Block Ciphers

Modes of Encryption

Why do we need Modes?

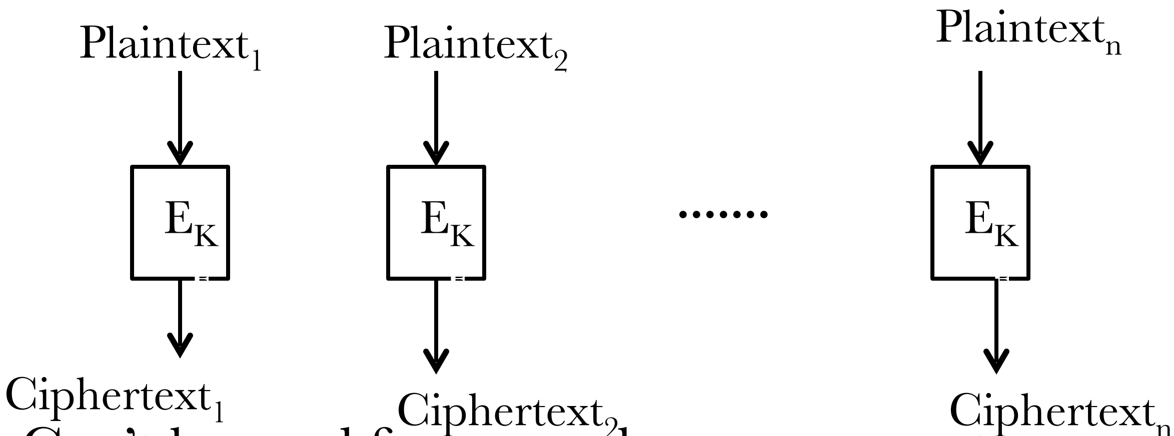
- Recollect

$$\underbrace{E_K}_{\text{k bits}} \quad \underbrace{(M)}_{\text{l bits}} \quad = \underbrace{(K, M)}_{\text{l bits}}$$

$$E : \{0,1\}^{k} \text{ x } \{0,1\}^{l} \rightarrow \{0,1\}^{l}$$

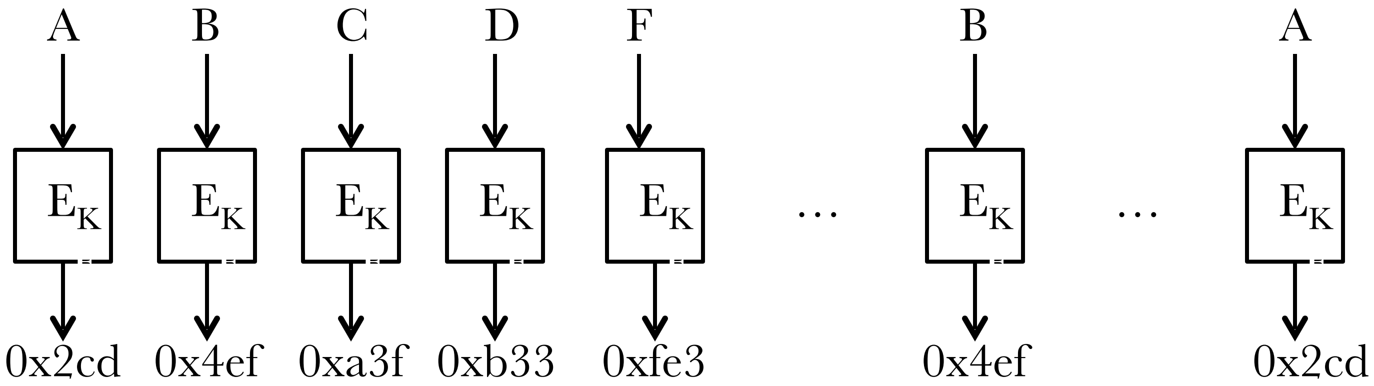
- What if M is greater than *l* bits?
- Modes tell us how to apply E_K to M, when M is:
  - Long sequence of data blocks
  - Data stream

Electronic Code Book (ECB) Mode



- Can't be used for general-purpose crypto
- Main weakness: If $Plaintext_1$ = $Plaintext_2$, then $Ciphertext_1$ = $Ciphertext_2$ (i.e., deterministic encryption)
- Not IND-CPA secure (or even IND-KPA secure)
- Zoom used AES in ECB mode until April'20!

