POIS-8

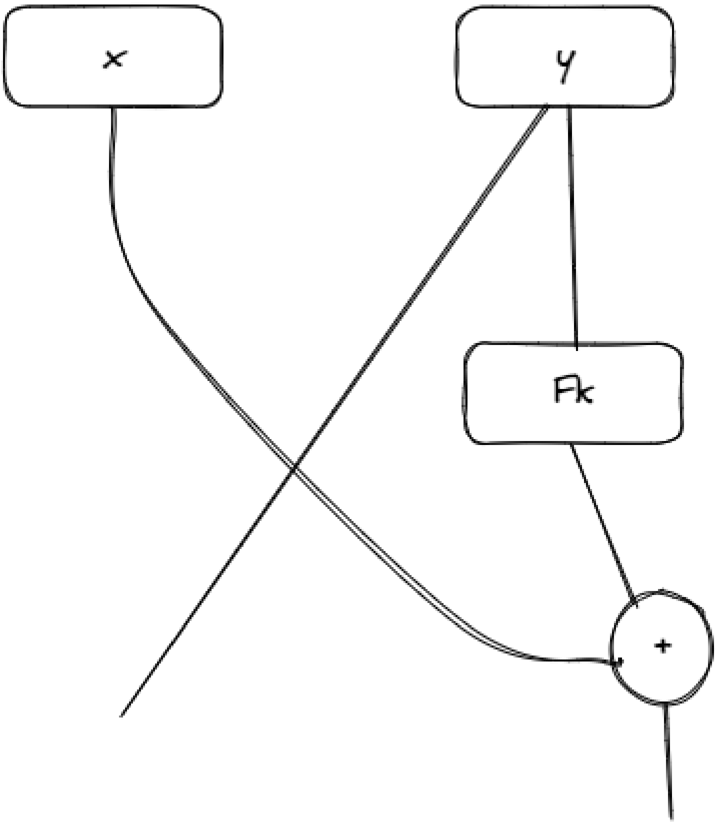
31/01/2023 (Tue)

PseudoRandom Permutations

Pseudo Random Permutations are basically pseudorandom functions, which are just one-one. Note that these are invertible.

Converting Pseudo Random Function to a Pseudo Random Permutation

Fiestel Network



This network converts a pair (x, y) to $(F_k(y) \otimes x, y)$. Notice that the function is invertible, since, we can get back y and x (we can get back x by passing y through the PRF, and then XORing with the first output).

Turns out , you need to perform this operation atleast 3 times, and at most 4 times, in order to create a pseudo random permutation. This gives rise to tools such as S-boxes.

Data-Integrity

Say, two parties are communicating A and B. A sends a message to m to B, and B recieves m'. How do we ensure that $m = m'$. Note that the eavesdropper, is listening in to the conversation, therefore we can't perform any backward communication.

The idea is that we attach a tag t to the message, making it a pair $< m, t >$, and send that over the channel, and ensure that the problem of creating a valid m, t pair is extremely hard. Any change in the message, should also change the bit as well.

MACs

Assume that we have a MAC, which can generate these tags
$MAC_k(m) = t$
Also assume that we have a verifier, which can verify whether a given message-tag pair is valid or not
$Vrf_k(m, t) = Yes/No$

Assume that the adversary has access to the MAC oracle, and can generate message tag pairs on command, however, assume that he perform Q such queries, and obtains multiple such message-tag pairs, the job of the adversary would be to create a message-tag pair from scratch. without the use of the oracle. For data integrity we have

$$P(Vrf_k(m, t) = Yes \mid m \notin Q) \leq negl(n)$$

Meaning that it is almost impossible for the adversary to create a message tag pair which passes the verifier, without it being a member of the set of queries to the oracle.

Replay Attacks, and why they are not that important

Replay Attack is essentially sending the same message over and over again, could be used in transaction frauds, since even if the adversary cannot change the message, he can replay it over and over again, this can be treated by just adding the timestamp to the message, and does not need any changes in the security layer.

Example of Basic-MAC which doesn't work

Assume that you have a small message block m_i where i ranges from 1 to k
$|m_i| = n$
And your full message m, is just a combination of these small message blocks.
$m = m_1 m_2 m_3 \ldots m_k$
We define our tag in the following way
$t = \otimes_k F_k(m)$ (Basically XOR all messages after passing through the pseudo random function)

This wouldn't work, there are two possible attacks
i) A permutation attack is possible, notice that $m_1 m_2 \ldots m_k$ has the same tag, as any other permutation of it
ii) Also notice that if two messages, have the same tag, then their XOR message also has the same tag.

Variable length MAC

Check from book (After page 126)

CBC MAC

Check from book (After page 126)