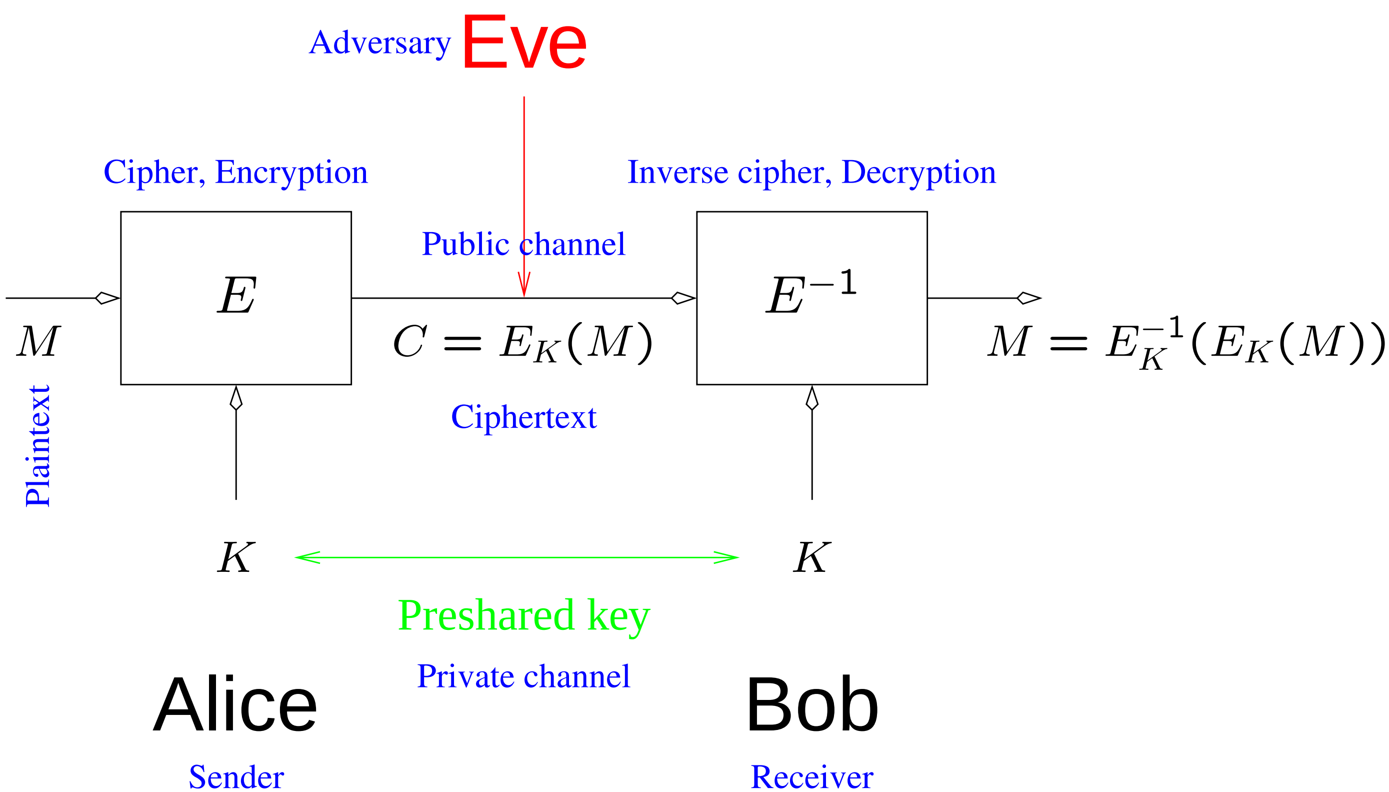
Lecture 3: Modes of Operation

Helger Lipmaa
Helsinki University of Technology
helger@tcs.hut.fi

Reminder: Communication Model



Reminder: Block Ciphers

- Usually a permutation $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$
- n is the block length, k is the key length
- Exhaustively searching k-bit keys takes 2^k time units
- Storing sufficient amount of plaintext-ciphertext pairs takes 2^n memory units. Birthday attack: $2^{n/2}$ memory units sufficient
- Recommendations: key $k \geq 80$ bits
- Recommendations: block $n \geq 128$ bits

Block cipher modes: Motivation

- A fixed block cipher work with a fixed block length
- One needs to encrypt arbitrary long messages
- Approach 1: design a new block cipher for every block length
- Must do new security evaluation for every cipher

