Alin Tomescu, CSE408
Tuesday, February 22nd, Lecture #7

Block ciphers and their operation modes

RoR security

Ideal world:

- Whenever Alice sends a message to Bob, she sends him $E_k(m)$ and she also sends $E_k(\$m)$ to Eve, where \$ replaces m with random bits.
    - That's to say Eve will always know that a message of a particular length was sent.

Real world:

- Whenever Alice sends a message $E_k(m)$ to Bob, Eve gets a copy of $E_k(m)$.
- Even can also provide a message m to Alice for her to send it encrypted as $E_k(m)$ to Bob.
    - Eve can now see $E_k(m)$.

In real or random security the goal is for the encryption scheme to make it impossible for Eve to distinguish whether she's in the ideal or in the real world.

Definition: An encryption scheme is a family of functions $E_k: \{0,1\}^* \rightarrow \{0,1\}^*$ is (t, q, ε)-real-or-random secure if $\forall$ algorithms A running in time t and making at most q bits of queries, then the advantage of A is:

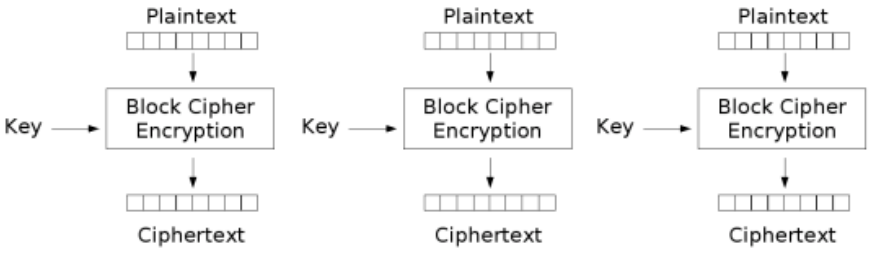
$$Adv\ A = \left|\Pr[A^{E_k} = 0] - \Pr\left[A^{E_k \circ \$} = 0\right]\right| \leq \varepsilon$$

Modes of operation for block-ciphers

A **mode of operation** takes a block-cipher (a PRF or a PRP usually) and converts it into an encryption scheme.

Electronic codebook mode (ECB)

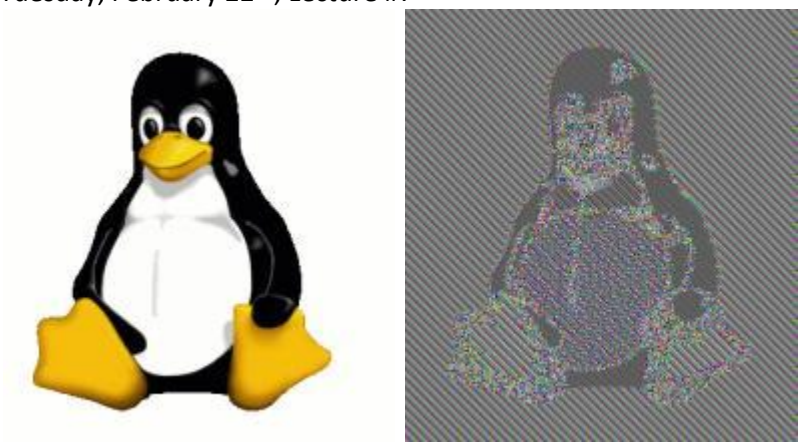
Last time we used a black-box F_k that took blocks of our plaintext P and converted them to blocks of ciphertext C. This was bad because identical plaintext blocks got encrypted to the identical ciphertext blocks.



Electronic Codebook (ECB) mode encryption

The end result would be that an attacker could distinguish patterns in the ciphertext when ECB mode is used. You can see very good example of this below:

Alin Tomescu, CSE408
Tuesday, February 22nd, Lecture #7

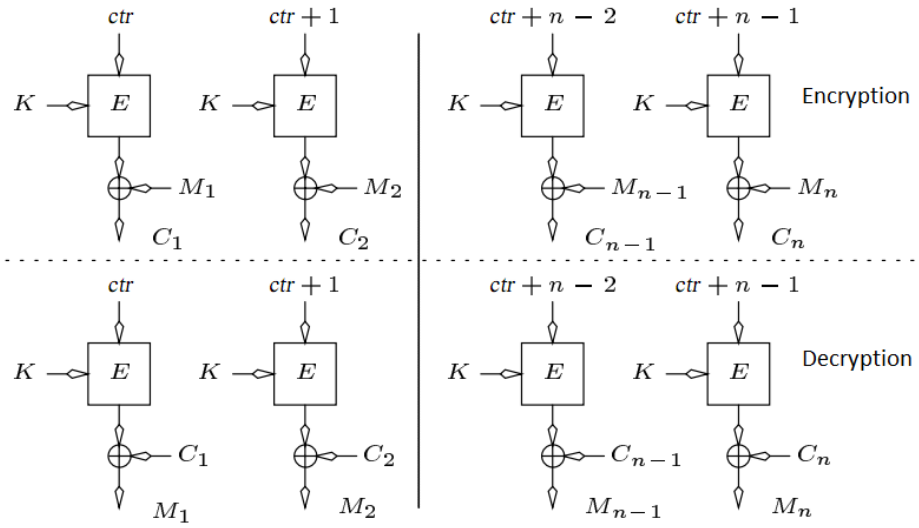


You couldn't really rely on ECB for encrypting a bitmap, for instance, since the encrypted bitmap would reveal **a lot of information** about the original bitmap.

