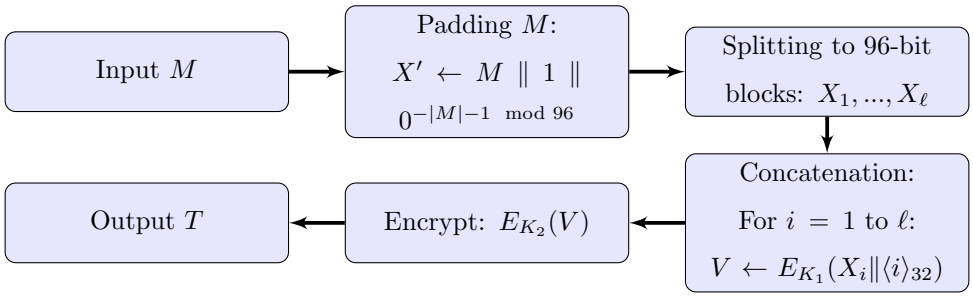
Assignment 1

Cristian Bassotto, Roberta Chissich

11th September 2024



(1a) What is the key size of LightMAC?

The key size of LightMAC is given by the sum of the sizes of the two AES-128 keys, K_1 and K_2, each of which is 128 bits:

$$\text{Key Size} = 128\,\text{bits} + 128\,\text{bits} = 256\,\text{bits}.$$

Thus, the key size of LightMAC is **256** bits.

(1b) Is this construction a Wegman-Carter MAC or a Protected Hash MAC?

This construction is closer to a Protected Hash MAC (PH-MAC).

A Wegman-Carter MAC utilizes a universal hash function combined with a secret key to produce a tag. Since LightMAC (AES)encrypts the checksum V using the second independent key K_2 after XORing the encrypted blocks, it aligns with the structure of a Protected Hash MAC, which combines the secret key with the message and then applies a hash function.

(1c) What can you conclude about the corresponding intermediate values V and V'?

If we find two messages M and M' of the same length such that LightMAC(M) = LightMAC(M'), then we can conclude that the corresponding intermediate values V and V' are equal if K_2 and K_2' are equal. Given the simplified version of our LightMAC we can say that $T = LightMAC(M) = LightMAC(M') = T'$, so, given $T = E_{K_2}(V)$ and $T' = E_{K_2'}(V')$, we assume that $E_{K_2}(V) = E_{K_2'}(V')$. This is because the LightMAC construction computes the tag T by first computing the checksum V, which is the XOR of the encrypted message blocks, and then encrypting V using the key K_2 to get the

final tag T. Since the AES-128 is a bijective operation given a key, we can only conclude something about V and V' if the key is the same, and so we have $E_{K_2}(V) = E_{K_2}(V') \rightarrow V = V'$.

(1d) Describe a forgery for LightMAC using this additional knowledge, and explain why this forgery is valid.

The core idea behind the LightMAC forgery attack is to find a different message M'' and its corrisponding tag that is considered as a valid tag. Since that we already assume the equivalence between $LightMAC(M) = LightMAC(M')$, then also $V = V'$ seince we will have the same key K_2.
So, from the problem statement, we have an arbitrary-length message X, and we know that

$$\text{LightMAC}(M \parallel 1 \parallel 0^{-|M|-1 \mod 96} \parallel X) = T^*.$$

From the structure of the LightMAC, we know that before computing the tag, we compute V from each 96-bit block of the message. Now we call V_M the value calculated before $T*$:

$$V_M = \bigoplus_{i=1}^{|M|/96} E_{K_1}(M_i \parallel < i >_{32}) \bigoplus_{i=|M|/96+1}^{l} E_{K_1}(X_i \parallel < i >_{32})$$

But the first part is just the V calculated inside LightMAC(M) before, and since we discover that $V = V'$, then we can substitute V with V' and the final V_M will remain invariate. This is possible also because the initial V and V' values where also calculated using the same initialization of the counter.
So, chnaging V and V' means that we can also substitute the message M with M' and the intermediate value V_M will not change.
So, at the end, our forgery attack will consist in using the new message $M_{\text{forged}} = M' \parallel 1 \parallel 0^{-|M'|-1 \mod 96} \parallel X)$ and using the same tag $T*$.
This forgery is valid because we managed to create a new message that reuse part of the previous ones, but it is not exactly as any of the previous ones, and the tuple $(M_{\text{forged}}, T*)$ is valid. Since intermediate values of both the known message and M_{forged} are equal, the LightMAC tags will be equal, even though the final messages are different.

(2a) Code part. link to the source

(2b) Argue whether $AES-128-OFB$ or $AES-128-OFB^{-1}$ can be implemented in parallel.

The problem in parallelizing AES-128-OFB encryption and decryption is the sequential dependency in the keystream generation process. Since that each keystream block depends on the previous one, this prevents the keystream generation from being fully parallelized.

However, the application of the keystream to the data (plaintext or ciphertext) can be parallelized, as each data block can be XORed with the corresponding keystream block independently.

