

43

UNIT - II

①

Block CIPHERS & PUBLIC KEY CRYPTOGRAPHY

Data Encryption Standard (DES) - ②

Block cipher Principles - ⑤

Block cipher modes of opn - ⑥

Advanced Encryption Standard (AES) - ⑫

Triples DES - ⑭

Blowfish - ⑰

RC5 algorithm - ⑳

Public-key cryptography - ㉓

Principles of public key crypto systems - ㉕

The RSA algorithm - ㉗

Key management - ㉚

Diffie-Hellman Key Exchange - ㉜

Elliptic curve arithmetic - ㉟

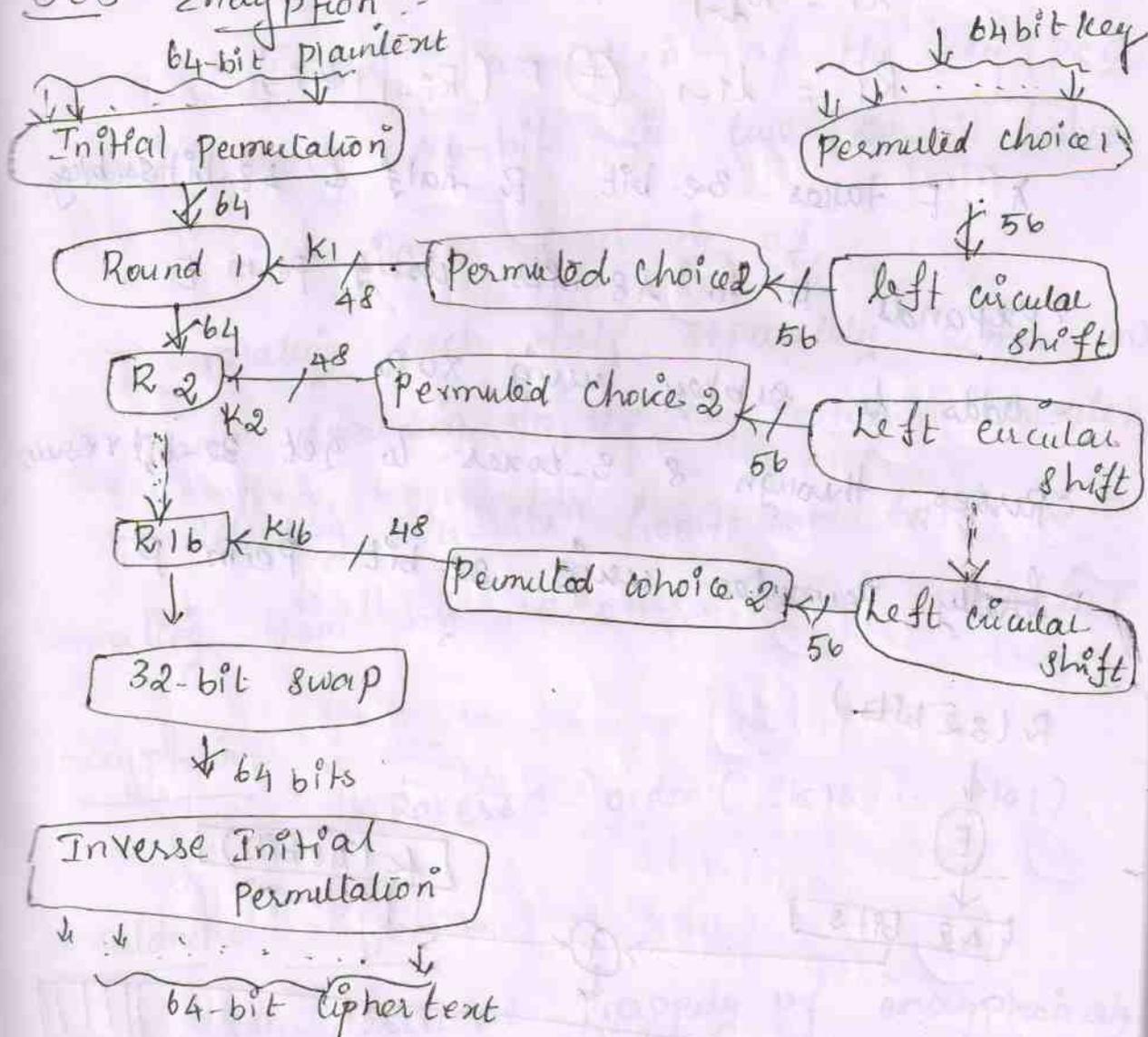
Elliptic curve Cryptography - ㊱

Block CIPHERS

Data Encryption Standard (DES):-

- * Proposed by NIST adopted in 1977.
- * It is a block cipher & encrypts 64-bits data using 56-bit key.

DES Encryption :-



Initial Permutation IP:-

- * 1st step of the data computation.
- * IP reorders the IP data bits.
- > Even bits to LH half, odd bits

Ex: IP (675a69b7 5e5a6b5a) ⁽³⁾

= (ff b 21 94 d 004 df6 fb)

DES Round Structure:

* uses two 32-bit L & R halves.

* Feistel cipher,

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

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$$

* F takes 32-bit R half & 48-bit subkey.

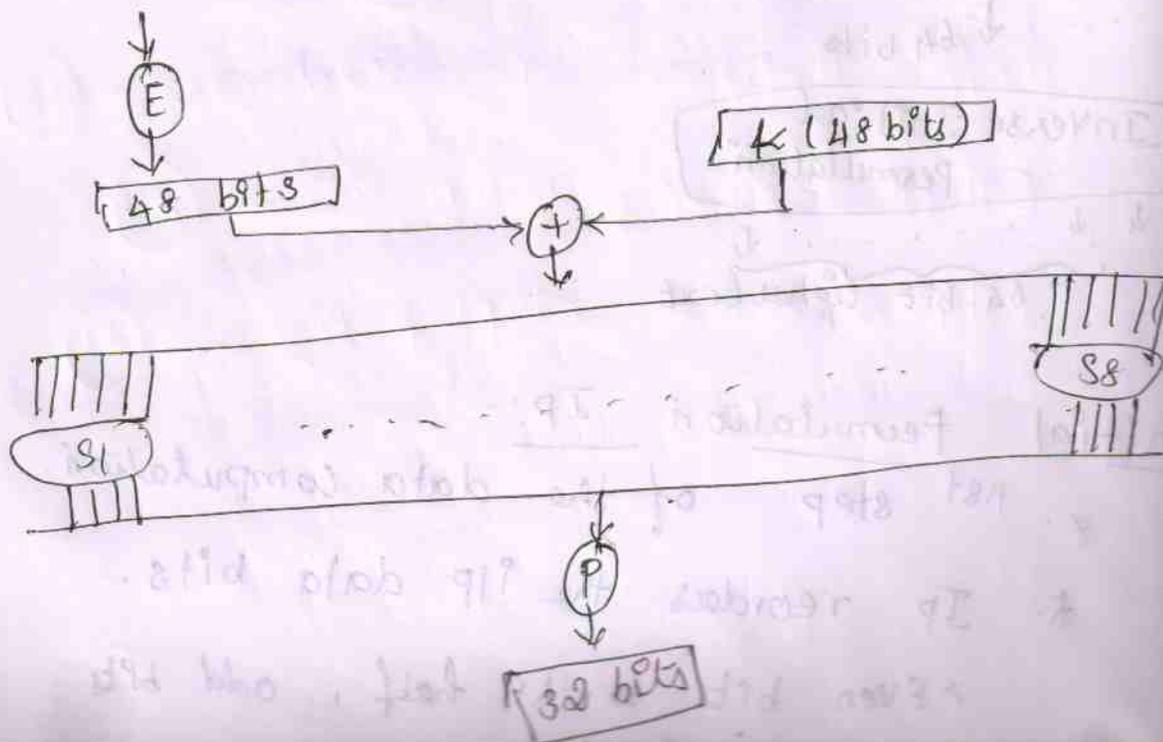
> expands R to 48-bits using Perm E

> adds to subkey using XOR

> passes through 8 S-boxes to get 32-bit result

> finally permutes using 32-bit Perm P.

R (32 bits)



③

Substitution Boxes 8:-

(4)

- * Each of the eight S-boxes is different.
- * Each S-box reduces 6 bits to 4 bits.
- * So, the 8 S-boxes implement the 48-bit to 32-bit contraction substitution.

DES Key Schedule:-

- * Forms subkeys used in each round.
 - > Initial permutation of the key (PC1) which select 56-bits in two 28-bit halves.
 - > 16 stages consisting of,
 - rotating each half separately either 1 (or) 2 places depending on the key rotation schedule.
 - selecting 24-bits from each half & permuting them by PC2 for use in round fn F.

