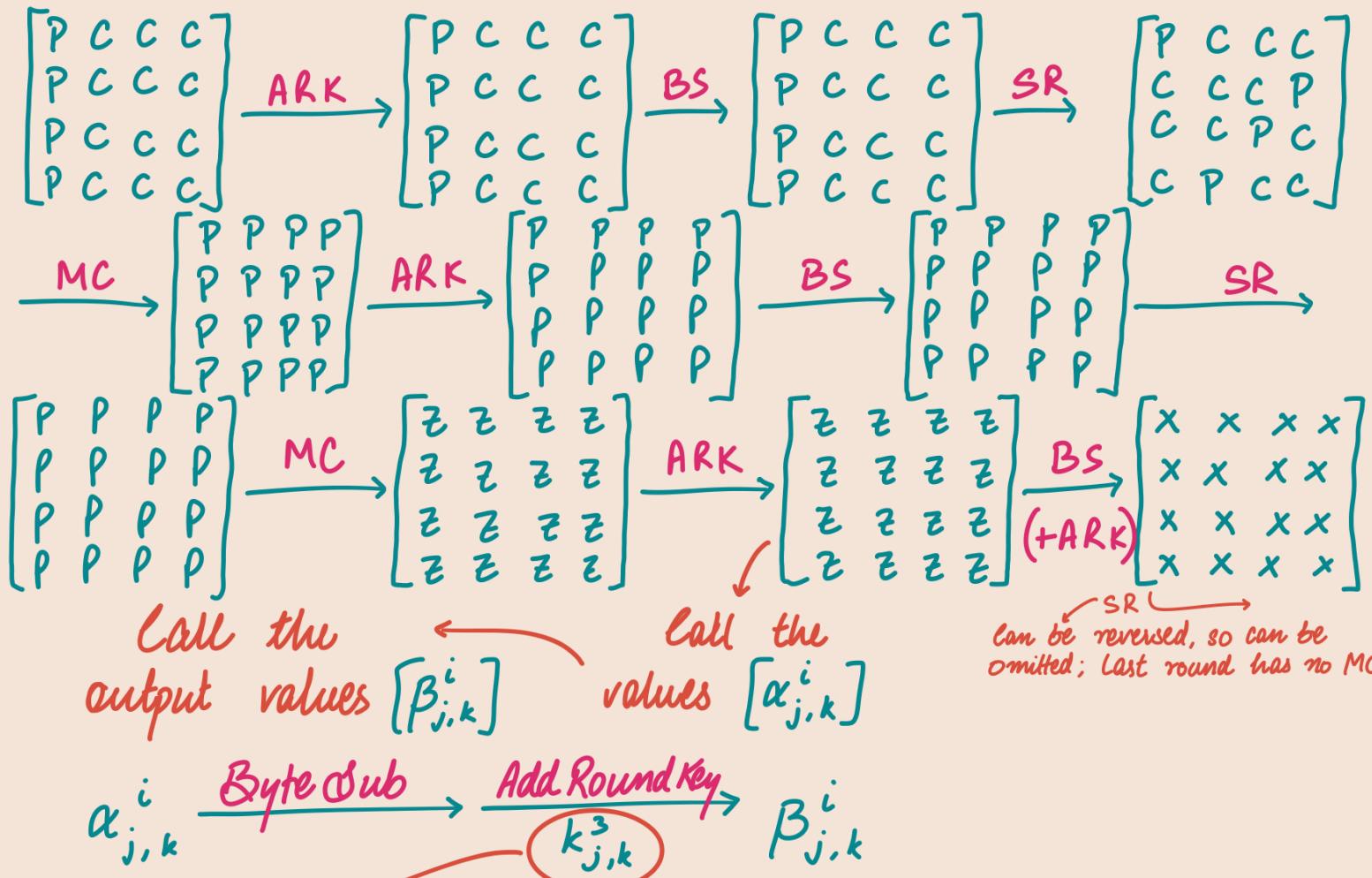
Sum still remains  
 zero  $\therefore P \subseteq Z, C \subseteq Z$

$(m_{11}\alpha_{i1}) + (m_{12}\alpha_{i2} + m_{13}\alpha_{i3} + m_{14}\alpha_{i4})$   
 Takes all 256 values w/ i changing i  $\Rightarrow$  does not change w/ i

$\Rightarrow$  Takes all 256 vals w/ changing i  $\Rightarrow P$

Now, we observe how these patterns of 256

values move with AES operations:



- Guess a value of  $k_{j,k}^3$ , compute  $\alpha_{j,k}^i + k_{j,k}^3$  & check if their sum is zero. If the guess is right, then the sum will be zero, other values of  $k_{j,k}^3$  are unlikely to give zero sum.
- Obtain a set of candidate  $k_{j,k}^3$ 's. Repeat with another input & take intersection till only one candidate remains.
- 256 brute-force attacks help yield one byte of  $k^3$ , i.e.,  $k_{j,k}^3$ . To get the complete 128 bits of  $k^3$ , only need  $256 \times 16 = 4096$  (128/8) brute-force attacks.
- For AES: 4 rounds, use  $\begin{bmatrix} P & C & C & C \\ C & C & C & C \\ C & C & C & C \\ C & C & C & C \end{bmatrix}$  as the inputs.