Counter mode (CTR)

Counter mode basically turns a **block cipher into a stream cipher**.

In counter mode, the **sender maintains a small amount of state**: the counter. To encrypt the plaintext, which is essentially a set of blocks $P = \{P_1, P_2, \dots P_n\}$, we will take our block cipher F_K, feed it an IV and obtain a small key, and the we will XOR the key with P_1 obtaining C_1. We then increment the IV, feed it to F_k obtain another small key, XOR it with P_2 and obtain C_2. We repeat the process until we finally get the ciphertext $C = \{C_1, C_2, \dots C_n\}$.



The **IV will be part of the ciphertext**, since the receiver needs it to start decrypting the ciphertext.

In order to keep this secure, you have to **keep track of the IV properly**. To encrypt the n^{th} block you need to feed E_K $IV_n = IV_{initial} + (n-1)$. Using the same IV would defeat the purpose of encrypting identical plaintext blocks to different ciphertext blocks.

What happens if **one of the packets gets lost** (i.e. when the encryption scheme is used along a network connection)? Even if some packets get lost, the recipient can resync with the sender by just looking at the IV in the package.

Alin Tomescu, CSE408
Tuesday, February 22nd, Lecture #7

What happens if we **reuse an IV**? Lots and lots of bad stuff (see WEP). We better make this IV really big, at least 128 bits. If the IV was small, in the real world an attacker could keep sending the same message to Alice and she would encrypt it until she runs out of IVs. The attacker would know Alice ran out of IVs when she sees the same ciphertext being transmitted again.

F_K does not need to be invertible.

Theorem:

If F_k is a (t, q, ε)-secure PRF then CTR^{F_k} (counter mode built on F_k) is a $\left(t - O(q), \min\left(q, \left|IV_{space}\right|\right), \varepsilon\right)$-RoR-secure encryption scheme.

Proof:

We want to prove that $CTR^{F_k} \sim CTR^{F_k} \circ \$$

F_k is a (t, q, ε)-secure means $F_k \overset{t,q}{\underset{\varepsilon}{\sim}} Random_f$

By DPI $CTR^{F_k} \overset{t - O(q), q}{\underset{\varepsilon}{\sim}} CTR^{Random_f}$

As long as the IV never wraps, $CTR^{Random_f} \overset{\infty, \left|IV_{space}\right|}{\underset{0}{\sim}} CTR^{F_k} \circ \$$

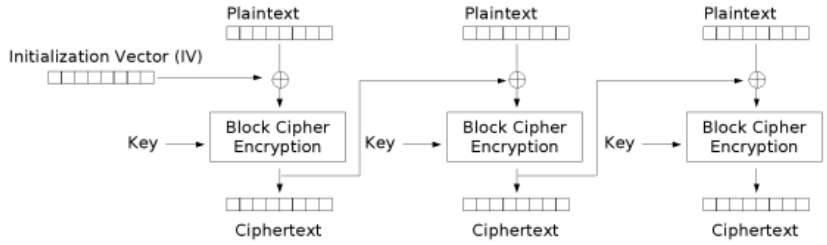
By transitivity we take the minimum on the time and the queries, and the sum of the epsilons and we get CTR^F to be a $(t - O(q), \min\left[q, \left|IV_{space}\right|\right], \varepsilon)$-RoR-secure encryption scheme.

Later on we will show how to make this secure against CCA attacks, since you can take any CPA-secure system and turn it into a CCA-secure system.

Cipher block chaining (CBC)

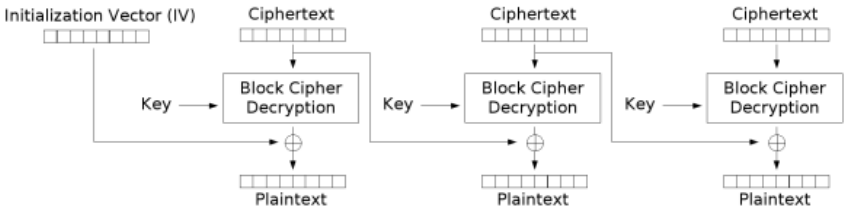
In CBC mode, we have blocks of plaintext P_i, we will have an invertible function F_k, and an IV.

To encrypt a message, we take P_1 and XOR it with the IV, feed the result to F_k which gives us C_1. We then repeat the process for P_2 except we use C_1 as the IV. So in CBC mode, the IV for the next block will be the previous ciphertext.



Cipher Block Chaining (CBC) mode encryption

Alin Tomescu, CSE408
Tuesday, February 22nd, Lecture #7



Cipher Block Chaining (CBC) mode decryption

$$C_i = F_k(P_i \otimes C_{i-1}), C_0 = IV$$

$$P_i = F_k^{-1}(C_i) \otimes C_{i-1}, C_0 = IV$$

What happens if C_i gets corrupted? It's going to screw up $P_i \; and \; P_{i+1}$ but P_{i+2} will decrypt correctly. Encryption is not parallel but decryption is. F_K needs to be invertible.

How do you choose the IV? Pick it randomly for each message, a.k.a. $CBC^{\$}$, or pick it randomly for the first message and use C_n as IV for the next message and repeat.