Decryption:

- * Reverse order ($S_{k16} \dots S_{k1}$).

Avalanche Effect:-

- * Key desirable property of encryption alg.
- * where a change of one i/p (or) key bit results in changing approx half o/p bits.
- * Making attempts to "home-in" by guessing keys impossible.

Strength of DES - key size.

(5)

> 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values.

> brute-force search looked hard.

Analytic Attacks:-

> differential cryptanalysis

> linear cryptanalysis

> related key attacks.

Block cipher principles:-

* Basic principles still like Feistel in 1970's.

x. Number of Rounds.

↳ more is better, Exhaustive search best attack.

x. Function f :

↳ provides "confusion" is non-linear, avalanche.

> have issues of how S-boxes are selected.

* key schedule

> complete subkey creation, key avalanche

Block cipher modes of operation - (6)

* Block cipher encrypts fixed size blocks.

Ex: DES encrypts 64-bit blocks.

* NIST SP 800-38A defines 5 modes.

↳ block & stream modes.

Modes of Opn:-

> Electronic code book (ECB) } Blk

> Cipher Block chaining (CBC) } Blk

> Cipher Feedback (CFB) } Stream

> output Feedback (OFB) } Stream

> Counter (CTR).

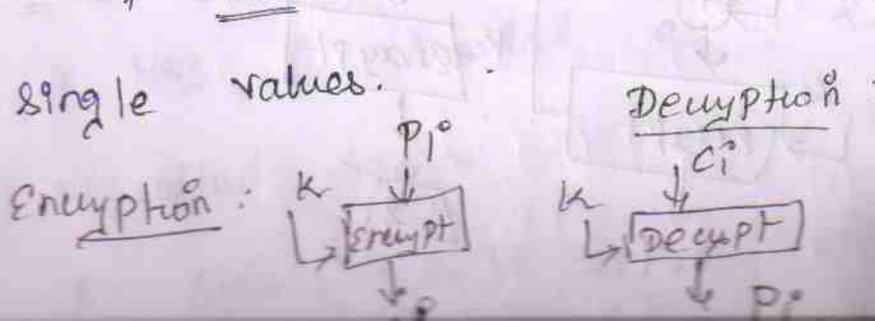
Electronic codebook Book (ECB):-

* Msg is broken into independent blks that r encrypted.

* Each blk is a value which is substituted like a codebook, hence name.

* Each blk is encoded independently of the other blks. $C_i = E_k(P_i)$.

* Uses: Secure transmission of single values.



Adv & limitations of ECB: (7)

- * Msg repetitions may show in ciphertext.
 - > If aligned with msg blks.
 - > Particularly with data such graphics (or) with msg that change very little, which become a code-book analysis pbm.
- * Weakness, is ~~not~~ independent.
- * ~~Not~~ vulnerable to cut-and-paste attack.
- * Main use is sending a few blks of data.

Cipher Block Chaining (CBC):

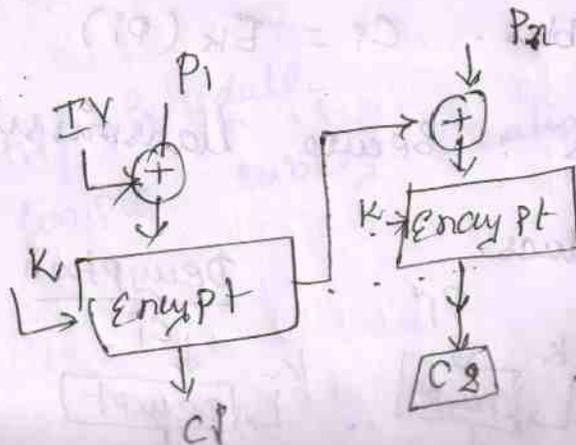
- * Msg is broken into blks.
- * Each previous cipher block is chained with current plaintext blk.

$$C_i = E_k (P_i \oplus C_{i-1})$$

$$C_1 = IV$$

uses: bulk data encryption, authentication.

Encryption:



Adv & dis adv:-

* A ciphertext blk depends on all blks before it. any change to a blk affects all following ciphertext blks ... avalanche effect.

Disadv:

- * need Initialization vector (IV)
 - > which must be known to sender & receiver
 - > Integrity must be checked.

Stream modes of opn:-

- * blk modes encrypt entire blk.
- * may need to operate on smaller units
 - ↳ Real-time data.
- * Convert blk cipher into stream cipher
 - > cipher feedback (CFB) mode
 - > output " (OFB) "
 - > counter (CTR) "
- * use blk cipher as some form of

Pseudo-random no generator ... Vernam cipher

Cipher Feedback (CFB):-

- * msg is treated as a stream of bits
- * added to the o/p of the blk cipher

* stds [128, 192, 256, 512, 1024]

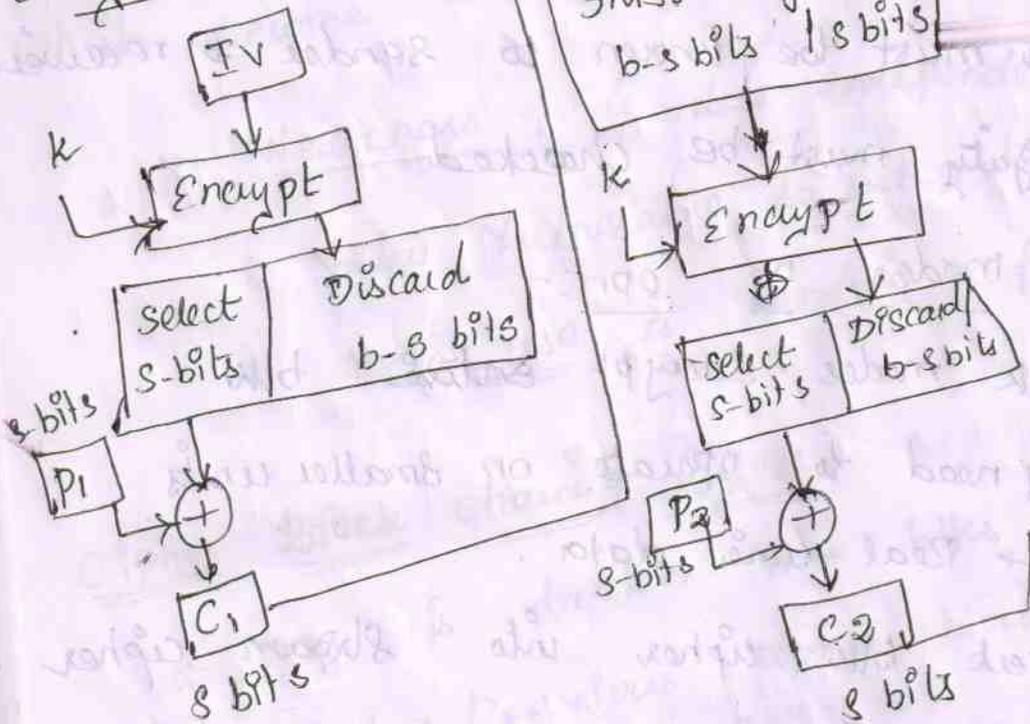
$$C_i = P_i \text{ XOR } E_k(C_{i-1})$$

(9)

$$C_{-1} = IV$$

uses: Stream data Encryption, authentication

Encryption:



Adv & Limitations:-

- * Data arrives in bits / bytes.
- * Limitation is need to stall while doing encryption after every s-bits.
- * Errors propagate for several bits after the error....

output Feedback (OFB) :-

- * O/P of cipher is added to msg.

$$O_i = E_k(O_{i-1})$$

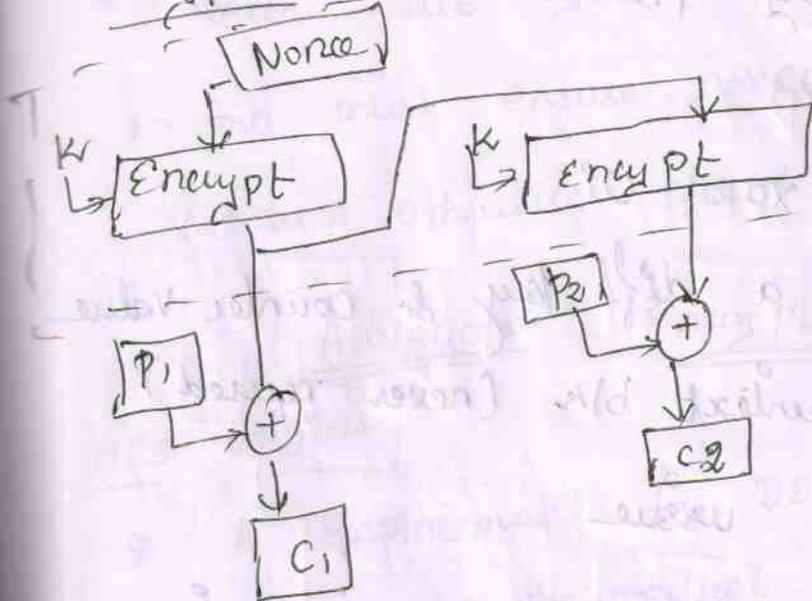
$$C_i = P_i \text{ XOR } O_i$$

* FB is independent of msg (10)

uses: Stream encryption on noisy channels.

why noisy channels?