Block cipher modes: Motivation

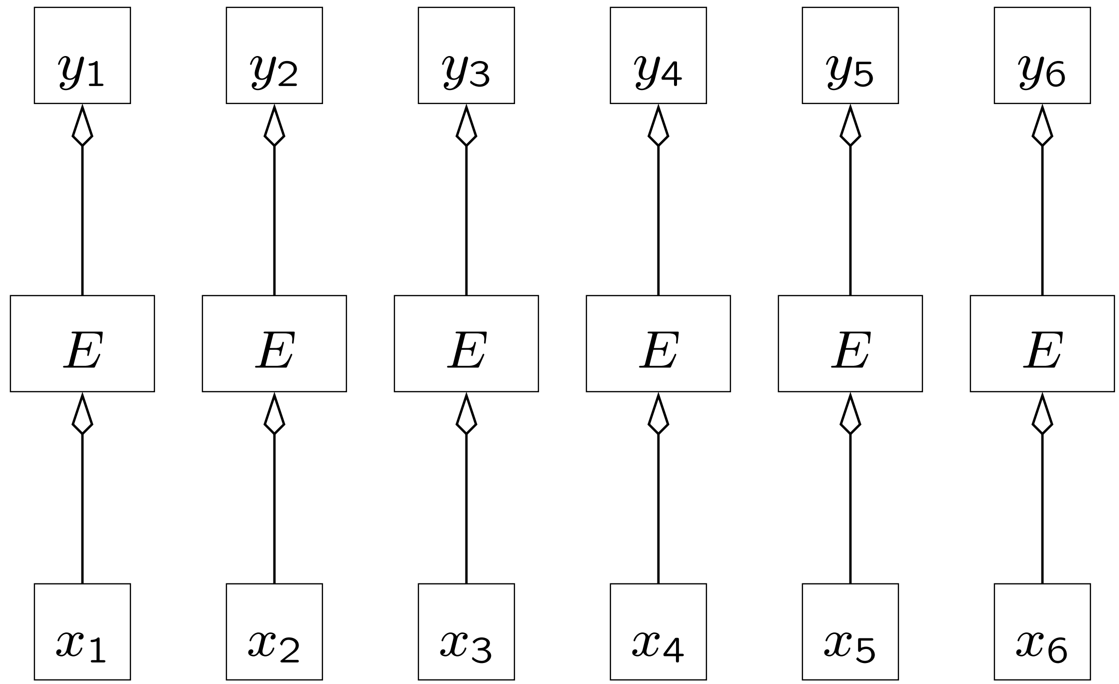
- Approach 2 (block cipher modes): use a block cipher E in an higher level protocol Π

- Hopefully can do a security reduction: if E is secure then Π is secure

- Modus ponens: If A and $A \Rightarrow B$ then B

- For this, one must use block cipher modes

ECB: Electronic Codebook



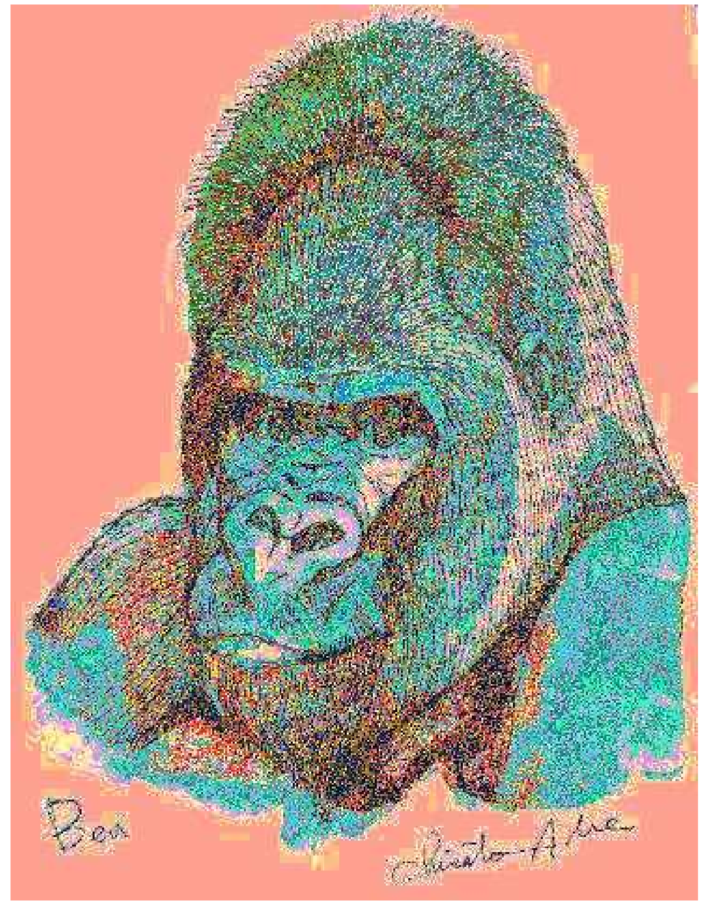
Simplest mode! (Also, already seen in the first lecture)

