ЗАШТИТА ПОДАТАКА

Симетрични алгоритми заштите

додатак

Преглед

- Биће објашњено:
  - Модови функционисања блок алгоритама
  - Triple-DES
  - RC4

Модови функционисања

- блок алгоритми шифрују блокове фиксне величине
- нпр. DES шифрује 64-битне блокове са 56-битним кључем
- потребно је то некако искористити у пракси, с обзиром да обично постоји произвољна количина информација које је потребно шифровати
- четири решења су дефинисана за DES у ANSI стандарду **ANSI X3.106-1983 Modes of Use**
- сада постоје 5 модова функционисања за било који симетрични блок алгоритам (укључујући и 3DES и AES)
- постоје **block** и **stream** модови

Модови функционисања (2)

Мод функционисања	Типична примена
Electronic Codebook Book (ECB)	Сигуран пренос појединачних вредности (нпр. кључева)
Cipher Block Chaining (CBC)	Мод опште намене за блоковски пренос података Аутентикација
Cipher FeedBack (CFB)	Мод опште намене за пренос података на нивоу тока Аутентикација
Output FeedBack (OFB)	Мод за пренос података на нивоу тока код канала са шумовима (нпр. сателитска комуникација)
Counter (CTR)	Мод опште намене за блоковски пренос података Погодан када се захтева велика брзина

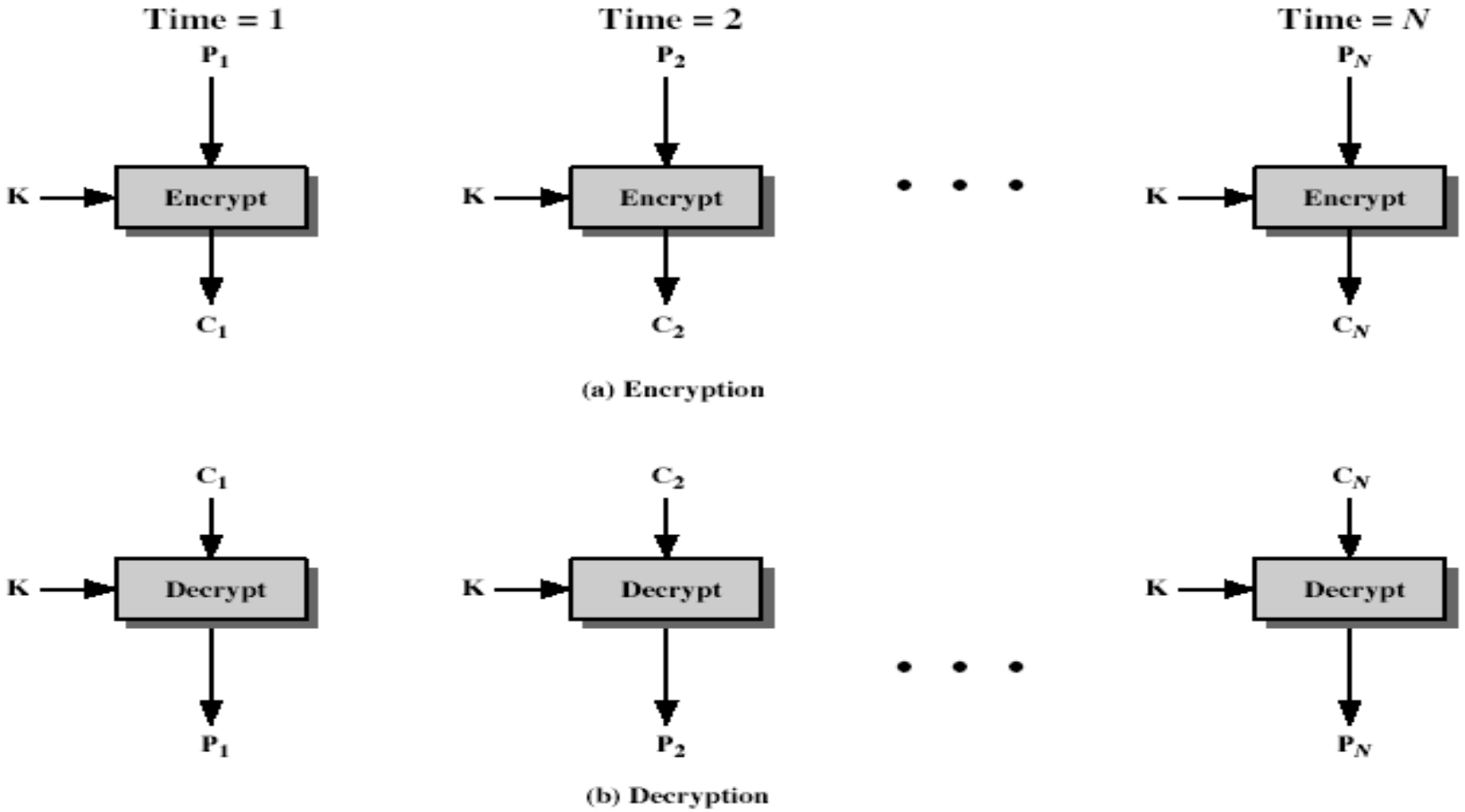
Electronic Codebook Book (ECB)