This is called Square Attack.

It works upto 4-rounds. It fails beyond that.

For 6-round AES,  $\exists$  an attack taking time  $\approx 2^{126}$  operations, better than Brute force attack by a factor of 4.

## Key Exchange

[~1977] Diffie Hellman: Uses exponentiation  
1<sup>st</sup> method

- Method:
- 1) Anubha and Braj agree on a large prime  $p$  & a generator  $g$  of  $\mathbb{F}_p^*$
  - 2) Anubha picks a random number  $a$ ,  $0 < a < p-1$ , computes and sends  $g^a \pmod{p}$  to Braj
  - 3) Braj picks a random number  $b$ ,  $0 < b < p-1$ , and sends  $g^b \pmod{p}$  to Anubha.
  - 4) Anubha computes  $(g^b)^a \pmod{p}$   $\hookrightarrow$  Equal Braj computes  $(g^a)^b \pmod{p}$   
 $\hookrightarrow$  This is the key

What does Ela know?

What does Ela want?

$p, g, g^a \pmod{p}, g^b \pmod{p}$

$g^{ab} \pmod{p}$

Can be used to find  $a$   
Keep multiplying with  $g^{-1}$  till we get 1.

$\hookrightarrow$  Requires  $a$  multiplications

Want  $a \sim 2^{128} \Rightarrow p \sim 2^{128}$

$\therefore$  Simply multiplying with  $g^{-1}$  does not work  
Need to find some other method.

## Discrete Log Problem

Given a finite group  $G$ , and  $g, h \in G$   
find  $a$  such that  $h = g^a$ .

Believed to be hard for group  $F_p^*$   
when  $p$  is large.

Fastest known algorithm  
time  $2^{O(\log^{1/3} p \log \log^{2/3} p)}$

- For a  $p$  1024 bits long, the best algorithm for discrete logarithm problem takes  $2^{4 \cdot (1024)^{1/3}} (10)^{2/3} \approx 2^{180}$  operations

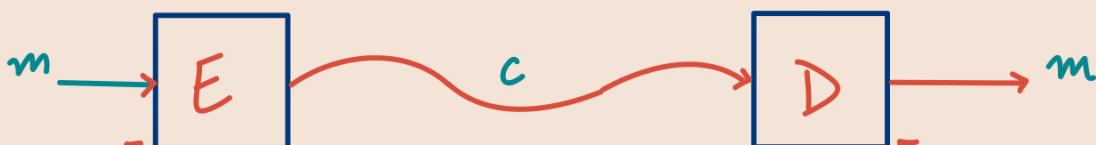
Efforts required by Anubha & Braj :

- Find a large prime  $p$  (1024 bit) [ $\sim 1024$  expected picks]
- Find a generator  $g$  of  $F_p^*$   
Choose  $P$  such that  $p-1 = 2q$ ,  $q$  prime  
Sophie-Germaine primes.

Conjecture: In  $[1, n]$ , there are  $\sim \frac{n}{\log^2 n}$  Sophie-Germaine primes.

- For Sophie-Germaine prime  $p = 2q + 1$ ,  
 $F_p^* = G_2 \times G_a$   $\rightarrow$  Either  $-a$  or  $+a$ , is a generator  
of order  $2q$  for  $a \neq \pm 1$

## Public-Key Encryption



$k_E$  → Needs to be secret  
in private key encryption, can be public in public key encryption

- Possibility discussed by Diffie-Hellmann in 1976
- Public-key encryption solves key-exchange problem
- Chosen plaintext attack can be carried out easily.
- \* Two distinct map requirements: One for the keys, one for texts. Finding  $k_D$  from  $k_E$  should be hard.

One-way maps:

Discrete-Log :  $e \rightarrow a^e \pmod{n}$

Factoring :  $(p, q) \rightarrow pq$

Computing the map ( $\rightarrow$ ) is easy but, inverting the map ( $\leftarrow$ ) is difficult.

## RSA

- \* Rivest-Shamir-Adleman, 1977
- First public-key encryption algorithm
- Uses the previous two one-way maps.

## Key Generation

- 1.) Choose two large primes  $p$  &  $q$
- 2.) Compute  $n := pq$
- 3.) Pick a random  $e$ ,  $1 < e < (p-1)(q-1)$ , such that  $\gcd(e, (p-1)(q-1)) = 1$
- 4.) Compute  $d$  such that  $de \equiv 1 \pmod{(p-1)(q-1)}$

Public Key  $(k_E) = (n, e)$

Private Key  $(k_D) = d$

## Encryption

- 1.) Break plaintext into blocks such that each block is a number between  $1 \leq n$
- 2.) Let  $m$  be a number corresponding to a block. Compute ciphertext block  
 $c = m^e \pmod{n}$

## Decryption

- 1.) Let  $c$  be the number corresponding to a ciphertext block. Compute  
 $m = c^d \pmod{n}$

