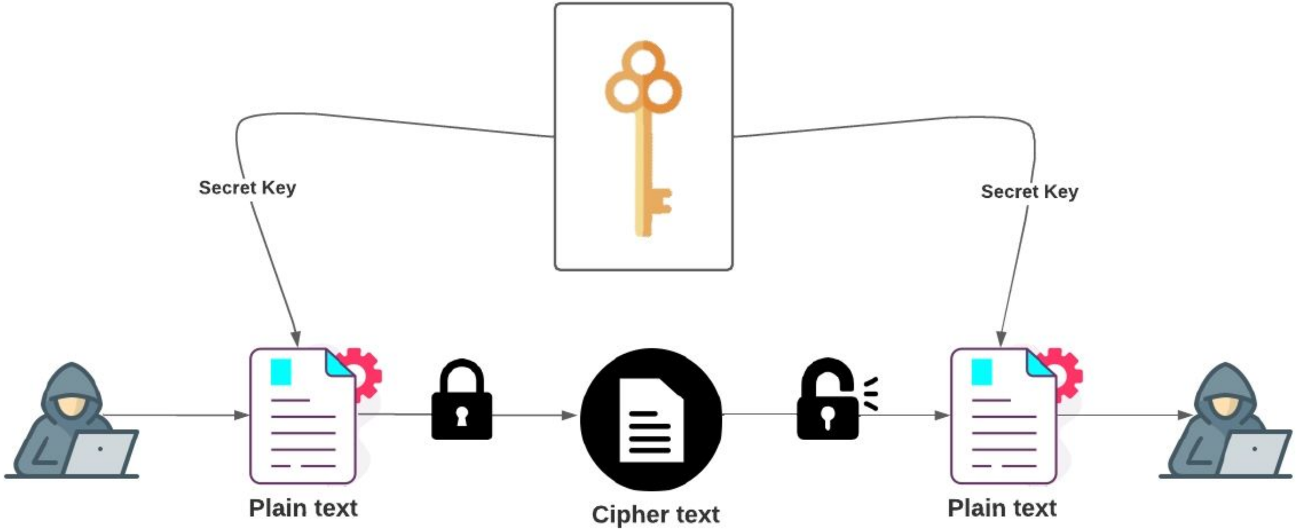
Unit 3 Cryptography



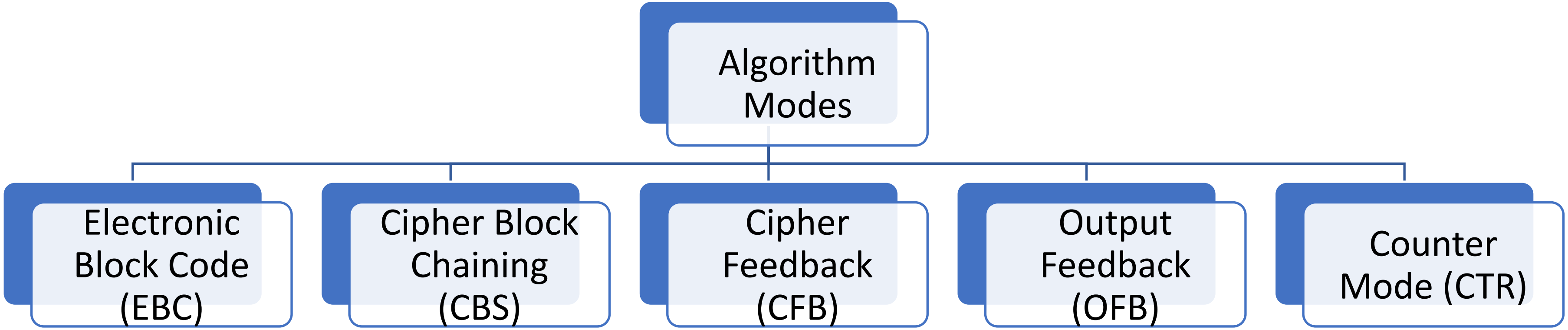
Prepared by : Prof. Ruchika Rami

Subject : 222301503 Fundamentals of Network Security

Cryptography & Network Security

- Cipher Modes
  - Electronic Code Book Mode
  - Cipher Block Chaining Mode
  - Cipher Feedback Mode
  - Stream Cipher Mode
  - Counter Mode
- Public Key Algorithm
  - RSA
- Digital Signatures
  - Symmetric-key
  - Public key
  - Message Digest (SHA, MD5)