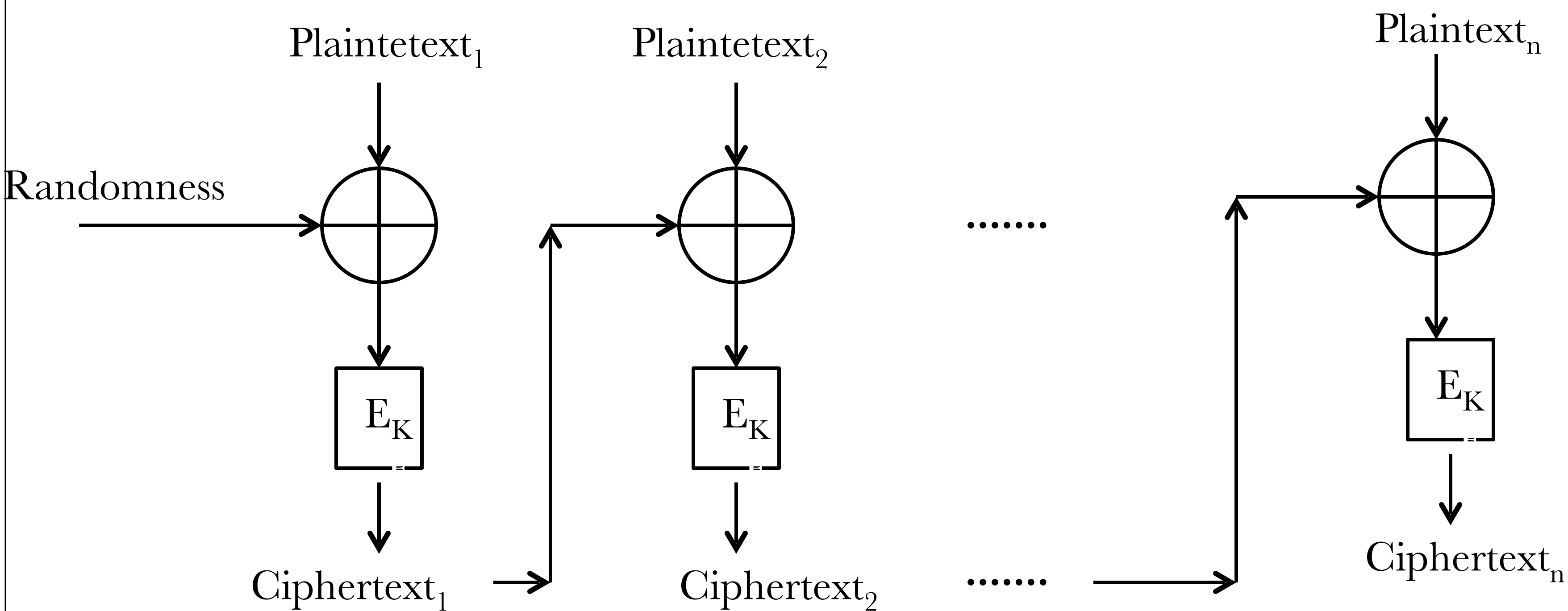
ECB Weakness



- Solution: randomize the plaintext messages
- Used for transmitting short, non-repeating plaintext, e.g., key, IV

Stateless Cipher Block Chaining (CBC) Mode

- Corrects weakness of ECB mode



- Randomness —> "Initialization vector (IV)"

Used in SSL/TLS, 3.0/1.0

Stateless CBC Mode

- All message blocks same size (as in ECB)

- What if message blocks run short? Especially last one
  - E.g., M = 429 bits, block size = 128
  - $M_1 = 128$, $M_2 = 128$, $M_3 = 128$, $M_4 = 45$
  - Set $M_4 = 45 + 83 = 128$ bits
  - 83 bits padding

- How will receiver know?
  - Send info about padding along with encryption

Stateless CBC Mode

- Initialization Vector (IV) – randomness fed in to first block
  - At times, secret between sender and receiver, although not always
  - Must be random (at least computationally random), unpredictable, not sufficient to be *distinct (yet predictable*!
  - Can be sent through ECB encryption
- Stateless model: *Fresh* IV for every message $M=(m_1,\ldots m_n)$

- Where does IV come from?
- Good source of randomness
  - E.g., PRFs, PRNG's implemented in software or hardware: timestamps, keyboard strokes, non-repeating counters, ….

