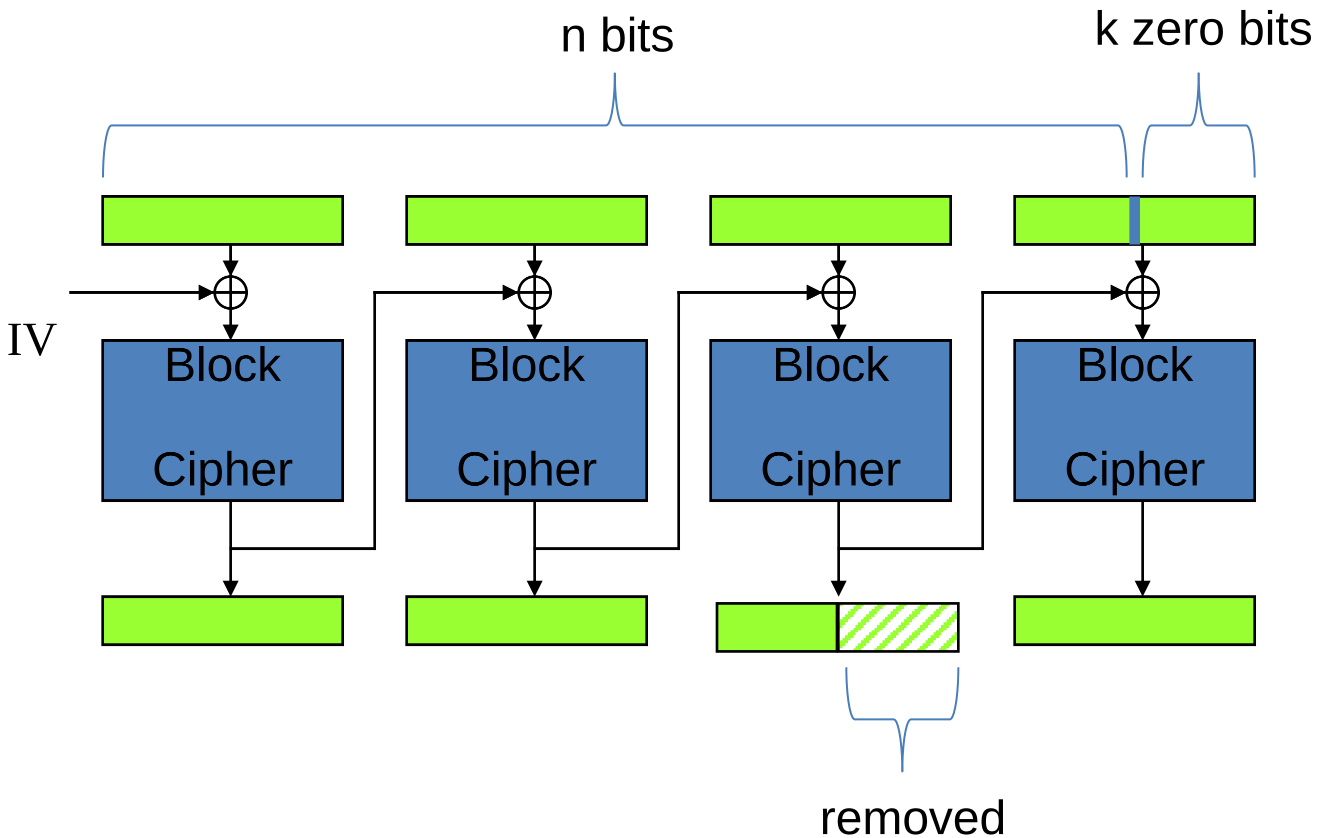
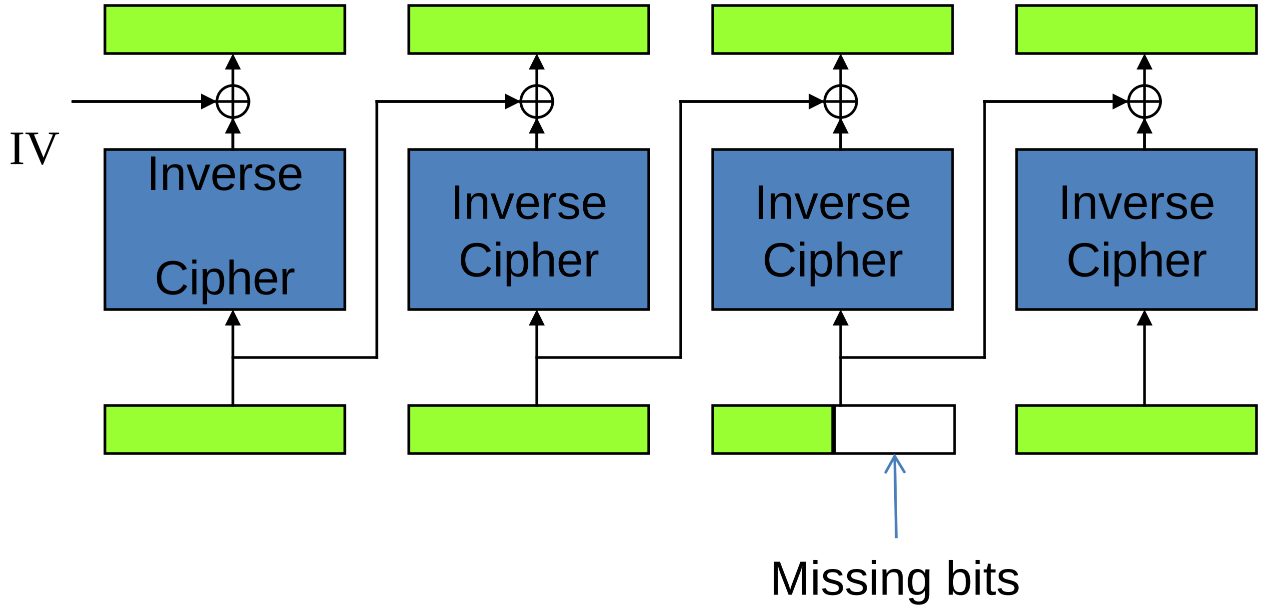
Quiz0926