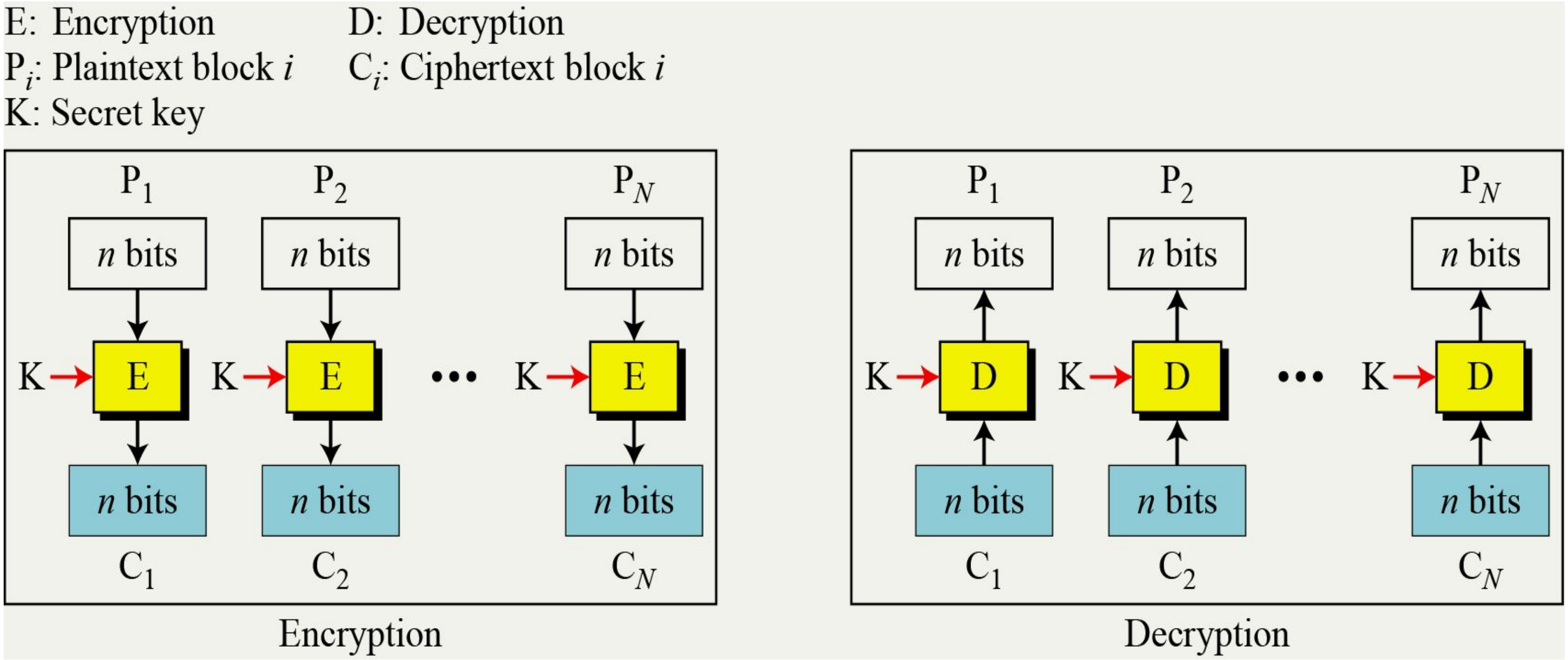
Cipher Modes



Electronic Block Code (EBC)

- Message is broken into **independent blocks** which are encrypted
- Each block is a **value** which is substituted, **like a codebook**, hence name
- each block is **encoded independently** of the other blocks
- **Uses:** secure transmission of single values

Electronic Block Code (EBC)



Electronic Block Code (EBC)

- **Benefits**

✓ Parallel encryption of blocks of bits is possible, thus it is a faster way of encryption.

✓ Simple way of block cipher.

- **Problems**

✓ message repetitions may show in ciphertext

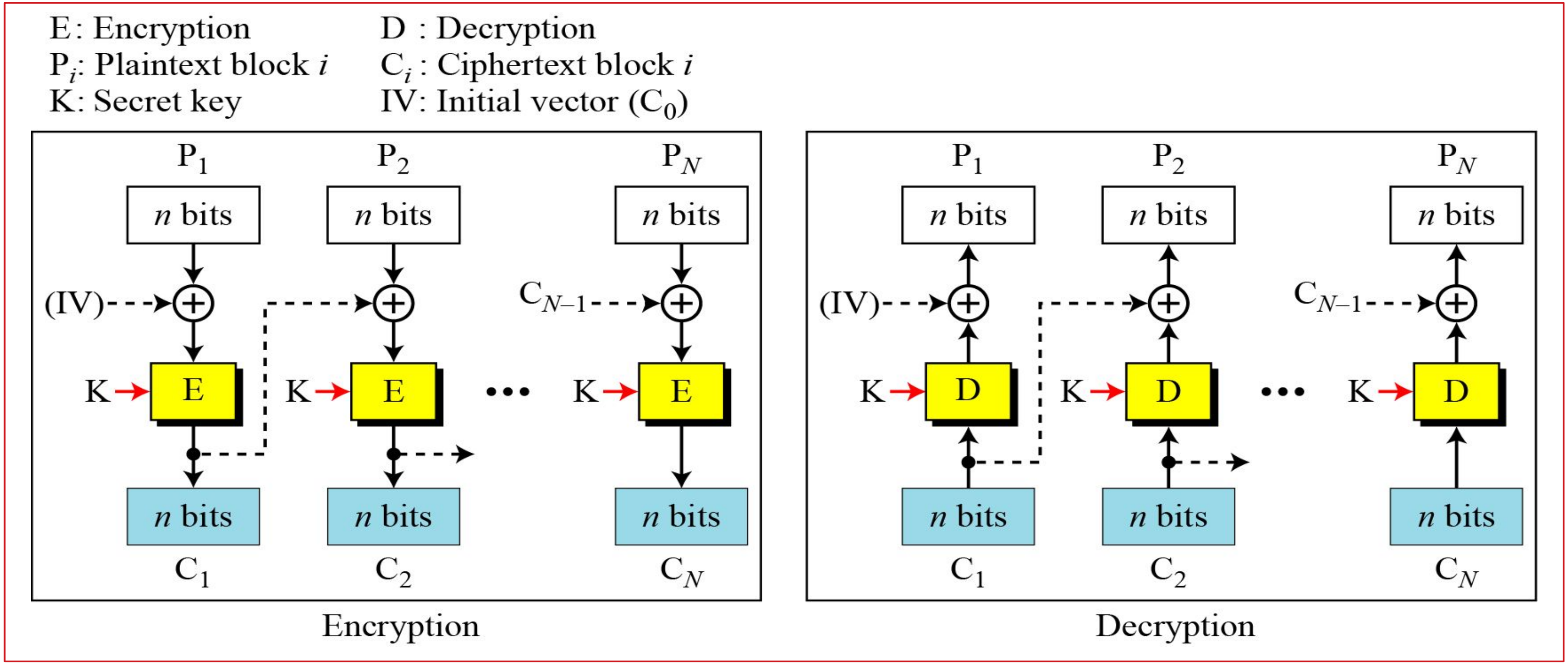
✓ weakness is due to the encrypted message blocks being independent