## Analysis (E + D)

$$\begin{aligned}
 c^d \pmod{n} &= (m^e)^d \pmod{n} = m^{ed} \\
 &= m^{1+k(p-1)(q-1)} \pmod{n} \\
 &= m * (m^{(p-1)(q-1)})^k \pmod{n} \\
 &\stackrel{\text{provided } m \in \mathbb{Z}_n^*}{=} m \pmod{n}
 \end{aligned}$$

Set of nos.  $< n$   
 that are co-prime to  $n$

$$|\mathbb{Z}_n^*| = (n-1) - \underbrace{(p-1)}_{\text{Factors of } p} - \underbrace{(q-1)}_{\text{Factors of } q} = n-p-q+1 = (p-1)(q-1)$$

So, encryption + decryption works if  $m \in \mathbb{Z}_n^*$ ,  
 but what if  $m \notin \mathbb{Z}_n^*$ ?

Then,  $\gcd(m, n)$  is either  $p$  or  $q$ .

Then,  $(p-1)(q-1)$  can be computed easily.

Then,  $d$  can be computed easily.

Numbers not in  $\mathbb{Z}_n^* = p-1 + q-1$

$$\Rightarrow P[m \notin \mathbb{Z}_n^*] = \frac{p-1+q-1}{n-1} \approx \frac{p+q}{n} = \frac{1}{p} + \frac{1}{q}$$

Very small if  $p$  &  $q$  are large

## Security of $k_D$

Encryption Key :  $(n, e)$   
Decryption Key :  $d$

$n$  can be factored  $\Leftrightarrow (p-1)(q-1)$  can be computed

$\Rightarrow$  is trivial.  $\Leftarrow n = pq$  known, can find  $p+q$ , then finding  $p, q$  same as solving a quadratic equation.

$(p-1)(q-1)$  can be computed  $\Leftrightarrow d$  can be computed

## Factoring $n$

1.) School Method

2.) Quadratic Sieve method

3.) Number Field Sieve method

$$2^{\frac{O(\log n \log \log n)^{1/2}}{2^{O(\log^{1/3} n \log \log^{2/3} n)}}}$$

Taking  $n > 1024$  bit is recommended as a safeguard.

## Time Complexity

### Key Generation

$p, q$  : Done by randomly choosing a number of size  $l$  & testing if it is prime.  
 $d$  : Computing from  $e$  can be done via extended GCD.

### Encryption

$m^e \pmod{n}$  :  $O(\log e) = O(\log n)$  multiplications  
 $= O(\log^2 \log \log n)$

→ Encrypting a plaintext, even though efficient, takes non-trivial time.

RSA is used primarily for exchanging small amount of information, like AES keys, and AES used for bulk encryption.

## Decryption

→ Same as Encryption

- \* If  $d$  is not known, computing  $m$  from  $c$  has no method in general.
- This makes RSA encryption very secure in general. However, one needs to avoid several special cases when choosing the key, where the system becomes insecure.

## Cryptanalysis: Prime Factors

$n = pq \longrightarrow$  If  $p$  is small, then only  $O(p)$  operations required:

If  $p-q$  is small, then only  $O(p-q)$  operations required:  $-(p-q)/2 \leftarrow \sqrt{n} \rightarrow + (p-q)/2$

- \* In summary, both  $p$  &  $q$  should be large but not close to each other.

## Smooth Primes

Composite number  $m$  is  $k$ -smooth if all prime factors of  $m$  are  $\leq k$ . Prime number  $p$  is  $k$ -smooth if  $p-1$  is  $k$ -smooth.

Let  $n = pq$  & suppose  $p$  is  $k$ -smooth but  $q$  is not.

Let  $T = (k!)^{\log n}$

Lemma:  $\gcd(a^T - 1, n) = p$

$T$  is  $\frac{(q-1)}{k}$  &  $T \rightarrow$  Some factor of  $q-1$  is  $> k$

$(p-1)/T \rightarrow$  Every prime divisor of  $p-1$  is  $\leq k$  w/ index at most  $\log p \leq \log n$

for most  $a$ 's

Proof: Suppose  $a \in \mathbb{Z}_n^*$ . Then,  $a^T = a^{r(p-1)} = 1 \pmod{p}$

But,  $a^T \pmod{q} \neq 1$  whenever  $a$  is a generator of  $\mathbb{F}_q^*$   $\because |\mathbb{F}_q^*| = q-1 + T$

Most  $a$ 's are generators of  $\mathbb{F}_q^*$ , it follows that  $a^T - 1$  is divisible by  $p$  but not  $q$ .

Hence, both  $p$  &  $q$  should not be  $k$ -smooth for small  $k$ .

Best strategy is to choose  $p$  &  $q$  such that both  $(p-1)/2$  and  $(q-1)/2$  are primes.

Choose Oppie Germain primes.  $\xrightarrow{\text{prob. is } \approx \frac{1}{(kn)^2}}$

## Integer Lattice

Given a set of LI vectors  $v_1, \dots, v_D \in \mathbb{R}^D$ , integer lattice generated by them is:

$$L = \left\{ \sum_{i=1}^D a_i v_i \mid a_i \in \mathbb{Z} \right\}$$

$L$  is a vector space consisting of integer linear combinations of vectors  $v_1, \dots, v_D$ .