Stateful Cipher Block Chaining (Chained-CBC) Mode

- Subtle changes to stateless CBC…
- Also called "chained" CBC mode

- Idea: We are chaining ciphertexts for blocks of message $M = (m_1,..m_n)$: $c_i = E_K(c_{i-1} \oplus m_i)$
  - So why not use last cipher text block c_n as IV for *different, fresh* message $M' = (m_1',\ldots m_n')$?
  - Seems logical and harmless right?
  - … Results in mode that is IND-CPA-insecure (and hence doesn't even meet min. standard of security)

Stateful CBC Mode

- IND-CPA attack game:
  - Adversary picks (m_1^0, m_1^1), gives to challenger
  - m_1^b is used by challenger as m_1 in encrypting $M = (m_1, m_2, m_3)$
  - IV, c_1, c_2, c_3 is given to adversary
  - Adversary picks another m_4 thus: $(m_4 = IV \oplus m_1^0 \oplus c_3)$, gives to challenger*
  - m_4 used in $M' = (m_4, m_5)$. First block is $E_K(M') = E_K(m_4 \oplus c_3)$
  - c_4, c_5 is given to adversary,
  - Easy to verify $m_1 = m_1^0$ iff $c_4 = c_1$

*In IND-CPA game, adversary can query (submit messages to) challenger as many times as she wants

Stateful CBC Mode

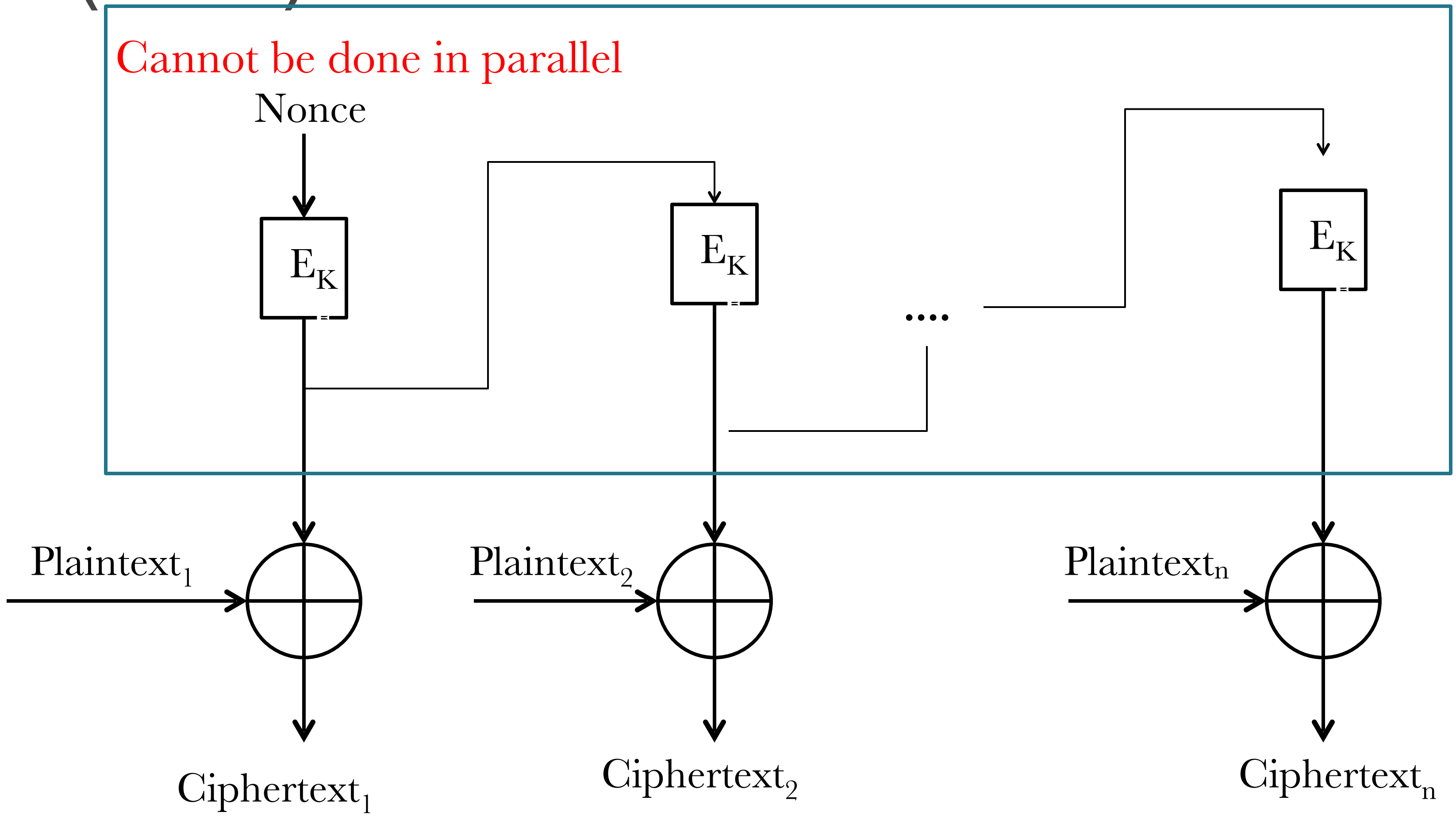
- Stateless is secure, but stateful isn't?! Whence came the problem?
- Vulnerability arises since:
  - Stateful adversary knows IV of second message (stateless adversary doesn't, fresh IV chosen per message)
  - Successive IVs contain previous ciphertexts