Insecurity of ECB

- If $y_i = y_j$ for two different ciphertext blocks then we know that $x_i = x_j$. Works also across different messages

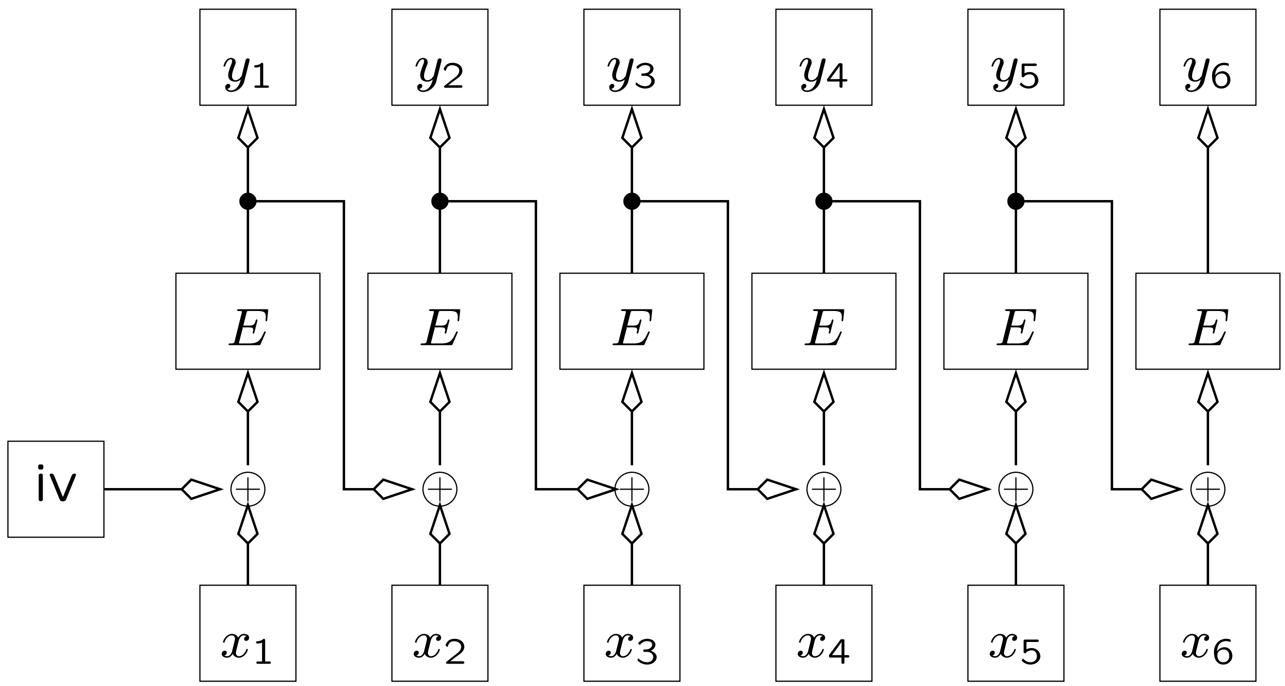
  - ⋆ Simplifies statistical analysis (see slides 30-31 of Lecture 1)
  - ⋆ Makes it possible to spot repetitions ("Attack!")
  - ⋆ Absolutely no authentication: swapping two ciphertext blocks corresponds to swapping to two plaintext blocks
  - ⋆ Most amusing: visual cryptanalysis

Low-Intelligence ECB Cryptanalysis



Give her a banana, and she will decrypt it...