These vectors form a basis of  $L$ .

Ex:  $L$  of  $(1, 0)$   $(1, 1)$  is  $\mathbb{Z}^2$

Volume of a Lattice: Denoted by  $v(L)$ , is

defined as  $|\det(v)|$  where  $v = [v_1, v_2, \dots, v_D]$

Lemma:  $v(L)$  is independent of the basis

Proof: Let  $u, \dots, u_D$  be another basis.

Then  $U = [u, \dots, u_D] = AV$  and  $V = BU$

with  $A, B \in \mathbb{Z}^{D \times D}$ .  $\therefore \{u_i\}_{i=1}^n$  are LI

$\therefore \det(U) \neq 0 \Rightarrow U^{-1}$  exists, do,

$$U = AV = ABU \Rightarrow I = AB \Rightarrow \det(A)\det(B) = 1$$

$$\Rightarrow \det(A) = \det(B) = \pm 1$$

$$\Rightarrow |\det(U)| = |\det(v)|$$

Shortest Vector of lattice  $L$  is the minimum length non-zero vector in  $L$ .

Length of shortest vector is denoted as  $\lambda_1(L)$ .

- Finding shortest vector of a lattice is known to be a hard-to-solve problem.
- Even finding a vector of length  $w/\sqrt{2}\lambda_1(L)$  is known to be hard [Ajtai-Micriano]
- However, it is possible to find a vector of length  $w/\sqrt{2}^{(D-1)/2}\lambda_1(L)$ .

Minkowski's Theorem

For any lattice  $L$ ,  $\lambda_1(L) \leq \sqrt{D} v(L)^{1/D}$

Lenstra-Lenstra-Lovasz ( $L^3$ ) Algorithm

Given a lattice  $L$ , it computes a vector  $v$ , in time polynomial in  $D$ , such that,  $|v| \leq 2^{(D-1)/2}\lambda_1(L)$  ( $\leq 2^{(D-1)/2}\sqrt{D} v(L)^{1/D}$ )

## Small $e$

- In order to save time during encryption,  $e$  may be chosen small.

$e=1$        $\times$       Trivial

$e=2$        $\times$        $\gcd(e, \phi(n) = (p-1)(q-1)) \neq 1$

$e=3$        $\checkmark$       Smallest case possible

- But, for small  $e$ , the system can be broken without  $d$ .

Set  $e=3$ .

- Suppose that  $m$  is a 128 bit key of AES, and  $n$  is 1024 bits long.

Then  $c = m^e = m^3 \pmod{n} = m^3$  over integers and can be computed easily.

Assume that  $m = k + x$  where  $x < 2^t$  and  $k$  is known. Then,

$c = m^3 = (k+x)^3 = x^3 + 3kx^2 + 3k^2x + k^3 \pmod{n}$  gives a degree 3 polynomial whose root ( $< 2^t$ ) mod  $n$  needs to be computed.

$$P(x) = x^3 + 3kx^2 + 3k^2x + k^3 - c$$

- Define lattice  $L \subseteq \mathbb{R}^6$ :

$$L = \left[ \begin{array}{cccccc} 1 & 3k & 3k^2 & k^3 - c & 0 & 0 \\ 0 & 1 & 3k & 3k^2 & k^3 - c & 0 \\ 0 & 0 & 1 & 3k & 3k^2 & k^3 - c \\ 0 & 0 & 0 & n & 0 & 0 \\ 0 & 0 & 0 & 0 & n & 0 \\ 0 & 0 & 0 & 0 & 0 & n \end{array} \right] \rightarrow \begin{cases} x^3 P(x) \\ x^2 P(x) \\ x P(x) \\ P(x) \\ nx^2 \\ nx \\ n \end{cases} \quad \text{6 basic vectors}$$

$v(L) = n^3$

- Use  $L^3$ -algorithm to find a vector in  $L$  of length,  $\leq 2^{(6-1)/2} \sqrt{6} n^{3/6} = \sqrt{192} n$
- construct  $\Phi(x)$  as such:

$$r_1 \cdot x^2 P(x) + r_2 \cdot x P(x) + r_3 \cdot P(x) + r_4 \cdot nx^2 + r_5 nx + r_6 n$$

$\hookrightarrow Q(m) \equiv 0 \pmod{n}$

Also, using L<sup>3</sup>-algorithm, can find a vector  $r$  s.t.  $|r| \leq \sqrt{192n} \Rightarrow |r_i| \leq \sqrt{192n}$ .

Also note that  $\because r \in \mathbb{Z}$ , it represents a polynomial  $Q(x)$  s.t. each co-efficient is  $\leq \sqrt{192n}$ , and so, for any  $x < 2^l$ , we have,

$$|Q(x)| \leq \sqrt{192n} (x^5 + x^4 + x^3 + x^2 + x + 1) \\ \leq \sqrt{192n} \cdot 2^{6l} \quad (\because x < 2^l)$$

$\because x < 2^l$   
 do,  $Q(x)$  can be treated as a normal polynomial and the problem reduces to finding one of its root.

$$\leq n \quad \text{for } l < \frac{1}{12}(\log n - \log 192)$$

$$\Rightarrow x < n^{1/12}$$

$\therefore$  If unknown part is  $< n^{1/12}$ , the complete message can be recovered.

