Message Authentication Code

Ali El Kaafarani

[1]Mathematical Institute
[2] PQShield Ltd.

Outline

1. CBC-MAC
2. Authenticated Encryption
3. Padding Oracle Attacks
4. Information Theoretic MACs

Basic CBC-MAC for fixed-length messages

Definition

Let F be a pseudorandom function. The basic CBC-MAC can be defined as follows:

- Mac$(k \in \{0,1\}^n, m)$*: it takes a key k and a message m of length $n \cdot L$ where $L = \ell(n)$ and does the following;*
  - *parses m as $m_1, \cdots, m_L$, where $|m_i| = n$*
  - *initializes $t_0 \leftarrow 0^n$, and for $i = 1, \cdots, L$ Do*

$$t_i \leftarrow F_k(t_{i-1} \oplus m_i)$$

  - *outputs the tag t_L.*
- Verify$(k \in \{0,1\}^n, m, t)$*: if $|m| = n \cdot \ell(n)$ and $t =$ Mac(k, m) output 1, output 0 otherwise.*

CBC-MAC

⚠ The previous construction is only secure for fixed-length messages!

- There are ways to modify the construction to handle arbitrary-length messages.
- For example, one can change the key generation to choose two uniformly independent keys, $k_1, k_2 \in \{0,1\}^n$. The authentication will be done in two steps. First, it computes $t_1 \leftarrow$CBC-MAC(m, k_1), and then it outputs the final result as $t \leftarrow F_{k_2}(t_1)$.

Differences between CBC-MAC and CBC-mode encryption

There are two main differences:

- CBC-mode encryption has a random IV whereas CBC-MAC has a fixed one (i.e. 0^n) and they are only secure under these conditions
- CBC-mode encryption outputs all the intermediate values c_i as parts of the ciphertext whereas CBC-MAC only outputs the final tag t_L (and only secure in this case).

Outline

Authenticated Encryption

- A way to achieve both secrecy and integrity at the same time.
- No standard terminology or definitions yet.
- CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness.
  `http://competitions.cr.yp.to/caesar.html`
- The level of secrecy that we want: CCA-secure
- The level of integrity: existential unforgeability under chosen-message attacks.

Unforgeable Encryption

We define the unforgeable encryption experiment as follows:

- KeyGen(n): output a key k.
- Adversary's capabilities: access to an encryption oracle Enc$(k, \cdot)$. All his queries will be stored in Q
- Adversary's output: a ciphertext c.
- Winning conditions: compute $m \leftarrow \mathsf{Dec}(k, c)$ and output 1 if the following hold
  - $m \neq \perp$
  - $m \notin Q$

Definition

A private key encryption scheme S is unforgeable if for all PPT adversaries A*, we have* $\Pr[\mathsf{PrivK}_{\mathsf{A},S}^{Unforg}(n) = 1] \leq \mathsf{negl}(n)$

Authenticated Encryption: A Definition

Definition

A private-key encryption scheme is an authenticated encryption scheme is it is both CCA-secure and Unforgeable.

- Does any *random* combination of a secure encryption scheme and a secure message authenticated code yield an authenticated encryption scheme?

Authenticated Encryption: A Definition

Definition

A private-key encryption scheme is an authenticated encryption scheme is it is both CCA-secure and Unforgeable.

- Does any *random* combination of a secure encryption scheme and a secure message authenticated code yield an authenticated encryption scheme?
- The answer is NO!

Authenticated Encryption: A Definition

Definition

A private-key encryption scheme is an authenticated encryption scheme is it is both CCA-secure and Unforgeable.

- Does any *random* combination of a secure encryption scheme and a secure message authenticated code yield an authenticated encryption scheme?
- The answer is NO!
- Lesson: you can't just combine two secure cryptographic modules/tools and expect the combination to be automatically secure!

Authenticated Encryption: How to combine MAC and ENC?