- порука се дели у независне блокове који се шифрују (уз допуну последњег блока по потреби)
- сваки блок представља вредност која је замењена, као некакав шифрарник, отуда и име
- сваки блок се шифрује независно у односу на остале блокове

$$C_i = E_K (P_i)$$

- употреба: сигуран пренос појединачних вредности

Electronic Codebook Book (ECB)



Предности и ограничења ECB мода

- понављања у поруци могу се видети у шифрованом тексту (исти блок шифрован увек даје исти резултат)
  - нпр, ако се ради о графичким елементима
  - или неке друге поруке које се јако мало мењају
- главна употреба ако се шаље неколико блокова података, нпр. кључ за шифровање

Cipher Block Chaining (CBC)

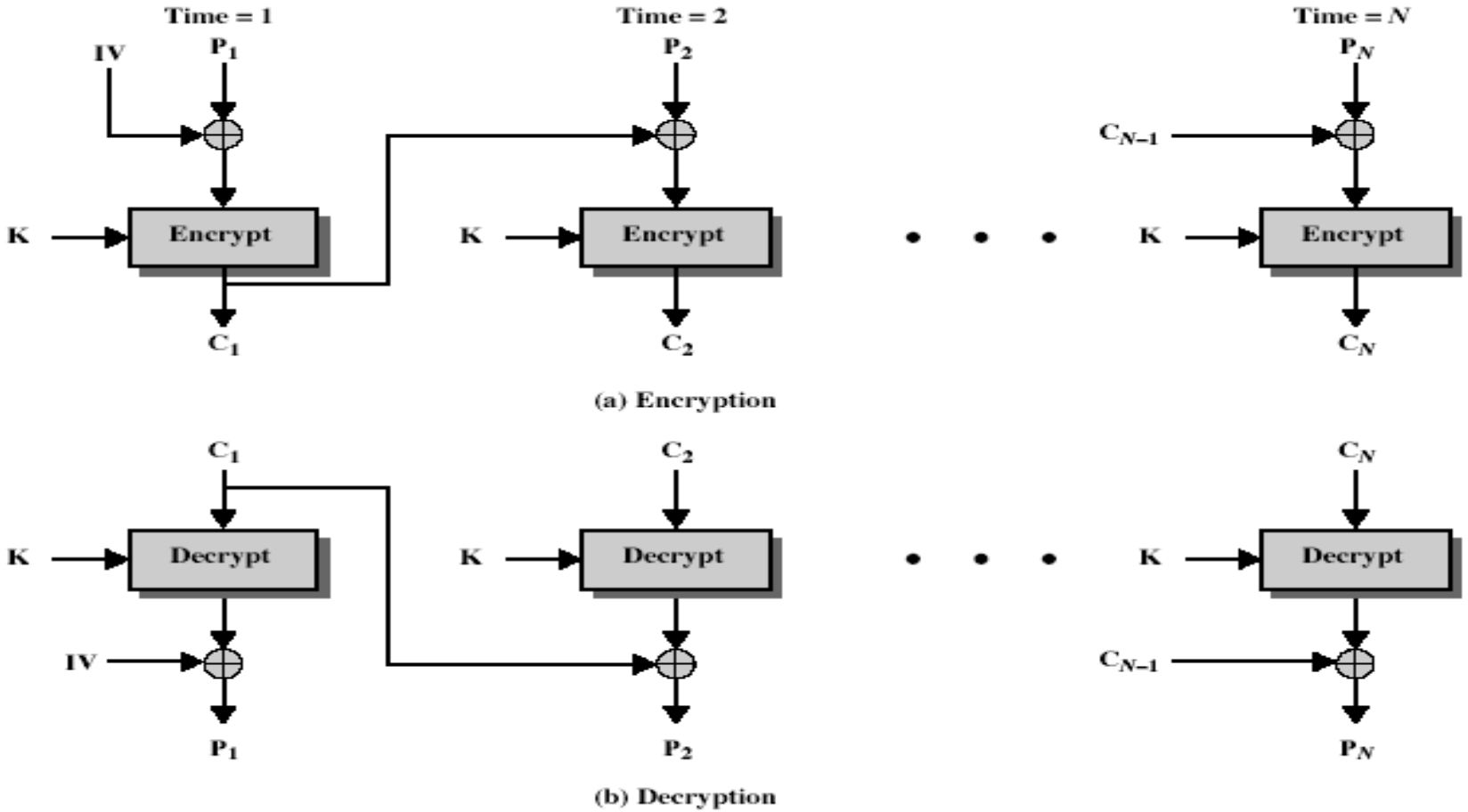
- порука се дели у блокове
- али блокови се шифрирају повезано
- сваки претходни шифрирани блок се уланчава са тренутним блоком поруке
- користи се почетна вредност (Initial Value - IV) да се започне поступак