But, this bound of  $n^{1/12}$  can be improved to  $n^{1/2}$  with the choice of more sophisticated polynomials.

So, a small  $e$  cannot be chosen without compromising security.

## Small $d$

Let  $d = O(n^\epsilon)$ ,  $\epsilon > 0$ . Typically,

$e = \sqrt{n}$  in such a case

$$\begin{aligned} \text{unknown } de &= 1 \pmod{\phi(n)} = (p-1)(q-1) \\ &= 1 + r(p-1)(q-1) \end{aligned}$$

$$= 1 + r(n+1) - r(p+q) \quad \text{unknown but } O(n^{1/2})$$

$$\rightarrow r = \frac{de-1}{(p-1)(q-1)} = O(n^{1+\epsilon-1}) = O(n^\epsilon) \Rightarrow r = O(n^\epsilon)$$

$$\rightarrow 1 + r(n+1) - r(p+q) = 0 \pmod{e}$$

$\underbrace{r = O(n^\epsilon)}_{\Rightarrow r \leq K} \leq L \quad \underbrace{= O(n^{1/2+\epsilon})}_{\leq L} \leq L$

Let  $R(x, y) = Ly - (n+1)Kx - 1$ . Goal is to find  $(r, s)$  s.t.  $\begin{cases} R(r/k, s/l) = 0 \\ r/k \leq 1, s/l \leq 1 \end{cases}$  mod e

We use  $K^i x^j P(x, y)$  for additional vectors: there will be  $t$  such vectors for  $0 \leq i < t$  with terms  $x^i y^j$  &  $x^j$  for  $0 \leq i < t$ ,  $0 \leq j \leq t$ .  $\therefore$  There are a total of  $2t+1$  terms.

We can get the same number of vectors by taking additional polynomials  $K^i x^j e$ ,  $0 \leq i < t$ . All the additional polynomials are 0 mod e. These polynomials give rise to a lattice  $L$  in  $\mathbb{Z}^{2t+1}$ .

Volume of  $L$  equals

$$\prod_{i=0}^{t-1} (K^i L) \prod_{i=0}^t (K^i e) = L^t K^{t^2} e^{t+1}$$

Then, the vector computed by  $L^3$ -algorithm has length at most

$$L = 2^t \sqrt{2t+1} L^{t/(2t+1)} K^{t^2/(2t+1)} e^{t+1/2t+1}$$

Using  $L = C_L n^{1/2 + \epsilon}$ ,  $K = n^\epsilon$ ,  $\epsilon < n$ , we get

$$L < 2^t \sqrt{(2t+1)C_L} n^{t/2(2t+1) + \epsilon t/(2t+1) + \epsilon t^2/(2t+1) + \frac{t+1}{2t+1}}$$

$$= 2^t \sqrt{(2t+1)C_L} n^{(2\epsilon t^2 + 2\epsilon t + 3t + 2)/2(2t+1)}$$

If  $(2t+1)L < e$ , then we have the desired property: polynomial defined by the  $L^3$  vector is zero over integers for

$$x = r/k, y = s/l$$

This requires  $n^{(1)} < n \Rightarrow \epsilon < 1/2(t+1)$

But, a bivariate polynomial has infinitely many roots  $\rightarrow$  Identify another polynomial w/ same property using 2<sup>nd</sup> smallest vector computed by  $L^3$  algorithm: this

vector is linearly independent from the 1<sup>st</sup> one  
 & has length bounded by  $2^{(\Delta-1)/2} \sqrt{\Delta} v(L)^{1/(n-1)}$   
 $\rightsquigarrow e < Y_2(t+1) - Y_t(t+1)$

We get two bivariate polynomials  $R_1$  &  $R_2$   
 with  $(r/K, s/L)$  as common root.  
 GCD likely to give a univariate polynomial  
 in  $x$  w/ root  $r/K$ .

can find  $r$  &  $s$ . can find  $d$ .  
 If  $d < n^{1/20}$ , it can be computed  
 from  $e$  and  $n$ .

With choice of more involved polynomials,  
 one can show the same result for  $d < n^{0.212}$   
 Hence, neither  $e$  nor  $d$  should be  
 chosen small.

- 1.)  $p, q$  close but not too close
- 2.)  $p, q$  both Sophie-Germain primes
- 3.)  $d, e = \mathcal{N}(n)$

## El-Gamal Cryptosystem

### Key Generation

- 1.) Let  $(G, \cdot)$  be a commutative finite group,  
 $|G| = n$ .
- 2.) Let  $g \in G$ . Let  $\text{order}(g) = k$
- 3.) choose a number  $e, 1 < e < k$

Public Key :  $(g, g^e, k)$   
 Private Key :  $e$

## Encryption

- Let plaintext  $m \in G$
- 1.) Choose a random  $r, 1 < r < k$
  - 2.) Output  $(g^r, g^{er} \cdot m)$

## Decryption

- Let ciphertext  $(h_1, h_2)$
- 1.) Output  $h_2/h_1^e (= g^{er} \cdot m / (g^r)^e = m)$

## Security

Ella knows  $(g, g^e, k, g^r, m \cdot g^{er})$

- She needs to compute  $g^{er}$
- She essentially needs to solve discrete-log problem in the group  $G$ .