Encryption:



Adv & limitation:

- * Needs an IV which is unique for each use.
 - ↳ If ever reuse attacker can recover o/p.
 - ↳ OTP
- * Can Pre-compute
- * Bit errors do not propagate
- * More vulnerable to msg stream modification.
 - ↳ change arbitrary bits by changing ciphertext.
- * Sender & receiver must remain in sync
- * only use with full blk FB.
 - ↳ CFB64 (or) CFB128.

Counter (CTR):-

(11)

- * a "new" mode, though proposed early on
- * like to OFB but encrypts counter value rather than any fb value.

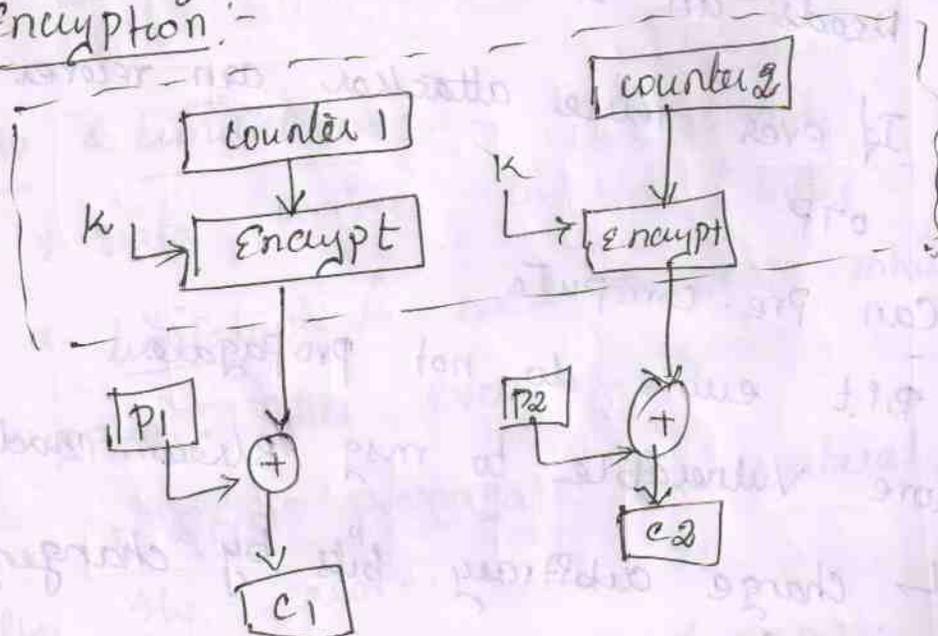
$$O_i = E_k(P)$$

$$C_i = P_i \text{ XOR } O_i$$

- * Must have a diff key & counter value for every plaintext blk (never reused)
↳ OTP issue

- * Uses: high-speed n/w encryption.

Encryption:-



Adv & limitations:-

- * Efficiency
↳ Can do Parallel encryption. In the (a) 50
- ↳ can preprocess in advance of need

- * Good for bursty high speed links. (12)
- * Random access to encrypted data blks.
- * Provable security (good as other modes).
- * Never have cycle less than 2^b .
- * But must ensure never reuse key/counter values, otherwise could break (cf OFB).

Advanced Encryption Standard (AES)

AES origins:-

- * A Replacement for DES was needed.
 - > Have theoretical attacks that can break it.
 - > Have demonstrated exhaustive key search attacks.
- * Can use Triple-DES - but slow - has small blocks.

AES cipher:-

- * AES cipher proposed by Rijmen-Daemen in Belgium.
- * The key length: 128, 192 & 256. The block length is 128-bit data.
- * An iterative rather than Feistel cipher
 - > Processes data as blk of 4 cols & 4 bytes
 - > operates on entire data blk in every

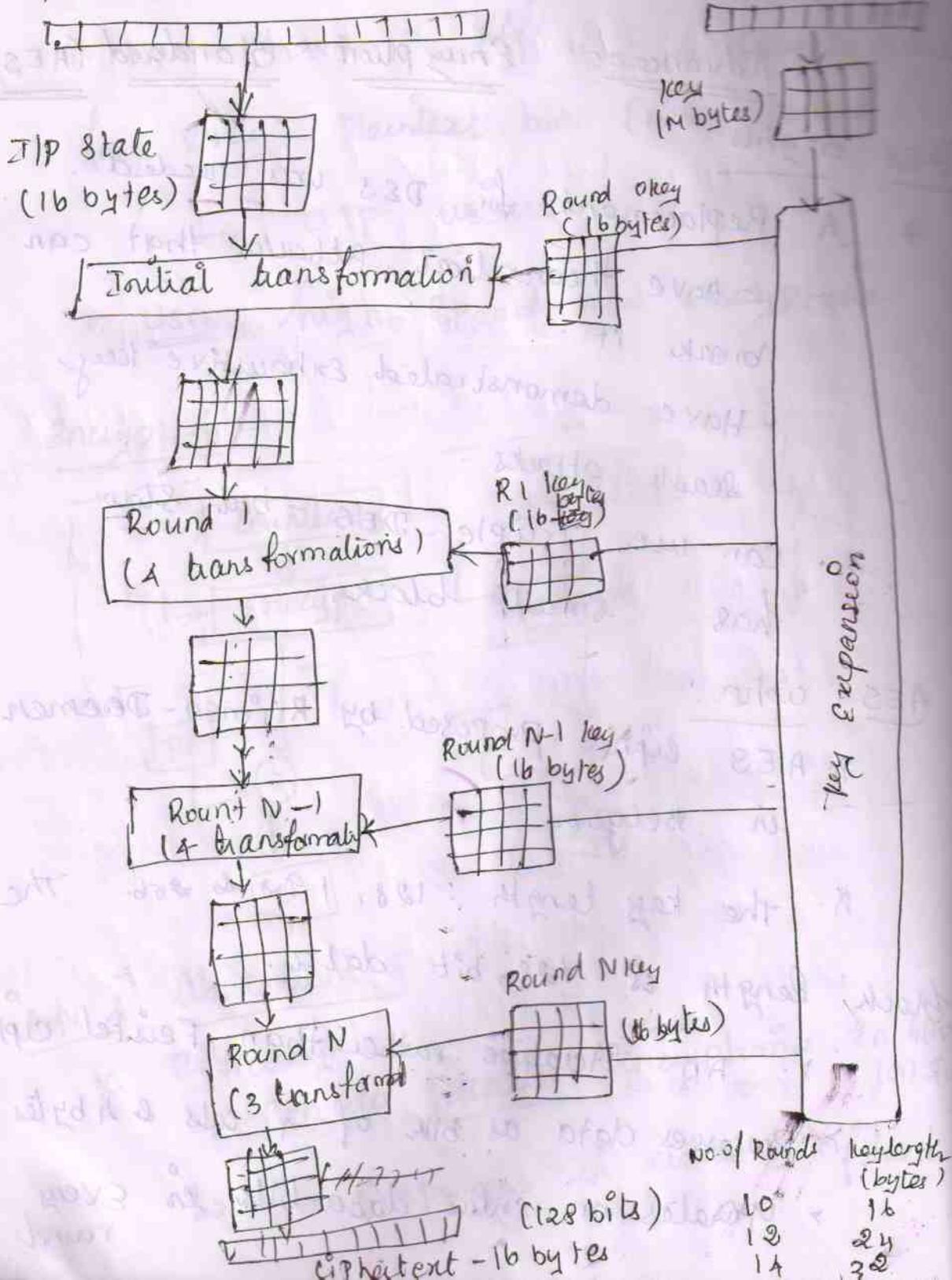
> Resistance

> Speed & Code Compactness

> design simplicity.

AES Encryption Process :-

Plaintext - 16 bytes (128 bits)



AES Structure:-

Dia: 178

(14)

- * Data blk of 4 cols of 4 bytes is state
- * key is expanded to array of words.
- * has 9/11/13 rounds in which states,
 - > byte substitution (1 S-box used on every bytes)
 - > shift rows (permute bytes b/w groups / cols)
 - > mix columns (subs using matrix multiply of groups)
 - > add round key (XOR state with key material).
 - > view as alternating XOR key & scramble data bytes.