✓ main use is sending a few blocks of data

Cipher Block Chaining (CBC)

- Message is broken into **blocks**
- Linked together in encryption operation
- Each **previous cipher blocks** is **chained** with **current plaintext block**, hence name
- Use **Initial Vector (IV)** to start process
- **Uses:** bulk data encryption, authentication

Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC)

- Problems
  - ✔ A ciphertext block depends on all blocks before it
  - ✔ Any change to a block affects all following ciphertext blocks
  - ✔ Need Initialization Vector (IV)
  - ✗ which must be known to sender and receiver
  - ✗ if sent in clear, attacker can change bits of first block, and change IV to compensate

Cipher Feedback (CFB)

- At times we need to use block cipher that works as streams cipher Eg. Telnet
- Generic Steps

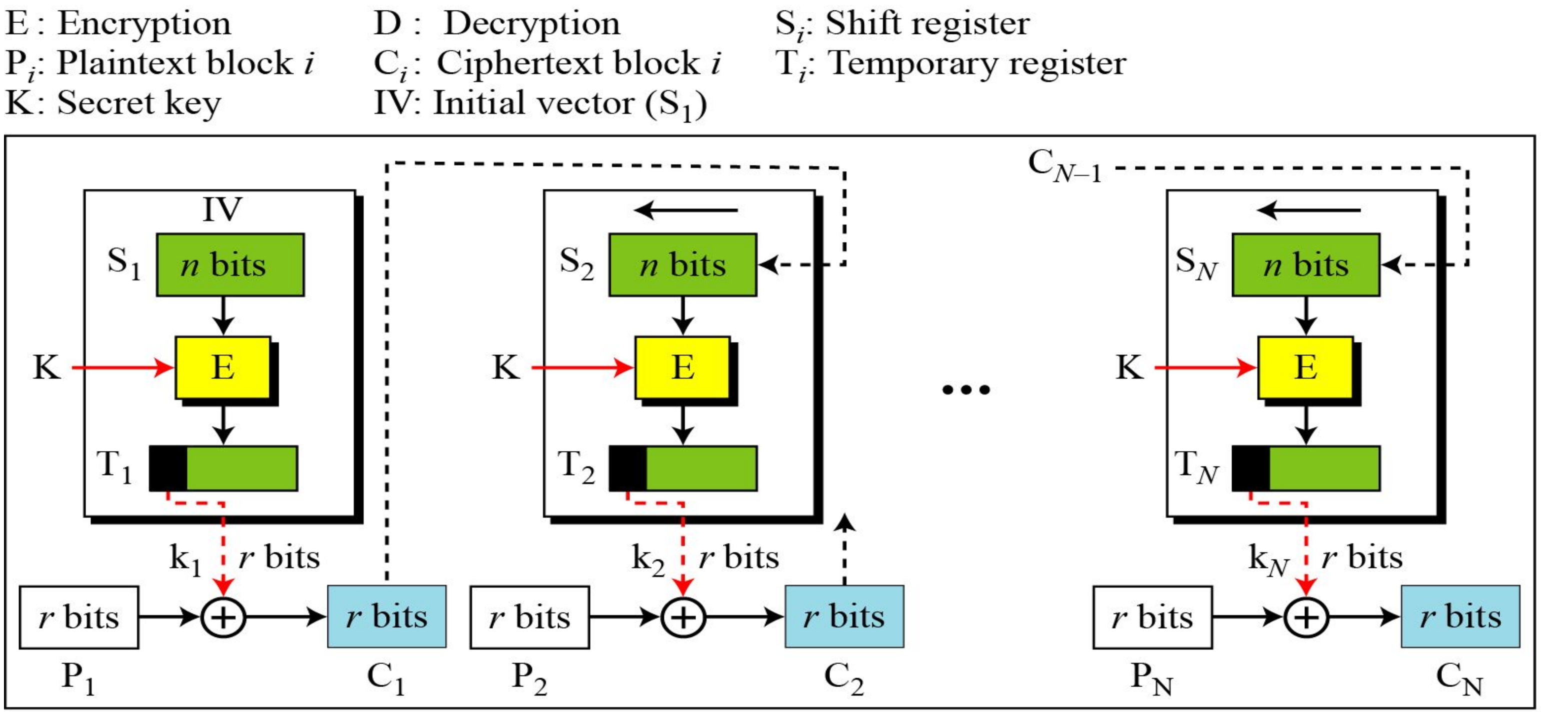
✔ Take 64 bit **Initial Vector (IV)** store it in **shift register**, Encrypt it to get EIV