Groups for which discrete log is hard to solve provide secure public-key encryption.

Eg:  $G = (F_p^*, *)$ ,  $G = (F_p, +)$

$\xrightarrow{\text{Good}}$   $\xrightarrow{\text{Bad}}$

$2^{O(\log^{1/3} p \log \log^{2/3} p)}$

## Elliptic Curves

$$y^2 = x^3 + Ax + B, \quad 4A^3 + 27B^2 \neq 0$$

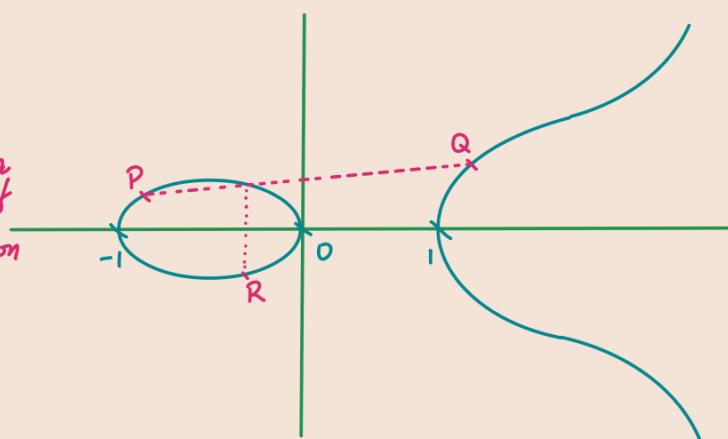
Consider  $y^2 = x^3 - x$

$P + Q := R$	}	Connect $P - Q$ using a line, the reflection of the third point of intersection
$P + Q := \infty$		$P$ & $Q$ reflection each other

$P + P := S$	}	Reflection of point of tangent at $P$
$P + \infty := P$		

  
 $\infty + \infty := \infty$



$E(\mathbb{R}) :=$  Set of points on the curve over real numbers  $\cup \{\infty\}$

$E(\mathbb{Q}) :=$  Set of rational points on curve  $\cup \{\infty\}$

$E(F_p) :=$  Set of points satisfying

$$y^2 = x^3 + Ax^2 + B \pmod{p},$$

$$4A^3 + 27B^2 \neq 0 \pmod{p},$$

$\cup \{\infty\}$

- \*  $E(\mathbb{Q})$  was used in proving Fermat's Last Theorem  
 $E(\mathbb{C})$  is donut-shaped.

## Elliptic Curves over $F_p$

Theorem (Hasse):  $p+1-2\sqrt{p} \leq |E(F_p)| \leq p+1+2\sqrt{p}$

Theorem:  $E(F_p)$  is either cyclic, or a product of two cyclic groups.

- Until now, there is no algorithm that can solve discrete log problem in  $E(F_p)$  in time  $2^{O(\log |E(F_p)|)}$ .
- \* El Gamal system based on elliptic curves is called Elliptic Curve Cryptography (ECC).

$\Rightarrow$  160 bit ECC  $\sim$  2048 bit RSA

## Quantum Computers

- Store information in a sequence of qubits.
- Each qubit is a superposition of values 0 & 1, until measured.
- Unitary transformations can be applied on

these without measurement.

Theorem (Grover): Given a list of  $n$  unsorted numbers, a quantum computer can search for a specific number in the list in time  $\sqrt{n}$ .

Theorem (Shor): Integer factoring & Discrete log, both can be solved efficiently by quantum computers.

### Learning with Errors (LWE)

Consider  $\mathbb{R}^n$ ,  $n$ -dimensional real space and suppose one is given  $u_1 \cdot s, u_2 \cdot s, \dots, u_n \cdot s$  for  $u_1, u_2, \dots, u_n \in \mathbb{R}^n$  &  $s \in \mathbb{Z}^n$ .

P1: Compute  $s$  from  $\{u_i, u_i \cdot s\}_{i=1}^{i=n}$   
Easy if  $u_i$ 's are LI

P2: Compute  $s$  from  $\{u_i, u_i \cdot s + e_i\} : 1 \leq i \leq n$   
where  $e_i$  is a random small error.  
≡ Find closest vector in the lattice

$[u_1 \dots u_n]$  to  $v$

→ Closest Vector Problem (CVP)

Theorem: Finding closest lattice vector is NP-hard.

