Message Authentication Code

Federico Pintore[1]

[1]Mathematical Institute

Outline

Outline

Basic CBC-MAC

Definition

Let F be a pseudorandom function. The basic CBC-MAC is 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 $i = 1, \cdots, L$ computes*

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

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

CBC-MAC

⚠ The previous construction is only secure for messages of given lenght!

CBC-MAC

⚠ The previous construction is only secure for messages of given lenght!

There are ways to (securely) modify the construction to handle arbitrary-length messages.

CBC-MAC

⚠ The previous construction is only secure for messages of given lenght!

There are ways to (securely) modify the construction to handle arbitrary-length messages.

- prepend $|m|$, encoded as an n-bit string, to the message m;

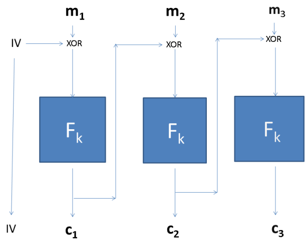
CBC-MAC

⚠ The previous construction is only secure for messages of given lenght!

There are ways to (securely) modify the construction to handle arbitrary-length messages.

- prepend $|m|$, encoded as an n-bit string, to the message m;
- change the key generation to choose two uniform, independent keys, $k_1, k_2 \in \{0,1\}^n$. Then $t_1 \leftarrow$CBC-MAC(m, k_1) is computed and the output tag is $t \leftarrow F_{k_2}(t_1)$.

CBC-MAC and CBC-mode encryption



- 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 primitive to achieve both secrecy and integrity simultaneously.
- No standard terminology or definitions yet.
- CAESAR - Competition for Authenticated Encryption: Security, Applicability, and Robustness.
  `http://competitions.cr.yp.to/caesar.html`
- Level of secrecy that we want: *CCA-security*.
- Level of integrity: a variant of *existential unforgeability under chosen-message attacks* for encryption schemes.

Unforgeable Encryption

We define the **unforgeable game** for an encryption scheme $S = (\text{KeyGen}, \text{Enc}, \text{Dec})$ as follows:

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

Definition

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

Authenticated Encryption: A Definition

Definition

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

- Not any combination of a secure encryption scheme and a secure would 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.

- Not any combination of a secure encryption scheme and a secure would yield an authenticated encryption scheme.
- Lesson: you can't just combine two secure cryptographic modules/tools and expect the combination to be automatically secure!

Authenticated Encryption from secure MAC and ENC

Any authenticated encryption is also CCA-secure.

- there exist CCA-secure encryption schemes that are not unforgeable;
- we do not really have an encryption which is only CCA secure and more efficient than authenticated encryptions.

We are only interested in combining a CPA-secure encryption with a secure MAC.

Authenticated Encryption: How to combine MAC and ENC?

- Mac and Enc: compute them independently and in parallel,

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

- Enc then Mac:

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

- Mac then Enc:

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

MAC and Encrypt

If the MAC is deterministic (like most MACs used in practice), the scheme is not even CPA-secure!

- CPA security implies CPA security for multiple encryptions;
- if the attacker submits (m, m) and (m, m'), from the challenge ciphertexts can easily guess which messages were encrypted.

Encrypt then MAC: formal description

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

- $\text{KeyGen}'(n)$: chooses **independent**, uniform keys $k_e, k_m \in \{0,1\}^n$.
- $\text{Enc}'(k_e, k_m, m)$: computes $c \leftarrow \text{Enc}(k_e, m)$ and then $t \leftarrow \text{Mac}(k_m, c)$. The ciphertext is (c, t).
- $\text{Dec}'((c,t), k_e, k_m)$:
  - if $\text{Verify}(k_m, c, t) = 1$ then outputs $\text{Dec}(k_e, c)$
  - otherwise, outputs $\perp$.

Encrypt then MAC

In this case: CPA-secure S + strongly secure MAC $\Longrightarrow$ CCA-security and integrity of S'.