- Mac and Enc: compute them in parallel,

$$c \leftarrow \mathsf{Enc}(k_1, m) \text{ and } t \leftarrow \mathsf{Mac}(k_2, m)$$

- Mac then Enc:

$$t \leftarrow \mathsf{Mac}(k_2, m) \text{ then } c \leftarrow \mathsf{Enc}(k_1, m||t)$$

- Enc then Mac:

$$c \leftarrow \mathsf{Enc}(k_1, m) \text{ then } t \leftarrow \mathsf{Mac}(k_2, c)$$

MAC and Encrypt

- This combination violates the secrecy of the scheme even if Enc is secret. Why?

MAC and Encrypt

- This combination violates the secrecy of the scheme even if Enc is secret. Why?
- Remember, MAC doesn't provide any secrecy, and yet you are sending the tag t in the clear!
- MAC can be deterministic (like most MACs used in practice)
- The scheme is not even CPA-secure in this case!

MAC then Encrypt

- This combination is not guaranteed to be an authenticated encryption either!
- $m||t$ has to be padded in a specific way to get a multiple of the block length.
- The decryption may now fail for two different reasons: incorrect padding or invalid tag! (Note that the padded part is not protected under the tag scheme!)
- What if the attacker can distinguish between the two errors?
- Okay, we return a single error message in both cases (even though it is not ideal!)
- What about the difference in time to return each of them? (Some attacks on *Secure Socket Layer (SSL)* were based on this idea!)
- Result: padding-oracle attack (in details soon)!

Encrypt then MAC

- The MAC should be *strongly secure*.
- This guarantees that the adversary can't produce any *new* valid ciphertext. (i.e. not obtained from the encryption oracle)
- This way, the adversary cannot benefit from the decryption oracle of the CCA game.
- Therefore, CPA security of the encryption scheme S is enough.

Encrypt then MAC

- The MAC should be *strongly secure*.
- This guarantees that the adversary can't produce any *new* valid ciphertext. (i.e. not obtained from the encryption oracle)
- This way, the adversary cannot benefit from the decryption oracle of the CCA game.
- Therefore, CPA security of the encryption scheme S is enough.
- in this case: CPA-secure S + strongly secure MAC $\Longrightarrow$ CCA-security+integrity

Encrypt then MAC: Generic construction

Given a private-key encryption scheme $S = (\mathsf{Enc}, \mathsf{Dec})$ and a message authentication code $MAC = (\mathsf{Mac}, \mathsf{Verify})$, we define a private-key encryption scheme $S' = (\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- $\mathsf{KeyGen}'(n)$: choose **independent**, uniform keys $k_{enc}, k_{mac} \in \{0,1\}^n$.
- $\mathsf{Enc}'(k_{enc}, k_{mac}, m)$: compute $c \leftarrow \mathsf{Enc}(k_{enc}, m)$. Then compute $t \leftarrow \mathsf{Mac}(k_{mac}, c)$. The ciphertext will then be (t, c).
- $\mathsf{Dec}'(c, t, k_{enc}, k_{mac})$:
  - if $\mathsf{Verify}(k_{mac}, c, t) = 1$ then output $\mathsf{Dec}(k_{enc}, c)$
  - otherwise, output $\perp$.

Authenticated Encryption: an Application and Potential attacks

- It is used to offer secure communication sessions.
- It is not enough on its own to provide secure sessions, here are some possible attacks:
- Re-ordering attack: change the order in which the message were supposed to be delivered (force c_2 to arrive before c_1)
- Replay attack: to replay a previously sent valid ciphertext
- Reflection attack: to change the direction of the message and resend to the sender instead of the receiver.
- Solutions: use *counters* for the first two problems, and different encryption keys for different directions, i.e. $K_{A\to B} \neq K_{B\to A}$.