### Kyber Public Key Encryption

- Based on LWE
- During decryption, if the result is  $> q/2$  then  $m=1$  otherwise if  $< q/2$ ,  $m=0$

## Digital Signatures

Properties required:

- (1) Hard to forge
- (2) Easy to verify
- (3) Easy to sign

### RSA Signature:

Public Key:  $(n, e)$

Private Key:  $d$

Suppose document  $m$  is to be signed,  
assume  $m < n$

Signing:  $s = m^d \pmod{n}$

Verification: Check if  $m = s^e \pmod{n}$

Signed document:  $(m, s)$

► What if message is larger?

$m \stackrel{?}{=} m_1, m_2, \dots, m_k, \quad m_i < n$

$s_i = m_i^d \pmod{n}$

Signature:  $(m, s_1, s_2, \dots, s_k) \times$

If  $m, m_2, \dots$  removed,

then  $s$  does not change much.

## Secure Hash Functions

Function  $h: \{0, 1\}^* \rightarrow \{0, 1\}^l$  is a secure hash function if:

1) Computing  $h(x)$  is easy

2) Given  $x$ , finding  $y \neq x$  with  $h(y) = h(x)$  is hard

3) Given  $h(x)$ , finding  $x$  is hard

4) Finding  $x \neq y$  such that  $h(x) = h(y)$

is hard.

## RSA Signature

Signing:  $s = h(m)^d \pmod{n}$

choose  $l$  such that  $2^l < n$

Signed document:  $(m, s)$

Verification: Check if  $h(m) = s^e \pmod{n}$

- ★ Other Public Encryption methods like ECC [Elliptic curve cryptography] can also be used for Digital Signatures.

## Identifying Secure Hash Functions

In 1991, Ron Rivest proposed MD5

- ★ Subsequently, SHA-1 & SHA-2 were proposed.

MD5 was broken in 2005; So was SHA-1 within a few years.

In 2006, NIST started a contest to identify a new secure hashing algorithm

In 2012, SHA-3 came into being

## SHA-3

Key parameters:  $r, b, d \in \mathbb{Z}, b > r$   
and an auxiliary parameter  $c := b - r$

Key component: Function  $f$   
 $f: \{0, 1\}^b \rightarrow \{0, 1\}^b$

## The Algorithm:

- 1.) Break message  $m$  into blocks of  $r$

- bits,  $m = m_1, m_2, \dots, m_k$
- 2.) Let  $s_0 := 0^b$
  - 3.) Let  $s_i := f(s_{i-1} \oplus m_i 0^c)$ ,  $1 \leq i \leq k$
  - 4.) Let  $s_{k+i+1} := f(s_{k+i})$ ,  $0 \leq i \leq \lceil d/r \rceil$
  - 5.) Let  $z_i$  be first  $r$  bits of  $s_{k+i}$
  - 6.) Output  $z_0 z_1 \dots$  truncated to  $d$  bits

## Function $f$

- Typically,  $b = 1600$
- View input  $s_i$  as  $5 \times 5$  matrix of 64-bit strings.
- Let  $a[i][j][k]$  denote  $k^{\text{th}}$  bit of string at the  $(i, j)^{\text{th}}$  cell of the matrix
- Do following operations 24 times:

- $\delta$ :  $a[i][j][k] \leftarrow a[i][j][k] \oplus \sum_{u=0}^{u=4} \{ a[u][j-1][k] \oplus a[u][j+1][k] \}$
- $\rho$ : Bitwise rotate each  $a[i][j]$  by a different triangular number  $0, 1, 3, 6, 10, \dots$
- $\pi$ :  $a[3i+2j][i] \leftarrow a[i][j]$
- $\chi$ :  $a[i][j][k] \leftarrow a[i][j][k] \oplus \{ \neg a[i][j+1][k] \wedge a[i][j+2][k] \}$
- $\iota$  (iota):  $a[0][0] \leftarrow a[0][0] + \text{Round constant}$

$0: 4\ 6\ 2$   
 $1: 0\ 8\ 2$   
 $2: 1\ b\ 3$   
 $3: 2\ 8\ 4$   
 $4: 3\ 6\ 0$

- $\delta$ : Bringing effect of other columns
- $\rho$ : change depth of each entry
- $\pi$ : Shuffle
- $\chi$ : Only non-linear operation

## 2: Break the symmetry

### Typical parameter Values

1)	$d = 224,$	$r = 1152,$	$c = 448$
2)	$d = 256,$	$r = 1088,$	$c = 512$
3)	$d = 384,$	$r = 832,$	$c = 768$
4)	$d = 512,$	$r = 576,$	$c = 1024$

### Public Key Infrastructure (PKI)

#### Certification Authorities

- Root CAs: Symantec, DigiCert, Comodo, ...
- National / Intermediate CAs: NIC, Armed Forces, ...  
(Indian Examples)
- CAs: Banks, Financial Institutions, ...

