CAS CS 357

In Class Note 8

1. How do we define a symmetric encryption scheme

Plaintext m key k cipher text ck=Enck(m)

Where Enck(m) = c and Deck(c) = m’

1. Goal is for A and B to send messages in a way that only they can read them
2. Symmetric 🡪 key shard by A and B are the same (that is used to read message)
3. A sends Enck(m) = cb to B where c is the plain text
4. B decrypts the c and gets back the message (Deck(m)=m)
5. The Caesar Cipher and breaking it
6. Example

F XQB QMB ZMBBPB

1. Step 1: find the most familiar letter in English: e and plug it in

F XQB QMB ZMBBPB (B to E)

F XQE QME ZMEEPE

1. Step 2: English letter that can come alone: A or I

F XQE QME ZMEEPE (F to A/I)

A/I XQE QME ZMEEPE

1. Step 3: English word ends with E that is often used: the

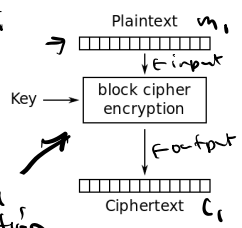
A/I XQE QME ZMEEPE (XQ or QM equals TH🡪in this case Q=T, M=H)

A/I XQE THE ZHEEPE

1. Guess the remaining words

A/I XQE THE ZHEEPE

I ATE THE CHEESE

1. What did our attack do?
2. Plaintext recovery and partial key recovery (these should not be allowed for perfect security)
3. Partial key recovery 🡪 Found out that I maps to F, B maps to E (etc.), recovered part of the key
4. Intuition – Electronic code book mode (ECB mode)
5. 
6. In this example, 12 bit inputs go through key and outputs 12 bit
7. Famous example of penguin being encrypted with key
8. 
9. 
10. Because we are using the same key, the encrypted version of penguin is somewhat similar to the image of penguin. All the black pixels in penguin go through the same key and produces the same output. Similarly, other pixels of same color go through the same key and produce the same output in a way that we can guess that the encrypted image is somewhat a penguin
11. This arises a problem since we have plaintext recovery
12. In order to reach perfect security, the encrypted penguin should be random dots that other people do not know what the picture represents 🡪 every dot and pixel should be independent from each other
13. What the adversary does (attack severity)
14. Key recovery: at the end of the attack, the adversary produces entire key k
15. Full plaintext recovery: adversary has some ciphertext that outputs m, such that Deck(c) = m
16. Recovery of a single bit of the plaintext: adversary has some ciphertext that outputs b such that b is the ith bit or m1=Deck(c1)
17. Distinguishing attack: adversary gets to pick two messages that are not identicall and outputs them to gamemaster. Gamemaster, who knows the key, flips a coin and choose either of the two messages and then encrypt it (Enck(mb)) = c and gives it back to adversary. Adversary’s job is to produce b’ where b’=b. The adversary does not know which of the two messages he/she received, therefore, the expected probability of winning is 0.5. The adversary wins if Pr[b=b’] > ½
18. Attack Severity:

P = There exists adversary that can perform key recovery attack

Q = There exists adversary that can perform a recovery of one bit of the plaintext

P implies Q ( P 🡪 Q): if adversary can recover the whole text, he can recover one bit

Not Q implies not P (not Q 🡪 not P): if adversary cannot recover one bit of the plaintext, the adversary cannot perform key recovery attack

In other words, if the scheme is secure against recovery of one bit of plaintext, it is secure against key recovery (This implies that if we block the simplest attack, we can block more difficult attacks)

Similarly, security against distinguishing attack implies security against all the other attacks we have seen

1. What adversary knows
2. What adversary knows: Ciphertext Only Attacks

Adversary has access to ciphertext prior to attack. All the adversary knows is a ciphertext

1. What adversary knows: Known Plaintext Attacks

In addition to ciphertexts, the adversary knows plaintexts (knows the text and the encrypted version of the text – ciphertext)

1. What the adversary knows: Chosen Plaintext Attacks

Allows the adversary to choose pairs of plaintext and ciphertext until he thinks he is ready to play the distinguish attack game