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

$$C_{-1} = IV$$

- употреба: шифровање велике количине података (bulk data), аутентикација (authentication)

Cipher Block Chaining (CBC)



Предности и ограничења CBC мода

- дешифровање

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

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

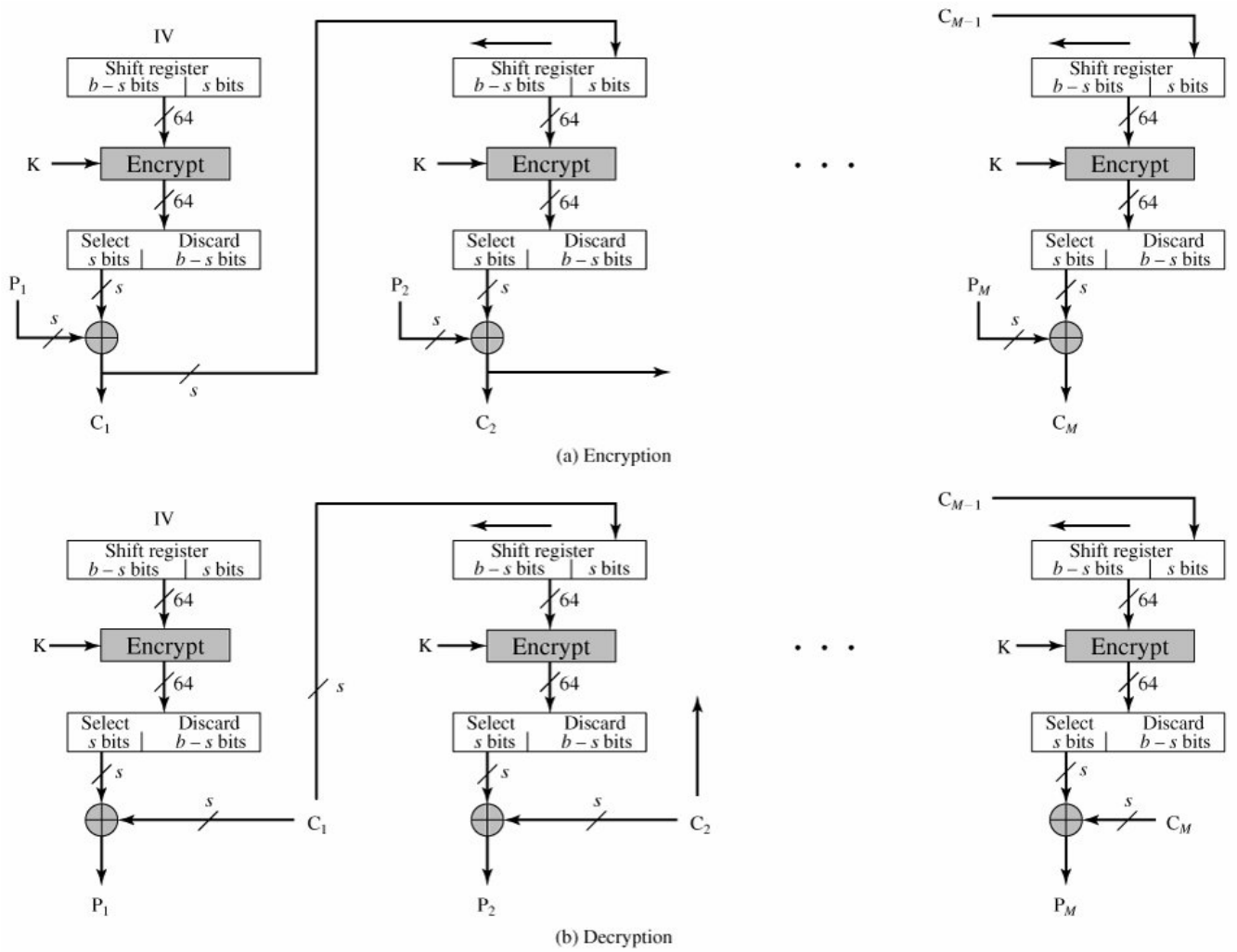
  $C_{i-1} \text{ XOR } D_K(C_i) = C_{i-1} \text{ XOR } P_i \text{ XOR } C_{i-1} = P_i$