(2c) Explain that this makes the OFB mode insecure.

The security of the OFB mode relies on the randomness of the IV. The IV is used to generate the keystream by encrypting it with the block cipher (AES in this case).

If the IV is replaced with a user-chosen nonce, in other words if it is not random, then every time the same nonce is used with the same key, the same keystream will be generated.

When the same keystream is used for multiple plaintexts, an attacker can XOR the ciphertexts together to eliminate the key stream and potentially recover information about the plaintexts (also implemented in the previous notebook in point 2a).

(3a) Mount a forgery attack in $q_m = 3MAC$ queries and $q_v = 1VFY$ query.

The Wegman-Carter MAC (WC-MAC) function is designed to be secure against forgery attacks, provided that the adversary does not query the WC_K function for repeated nonces.

This is because WC-MAC function is deterministic, so the same nonce will produce the same MAC tag for the same message. During the forgery, this property might be exploited to derive relationships between the messages and the MAC tags, by reusing the same N.

Query1: Request MAC of a M_1 using N_1. Receiving $MAC(M_1, N_1) = T_1$.

Query2: Request MAC of a $M_2 \neq M_1$ using N_1. Receiving $MAC(M_2, N_1) = T_2$.

Query3: Request MAC of a M_1 using $N_2 \neq N_1$. Receiving $MAC(M_2, N_2) = T_3$.

Then we can compute the new tag using

$$T_1 \oplus T_2 \oplus T_3 = H_L(M_1) \oplus H_L(M_2) \oplus H_L(M_2) \oplus F_R(N_2) = H_L(M_1) \oplus F_R(N_2)$$

So we make 1 VFY query with message M_1 and using N_2 instead of N_1 used before. We can compute the correct tag T_4 as $T_4 = T1 \oplus T_2 \oplus T_3$. The forgery is valid since the combination of message and nonce is never used before, and the tuple (M_1, T_4) is valid.

(3b) Prove that H_L is 2^{-n}XOR-universal.

A family of hash functions H is said to be 2^{-n} XOR-universal if for any distinct messages M, M' in

the domain and any t in the range, the following holds:

$$\Pr[H_L(M) \oplus H_L(M') = t] \leq 2^{-n}$$

So, now we need to calculate the probability that $H_L(M) \oplus H_L(M') = t$. We know that H_L is defined as $H_L(M) = L \otimes M$, so the previous statement will become

$$L \otimes M \oplus L \otimes M' = L \otimes (M \oplus M')$$

Since M and M' are distinct messages, $M \oplus M' \neq 0$. From this statement and knowing that, over finite fields, multiplications can be inverted the same way that it can in the field of real numbers, we can conclude that:

$$L \otimes (M \oplus M') = t \implies L = t \otimes (M \oplus M')^{-1}$$

So, for each L choosen in the finite field, it exist an unique value such that we have a collision. Since we can choose over 2^n values for L (since we pick from a uniform distribution), then the probability to choose a specific L is $\frac{1}{2^n}$. So:

$$\Pr[H_L(M) \oplus H_L(M') = t] = 2^{-n}$$

that satisfies the ipothesis.

(3c) Describe a forgery attack in only $q_m = 2MAC$ queries and $q_v = 1VFY$ query.

Query1: Request MAC of a M_1 using N. Receiving $MAC(M_1, N) = T_1 = H_L(M_1) \oplus F_R(N)$.

Query2: Request MAC of a $M_2 \neq M_1$ using N. Receiving $MAC(M_2, N) = T_2 = H_L(M_2) \oplus F_R(N)$.

Then we can combine the resulted tags as $T_1 \oplus T_2 = H_L(M_1) \oplus H_L(M_2)$ and we can notice that $F_R(N)$ simplifies. Now we expand H_L function and we reach $T_1 \oplus T_2 = L \otimes (M_1 \oplus M_2)$.

From the previous exercise 3a we know that over finite fields multiplications can be inverted if the denominator is nonzero. This is our case since $M_1 \oplus M_2 \neq 0$. So, now, we can compute $L = \frac{T_1 \oplus T_2}{M_1 \oplus M_2}$ and use it to get the final tag.

Now, to compute the final message and tag for the verification we only miss $F_R(N)$. We can calculate it taking the first query:

$$H_L(M_1) \oplus F_R(N) = T_1 \Rightarrow L \otimes M_1 \oplus F_R(N) = T_1 \Rightarrow F_R(N) = T_1 \oplus L \otimes M_1$$

Now we have everything to compute the final tag given an arbitrary message M_f:

$$T_f = H_L(M_f) \oplus F_R(N) = L \otimes M_f \oplus F_R(N) = L \otimes (M_f \oplus M_1) \oplus T_1 = \frac{T_1 \oplus T_2}{M_1 \oplus M_2} \otimes (M_f \oplus M_1) \oplus T_1$$

Then we can make the verification with the tuple (M_f, T_f). For this forgery we can use any new message, taking care that is not one of the previously used.