Problem 1


n bits
k zero bits
IV
Block Cipher
Block Cipher
Block Cipher
Block Cipher
removed

Problem 1


IV
Inverse Cipher
Inverse Cipher
Inverse Cipher
Inverse Cipher
Missing bits

Problem 1

- Let b $= \left\lceil \frac{n}{m} \right\rceil$ be the number of blocks.
- Plaintext $P_0, P_1, \ldots, P_b$, ciphertext $C_0, C_1, \ldots, C_b$.
- We care about C_{b-1}, C_b, P_{b-1} and P_b.
- We know k, the number of bits removed from the penultimate block, since $k = m - (n \bmod m)$.
- Recall that for CBC decryption, we have plaintext block

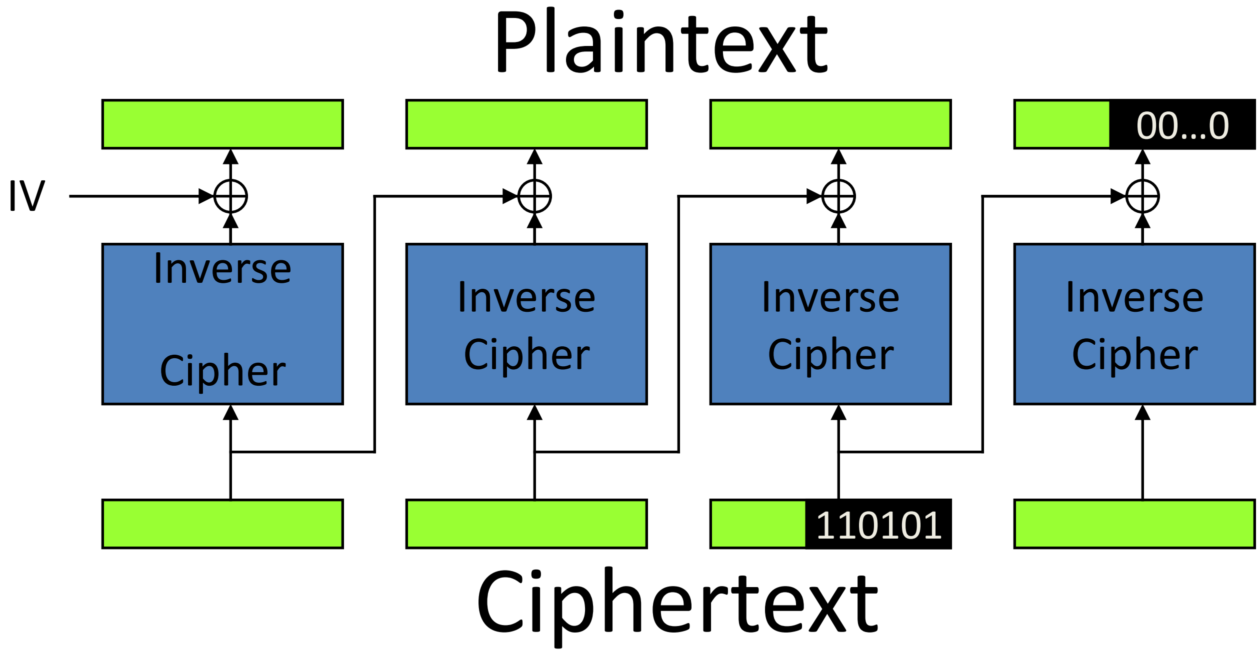
$$P_i = \text{Decrypt}(K, C_i) \otimes C_{i-i}$$

Problem 1

$$P_i = \text{Decrypt}(K, C_i) \otimes C_{i-i}$$

1. Compute $X_b = \text{Decrypt}(K, C_b)$ (intermediate value of final block)
2. We also know $X_b = P_b \, XOR \, C_{b-1}$ if we have all the bits in C_b.
3. Finally, we know the last k bits of P_b are 0 (pad).
4. So for each of the padding bits $P_{b,m-k+1}, \ldots, P_{b,m}$ we have $X_{b,i} = P_{b,i}$ XOR $C_{b-1,i}$ for $i = m - k + 1, \ldots, m$
5. Since $P_{b,i} = 0$, then $X_{b,i} = C_{b-1,i}$

Problem 1: Ciphertext Stealing



Problem 2

- Decrypt a k-block segment in the middle of a long CBC-encrypted ciphertext.
  - What is the minimum number of blocks of ciphertext that need to be decrypted?
  - Which blocks do you need to decrypt and how will you decrypt them?