CBC: Cipher Block Chaining



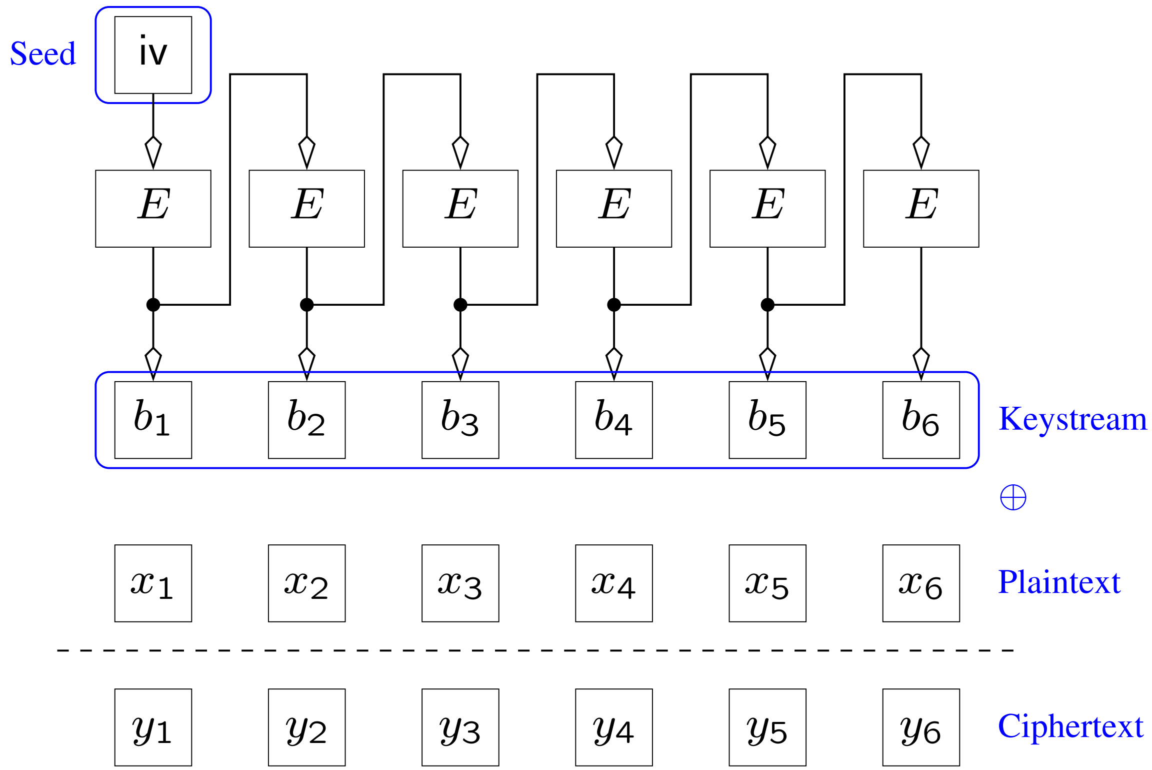
$y_i = E_K(y_{i-1} \oplus x_i)$, and iv is random (unpredictable)

Think about how to decrypt!

Why CBC might be a good mode?

- If iv is chosen randomly then the same message block will have different corresponding ciphertext block with a high probability
- Thus, no “recognition” and “banana” attacks
- If E is pseudorandom and iv is randomly chosen, then already the first ciphertext block looks random, and this randomness carries over to the next ciphertext blocks
- No authentication (still), but this is also not the goal

OFB: Output Feedback Mode



Stream cipher(!): First generate a key stream (b_i) from iv by using a block cipher, then compute $y_i = x_i \oplus b_i$