✔ Perform XOR of Leftmost **r** bits of Plain text (PT) and Leftmost **r** bits of **EIV** call it **Ci**

✔ Shift left **IV by r bits** and insert **Ci to its right**

✔ Repeat step 1 to 3

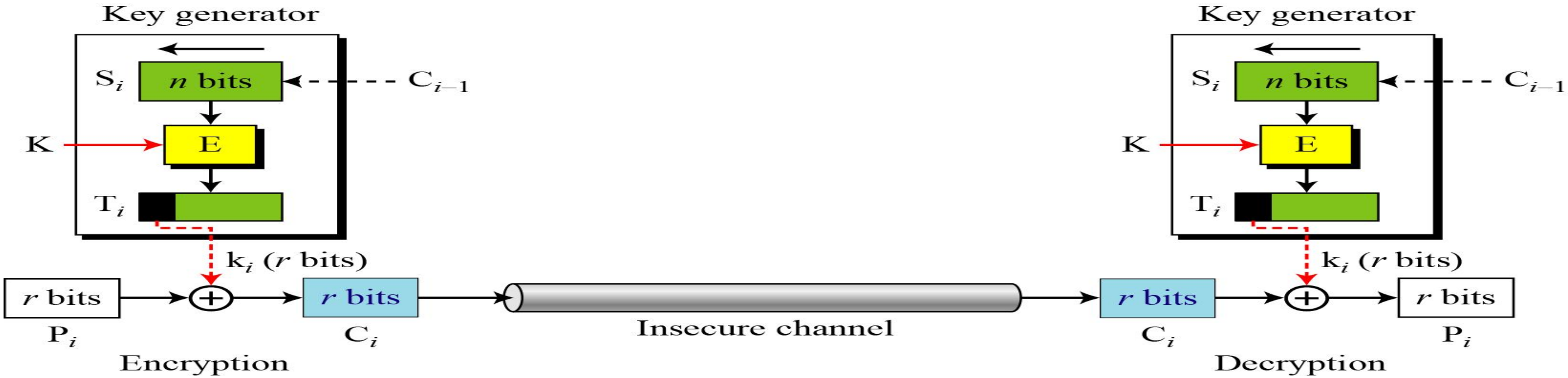
Cipher Feedback (CFB)



Encryption

Cipher Feedback (CFB)

- Message is treated as a stream of bits
- Added to the output of the block cipher
- Result is feedback for next stage (hence name)
- Standard allows any number of bit (1,8, 64 or 128 etc) to be feedback
- denoted CFB-1, CFB-8, CFB-64, CFB-128 etc



Cipher Feedback (CFB)

- Problems
- ✔ Appropriate when data arrives in bits/bytes
- ✔ Most common stream mode
- ✔ Errors propagate for several blocks after the error

Output Feedback (OFB)