- сваки блок за шифровање зависи од свих осталих блокова
- свака промена у поруци, утиче на промену у свим блоковима
- потребна је почетна вредност (**Initial Value** - IV) коју знају и пошиљалац и прималац

Cipher FeedBack (CFB)

- порука се третира као ток битова
- додаје се на излаз неког блок алгоритма
- резултат је feed back за следећи корак
- стандард дозвољава било који број битова (1,8,64,...) да буде feed back
  - називају се CFB-1, CFB-8, CFB-64 итд.
- $C_i = P_i \text{ XOR } E_K(C_{i-1})$; $C_{-1} = IV$
- употреба: шифровање тока података (stream data), аутентикација (authentication)

Cipher FeedBack (CFB)



Предности и ограничења CFB мода

- најчешћи када се ради о току података
- грешке пропагирају кроз неколико блокова док се не схвати да се ради о грешци

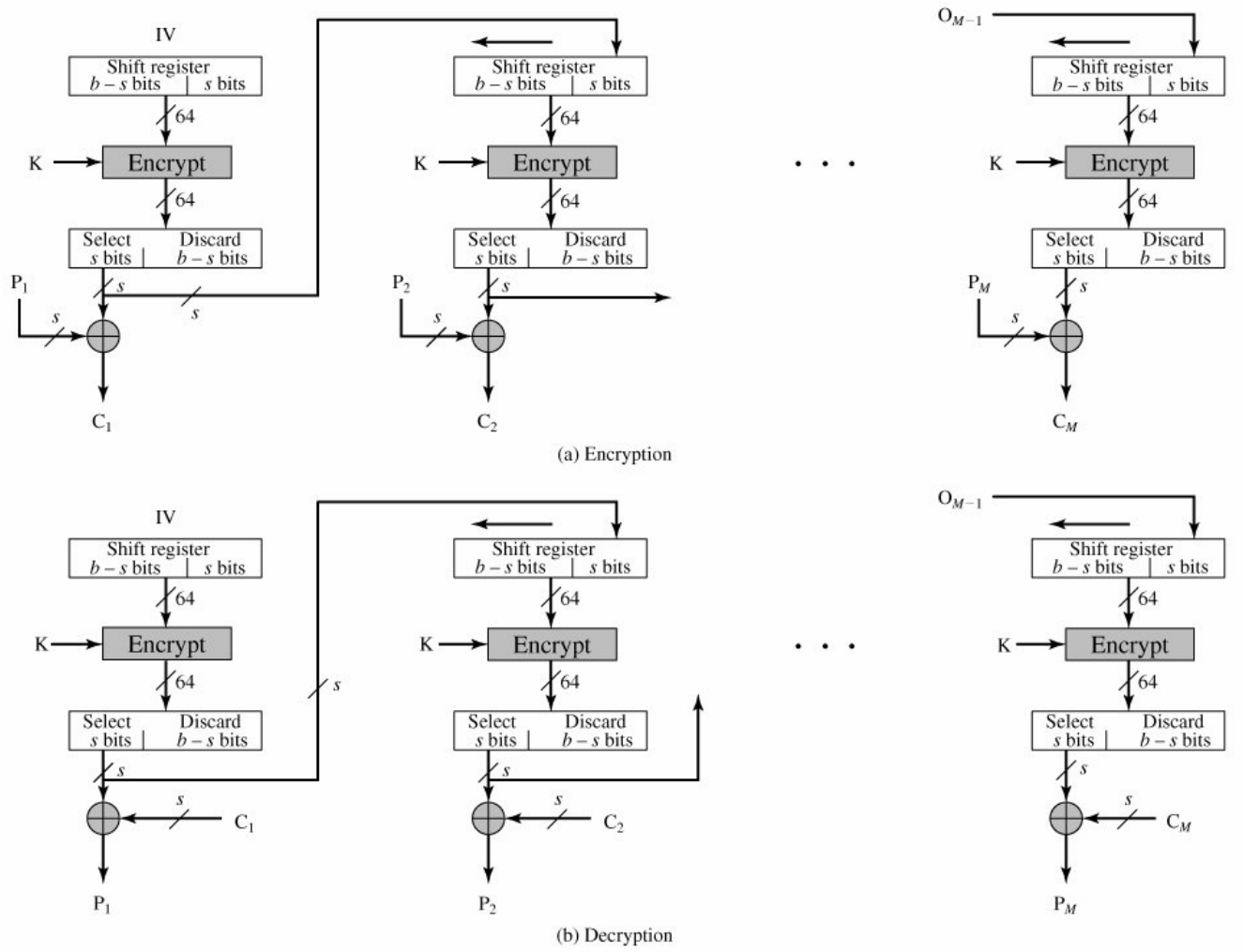
Output FeedBack (OFB)

- порука се третира као ток битова