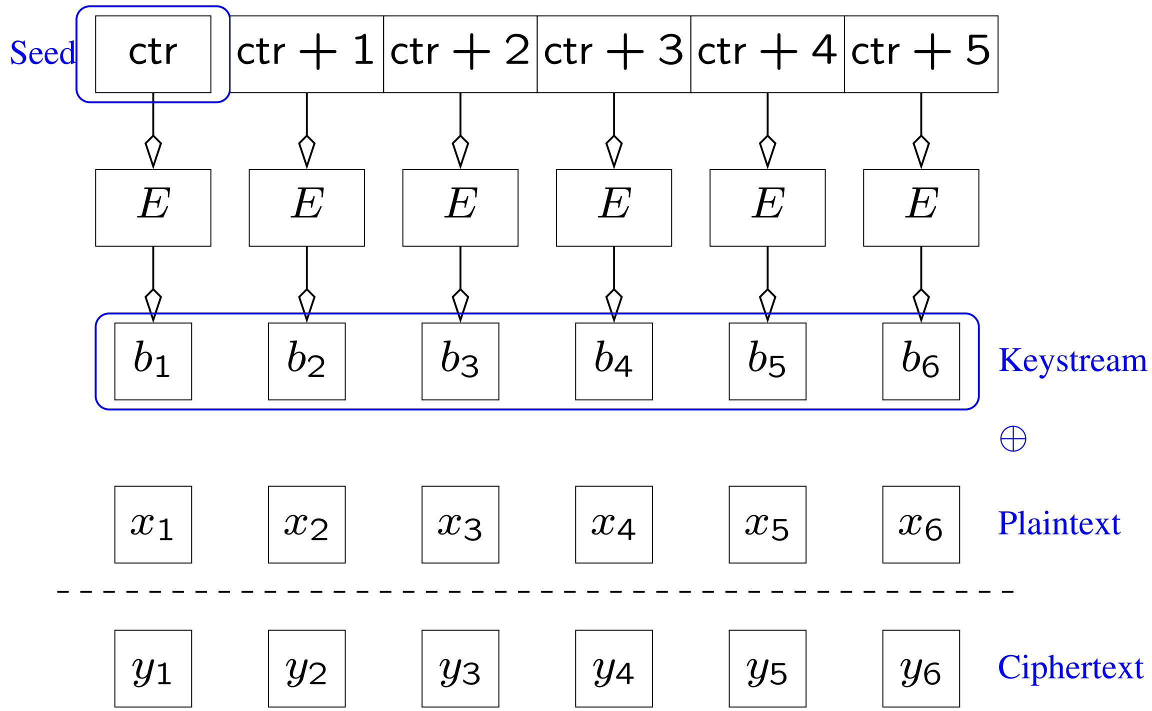
Why OFB is better than ECB, CBC?

- The same reasons as for CBC for being better than ECB

+ Keystream can be generated in advance
  - ⋆ “lunchtime” encryption
  - ⋆ Online one only must do XOR-s

+ Plaintext length can be arbitrary (in CBC, it must divide by n)

CTR: Counter Mode



As well as OFB, CTR mode is a stream cipher

Why CTR is better than ECB, CBC, OFB?

- The same reasons as for OFB for being better than ECB or CBC

+ Keystream generation can be parallelized
  - ⋆ Encryption and decryption can be fully parallelized

- With CTR you do not have to implement the decryption routine

- With CTR you can encrypt or decrypt in a random-access fashion

Note on authentication

- OFB and CBC modes are stream ciphers
- They have the same weakness as the common stream ciphers: by changing some ciphertext bits we introduce known changes to the plaintext bits
- Thus, weaker authentication
- However, this is sloppy thinking! Also CBC does not provide full authentication (it's only "somewhat" less manipulable)
- For full authentication, one must use proper authentication primitives

- This is not a goal of the (encryption) mode!

Note on error-correction

- If by some reason, a few bits of the ciphertext are changed, one would still like to be able to recover "most of the plaintext"

- Possible in OFB and CTR (as well as in common stream ciphers), since only the ith plaintext bit depends on the ith ciphertext bits. Not possible in CBC

- Sloppy thinking again in most situations. One can use proper error-correction codes to protect against induced errors

- This is not a goal of the (encryption) mode!

Block cipher modes: Goals

- Recall that a block cipher E is a family of permutations on short blocks. In particular, E_k is deterministic for every key

- This is not sufficient in real life: We need to encrypt arbitrary long messages, and we need to have randomness

  ⋆ Otherwise one can simply detect whether two plaintexts are equal ("banana attacks")

- Block cipher mode is an example of real-life cryptosystems

- We can encrypt long messages, and IV/$\mathtt{ctr}$ takes care of randomness

Block cipher modes: Security

- CTR, OFB and CBC modes are provably secure if used with provably secure ciphers
  - ⋆ Show why CTR together with shift cipher is weak!
- AES, DES, ... are not provably secure: they are only secure against known attacks
- Reduction works backwards: If $\neg B$ and $A \Rightarrow B$ then $\neg A$
- E.g.: an attack against CTR-AES also breaks AES

Provable security and reductionism

- To define, what is a primitive (block cipher, mode, ...), one must define its syntax and security.

- The definition of security is actually a definition of what constitutes an attack against this primitive.