Comments on AES:-

1. An iterative rather than Feistel cipher.
 2. key expanded into array of 32-bit words.
 - ↳ four words from round key in each round.
 3. 4 different stages r used as shown
 4. has a simple structure.
 5. Only AddRoundKey uses key.
 6. AddRoundKey a form of Vernam cipher.
 7. Each stage is easily reversible
 8. decryption uses keys in reverse order
 9. " does recover Plaintext
- Each round has only 3 stages.

AES arithmetic :-

(15)

* Uses arithmetic in the finite field

$GF(2^8)$ with irreducible polynomial

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

which is (100011011) or $\{11b\}$.

Ex.

$$\{02\} \cdot \{87\} \pmod{\{11b\}} = (100001110) \pmod{\{11b\}}$$

$$= (100001110) \text{ XOR } (100011011)$$

$$= (00010101).$$

AES key expansion

* Takes 128-bit (16-byte) key & expands into

away of $114 | 152 | 160$ 32-bit words.

* Start by using key into first 4 words.

* then loop creating words that depend on values in previous & 4 places back.

> in 3 of 4 cases just XOR these together

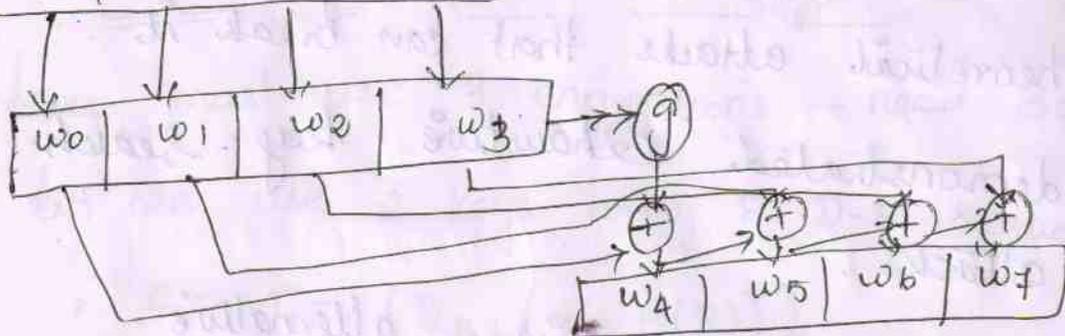
> 1st word in 4 has rotate + S-box +

XOR round constant on previous,

before XOR 4th bits back.

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

(16)



Implementation Aspects:-

* Can efficiently implement on 8-bit CPU.

> byte substitution works on bytes using a table of 256 entries.

> Shift rows is simple byte shift.

> add round key works on byte XOR's.

> Mix columns requires matrix multiply in $GF(2^8)$ which works on byte values, can be simplified to use table

lookups & byte XOR's.

* Designers believe this very efficient

implementation was a key factor in its

selection as the AES cipher.

Triple DES :-

(17)

Multiple Encryption & DES :-

- * Clear a replacement for DES was needed
 - > theoretical attacks that can break it.
 - > demonstrated exhaustive key search attacks.

* AES is a new cipher alternative.

* Prior to this alternative was to use multiple encryption with DES implementations.

* Triple - DES is the chosen form.

Why not Double - DES?

> could use 2 DES encryptions on each blk

$$C = E_{K_2}(E_{K_1}(P)).$$

> concern at time of reduction to single

stage.

> Meet - in - the middle attack.

* works whenever use a cipher twice

$$X = E_{K_1}(P) = D_{K_2}(C).$$

* attack by encrypting P with all keys & store

* then decrypt C with keys & match

* (1) can show takes $O(2^{56})$ steps. (18)

* Requires known plaintext

Triple-DES with Two-keys :-

* Hence must use 3 encryptions - need 3 distinct keys.

* but can use 2 keys with E-D-E sequence

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

> Encrypt & decrypt equivalent in security

> if $K_1 = K_2$ then can work with single DES.

* Standardized in ANSI X9.17 & ISO 9796.

* no current known practical attacks.

↳ Several proposed impractical attacks might become basis of future attacks.

Triple-DES with Three-keys :-

* Although no practical attacks no two-key

Triple-DES have some indications -

* Can use Triple-DES with three-keys to avoid even those,

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$$

* has been adopted by some Internet appls,

Ex: PGP, S/MIME

(18) Blowfish (19)

* A symmetric blk cipher Blowfish.

* Designed by Bruce Schneier in 1993/94.

* Characteristics:-

- > fast imple- on 32-bit CPUs.
- > compact in use of memory
- > simple structure for analysis/imple
- > Variable security by varying key size.

* Uses a 32 to 448 bit key.

* Key Schedule consists of:

- > Initialize P-array & then 4 S-boxes using P_i .
- > XOR P-array with key bits (reverse as needed)
- > loop repeatedly encrypting data using current P & S and replace successive pairs of P then S values.
- > Requires 621 encryptions, hence slow in rekeying.
- > uses primitives:
 - > addition & XOR
- > data is divided into two 32-bit halves L0 & R0.

for $i=1$ to 16 do

$$R_i = L_{i-1} \text{ XOR } P_i;$$

$$L_i = F[R_i] \text{ XOR } R_{i-1};$$

Ex: $L_{17} = R_{16} \text{ XOR } P_{17};$

$$R_{17} = L_{16} \text{ XOR } P_{17};$$

where,

$$F[a, b, c, d] = ((s_1, a + s_{2,16}) \text{ XOR } s_3, c) + s_4, a$$

> key dependent 8 boxes and subkeys,

makes cryptanalysis ~~is~~ very difficult.

> Changing both halves in each round increases security.

> provided key is large enough, brute-force

key search is not practical, especially given the high-key schedule cost.

RC5 algorithm

* A proprietary cipher owned by RSADSI.

* designed by Ronald Rivest (of RSA).

* Used in various RSADSI products.

* Can vary key size / data size / no rounds

* very clean & simple design

* easy imple on various CPUs.

* RC5 is a family of ciphers RC5-w/r/b.

w = word size in bits (16 | 32 | 64) (21)

nb data = $2w$.

r = no. of rounds (0..255)

b = no. of bytes in key (0..255).

* Nominal version is RC5-32/12/16.

(ie) 32-bit words so encrypts

64-bit data blocks. using 12 rounds

with 16 bytes (128 bits) secret key.

* RFC 2040 defines 4 modes used by RC5.

> RC5 Block cipher, is ECB mode

> RC5-CBC, is CBC mode

> RC5-CBC-PAD, is CBC with padding

by bytes with value being the no. of padding byte

> RC5-CTS, a variant of CBC which is

the same size as the original msg,

uses ciphertext stealing to keep size same as

RC5 key expansion and encryption:-

> RC5 uses $2r+2$ subkey words (w-bits)

> subkeys are stored in array

$S[i], i = 0 \dots t-1$.

* key schedule consists of, (22)

> initializing S to a fixed pseudorandom value, based on constants e and Φ .

> the byte key is copied (little-endian)

into a c-word array L .

> a mixing opn then combines L and S to

form the final S array.

* Split IP into two halves A & B :

$$L_0 = A + S[0];$$

$$R_0 = B + S[1];$$

for $i=1$ to r do

$$L_i = ((L_{i-1} \text{ XOR } R_{i-1} \lll R_{i-1}) + S[2 \times i]);$$

$$R_i = ((R_{i-1} \text{ XOR } L_i) \lll L_i) + S[2 \times i + 1];$$

* Each round is like 2 DES rounds

* Note rotation is main source of non-linearity

* Need reasonable no. of rounds (sa: 12-16)

Public Key Cryptography

Why?

* Developed to address two key issues

① Key distribution

* how to have secure comms in general without having to trust a KDC with yr key.

② Digital signatures

* how to verify a msg comes intact from the claimed sender.

* Public invention due to Whitfield

Diffie & Martin Hellman at Stanford in

in 1976.