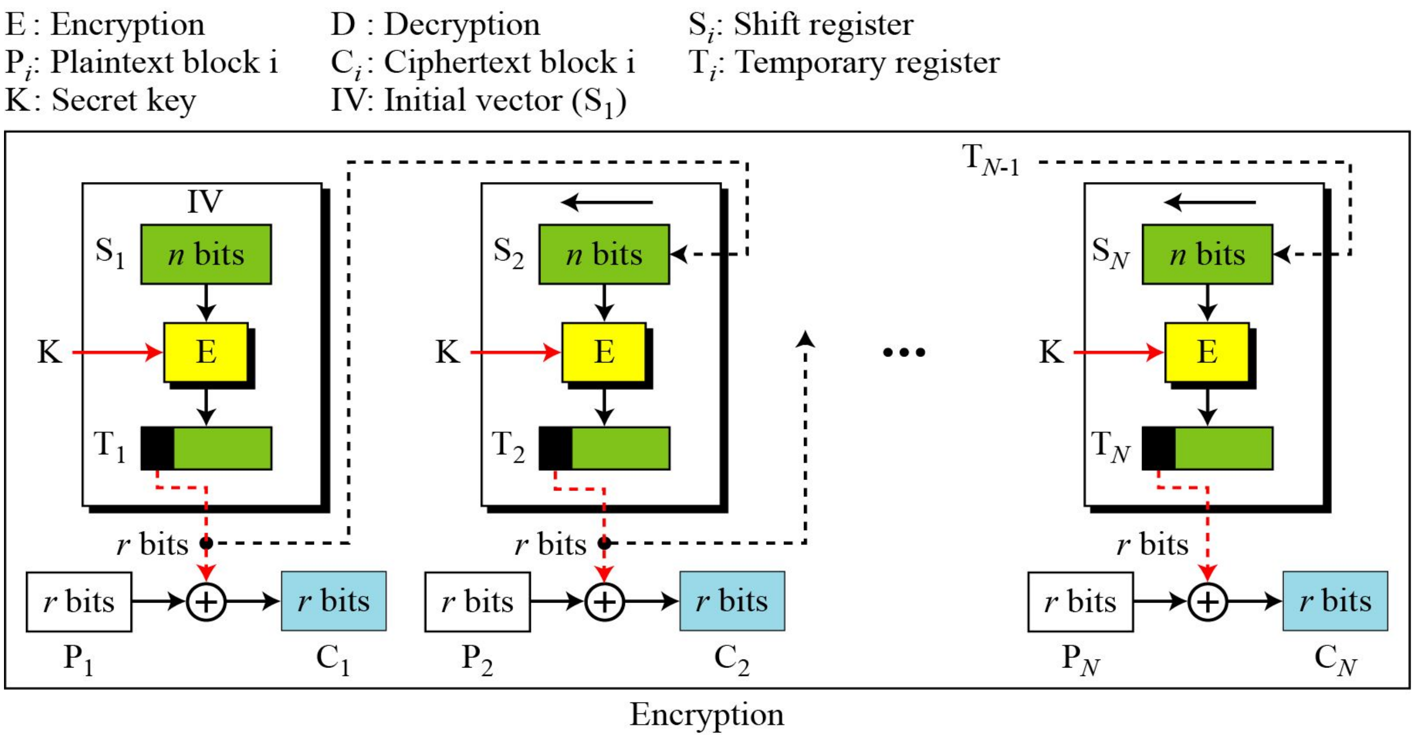
- At times we need to use **block cipher that works as streams cipher**
- In this mode **each bit** in the ciphertext is independent of the previous bit or bits.
- This avoids error propagation.
- **Generic Steps**

✓ Take 64 bit **IV** store it in **shift register**, Encrypt it to get **EIV** call it **Ti**

✓ Perform XOR of **Leftmost r bits of PT** and **Leftmost r bits of EIV** call it Ci

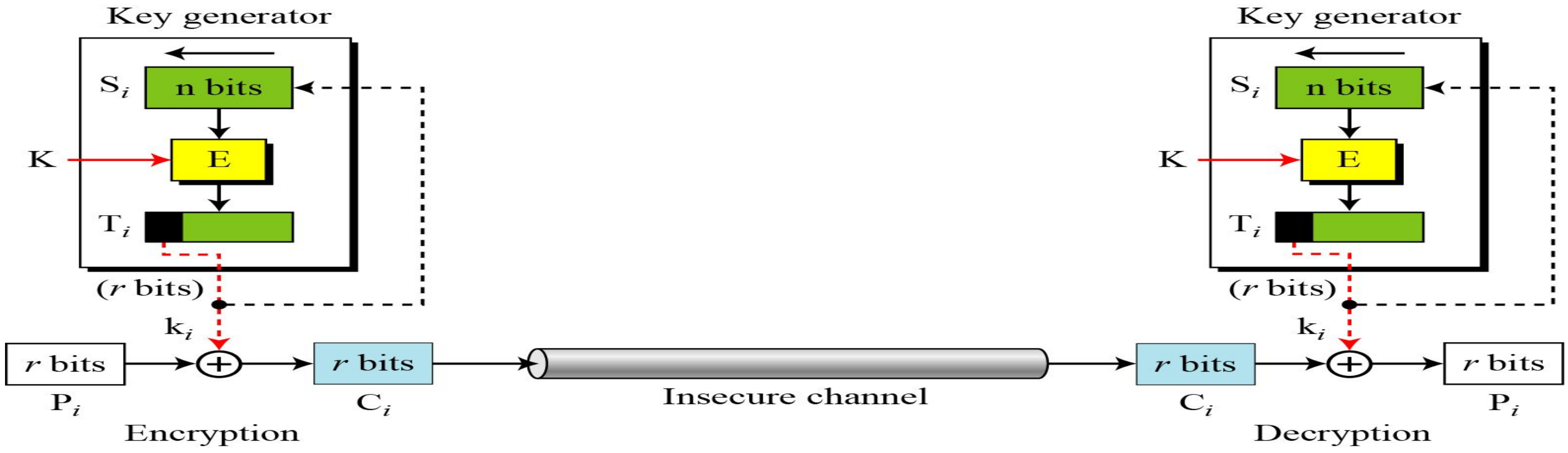
✓ Shift **left IV** by r bits and insert **r bits of EIV (Ti)** to its right

Output Feedback (OFB)



Output Feedback (OFB)

- Message is treated as a stream of bits
- Output of cipher is added to message
- Output is then feed back (hence name)