Outline

A Padding Oracle Attack

- In CBC mode, messages have to be multiple of the block length
- if they are not, we pad them. PKCS#5 is a famous and standardized approach.
- Assume that $|m| = n$ and block length= L (both in bytes). Let $m = r \cdot L + d$. Therefore, $b = L - d$ is the number of bytes that need to be padded to the message.
- Exceptionally, if $b = 0$, we pad L bytes, therefore $1 \leq b \leq L$.
- We append to the message the integer b represented in either 1-byte or two hexadecimal digits.
- if 1 byte is needed, we append 0x01 to the end of the message. If 3 bytes are needed, we append 0x030303
- The padded message which is called *encoded data*, will then be encrypted using CBC-mode encryption.

A Padding Oracle Attack

- Decryption in CBC mode: it first decrypts the ciphertext, it then checks on the correctness of the padding, and finally checks on the validity of the tag.

A Padding Oracle Attack

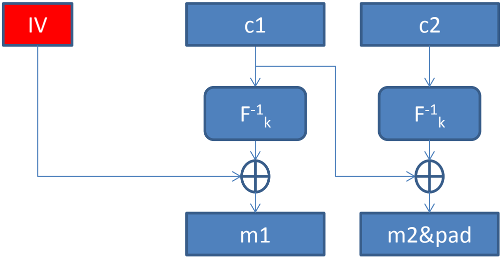
- Decryption in CBC mode: it first decrypts the ciphertext, it then checks on the correctness of the padding, and finally checks on the validity of the tag.
- You first read the values b of the last byte, and make sure it is the same value in the last b bytes.
- If the padding is correct, you drop the last b bytes and get the original plaintext, otherwise output “padding error”.
- This is a great source of information to the adversary, you can think of it as a *limited* decryption oracle.
- Adversaries can send ciphertexts to the server and learn whether or not they are padded correctly!
- This way the adversary can recover the whole message for any ciphertext of his choice.
-

A Padding Oracle Attack

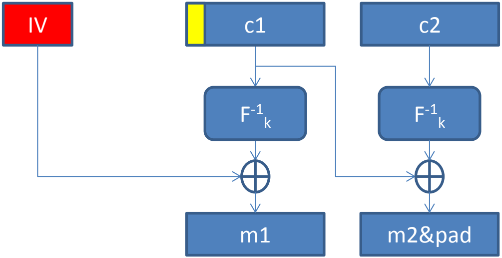
- We will take the example of a 3-block ciphertext, IV, c_1, c_2 that correspond to the message m_1, m_2 which is of course unknown to the attacker.
- By definition, $m_2 = F_k^{-1}(c_2) \oplus c_1$. The block m_2 should end with $\underbrace{0\mathrm{x}b \cdots 0\mathrm{x}b}_{b \text{ times}}$
- Key idea: if you let $c_1' = c_1 \oplus \Delta$, for any string Δ, and you try to decrypt the new cipher text IV, c_1', c_2 then you will get m_1', m_2', where $m_2' = m_2 \oplus \Delta$.
- Exploiting this, the adversary can learn b, and consequently the length of the original plaintext.
- The attacker starts with modifying the first byte of c_1 and sends the modified ciphertext, IV, c_1', c_2 to the receiver...

A Padding Oracle Attack

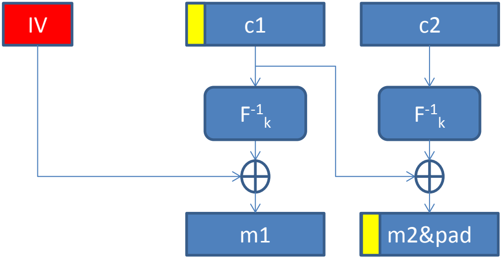
Step 1: find the length of the padded bytes b.