- излаз блок алгоритма се додаје поруци
- тај излаз је повратна вредност за следећи корак
- feedback је независан у односу на поруку
- може се унапред израчунати

$$C_i = P_i \text{ XOR } O_i$$
$$O_i = E_K(O_{i-1})$$
$$O_{-1} = IV$$

- употреба: шифрирање тока података за слање по каналима који су пуни шумова

Output FeedBack (OFB)



Предности и ограничења OFB мода

- користи се када је потребно дешифровати поруку и пре него што је цела стигла
- сличан је као CFB, али је повратна информација независна од поруке
- мора се наћи начин да се буде сигуран да су пошиљалац и прималац синхронизовани

Counter (CTR)

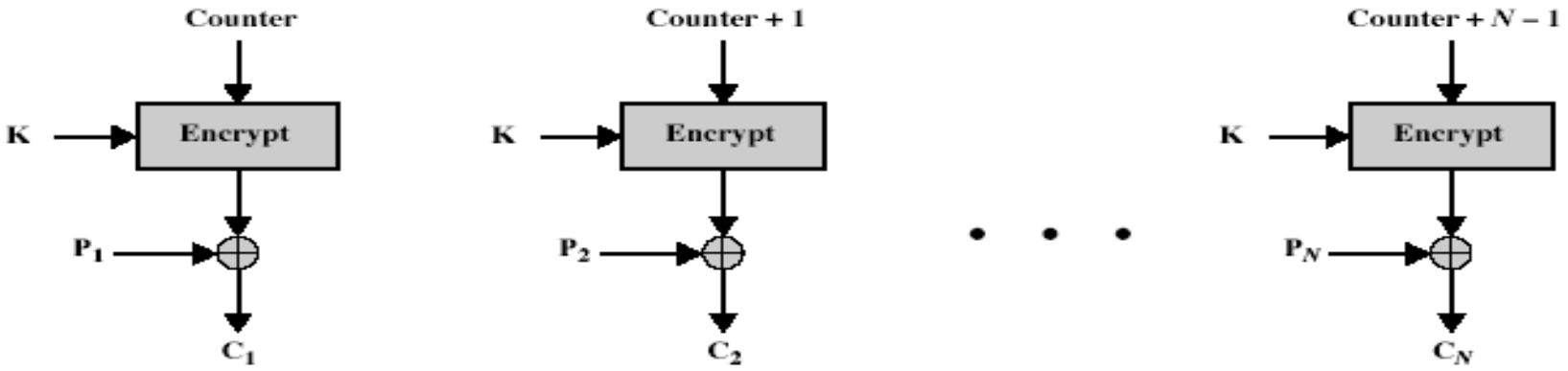
- сличан као OFB али шифрира се вредност бројача, а не повратне информације
- бројач мора бити једнаке дужине као блок података који се шифрује
- мора имати различиту вредност бројача за сваки блок који се шифрира