- The primitive is said to be (t, ε)-secure if no algorithm that takes $\leq t$ steps can break the primitive with probability $\geq \varepsilon$

Reminder: Message authentication codes (MACs)

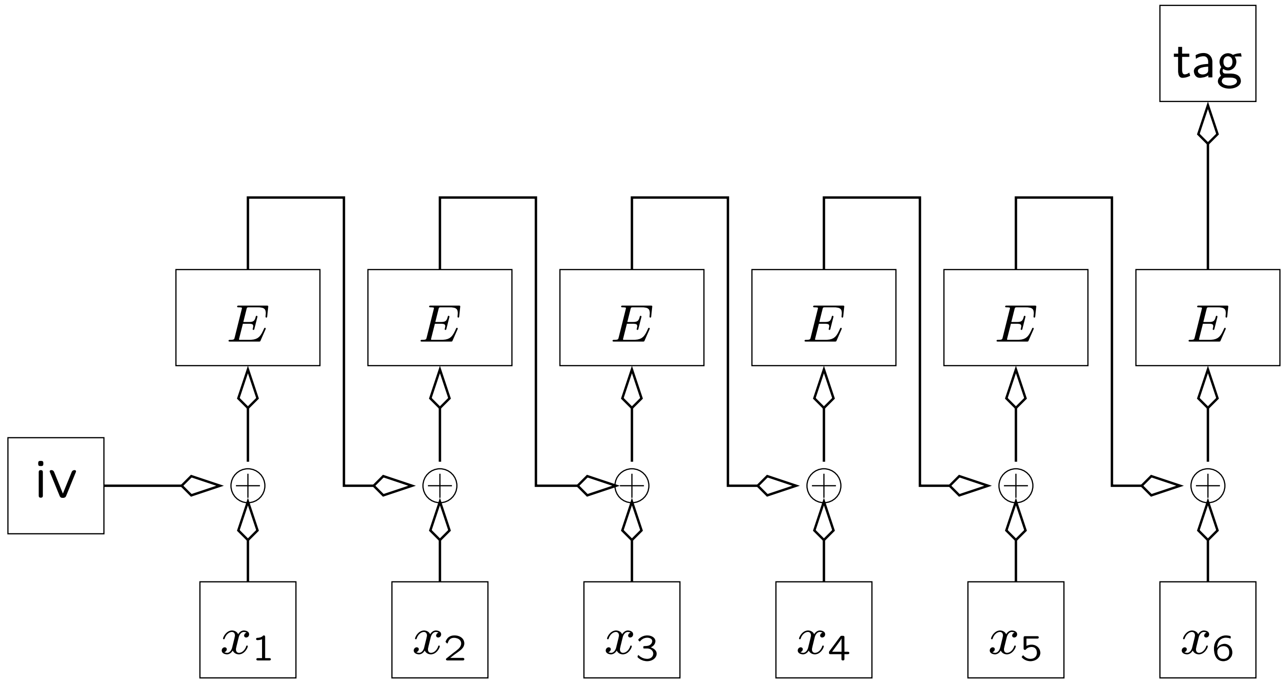
- Alice and Bob share a common private key K
- Symmetric authentication: Based on $\mathsf{MAC}_K(M)$, If Alice knows she has not sent M she knows that M was sent by Bob
- Provides no non-repudiation, but only data authentication
- Usually much-much faster than signature schemes

Security requirements

- It is computationally hard produce a MAC corresponding to a message for what the corresponding tag has not yet been seen, without knowledge of the private key

- We are not going into details, but formally this could be required to hold after chosen cipher-text etc attacks

Authentication mode: CBC MAC



As CBC, but only output the last block of ciphertext as the tag

Authentication mode: CBC MAC

- Block cipher with block length n
- Only secure if encrypting messages of fixed length mn
- Must use a different key for every m
- Recent constructions (Bellare, Rogaway, Iwata et alt) are more complicated but stay secure when MAC input has arbitrary length
- NB! One must use a different key for CBCMAC and for the used encryption mode