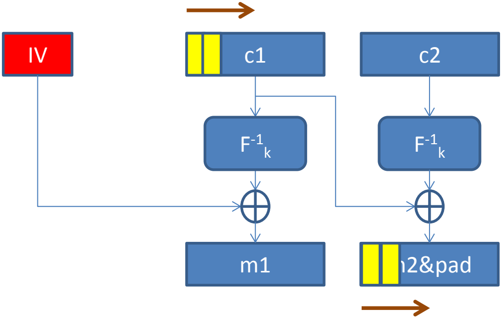
A Padding Oracle Attack



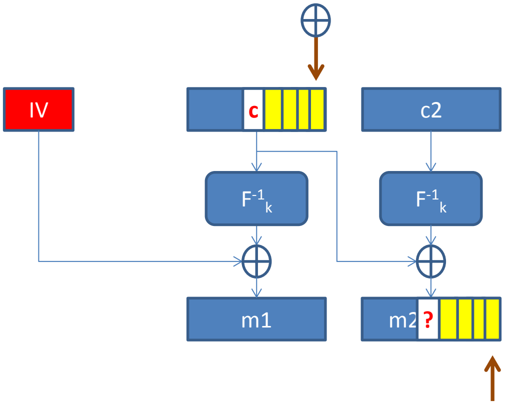
A Padding Oracle Attack

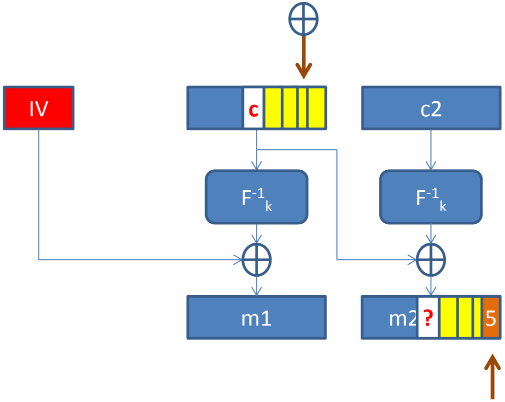


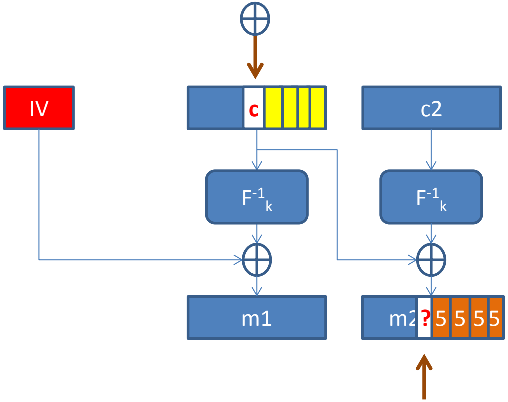
A Padding Oracle Attack

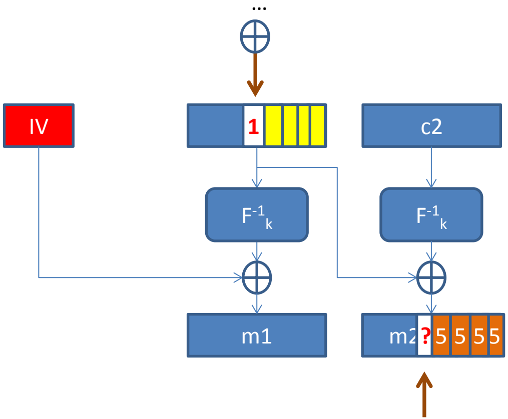


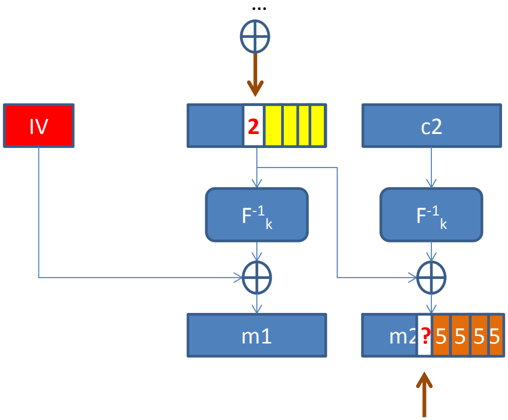
Second step, recover the plaintext byte by byte.