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

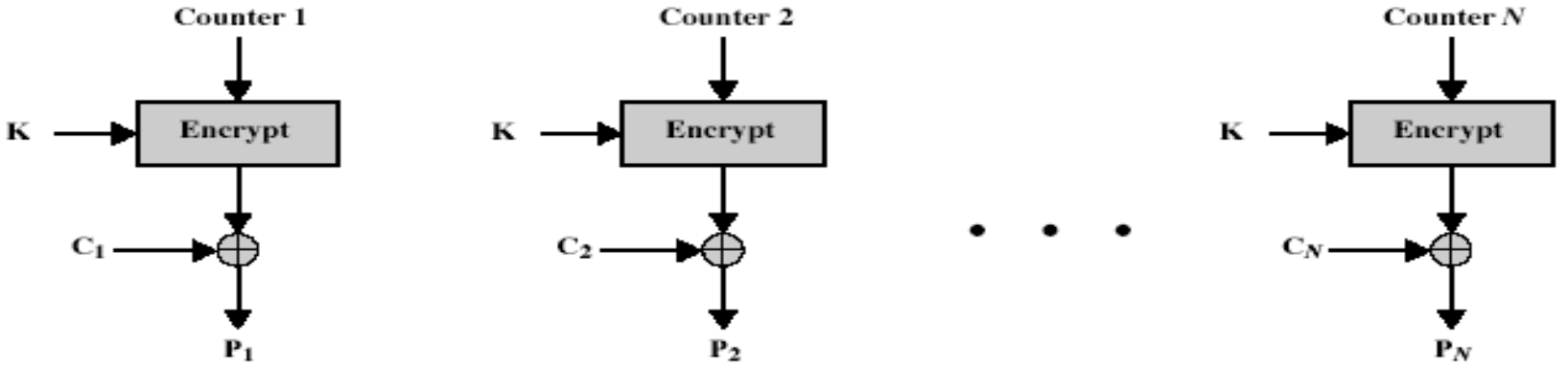
$$O_i = E_K(i)$$

- употреба: шифровање у мрежама велике брзине

Counter (CTR)