- Take-away point: Subtle modifications to well-known schemes, even seemingly benign ones, can be dangerous!
- …Yet stateful CBC (*still*) used in SSL/TLS, 3.0/1.0! Go figure!

Drawback of CBC

- Any problems?
- Forget ECB security issues, what about CBC?
- Not security problems, but efficiency…
- N^{th} encryption always depends on $N\text{-}1^{th}$ one
  - Any given ciphertext function of all preceding plaintexts
- No parallelism!

Stateless Output Feedback Mode (OFB)



OFB Mode

- Pros:
  - Errors are not propagated. Why?
  - Because plaintext/ciphertext of N^{th} block not fed in to $N+1^{th}$ block
  - Localized errors (if error in P_1, only C_1 is affected)
  - Stateful (chained) OFB is also IND-CPA secure! (why?!)*

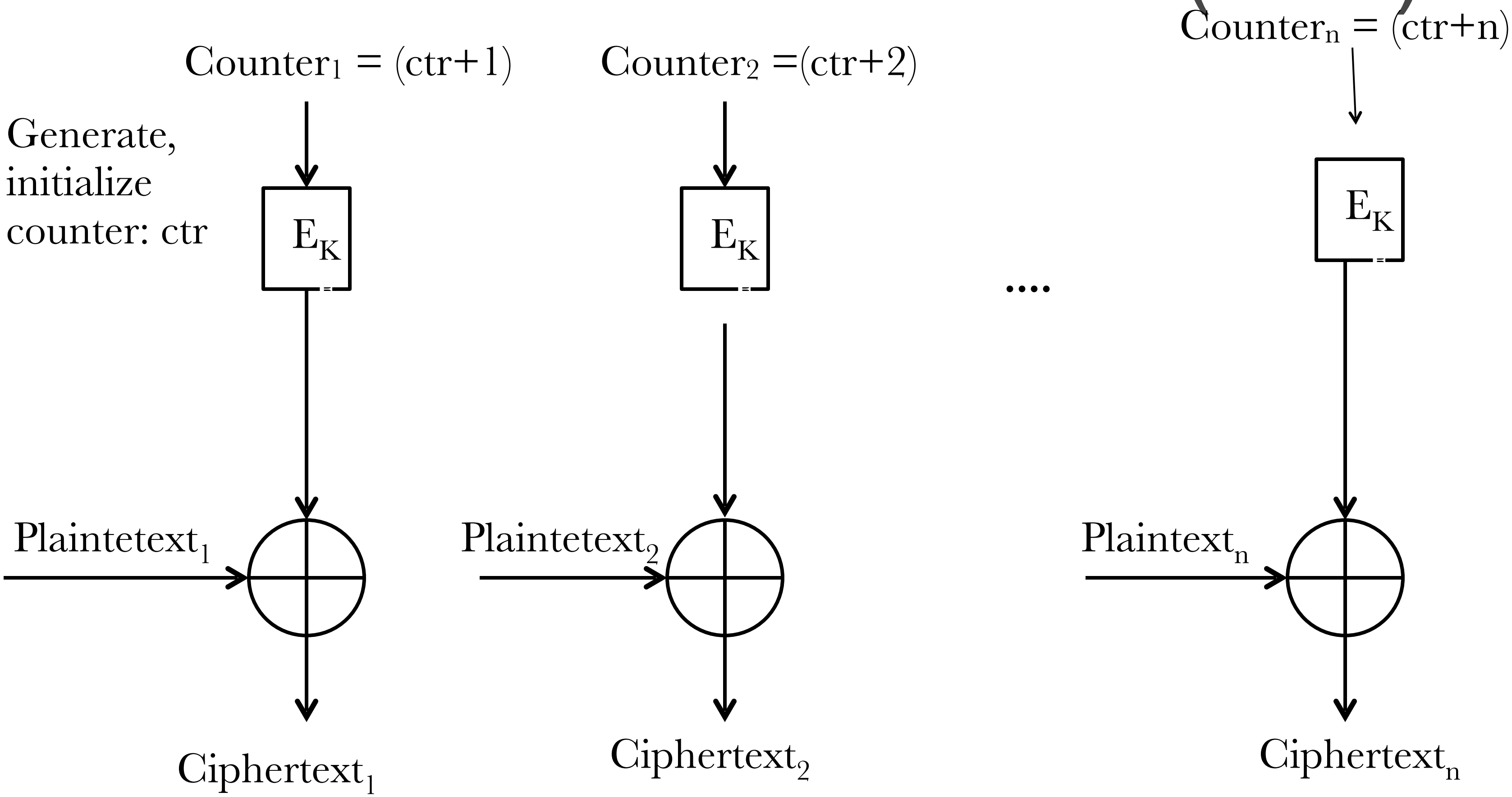
- Cons:
  - Vulnerable to message modification attacks
  - Change C_1 to C_1', P_1 gets changed to P_1'
  - Recovered plaintext is incorrect

*Because successive nonces do not contain previous ciphertexts, Adversary cannot set nonces that cancel out in IND-CPA game

OFB Mode

- Parallelism ?
- Somewhat parallizable …
- How?
  - Encrypt nonce
  - Compute all subsequent encryption blocks
  - XOR all plaintext blocks in parallel