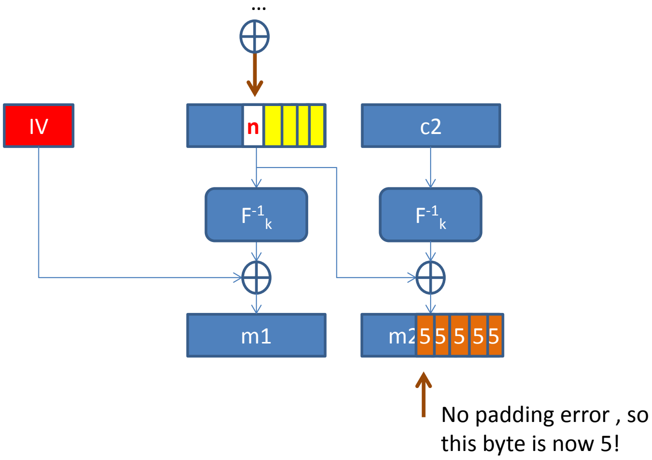


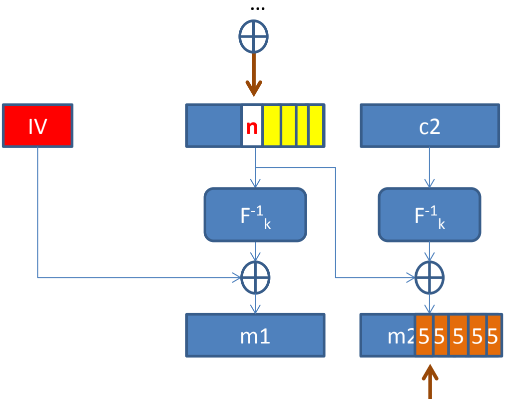

IV
c
c2
F-1k
F-1k
m1
m2
?
5


IV
c
c2
F-1k
F-1k
m1
m2 ? 5 5 5 5


...
IV
1
c2
F-1k
F-1k
m1
m2
?
5
5
5
5


...
IV
2
c2
F^{-1}_k
F^{-1}_k
m1
m2
?
5
5
5
5


...
IV
n
c2
F^{-1}_k
F^{-1}_k
m1
m2
5 5 5 5 5
No padding error , so
this byte is now 5!



No padding error , so this byte is now 5!
Simple computation will lead to finding the byte ?

Outline

Information Theoretic MACs

- All the MACs we have talked about so far have computational security, i.e. the adversary's running time are *bounded*
- Can we build a MAC that is secure even in the presence of *unbounded* adversaries?
- Note that we cannot get a perfectly secure MACs. Why?

Information Theoretic MACs

- All the MACs we have talked about so far have computational security, i.e. the adversary's running time are *bounded*
- Can we build a MAC that is secure even in the presence of *unbounded* adversaries?
- Note that we cannot get a perfectly secure MACs. Why?
- Clearly because adversaries can guess a valid tag with probability $1/2^{|t|}$, if t is the length of the scheme's tags.
- Back to unbounded adversaries: is information theoretic MACs achievable?
- Yes, BUT with a bound on the number of messages to be authenticated!

Information Theoretic MACs

As we want to put a bound on the number of the messages to be authenticated, let's start with most basic case, i.e. *only one message*. Here is the one-time message authentication experiment. Notice that here we drop the security parameter n, as we are dealing with unbounded adversaries!