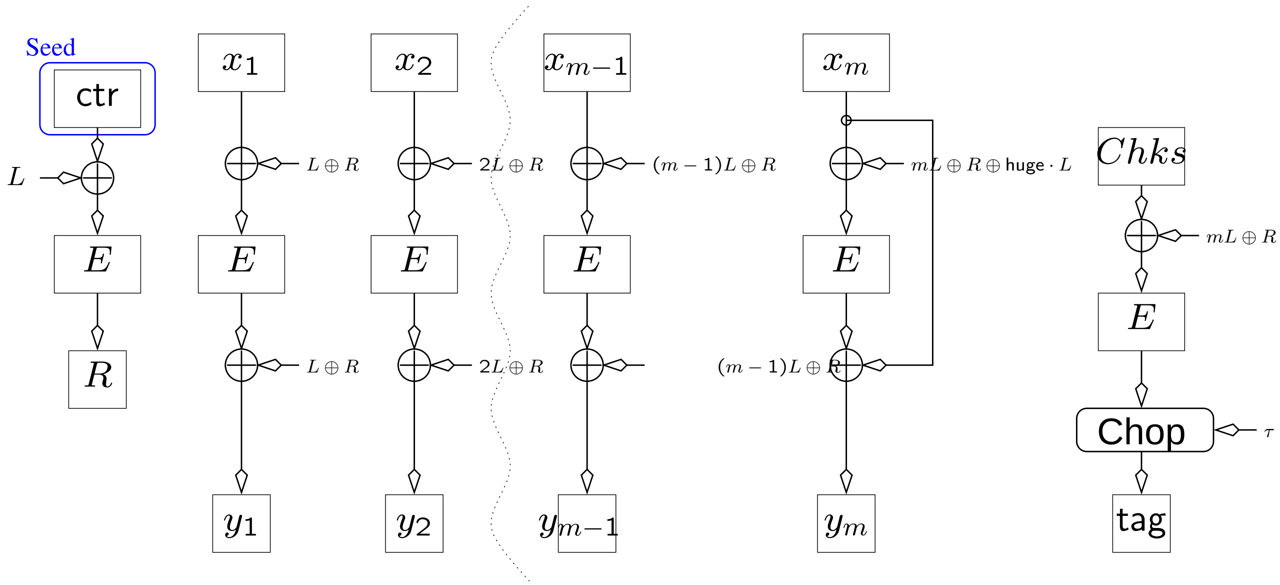
Quest for an Authenticated Encryption Mode

- CBC + CBCMAC, CTR + CBCMAC, . . . provide authentication and encryption, but
  - They need two different keys
  - They are twice slower than eiter CBC or CBCMAC by itself
- CBC with various checksums (wrong)
- PCBC in Kerberos (wrong)

Quest for an Authenticated Encryption Mode

- First correct solutions: IACBC, IAPM by Jutla (2000)
- Additional modes by Gligor, Donescu (2001)
- ... and OCB by Rogaway (2001)
- OCB is the most practical one, although difference in efficiency is not major
- All modes are covered by patents, and thus fast standarization cannot be expected

Authenticated encryption mode: OCB

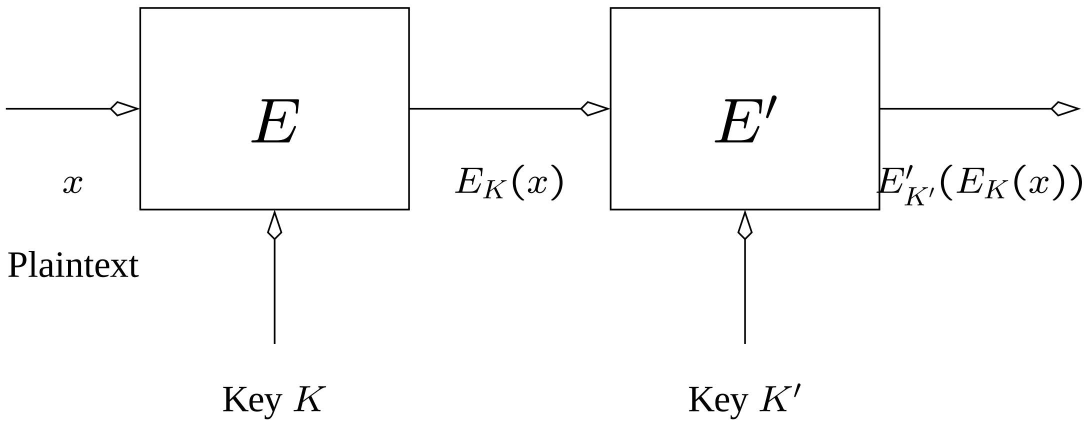


$$Chks = x_1 \oplus x_2 \oplus \cdots \oplus x_{m-1} \oplus (x_m||0^*) \oplus Pad$$
$$L = E_K(0)$$

As CBC, but only output the last block of ciphertext as the tag

Reminder: Product Ciphers

Idea: combine two weak ciphers to get a stronger cipher



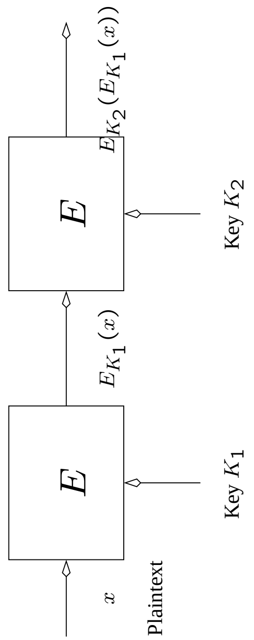
Tweak: Use the SAME cipher but with different keys

Yet another thing you can do with block ciphers

Multiple Encryption

- Idea: using the same cipher with possible different keys and multiple times could give increase in security
- In particular, possibly increases the effective key size
- Critical in the case of DES that has a key of $k = 56$ bits
- Does k-fold DES encryption with k different keys increase the effective key size k times?
- Not necessarily... even if E is a random permutation!