(a) Encryption



(b) Decryption

Предности и мане CTR мода

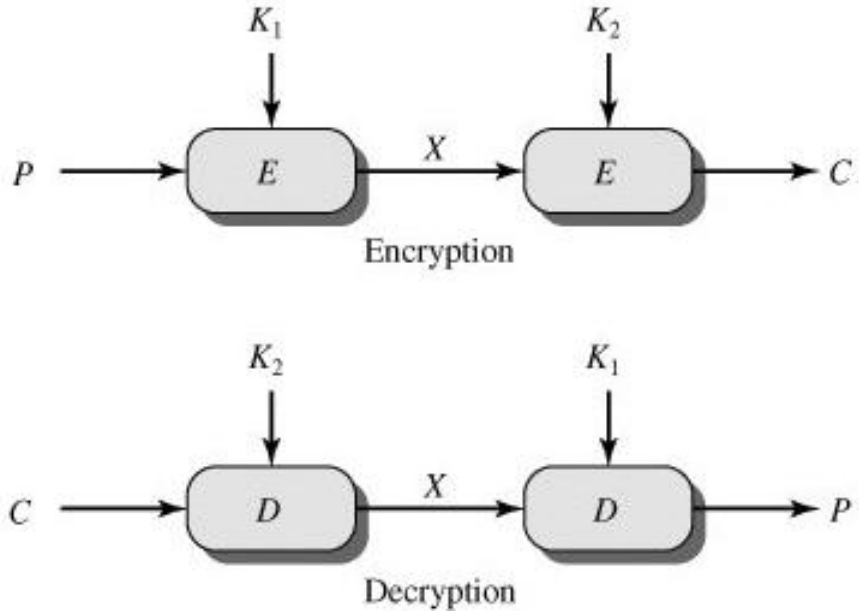
- ефикасност
  - може да паралелно ради шифровање
  - добар за линкове са великом брзином
- могуће претпроцесирање (припрема вредности за XOR унапред)
- произвољан приступ (може се дешифровати произвољан блок)
- доказана безбедност (једнако сигуран као и остали)
- једноставност (користи се само шифровање, нема уланчавања)
- мора се осигурати да комбинација кључ/бројач никада не буде два пута искоришћена у истој поруци

Triple DES

- пошто је била потребна замена за DES пре него што је смишљен AES коришћено је вишеструко шифровање са DES
- Triple-DES је изабран за тако нешто
- зашто не Double-DES?
- због meet-in-the-middle напада
  - чим се два пута користи шифра, овај напад успева

Meet-in-the-middle

- ако имамо
  - C = E(K2, E(K1, P))
- тада
  - X = E(K1, P) = D(K2, C)



Meet-in-the-middle

- За познати пар P и C напад изгледа на следећи начин:
- Шифровати P са свих 2^{56} могућих вредности кључа K1
- Направити табелу ових резултата и сортирати по X
- Дешифровати C користећи свих 2^{56} могућих вредности кључа, упоређујући сваки резултат са X из табеле

Triple-DES са два кључа

- дакле морају се користити 3 шифровања
  - изгледа да су потребна 3 кључа
- али могу се користити 2 кључа са E-D-E секвенцом
  - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
  - ако је $K1=K2$ може се радити са само једним DES, backward compatibility
- стандардизовано у ANSI X9.17 & ISO8732

