Lecture 3

Tuesday, January 31, 2017 1:59 PM

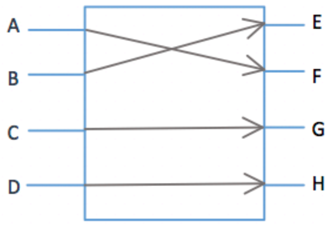
A *Block Cipher* is approximating a pseudorandom permutation. What does this mean ?

- Permutation = changing the order of something.
    - How many permutations on n-bit inputs ?
        - $(2^n)!$
        - Need $\log_2(2^n!)$ bits to represent what was chosen
- Random permutation = take all possible permutations that we can have and randomly pick one
    - Different than taking every input and choosing a random output for that input
        - This is not a permutation because we can have duplicates
        - Why is this a problem? Because encryption needs to satisfy the correctness property
    - Remember:

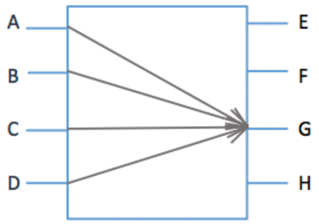
Correctness means $Dec_k(Enc_k(m)) = m$

When decrypting a message with the same key used to encrypt it, get the original message as the output

- We don't use random permutations to do encryption because it would not satisfy this property of correctness
- If something is only a function, it would not be able to have correctness.



(A) Permutation and Function



(B) Function but not Permutation

(A) can be used to encrypt and decrypt because it is both a permutation and a function. Every output corresponds to an input

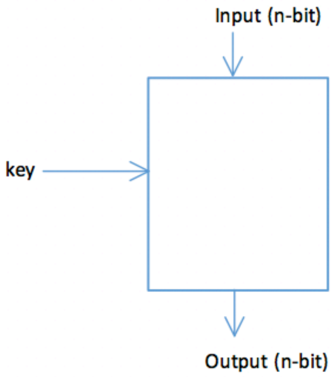
(B) is not a valid encryption scheme because it can't be used to decrypt the message. A unique plaintext does not correspond to a cipher text

- Pseudorandom permutation (PRP) = permutation that looks random but actually isn't

---

AES = Advanced Encryption Standard

- A Pseudorandom Permutation
- Algorithm specified by NIST, the algorithm itself Is public



Takes in a key (128-bits for example) and a 128-bit input and produces a 128-bit output

- Number of bits of input = number of bits of output

- 2^128 is considered a big number (128-bit security level); generally a difficult size to crack; can't brute force this

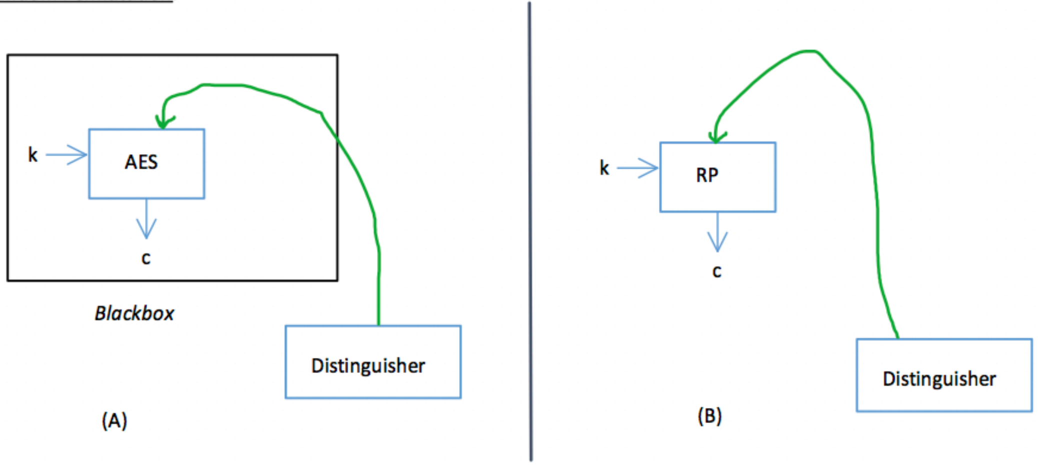
- Question is: How can we break AES?
  - Start with understanding the encryption scheme:

    $c = PRP_k(plaintext)$ or $Enc_k(m) = PRP_k(m)$

  - Let's start with this: given that the key has 4-bits how would you try to break it? *Brute force/trivially break the encryption scheme*
    - Try all possible 4 bit keys and plug into the inverse PRP
    - For all possible keys ($2^4 = 16$), compute $PRP^{-1}_k(c)$ and stop when a 'valid looking' message is found
- Essentially, we want to make sure that trying the brute force method for an n-bit key is close to impossible
  - aka why we use keys that are 128 bits long
  - ★ However, If lengthen the keys too much, it slows down PRP's
  - While there are better attacks, there exists these "easy ones".
  - Your encryption scheme should be able to withstand the easy attacks

---

PRP = Pseudorandom Permutation:



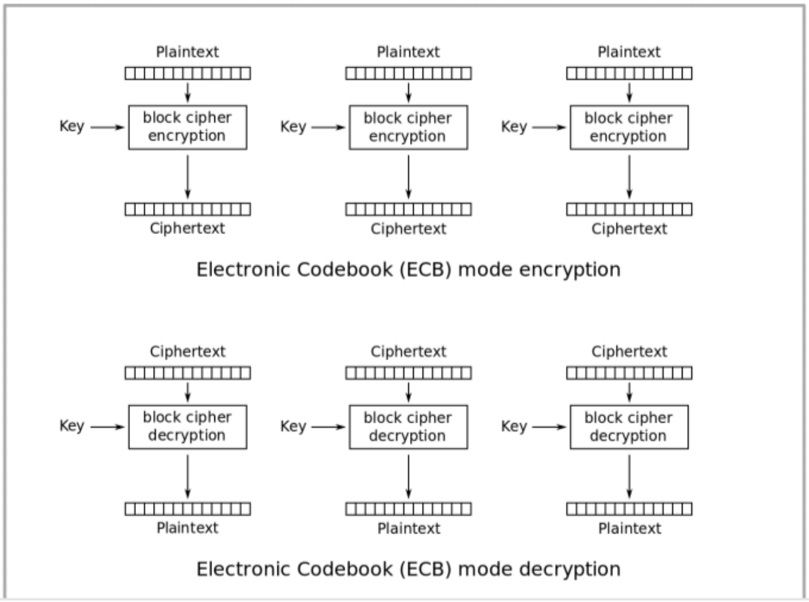
The person should not be able to distinguish between AES and a Random Permutation
- *Key is unknown*

Protocols are given that are used to exchange keys

[Side note to google later: Crypto competitions for AES (performance and security) Issue? NIST and NSA]

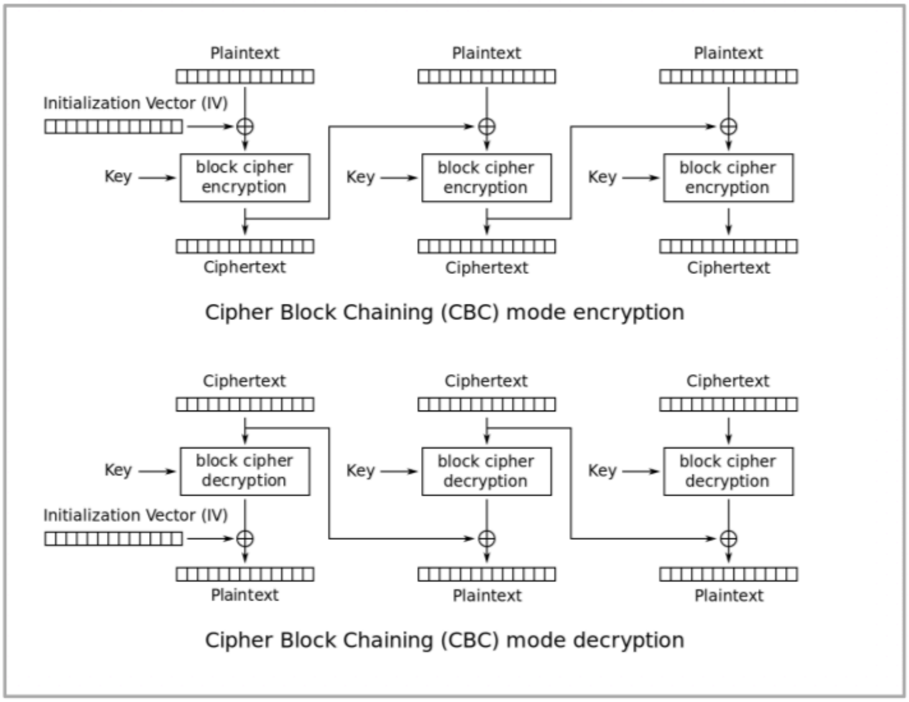
What happens if have X n-bit messages to send? Chain PRP's together