Public-key cryptography

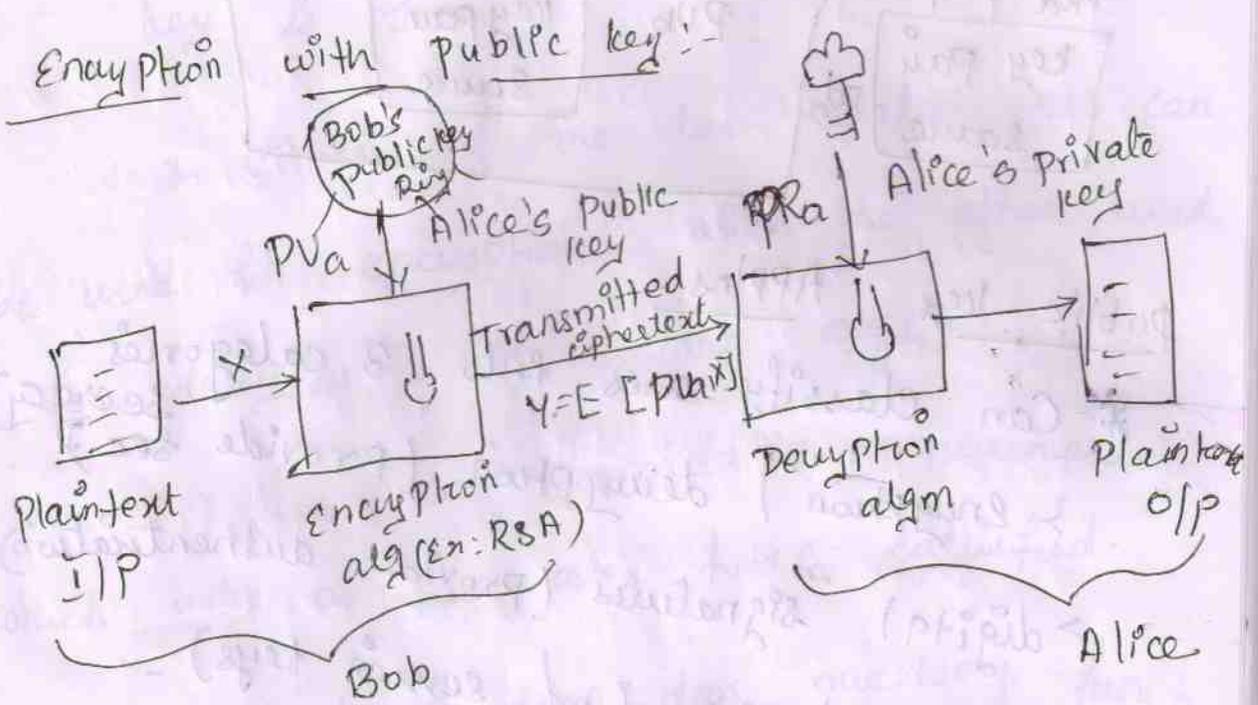
* Public-key / two-key / asymmetric crypt.

involves the use of two keys:

> Public key, which may be known by anybody, and can be used to encrypt msgs, & verify signatures.

> Private key, known only to the recipient, used to decrypt msgs, & sign (create) signatures.

X. Infeasible to determine private key from public is asymmetric coz, those who encrypt msgs (or) verify signature cannot decrypt msgs (or) create signatures.



Conventional Encryption

- Needed to work:
1. The same alg with the same key is used for encryption & decryption
 2. The S & R must share the alg & the key.

Security:

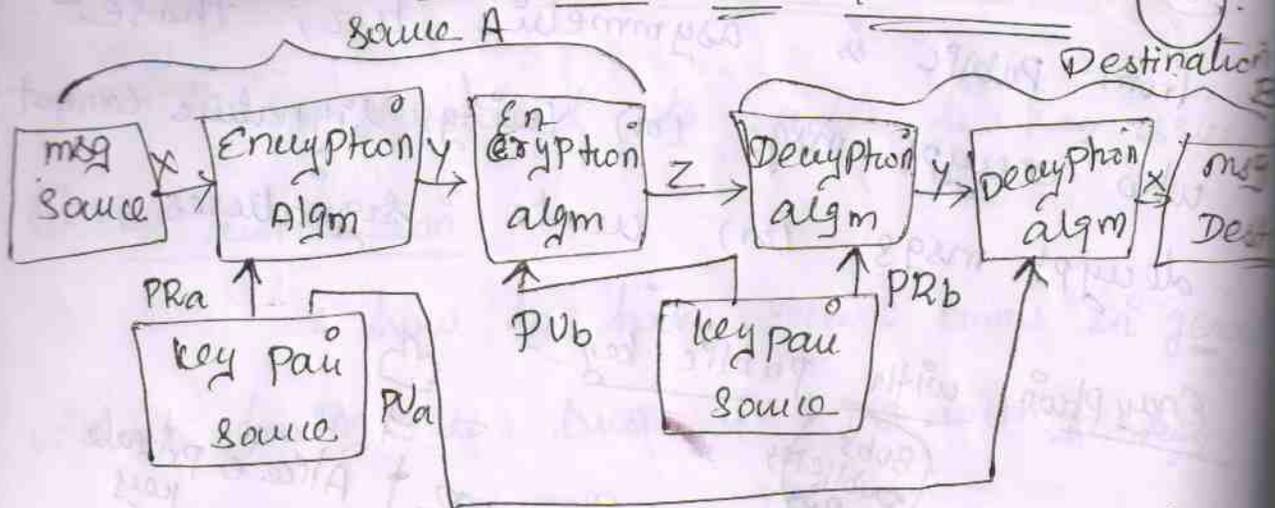
1. The key must be kept secret
2. It must be impossible (or) impractical to decipher any msg
3. Knowledge of the alg + samples of cipher must be

Public-key Encryption

- Needed to work:
1. one alg is used for encryption & decryption with a pair of keys, one for encryption & one for decryption
 2. The S & R must each have one of the matched pair of keys. (not the same one).

1. one of the 2 keys must be kept secret
 2. → "
- Knowledge of the alg + one of key + some of

Principles of Public Key Cryptosystems



Public-key APPLNS:-

* Can classify uses into 3 categories:

> encryption (provide ^{secret} security)

> digital signatures (provide authentication)

> key exchange (of session keys)

* Some algms suitable for all users, others are specific to one.

Algm	encryption / decryption	Digital signature	key exchange
RSA	Y	Y	Y
Elliptic-curve	Y	Y	Y
Diffie-Hellman	N	N	Y
DSS	N	Y	N

Public-key Requirements:-

* It rely on 2 keys,

to find decryption key knowing only algm
& encryption key. (26)

* It is computationally easy to en/decrypt
msg when the relevant (en/decrypt)
key is known.

* either of the two related keys can
be used for encryption, with the other used
for decryption (for some algs).

* these are formidable requirements
which only a few algs have satisfied.

* Need a trap-door one-way fun.

* One-way fun has,

$$Y = f(X) \text{ easy}$$

$$X = f^{-1}(Y) \text{ infeasible!}$$

* A trap-door one-way fun has,

$$Y = f_k(X) \text{ easy, if } k \text{ \& } X \text{ are known}$$

$$X = f_k^{-1}(Y) \text{ easy, if } k \text{ \& } Y \text{ "}$$

$$X = f_k^{-1}(Y) \text{ infeasible, if } Y \text{ known but } k \text{ not known.}$$

* A Practical public-key scheme depends
on a suitable trap-door one-way

Security of public-key schemes: - (27)

* like, private key schemes brute-force exhaustive search attack is always theoretically possible.

* but keys used are too large (> 512 bits)

* security relies on a large enough difference in difficulty b/w easy (en/decrypt) & hard (cryptanalyse) problems.

* More generally the hard problem is known, but is made hard enough to be impractical to break.

* requires the use of very large numbers

* hence, is slow compared to private

key schemes.

The RSA algorithm:

* By Rivest, Shamir & Adleman of MIT in 1977.

* Best-known & widely used public-key scheme

* Based on exponentiation in a finite

(Galois) field over integers modulo a prime.

→ Exponentiation takes $O((\log n)^3)$ ops (easy)

* Uses large integers (ex: 1024 bits).

* Security due to cost of factoring large no.

→ Factorization takes $O(e^{\log n \log \log n})$ ops (hard).

RSA Encryption / decryption :-

* To encrypt a msg M the sender:

> obtain public-key of recipient

$$P_U = \{e, n\}$$

> computes: $C = M^e \pmod n$, where $0 \leq M < n$

* To decrypt the ciphertext, C the owner:

> uses their private key $P_R = \{d, n\}$.

> computes: $M = C^d \pmod n$.

* Note that the msg M must be smaller than the modulus n (blk if needed).

RSA Key Setup :-

* Each user generates a public/private key pair by selecting two large

primes at random: p, q.

* Computing their system modulus.