- $\langle c, t \rangle$ is a valid ciphertext if Verify$(k_m, c, t) = 1$;
- if the MAC is strongly secure, then an adversary cannot generate a new ciphertext (i.e. not obtained from the encryption oracle);
- therefore, S' is unforgeable and the adversary cannot benefit from the decryption oracle of the CCA game;
- CPA-security of the encryption scheme S is enough.

Authenticated Encryption: an Application and Potential attacks

An authenticated encryption is not enough, on its own, to provide security over a communication session.

Possible attacks:

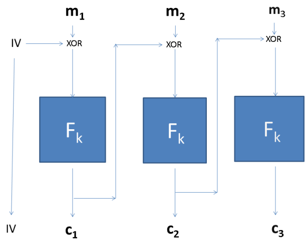
- Re-ordering attack: change the order in which the message were supposed to be delivered (force c_2 to arrive before c_1).
- Replay attack: replay a previously sent valid ciphertext.
- Reflection attack: change the direction of the message and resend to the sender instead of the receiver.

Solutions: use *counters* for the first two problems, and different encryption keys for different directions, i.e. $K_{A\to B} \neq K_{B\to A}$.

MAC then Encrypt

It is not guaranteed to be an authenticated encryption!

- the CBC mode encryption is CPA-secure but not CCA-secure;
- the padding oracle attack applies!



Outline

A Padding Oracle Attack

- In CBC mode, the number of bits of a message should be multiple of the block length

A Padding Oracle Attack

- In CBC mode, the number of bits of a message should be multiple of the block length
- if it is not, we pad the message. PKCS#5 is a famous and standardised approach.

A Padding Oracle Attack

- In CBC mode, the number of bits of a message should be multiple of the block length
- if it is not, we pad the message. PKCS#5 is a famous and standardised approach.
- Assume that $|m| = L$ and block length = t (both in bytes). Let $L = r \cdot t + d$. Therefore, $b = t - d$ is the number of bytes that need to be padded to the message.

A Padding Oracle Attack

- In CBC mode, the number of bits of a message should be multiple of the block length
- if it is not, we pad the message. PKCS#5 is a famous and standardised approach.
- Assume that $|m| = L$ and block length = t (both in bytes). Let $L = r \cdot t + d$. Therefore, $b = t - d$ is the number of bytes that need to be padded to the message.
- Exceptionally, if $b = 0$, we pad t bytes, therefore $1 \leq b \leq L$.

A Padding Oracle Attack

- In CBC mode, the number of bits of a message should be multiple of the block length
- if it is not, we pad the message. PKCS#5 is a famous and standardised approach.
- Assume that $|m| = L$ and block length = t (both in bytes). Let $L = r \cdot t + d$. Therefore, $b = t - d$ is the number of bytes that need to be padded to the message.
- Exceptionally, if $b = 0$, we pad t bytes, therefore $1 \leq b \leq L$.
- We append to the message the integer b represented in either 1-byte or two hexadecimal digits.

A Padding Oracle Attack

- In CBC mode, the number of bits of a message should be multiple of the block length
- if it is not, we pad the message. PKCS#5 is a famous and standardised approach.
- Assume that $|m| = L$ and block length = t (both in bytes). Let $L = r \cdot t + d$. Therefore, $b = t - d$ is the number of bytes that need to be padded to the message.
- Exceptionally, if $b = 0$, we pad t bytes, therefore $1 \leq b \leq L$.
- We append to the message the integer b represented in either 1-byte or two hexadecimal digits.
- Examples: 1 byte needed, then we append 00000001 to the end of the message; 2 bytes needed, then we append $00000010||00000010$.