Problem 2

- In CBC decryption, we have plaintext block
  $$P_i = \text{Decrypt}(K, C_i) \otimes C_{i-i}$$
- NOTE: Boundary case "C_{-1}" $=$ IV.
- Each plaintext block we want requires one decryption of the corresponding plaintext plus one XOR.
- So the minimum number of ciphertext blocks to be decrypted is k.
- If you want plaintext blocks $P_i, P_{i+1}, \ldots, P_{i+k-1}$, then you need ciphertext blocks $C_{i-1}, C_i, C_{i+1}, \ldots, C_{i+k-1}$.
  - If $i = 0$, instead of C_{i-1} you need the IV.

Problem 3

- H is a Merkle-Damgård hash function w/ compression function F. Black box takes inputs IV and y and outputs an x such that $F(IV, x) = y$.
- Show how by using the black box at most $2k$ times you can find a set of 2^k messages that all have the same hash value when input into the full hash function H.

Problem 3 – Solution 1

- Basic idea: find pairs of messages x_i, x_i' satisfying

$$F(IV_i, x_i) = F(IV_i, x_i') = y_i, i \ = \ 1, .., k$$
$$y_i = IV_{i+1}$$
$$IV_1 = IV$$

- Start at the end. Choose a random target output value y_k and a random input value $y_{k-1} = IV_k$. Call the black box twice with IV_k, y_k to generate x_k, x_k'.
- Now move back a block. We have y_{k-1}, choose random $IV_{k-1} = y_{k-2}$. Run the box twice, get x_{k-1}, x_{k-1}'.

Problem 3 – Solution 1

- We now have 4 two-block messages that hash to the same value when F is the compression function:
$$x_{k-1}x_k, x_{k-1}x'_k, x'_{k-1}x_k, x'_{k-1}x'_k$$
- Repeat this procedure k times and you'll have made $2k$ calls to the black box to generate k pairs x_i, x'_i.
- To generate 2^k messages that hash to the same value, make k-block messages where the ith block is either x_i or x'_i. Two choices per block, k blocks == 2^k.

Problem 3 – Solution 2

- The "fixed point" solution
- Choose a fixed value for IV. Now call the black box to find an x such that $F(IV, x) = IV$.
- Concatenate x as many times as you want, the hash will still be IV. So to get 2^k messages:
- $x, xx, xxx, xxxx, \ldots, xxx \ldots xxx$ (2^k total times)

Problem 4

- $G(x) = H(x) \parallel H'(x)$, $H(x)$ and $H'(x)$ are hash functions with n-bit outputs, so $G(x)$ has $2n$-bit outputs.
- Normally, with a birthday attack we would expect to have to generate $2^{2n/2} = 2^n$ messages to find a collision.
- However, $H(x)$ is badly broken (as in Prob. 3) so assume we can generate $2^{n/2}$ messages that all have the same hash value in $H(x)$.

Problem 4

- Now compute $H'(x)$ for each of the $2^{n/2}$ that have the same hash value in $H(x)$.
- By the birthday attack we expect to find a collision from those $2^{n/2}$ messages.

Problem 4

- Was it a good idea to construct $G(x) = H(x) \parallel H'(x)$?

Problem 4

- Was it a good idea to construct $G(x) = H(x) \parallel H'(x)$?
- Well, it depends…

Problem 4

- Was it a good idea to construct $G(x) = H(x) \parallel H'(x)$?

- Well, it depends…
- YES: At the cost of computing two hashes vs. one, you get resistance if one of H, H' breaks.

Problem 4

- Was it a good idea to construct $G(x) = H(x) \parallel H'(x)$?

- Well, it depends…
- YES: At the cost of computing two hashes vs. one, you get resistance if one of H, H' breaks, but…
- NO: However, $G(x)$ doesn't have the security margin you'd expect of a $2n$-bit hash function. It's only as strong as the better of its two components

Problem 5

- Alice → Bob: $m =$ "please pay the bearer \$1", $H(k, m)$.
- m is an exact multiple of H's block size (so you don't need to do any padding).

- What can Bob do?

Problem 5

- Note that k is only an input to the first application of $H's$ compression function (e.g. it's the IV to the hash of the first block of m)
- Bob can **append** data to m, create $m' = m$ || ",000,000", and compute $H(k, m')$ from $H(k, m)$.