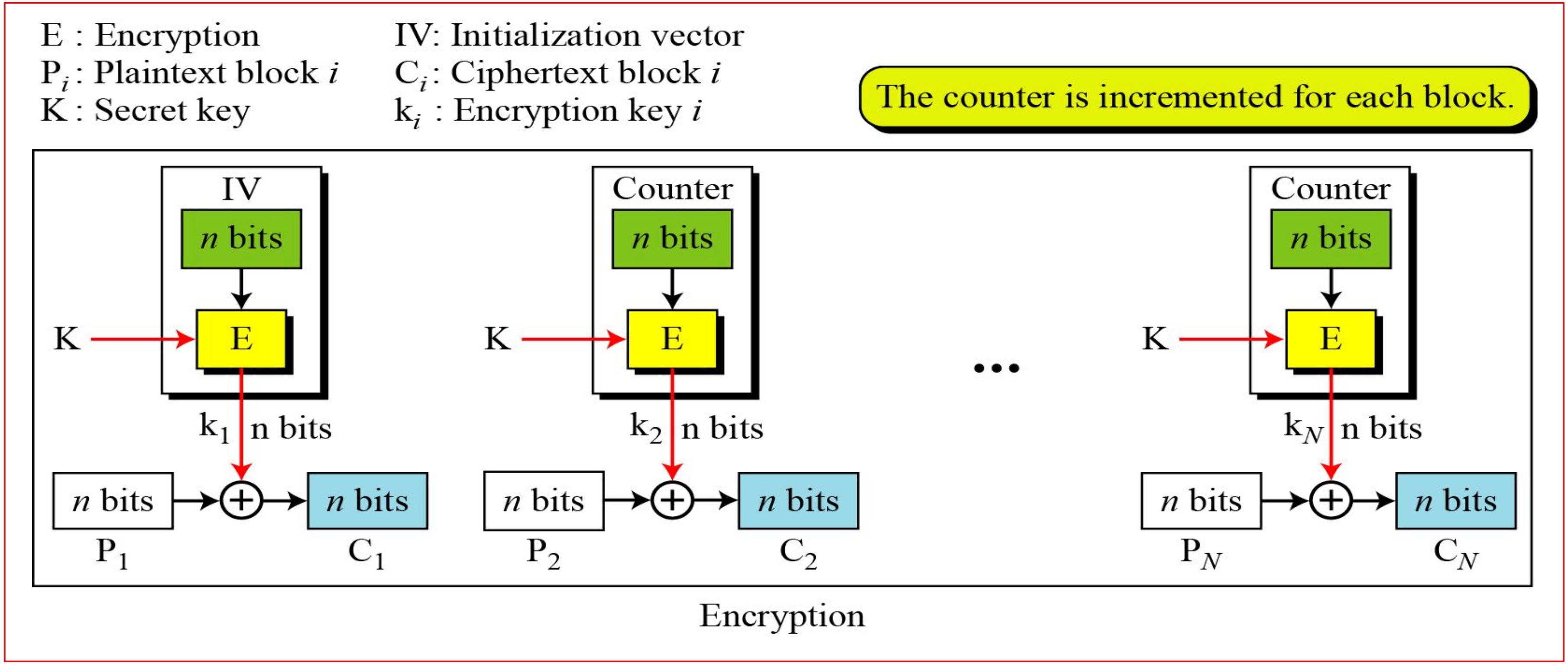
Output Feedback (OFB)

- **Problems**
  - ✔ bit errors do not propagate
  - ✔ more vulnerable to message stream modification
  - ✔ a variation of a Vernam cipher
  - ✔ sender & receiver must remain in sync
  - ✔ originally specified with m-bit feedback
  - ✔ subsequent research has shown that only full block feedback (ie CFB-64 or CFB-128) should ever be used

Counter Mode (CTR)

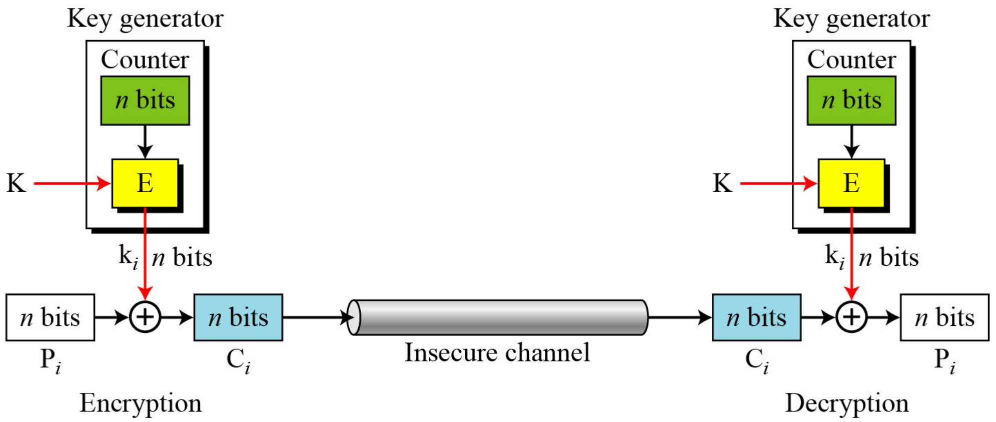
- At times we need to use block cipher that works as streams cipher
- In the counter (CTR) mode, there is no feedback.
- The pseudorandomness in the key stream is achieved using a IV = counter.
- No Chaining process is used.