(29)

$$n = p \cdot q$$

$$\phi(n) = (p-1)(q-1)$$

* Selecting at random the encryption key e

where, $1 < e < \phi(n)$, $\gcd(e, \phi(n)) = 1$

* Solve following eqn to find decryption key d .

$$e \cdot d = 1 \pmod{\phi(n)} \quad \& \quad 0 \leq d \leq n$$

* Publish their public encryption key:

$$PU = \{e, n\}$$

* keep secret private decryption key:

$$PR = \{d, n\}$$

why RSA works :-

* Base of Euler's theorem:

$$a^{\phi(n)} \pmod{n} = 1$$

where, $\gcd(a, n) = 1$

* In RSA have:

$$n = p \cdot q$$

$$\phi(n) = (p-1)(q-1)$$

> carefully choose e & d to be inverse mod $\phi(n)$.

$$1 + k \cdot \phi(n) \quad \text{for some } k$$

∴ hence,

$$C^d = M^{e \cdot d} = M^{1+k \cdot \phi(n)} = M' \cdot (M^{\phi(n)})^k$$

$$= M' \cdot (1)^k = M' = \underline{M \pmod n}$$

RSA Example - Key Setup:

1. Select primes: $p = 17$ & $q = 11$.
2. Calculate: $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p-1)(q-1) \Rightarrow 16 \times 10 \Rightarrow 160$.
4. Select e : $\gcd(e, 160) = 1$; choose $e = 7$.
5. Determine: d : $de = 1 \pmod{160}$ and $d < 160$
value is $d = 23$ since $23 \times 7 = 161 \Rightarrow 10 \times 160$.
6. Publish public key $PU = \{7, 187\}$.
7. Keep secret private key $PR = \{23, 187\}$.

RSA Example - Encryption / Decryption:

> Sample msg $M = 88$ (nb. $88 < 187$).

Encryption: $C = 88^7 \pmod{187}$

$$\boxed{C = 11}$$

Decryption: $M = 11^{23} \pmod{187}$

$$\boxed{M = 88}$$

RSA Security: - * Possible approaches to attacking

- RSA are: (31)
- * Brute - force key search:
 - * Infeasible give size of nos.
(i) Trying all possible Private keys
 - * Mathematical attacks
 - * The approaches to factor the product of two prime numbers.
 - * Timing Attack:
 - * Depends on the running time of the decryption algm.

Defense to Brute force Attack: -

- * Use large key space. (i) larger no. of bits in e in d the better secured but Pblms are,
 - > Increased computing power
 - > Factoring Pblm.

Timing attacks: -

- * Use constant exponentiation time
- * add random delays
- * blind values used in calculations

Key Management:-

Key mgmt & Distribution:-

* Symmetric schemes require both parties to share a common secret key.

Key hierarchy:-

> Session key:

↳ temporary key

↳ used for encryption of data b/w users.

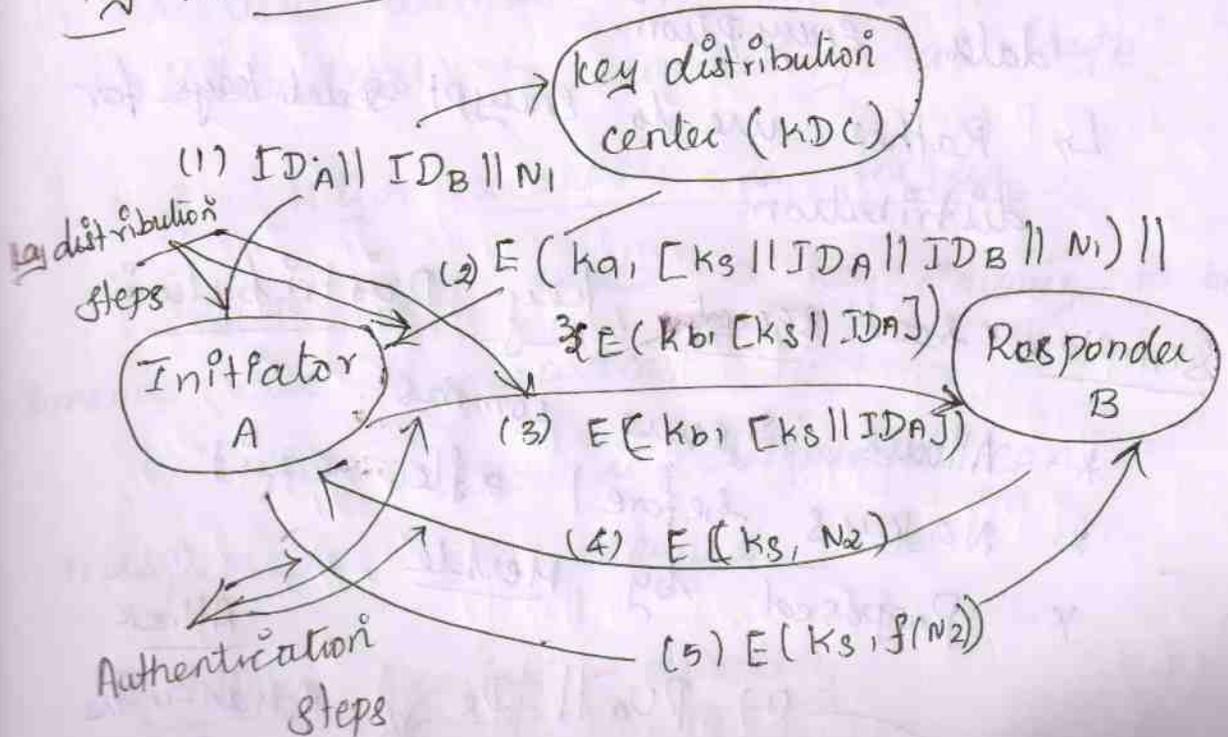
↳ for one logical session the discarded.

> Master key:

↳ used to encrypt session keys

↳ shared by user & key distribution center.

Key distribution scenario:-



key distribution issues:

(33)

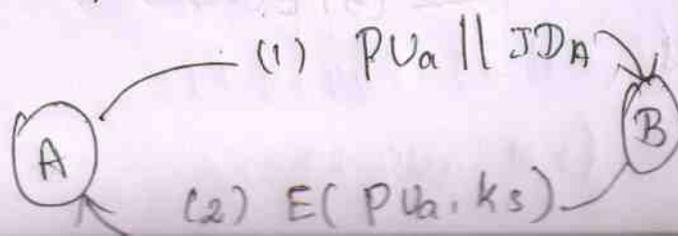
- * KDC's required large n/ws, but must trust each other.
- * Session key lifetimes should be limited for greater security.
- * Use of automatic key distribution on behalf of users, but must trust sys.
- * Use of decentralized key distribution.
- * Controlling key usage.

Symmetric key distribution using public keys

- * Public key cryptosystems are inefficient
↳ so almost never use for direct data encryption.
↳ Rather use to encrypt secret keys for distribution.

Simple secure key distribution:

- * Allow secure comms.
- * No keys before / after exist.
- * Proposed by Merkle.

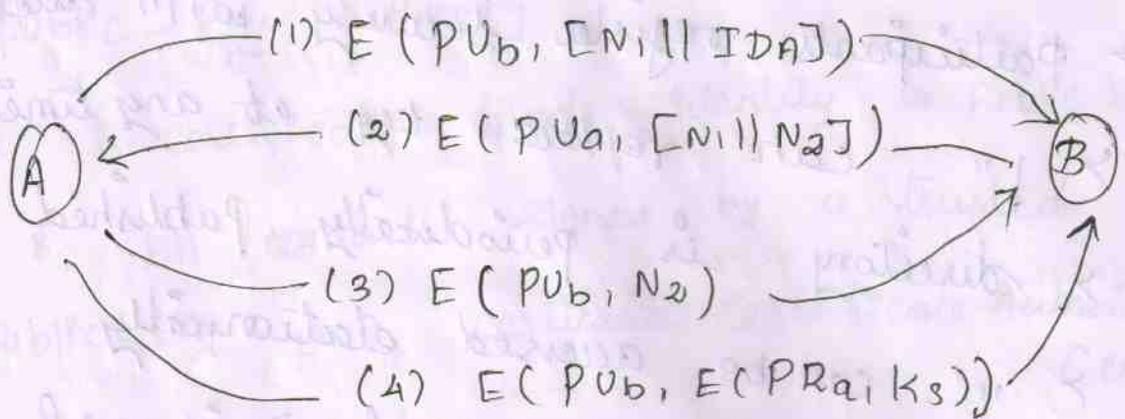


Attack
* Man-in-the-middle attack

Secret key distribution with confidentiality

& Authentication:-

(34)



Distribution of public keys:-

* Public announcement

* Publicly available directory

* Public-key authority

* Public-key certificates

Public Announcement:-

* Users distribute public keys to recipients
(or) broadcast to community at large.

* Major weakness is forgery.

> anyone can create a key claiming to be someone else & broadcast it.