- KeyGen : returns a key k
- Single tag query: adversary $\mathcal{A}$ sends a message m' and gets a tag on it t'
- Adversary's output: (m, t)
- Experiment's output: 1 iff

$$\text{Verify}(k, t) = 1 \text{ and } m \neq m'$$

Information Theoretic MACs

Definition

A message authentication code S is one-time ϵ-secure, if for all adversaries $\mathcal{A}$ (including unbounded ones):

$$\Pr[\mathsf{Mac}^{1-time}_{\mathcal{A},S} = 1] \leq \epsilon$$

Information Theoretic MACs

- We need to first define *strongly universal functions* (also called *pairwise-independent*).
- Given a keyed function $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$, where $h(k, m)$ is often written as $h_k(m)$. Informally speaking, we say that h is strongly universal if for any $m \neq m'$ and uniform key k, the images $h_k(m)$ and $h_k(m')$ are uniformly and independently distributed in $\mathcal{T}$.
- Formally speaking, $\forall m \neq m'$, and $\forall t, t' \in \mathcal{T}$, we have

$$\Pr[h_k(m) = t \wedge h_k(m') = t'] = 1/|\mathcal{T}|^2$$

where the probability is taken over uniform choice of $k \in K$.

Information Theoretic MAC: a construction from a strongly universal function

Given a strongly universal function $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$, we define a messages authentication code MAC with message space $\mathcal{M}$ as follows:

- KeyGen : output a uniformly chosen key $k \leftarrow \mathcal{K}$
- Mac(k, m): output the tag $h_k(m)$
- Verify(k, m, t): if $m \notin \mathcal{M}$ output 0, otherwise output 1 iff $t = h_k(m)$.

Information Theoretic MAC: a construction from a strongly universal function

Theorem

Given a strongly universal function $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$. A Message authentication code that is based on h with message space $\mathcal{M}$ is a one-time $1/|\mathcal{T}|$-secure MAC.

Proof.

Let $\mathcal{A}$ be an adversary against the MAC scheme. He queries m' and gets t'. He finally outputs the forgery (m, t). The probability that (m, t) is a valid forgery is the following:

$$\begin{aligned}
\Pr[\mathsf{Mac}^{1-time}_{\mathcal{A},S} = 1] &= \sum_{t'} \Pr[\mathsf{Mac}^{1-time}_{\mathcal{A},S} = 1 \wedge h_k(m') = t'] \\
&= \sum_{t'} \Pr[h_k(m) = t \wedge h_k(m') = t'] \\
&= \sum_{t'} \frac{1}{|\mathcal{T}|^2} \\
&= \frac{1}{\mathcal{T}}
\end{aligned}$$

□

Strongly Universal Function: a Concrete Construction

Example

Give $\mathbb{Z}_p$ for some prime p. Let $\mathcal{M} = \mathcal{T} = \mathbb{Z}_p$, and let $\mathcal{K} = \mathbb{Z}_p \times \mathbb{Z}_p$. we define a keyed function $h_{a,b}$ as

$$h_{a,b}(m) = a \cdot m + b \mod p$$

Theorem

For any prime p, the function h is strongly universal.

Information Theoretic MAC: its limitations

Theorem

If S is a one-time 2^{-n}-secure MAC with constant size keys, then $|k| \geq 2n$.

Proof.

Exercise. □

Information Theoretic MAC: its limitations

Theorem

If S is a ℓ-time 2^{-n}-secure MAC with constant size keys, then $|k| \geq (\ell + 1)n$.

Corollary

If the key-length of a given MAC is bounded, then it is not information-theoretic secure when authenticating an unbounded number of messages.

Further Reading (1)

- N.J. Al Fardan and K.G. Paterson.
  Lucky thirteen: Breaking the TLS and DTLS record protocols.
  In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 526–540, May 2013.
- J Lawrence Carter and Mark N Wegman.
  Universal classes of hash functions.
  In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 106–112. ACM, 1977.
- Jean Paul Degabriele and Kenneth G Paterson.
  On the (in) security of IPsec in MAC-then-Encrypt configurations.
  In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 493–504. ACM, 2010.

Further Reading (2)

- Ted Krovetz and Phillip Rogaway.
  The software performance of authenticated-encryption modes.
  In *Fast Software Encryption*, pages 306–327. Springer, 2011.
- Douglas R. Stinson.
  Universal hashing and authentication codes.
  Designs, Codes and Cryptography, 4(3):369–380, 1994.