Counter Mode (CTR)



Counter Mode (CTR)

- Similar to OFB but encrypts counter value rather than any feedback value
- Must have a different key and counter value for every plaintext block (never reused)
- **Uses**: h



Counter Mode (CTR)

- **Benefits & Problems**
  - ✓ can do parallel encryption in hardware or software
  - ✓ can reprocess in advance of need
  - ✓ good for burst high speed links
  - ✓ random access to encrypted data blocks
  - ✓ must ensure never reuse key/counter values

RSA Algorithm

- Ron Rivest, Adi Shamir and Len Aldeman have developed this algorithm (Rivest-Shamir-Adleman).
- It is a block cipher which converts plain text into cipher text and vice versa at receiver side.
- The algorithm works as follow

1. Select two prime numbers p and q where $p \neq q$.
2. Calculate $n = p * q$.
3. Calculate $\Phi(n) = (p-1) * (q-1)$.
4. Select e such that, e is relatively prime to $\Phi(n)$

    i.e. $(e, \Phi(n)) = 1$ and $1 < e < \Phi(n)$
5. Calculate $d = e \bmod \Phi(n)$ or $ed = 1 \bmod \Phi(n)$.
6. Public key = {e, n}, private key = {d, n}.
7. Find out cipher text using the formula,

    $C = P \bmod n$ where, $P < n$ and

    C = Cipher text, P = Plain text, e = Encryption key and

    n=block size.d
8. $P = C \bmod n$. Plain text P can be obtain using the given formula.
9. where, d = decryption key.

RSA Algorithm

Example: 1

1. Two prime numbers $p = 13, q = 11$.
2. $N = p * q = 13 * 11 = 143$.
3. $\Phi(n) = (13 - 1) * (12 - 1) = 12 * 10 = 120$.
4. Select $e = 13$, $\gcd(13, 120) = 1$.
5. Finding d:

    $e * d \bmod \Phi(n) = 1$

    $13 * d \bmod 120 = 1$

(How to find: $d * e = 1 \bmod \Phi(n)$ ➔ $d = ((\Phi(n) * i) + 1) / e$

$d = (120 + 1) / 13 = 9.30$ ($\because i = 1$)

$d = (240 + 1) / 13 = 18.53$ ($\because i = 2$)

$d = (360 + 1) / 13 = 27.76$ ($\because i = 3$)

$d = (480 + 1) / 13 = 37$ ($\because i = 4$))

RSA Algorithm

Example: 1

6. Public key = {13, 143} and private key = {37, 143}.

7. Encryption : Plain text P = 13. (where, P < n)

   $C = P^e \bmod n = 13^{13} \bmod 143 = 52.$

   $\boxed{\mathbf{C = 52}}$

8. Decryption:

   $P = C^d \bmod n = 52^{37} \bmod 143 = 13.$

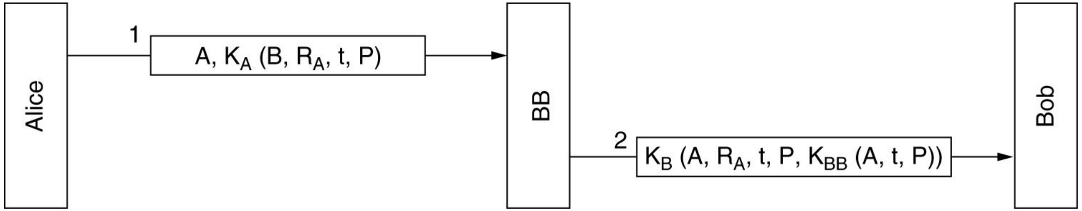
   $\boxed{\mathbf{P = 13}}$

DIGITAL SIGNATURES

- Symmetric-Key Signatures
- Public-Key Signatures
- Message Digests

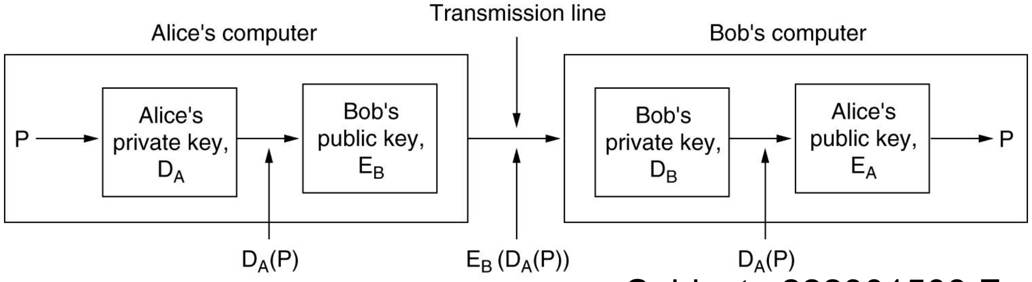
Symmetric-Key Signatures

- One approach to digital signatures is to have a central authority that knows everything and whom everyone trusts, say, Big Brother (BB).
- Each user then chooses a secret key and carries it by hand to BB's office.
- Thus, only Alice and BB know Alice's secret key, KA , and so on.
- When Alice wants to send a signed plaintext message, P, to her banker, Bob, she generates KA (B, RA , t, P), where B is Bob's identity, RA is a random number chosen by Alice, t is a timestamp to ensure freshness, and KA (B, RA , t, P) is the message encrypted with her key, KA . Then she sends it as depicted in Figure.
- BB sees that the message is from Alice, decrypts it, and sends a message to Bob as shown.
- The message to Bob contains the plaintext of Alice's message and also the signed message KBB (A, t, P). Bob now carries out Alice's request.