### Digital Money

#### Requirements:

- 1) Easily transferrable
- 2) Difficult to forge
- 3) Easy to verify
- 4) Double spending not possible

### Use-Case for Cryptocurrencies?

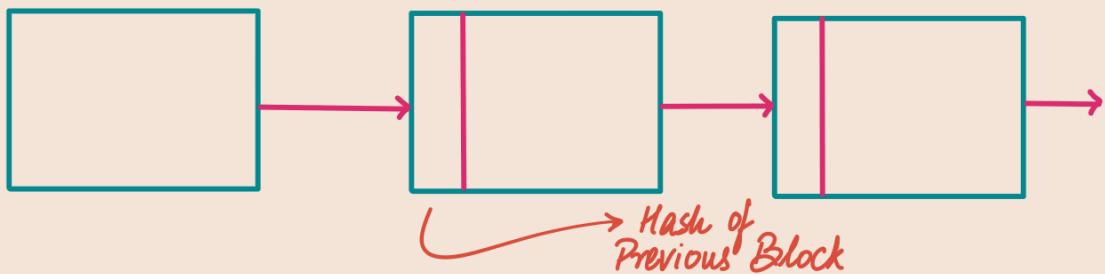
#### Typical digital money transactions:



# Bitcoin

- Published by Satoshi Nakamoto in 2008.
- Based on Blockchains

## Blockchain

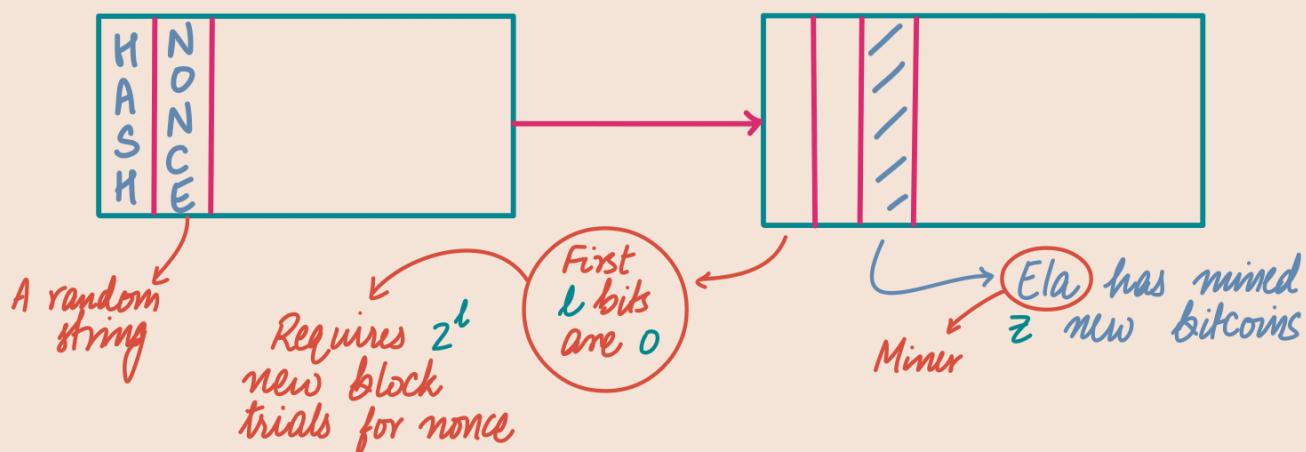


- Difficult to change order of the blocks.
- Difficult to change contents of the blocks while keeping the hash same.
- A transaction is a string of the form:
  - Anubha pays x bitcoins to Braj.These transactions are added to the blockchain.

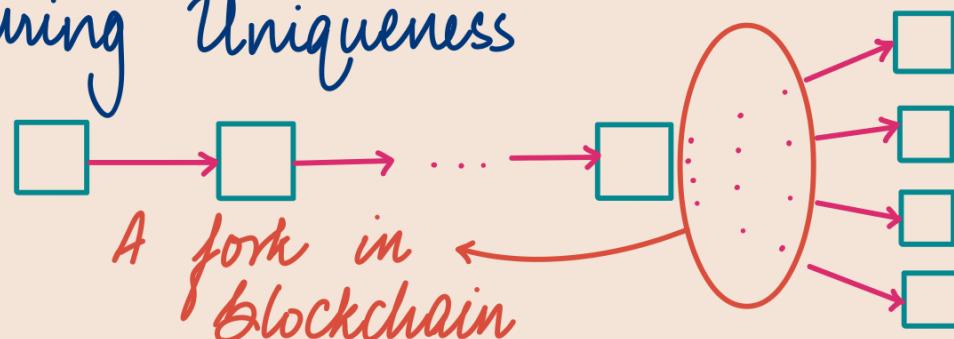
But, how is a unique blockchain maintained if it is managed by peers and not a centralized server / agency?

## Adding a Block

- Anyone can add a block by collecting new transactions on network.
- But before adding a new block, a puzzle needs to be solved.



## Ensuring Uniqueness



## Resolving Forks

- When one branch becomes 3+ blocks longer than others, remaining branches are removed.

## Zero-Knowledge Proofs

Anubha knows  $s$ . She wants to convince Brat that she knows  $s$  without revealing anything about  $s$ .

A protocol that achieves this is called Zero-Knowledge - Proof or Protocol

Theorem: All problems in NP have Zero-Knowledge Proofs.

Proof: NP has NP-complete problems as a subclass & 3-coloring can be shown to have a ZKP using cryptographically secure hash functions.