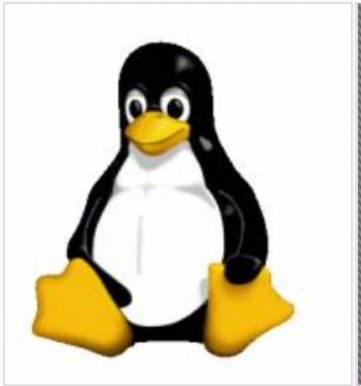
CAS CS 357

In-class Note 10

1. Several important crypto properties
2. Confidentiality 🡪 prevents man in the middle from read/accessing information
3. Integrity 🡪 prevents man in the middle from altering communications
4. Integrity does not imply confidentiality (you can still edit communications without knowing the information)
5. Confidentiality does not imply integrity (you can read the communications without changing them)
6. Block Cipher Review
7. Consisted of plaintext with n its, ciphertext with n bits, key with k bits
8. Block cipher is a permutation (every input corresponds to one output and every output corresponds to one input
9. The key decides which permutation pattern to use
10. Therefore, if the block size is tiny, the adversary can determine permutations easily by trying to give all possible combinations (try all the mapping out)
11. Trying all possible keys and mapping out every combination is brute-force attack
12. Therefore, the length of the block size and the key should be large
13. Computational infeasibility (“brute force”)
14. Schemes like one-time pad cannot be broken, even by an attacker that performs a brute-force attack, trying all possible keys
15. However, all future schemes that we will see can indeed be broken by such an attack
16. It is easy to point out that for a scheme with ƛ-bit keys:
17. The most direction computation procedure would be for the enemy to try all 2^ ƛ possible keys, one by one. Obviously, this is easily made impractical for all enemy by simply choosing ƛ large enough
18. We call ƛ the security parameter of the scheme. It is like a knob that allows the user to tune the security to any desired level
19. Increasing ƛ makes the difficulty of a brute-force attack grow exponentially fast. Ideally, when using ƛ-it keys, every attack will have difficulty roughly 2^ ƛ
20. However, sometimes faster attacks are inevitable: many schemes with ƛ-bit keys have attacks that cost only 2^ (ƛ/2)
21. For example, the birthday paradox requires ƛ/2 🡪 important that in these steps, we set ƛ higher than usual (such as 256, so that division by 2 will lead to 128 🡪 2^128 is a large number)
22. Important to note that attacks exist so that 2^128 can change to 2^80 (attacks that reduce security) and if these steps continue so that the number becomes too low, problems may occur
23. Polynomial time adversary
24. A program runs in polynomial time if there exists a constant c > 0 such that for all sufficiently long input strings x, the program stops after no more than O(lxl^c) steps
25. In other words, ƛ^100, 100\* ƛ, and etc. are all polynomial
26. Exponential such as 2^ ƛ is not polynomial time
27. Definition of CPA Secure Encryption
28. Suppose the key length is ƛ. The encryption scheme is secure if for every polynomial time adversary A, then Pr[Adv outputs \* = b] = 0.5 + negligible(ƛ)
29. Adversary can do polynomial time to try brute force attack but not exponential
30. Making 2^ ƛ very big that any computational device cannot run
31. As long as adversary is running in polynomial time, it is difficult to guess the key assuming that ƛ is high
32. Note that adversary CAN ask the oracle for the encryption of m0 and m1
33. Negligible
34. Negligible 🡪 ½ ^ (constant\* ƛ )
35. When ƛ is high, negligible is low
36. ECB mode 🡪 not CPA secure
37. ECB mode 🡪 substitution from one block to another block
38. EX) Penguin

🡪 🡪 🡪 🡪

1. How to prove not CPA secure 🡪 there is an adversary, does run in polynomial time, and guess the bit more than half of the time
2. Method:

Adversary 🡪 game master (prior to the game)

Adversary chooses m0 and m1 that does not equal to each other

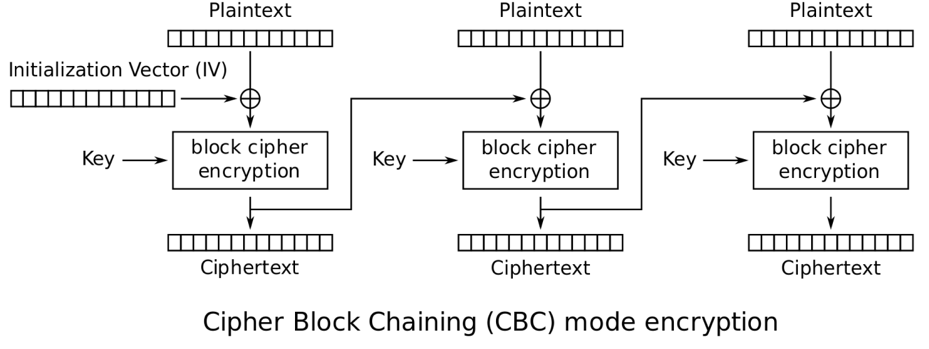
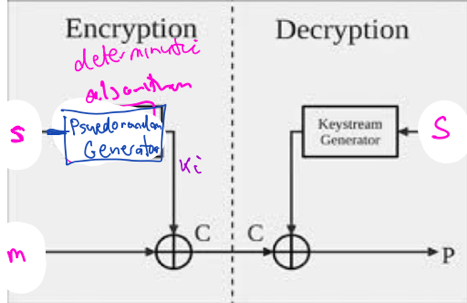
Adversary asks m0 to gamemaster and receives c0

Adversary asks m1 to gamemaster and receives c1

Game Starts:

Adversary gives m0, m1 to gamemaster and receives c\*

B = 0 if c\* = c0 or B=1 otherwise

1. Such method is possible because we know that pixels in the penguin are repeated within each other (the same color), allowing us send repetition of the same value, knowing that the key will the same all the time
2. CBC mode
3. 
4. CBC is CPA secure
5. CBC works by having an initialization vector with the length equal to plaintext that consists of random bits
6. The initialization vector is XORed with plaintext and then the block cipher runs to create ciphertext
7. The ciphertext is used as the initialization vector and is XORed with the next plaintext to produce the second ciphertext
8. This step is repeated until all plaintexts are used
9. Does stream cipher provide integrity (making sure that adversary cannot alter information)?
10. Stream Cipher 🡪 CPA secure
11. 
12. Stream cipher uses pseudorandom generator to produce keys 🡪 k1, k2, kn
13. These keys are XORed with messages m1, m2, mn to produce c1, c2 ,cn
14. Possible way to change the code 🡪 flipping the bit of c4 (for example) so that the result is the opposite of intended (note that the flipping of bits do not require confidentiality (you do not know the information you are changing))
15. Ensure integrity and authenticity by Message Authentication Code (MAC)
16. Since we are only targeting integrity and authenticity, we assume that confidentiality is achieved prior to MAC
17. Authenticity is property that information originated from its purported source
18. Sender (A) sends message to Receiver (B) and they share a symmetric key k
19. Tag T = MACk(m)
20. Sender sends the message m, associated with the tag t
21. Receiver does verification of the tag t using the key k
22. Verk(m,t) = 0 or 1 where 1 if valid tags, and 0 is not valid tag
23. The idea is that adversary cannot come up with a valid tag and therefore cannot change the message between the sender and receiver unless he knows the key
24. Correctness: Verk(MACk(m),m) = 1 🡪 validly MAC’d message verifies correctly
25. Security: 🡪 the adversary cannot forge the MAC without knowing the key k