Double Encryption: Attack



Man-In-The-Middle Attack:

- Assume attacker has a few known plaintext-ciphertext pairs (x_i, y_i), where $y_i = E_{K_2}(E_{K_1}(x_i))$
- Do for every possible key K:
  - ⋆ Let $A[K] := (K, E_K(x_1))$, and $B[K] := (K, D_K(y_1))$
- Sort arrays A and B on the values of the second coordinates
- Search both arrays for rows that match in second coordinate, (K'_1, z), $(K'2_, z)$. For every such row we know that $y_1 = E_{K'_2}(E_{K'_1}(x_1))$
  - ⋆ To eliminate wrong keys, test for every such (K'_1, K'_2) that $y_i = E_{K'_2}(E_{K'_1}(x_i))$ for $i = 2 \ldots$

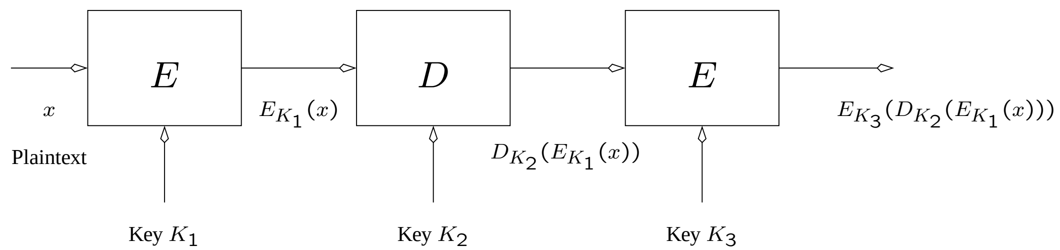
Double Encryption: Attack Analysis

- With an ideal cipher with DES's parameters ($k = 56$, $n = 64$), only every $1/256$th plaintext is present in table A, and in table B. Only every $1/2^{16}$th plaintext (2^{48} plaintexts) is present in both tables. Therefore, there are 2^{48} candidate keys (K_1', K_2')

- For every candidate key (K_1', K_2'), $\Pr[y_i = E_{K_2'}(E_{K_1'}(x_i))] = 2^{-64}$ for $i > 1$

- Thus testing for a single additional pair (x_2, y_2) should be sufficient with a h.p. to pick one a single candidate key

Triple Encryption "EDE"



Two common modes: 3EDE, 2EDE. In 3EDE, all keys are different. In 2EDE, $K_3 = K_1$. Best known are 3EDE-DES and 2EDE-DES.

Why 3EDE?

- Best attack: requires 4 KPC (known plain/ciphertext) pairs, 2^{112} time units and 2^{56} memory. Impractical!

- Security: 2EE-DES can be broken in $\approx 2^{56}$ space and $\approx 56 \cdot 2^{56}$ time. Practical

- Security: 2EDE-DES can be broken in 2^x KPC pairs, 2^{120-x} time and 2^x words of memory. "Almost practical" with $x \approx 50$

- Efficiency: applying more than 3 rounds gives additional security (but why?), and makes cipher less secure

Why 3EDE?

- EDE is better than EEE since fixing $K := K_1 = K_2 = K_3$ results in $y = E_K(D_K(E_K(x))) = E_K(x)$ (usual DES)

- 3EDE-DES (commonly known as 3DES) can be seen as a new cipher that is

  ⋆ 168-bit key

  ⋆ three times slower than DES,

  ⋆ with effective key length ≈ 112

  ⋆ reusing DES's hardware/software implementations

  ⋆ 3DES's security can be reduced to the security of DES

DESX

- Assume we have two keys, a 64-bit key K_1 and 56-bit key K_2
- Define $\mathsf{DESX}_{K_1,K_2}(x) := \mathsf{DESX}_{K_1}(x \oplus K_2) \oplus K_2$
- Exhaustive key search: 2^{120} time units
- Provable security assuming DES is secure
- Some loss due to differential/linear cryptanalysis. Breakable in $\approx 2^{89}$ steps. Still impractical
- Only marginally slower than DES