  - Compute all ciphertexts in parallel

Stateless Counter Mode (CTR)



Stateless CTR Mode

- Nice things about CTR mode:
- No chaining = parallelizable encryptions
- Prepare all counter encryptions beforehand
  - Assuming we have a way to store them safely!
- No dependencies. Can compute C_i 's in random order

- Parallelism ins't entirely possible with CBC, CFB, and OFB (ECB is fully parallelizable though)
- Stateful and stateless CTR mode are IND-CPA secure (why?)

Note about IVs

- Initial randomness fed in to first block
- $|IV| = l$; l is block size of E
- But IV-space is limited

- On avg., IV repeats after $2^{l/2}$ messages
- E.g., consider 64-bit DES, $l = |IV| = 64$
- After 2^{32} (≈4,300,000,000) encryptions, approx. 34GB of plaintext, repeated IV expected to occur
- Of course IV-space can be increased by increasing $|l|$

Integrity

- Until now only worried about adversary violating confidentiality (i.e., guessing m_i from c_i), not integrity
- E.g., what if adversary mangles any or all of $c_1, \ldots, c_n$?

- Different topic (MAC); different models of security (IND-CPA, etc. only for encryption)
- Up until now, Chapters 1, 3 done, we'll see MAC and associated security models in Chapter 4