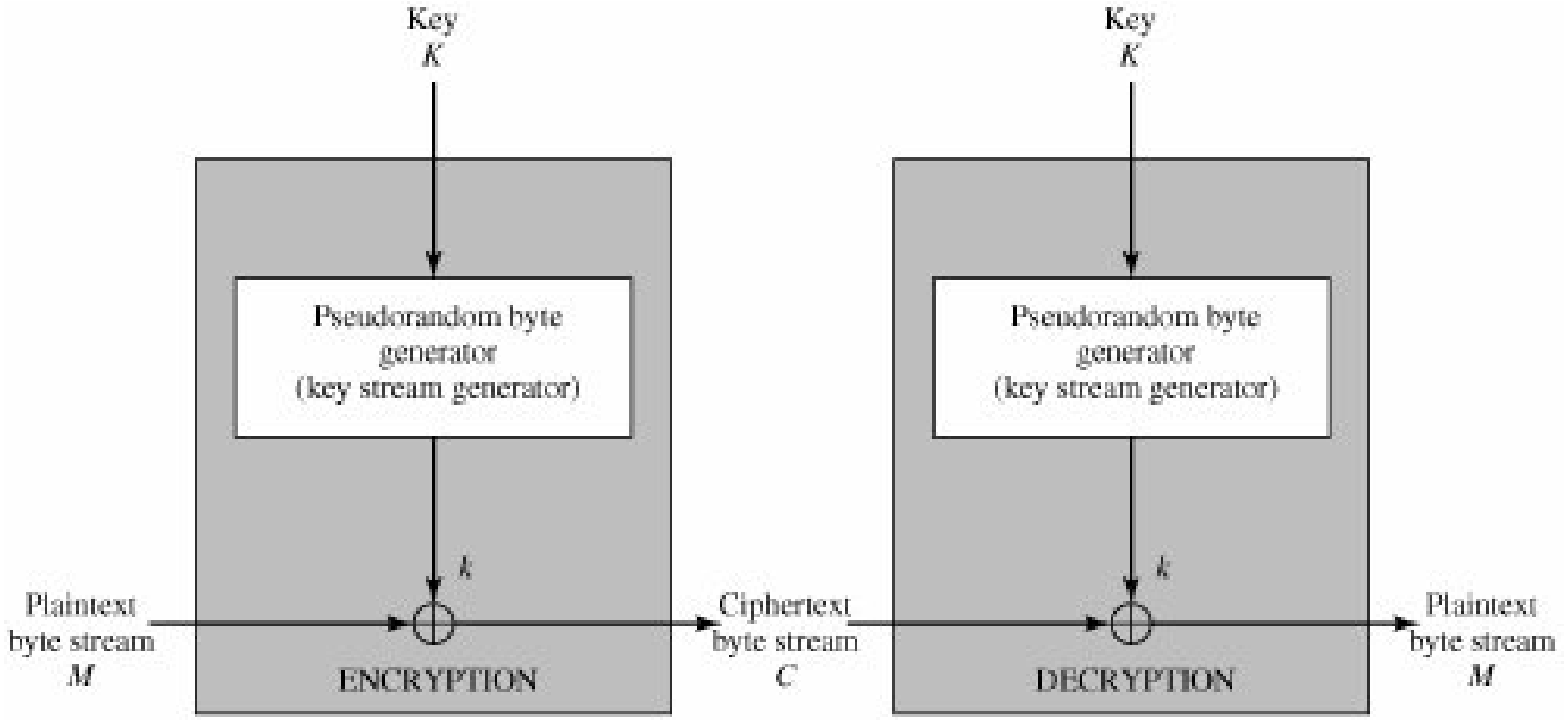
Triple-DES са 3 кључа

- иако се не зна за нападе на Triple-DES са два кључа, постоје неке назнаке да их има
- може се користити Triple-DES са 3 кључа да би се чак и то избегло
  - $C = E_{K3}[D_{K2}[E_{K1}[P]]]$
- неке интернет апликације користе ову технологију, нпр. PGP, S/MIME

Алгоритми за шифрирање тока

- процесирају поруку бит по бит (као ток)
- обично имају (псеудо) случајан **кључ тока**
- комбинује се са поруком бит по бит (XOR)
- случајност **кључа тока** уништава било какав статистички фактор у поруци
  - $C_i = M_i \text{ XOR } StreamKey_i$
- никад лакше...
- али никад не сме да се понови кључ тока

Алгоритми за шифрирање тока



Особине алгоритама за шифрирање тока

- неке особине које морају бити размотрене приликом дизајнирања су:
  - дуг период без понављања
  - статистички случајан
  - зависи од довољно великог кључа
  - велика сложеност

RC4

- у власништву RSA DSI
- Ron Rivest је дизајнирао, једноставно али ефикасно
- променљива величина кључа, бајтовски оријентисан алгоритам шифрирања тока података
- широко распрострањен (web SSL/TLS, wireless WEP)
- кључ променљиве дужине (од 8 до 2048 бита) формира случајну пермутацију свих 8-битних вредности
- користи се та пермутација да се шифрира улазни податак бајт по бајт

RC4 распоред кључа

- почиње са низом S бројева: 0..255
- користи се кључ да измеша овај низ
- S формира унутрашње стање (**internal state**) алгоритма
- за кључ k дужине l бајтова

```
for i = 0 to 255 do
   S[i] = i
j = 0
for i = 0 to 255 do
   j = (j + S[i] + k[i mod l]) (mod 256)
   swap (S[i], S[j])
```

RC4 шифровање

- алгоритам наставља да меша вредности низа
- сума измешаног пара одабира кључ тока
- за XOR са следећим бајтом поруке за шифровање/дешифровање

```
i = j = 0
for each message byte M_i
   i = (i + 1) (mod 256)
   j = (j + S[i]) (mod 256)
   swap(S[i], S[j])
   t = (S[i] + S[j]) (mod 256)
   C_i = M_i XOR S[t]
```

RC4 сигурност

- тврди се да је сигуран против свих познатих напада
  - направљене су неке анализе напада, али ниједна се није показала практичном
- резултат је веома нелинеаран
- како је RC4 алгоритам тока података, не сме се никада поновити исти кључ