> Untill forgery is discovered can masquerade as claimed user.

Publicly Available directory:-

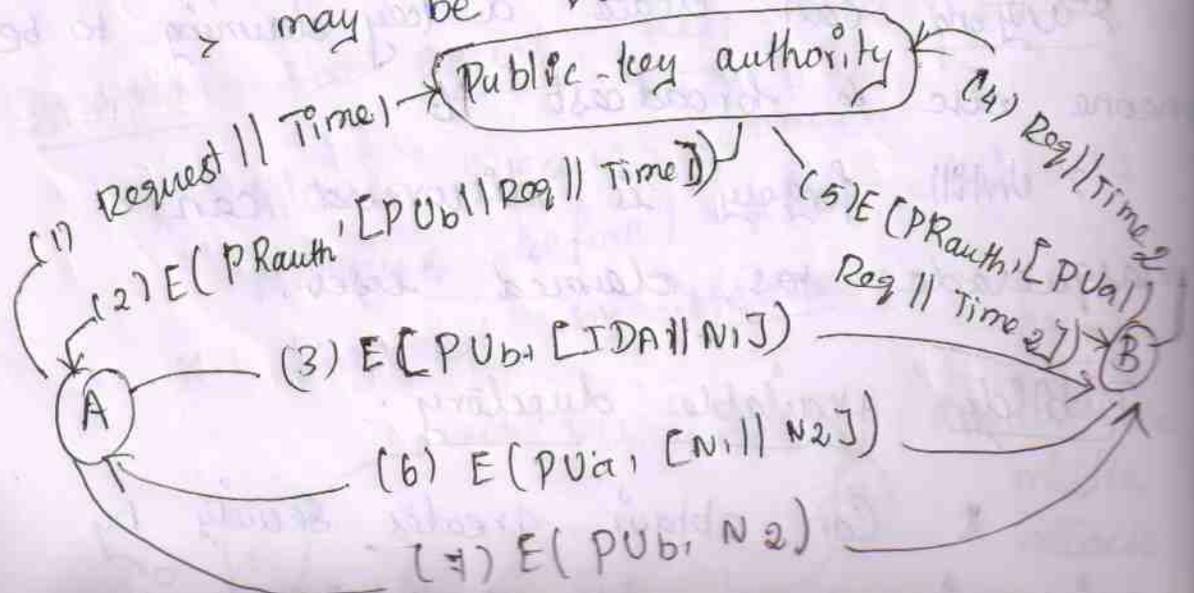
* Can obtain greater security by

* Properties :-

- > contains {name, public key} entries
- > Participants register securely with directory
- > " can replace key at any time.
- > directory is periodically published
- > " can be accessed electronically
- * Still vulnerable to tampering (or) forgery

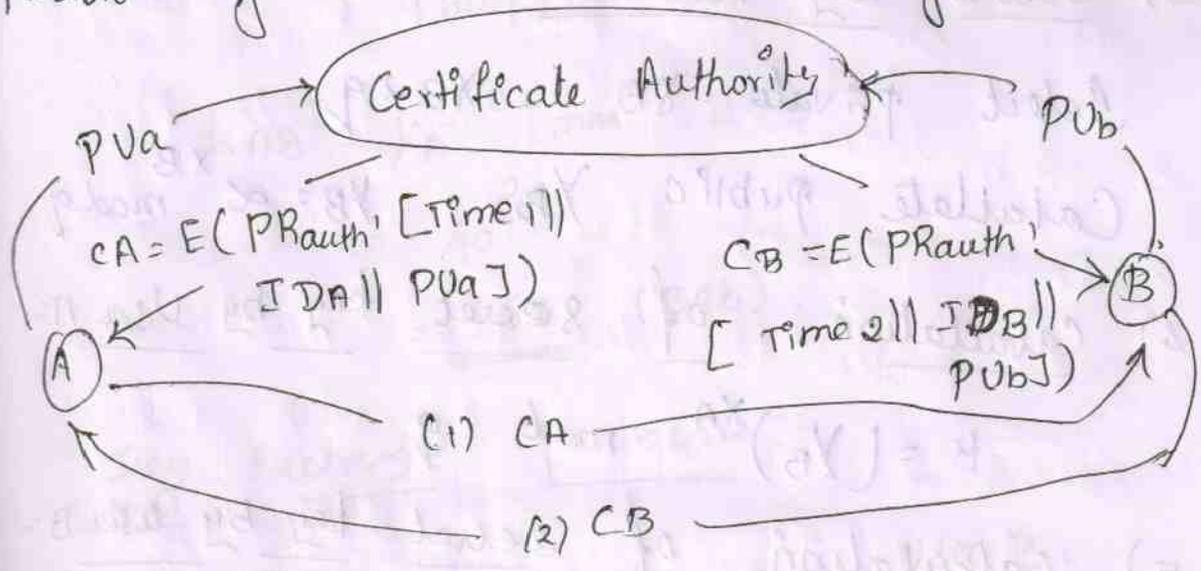
Public-key Authority :-

- * Improve security by tightening control over distribution of keys from directory.
- * Requires users to know public key for the directory. then users interact with directory to obtain any desired public key
- > does not require real-time access to directory when keys are needed.
- > may be vulnerable to tampering.



Public-key certificates:-

- * Allow key exchange without real-time access to public-key authority.
- * A certificate binds identity to public key.
- * All contents signed by a trusted public-key (or) certificate Certificate Authority (CA).
- * Can be verified by anyone who knows the public-key authority's public key.



Diffie Hellman key exchange:-

- * public-key scheme proposed by Diffie & Hellman in 1976.
- * This method is a practical method for public exchange of a secret key.
- * Used in a no. of commercial products.

Algorithm:-

(37)

1) Global public elements:-

q

Prime no

α

$\alpha < q$ and α a primitive root of q

2) User A key Generation:-

Select private X_A , $X_A < q$

Calculate public Y_A $Y_A = \alpha^{X_A} \text{ mod } q$

3) User B key Generation:-

Select private X_B , $X_B < q$

Calculate public Y_B $Y_B = \alpha^{X_B} \text{ mod } q$

4) Calculation of secret key by User A:-

$$K = (Y_B)^{X_A} \text{ mod } q$$

5) Calculation of secret key by User B:-

$$K = (Y_A)^{X_B} \text{ mod } q$$

Example:-

x. Users Alice & Bob who wish to swap keys.

x. agree on prime $q = 353$ and $a = 3$

x. select random secret keys:

chooses,

$$X_A = 97,$$

$$X_B = 233$$

* Compute respective public keys:

38

$$Y_A = 3^{97} \pmod{353}$$
$$= 40 \text{ (Alice)}$$

$$Y_B = 3^{233} \pmod{353}$$
$$= 248 \text{ (Bob)}$$

* Compute shared session key as,

$$K_{AB} = Y_B^{X_A} \pmod{353}$$

$$= 248^{97}$$

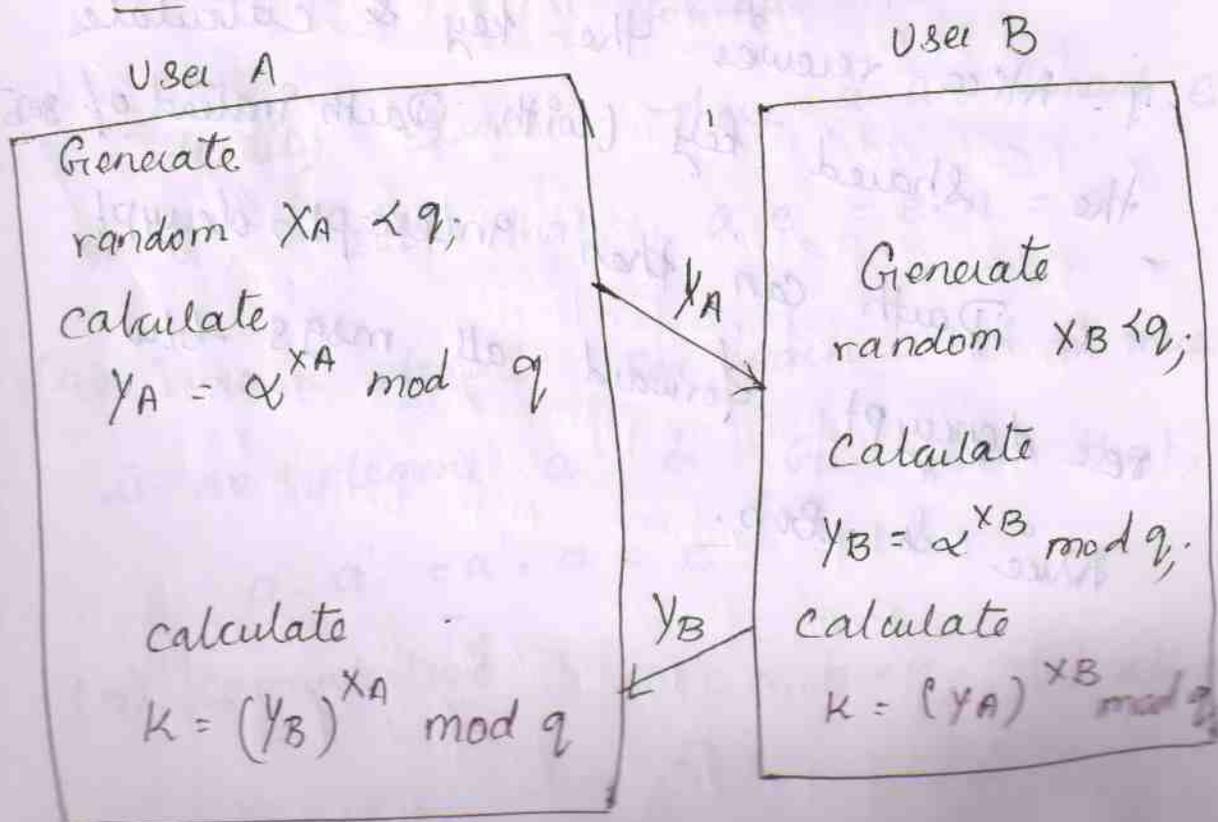
$$= 160 \text{ (Alice)}$$

$$K_{AB} = Y_A^{X_B} \pmod{353}$$

$$= 40^{233}$$

$$= \underline{160} \text{ (Bob)}$$

Key Exchange Protocols :-



Man-in-the-Middle Attack:- (39)

1. Dearth prepares by creating two private/public keys
 2. Alice transmits her public key to Bob.
 3. Dearth intercepts this & transmits his 1st public key to Bob. Dearth also calculates a shared key with Alice.
 4. Bob receives the public key & calculates the shared key [with Dearth instead of Alice]
 5. Bob transmits his public key to Alice
 6. Dearth intercepts this & transmits his 2nd public key to Alice. Dearth calculates a shared key with Bob.
 7. Alice receives the key & calculates the shared key (with Dearth instead of Bob)
- * Dearth can then intercept, decrypt, re-encrypt, forward all msgs b/w Alice & Bob.

Elliptic Curve Arithmetic

(40)

* ECC offer equal sec for a far smaller key size.

* confidence level in ECC is not yet as high as that in RSA.

Abelian Group:-

* A set of elements with a binary opn, denoted by \cdot that associates to each ordered pair (a, b) of elements in G an element $(a \cdot b)$ in G ,

(A1) Closure : If a and b belong to G , then $a \cdot b$ is also in G .

(A2) Associative $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

for all a, b, c in G .

(3) Identity element : There is an element e in G , such that, $a \cdot e = e \cdot a = a$.

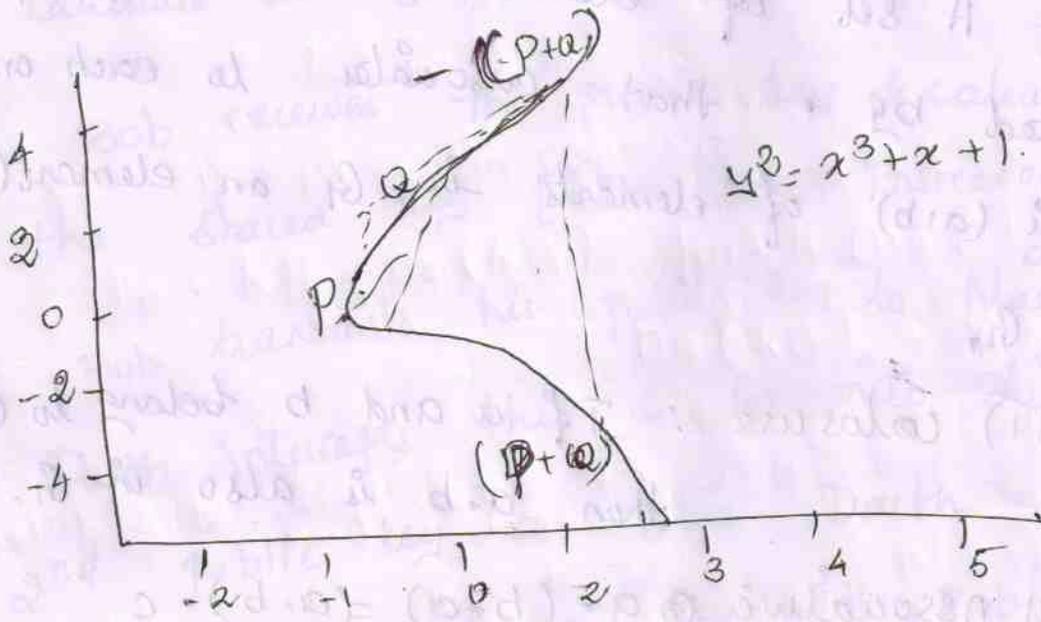
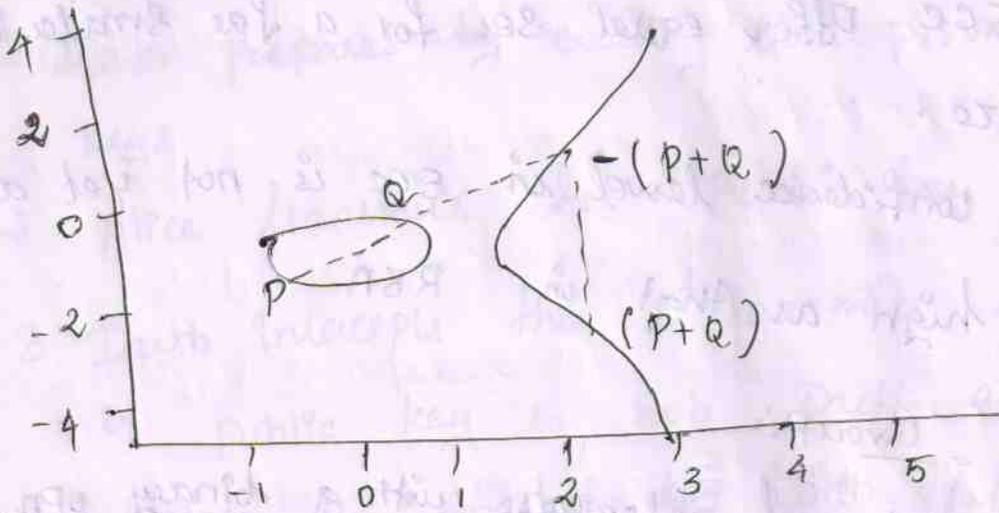
(4) Inverse element : For each a in G there is an element a' in G such that, $a \cdot a' = a' \cdot a = e$.

(5) Commutative : $a \cdot b = b \cdot a$ for all a, b in G .

Example:

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

(41)



ELLIPTIC CURVE CRYPTOGRAPHY:

- * ECC addition is analog of modulo multiplication
- * ECC repeated addition is analog of modulo exponentiation.
- * need "hard" problem equiv to discrete log

> $Q = kP$, where Q, P belong to a Prime curve.

> is "easy" to compute Q given k, P

> but "hard" to find k given Q, P

elliptic curve

Ex:

$E_{23} (9, 17)$.

(42)

ECC Diffie-Hellman key exchange:

① Global Public Elements:

$E_q(a|b)$ - elliptic curve with parameters $a|b \in \mathbb{F}_q$, where q is a prime (or) an integer of the form 2^m .

G - point on elliptic curve whose order is large value n .

User A key Generation:

Select private n_A $n_A < n$

Calculate public P_A $P_A = n_A \times G$

User B key Generation:

Select private n_B $n_B < n$

Calculate public P_B $P_B = n_B \times G$

Calculation of secret key by User A:

$$K = n_A \times P_B$$

Calculation of secret key by User B:

$$K = n_B \times P_A$$

ECC Encryption / Decryption:

x. Several alternatives, will consider

* Must 1st encode any msg M as a point on the elliptic curve P_m . (43)

* select suitable curve & point G as in D+1

* Each user chooses private key $n_A < n$.

* and computes public key $P_A = n_A G$

* to encrypt $P_m : C_m = \{kG, P_m + kP_b\}$,
 k random.

* Decrypt C_m compute :

$$P_m + kP_b - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

Comparable key sizes for equivalent security.

Symmetric scheme (key size in bits)	EC-based scheme size of n is bits	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360