Public-Key Signatures

- Public-key cryptography can make an important contribution in this area.
- Let us assume that the public-key encryption and decryption algorithms have the property that E(D(P)) = P, in addition, of course, to the usual property that D(E(P)) = P. (RSA has this property, so the assumption is not unreasonable.)
- Assuming that this is the case, Alice can send a signed plaintext message, P, to Bob by transmitting EB (DA (P)). Note carefully that Alice knows her own (private) key, DA , as well as Bob's public key, EB , so constructing this message is something Alice can do.
- When Bob receives the message, he transforms it using his private key, as usual, yielding DA (P), as shown in Fig. 8-19. He stores this text in a safe place and then applies EA to get the original plaintext.

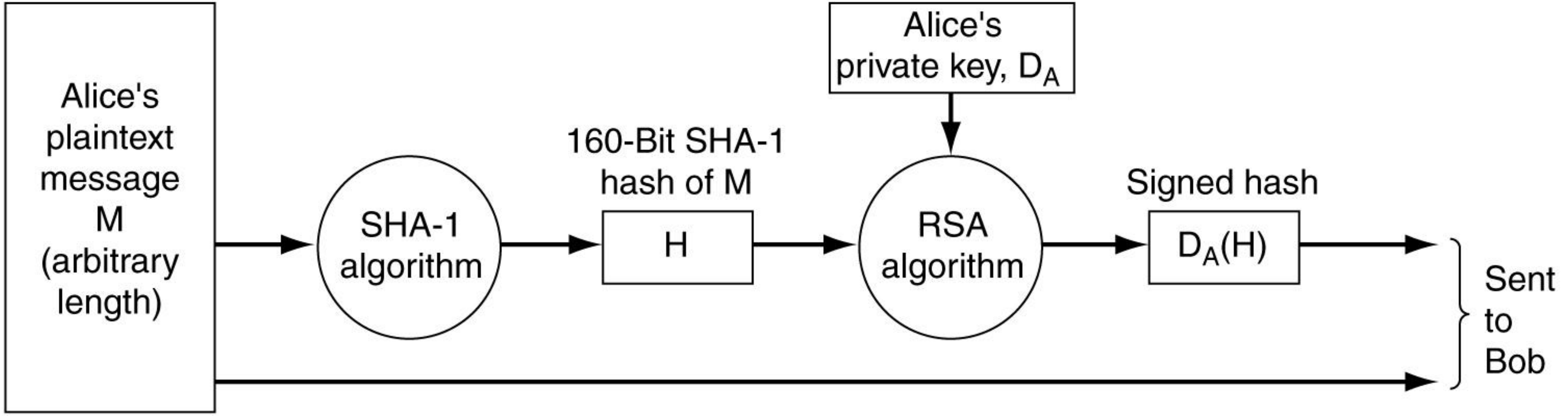


Message Digest

- This scheme is based on the idea of a one-way hash function that takes an arbitrarily long piece of plaintext and from it computes a fixed-length bit string.
- This hash function, MD, often called a message digest, has four important properties:
- 1. Given P, it is easy to compute MD(P).
- 2. Given MD(P), it is effectively impossible to find P.
- 3. Given P, no one can find P′ such that MD (P′) = MD(P).
- 4. A change to the input of even 1 bit produces a very different output.

Message Digest-SHA-1 and SHA-2

- A variety of message digest functions have been proposed. One of the most widely used functions is SHA-1 (Secure Hash Algorithm 1) (NIST, 1993).
- Like all message digests, it operates by mangling bits in a sufficiently complicated way that every output bit is affected by every input bit.
- SHA-1 was developed by NSA and blessed by NIST in FIPS 180-1. It processes input data in 512-bit blocks, and it generates a 160-bit message digest.



Message Digest-MD5

- MD5 (Rivest, 1992) is the fifth in a series of message digests designed by Ronald Rivest. Very briefly, the message is padded to a length of 448 bits (modulo 512).
- Then the original length of the message is appended as a 64-bit integer to give a total input whose length is a multiple of 512 bits.
- Each round of the computation takes a 512-bit block of input and mixes it thoroughly with a running 128-bit buffer.
- For good measure, the mixing uses a table constructed from the sine function.
- The point of using a known function is to avoid any suspicion that the designer built in a clever back door through which only he can enter.
- This process continues until all the input blocks have been consumed.
- The contents of the 128-bit buffer form the message digest.

THANK YOU