- Electronic Codebook (ECB)
  - Break it into multiple n-bit blocks and run the block cipher on each block using the same key for each
  - But, this is horribly insecure
    - Frequency analysis



- Cipher Block Chaining (CBC)
    1. To start: break up plaintext into blocks, have an Initialization Vector and xor them
    2. Then take key and the result and feed that into Block Cipher and get output
    3. Then use that result as the "initialization" vector for the next block

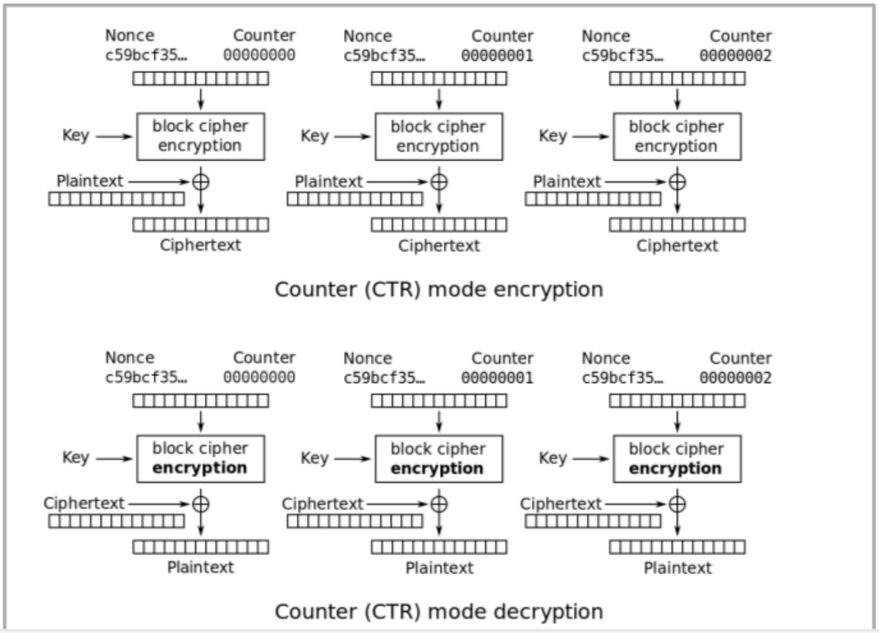
    - Is it vulnerable to length extension attack?
        - Not vulnerable to length extension because don't know the key and CBC mode needs the key
    - Does Bob need the IV?
        - Yes, he wouldn’t be able to decrypt it without it
    - IV has to be random but doesn't have to be secret (Not sure I understand why, so need to ask?)
    - How is the IV attained?
        - IV's are sent as part of ciphertext



Cipher Block Chaining (CBC) mode encryption

Cipher Block Chaining (CBC) mode decryption

- Counter Mode
    - Nonce = fresh random number (*random* being debatable because sometimes It's not random)
    - Very similar to the one-time pad
    - The counter makes it so that the keystream doesn't repeat
    - It's a parallel algorithm meaning could do every piece by itself



Counter (CTR) mode encryption

Counter (CTR) mode decryption