A Padding Oracle Attack

- In CBC mode, the number of bits of a message should be multiple of the block length
- if it is not, we pad the message. PKCS#5 is a famous and standardised approach.
- Assume that $|m| = L$ and block length = t (both in bytes). Let $L = r \cdot t + d$. Therefore, $b = t - d$ is the number of bytes that need to be padded to the message.
- Exceptionally, if $b = 0$, we pad t bytes, therefore $1 \leq b \leq L$.
- We append to the message the integer b represented in either 1-byte or two hexadecimal digits.
- Examples: 1 byte needed, then we append 00000001 to the end of the message; 2 bytes needed, then we append $00000010||00000010$.
- The padded message, which is called *encoded data*, will then be encrypted using CBC-mode encryption.

Authenticate then encrypt

Decryption: first, decrypt the ciphertext; then, check the correctness of the padding; finally, check on the validity of the tag.

Authenticate then encrypt

Decryption: first, decrypt the ciphertext; then, check the correctness of the padding; finally, check on the validity of the tag.

- Read the value b of the last byte, and check if it is the same value in the last b bytes.

Authenticate then encrypt

Decryption: first, decrypt the ciphertext; then, check the correctness of the padding; finally, check on the validity of the tag.

- Read the value b of the last byte, and check if it is the same value in the last b bytes.
- If the padding is correct, drop the last b bytes and get the original plaintext, otherwise output "padding error".

Authenticate then encrypt

Decryption: first, decrypt the ciphertext; then, check the correctness of the padding; finally, check on the validity of the tag.

- Read the value b of the last byte, and check if it is the same value in the last b bytes.
- If the padding is correct, 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 and learn whether or not they are padded correctly (receiving the padding error)!
- 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 corresponds to the message m_1, m_2 (unknown to the attacker).

A Padding Oracle Attack

- We will take the example of a 3-block ciphertext, IV, c_1, c_2, that corresponds to the message m_1, m_2 (unknown to the attacker).
- By definition, $m_2 = F_k^{-1}(c_2) \oplus c_1$. The block m_2 should end with $\underbrace{0\text{x}b \cdots 0\text{x}b}_{b \text{ times}}$

A Padding Oracle Attack

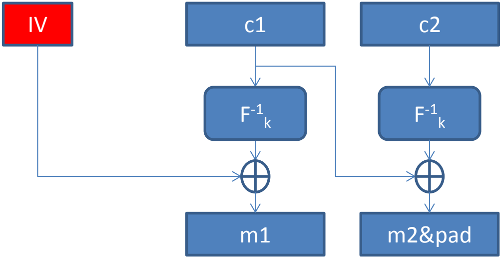
- We will take the example of a 3-block ciphertext, IV, c_1, c_2, that corresponds to the message m_1, m_2 (unknown to the attacker).
- By definition, $m_2 = F_k^{-1}(c_2) \oplus c_1$. The block m_2 should end with $\underbrace{0\text{x}b \cdots 0\text{x}b}_{b \text{ times}}$
- **Key idea**: given $c_1' = c_1 \oplus \Delta$, for any string Δ, if you try to decrypt the new ciphertext IV, c_1', c_2 then you will get m_1', m_2', where $m_2' = m_2 \oplus \Delta$.

A Padding Oracle Attack

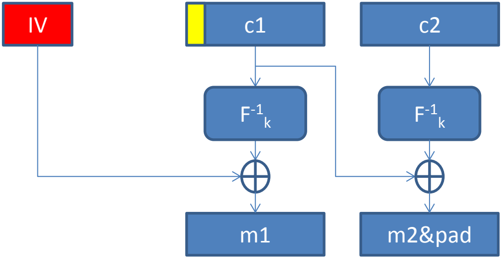
- We will take the example of a 3-block ciphertext, IV, c_1, c_2, that corresponds to the message m_1, m_2 (unknown to the attacker).
- By definition, $m_2 = F_k^{-1}(c_2) \oplus c_1$. The block m_2 should end with $\underbrace{0\text{x}b \cdots 0\text{x}b}_{b \text{ times}}$
- **Key idea**: given $c_1' = c_1 \oplus \Delta$, for any string Δ, if you try to decrypt the new ciphertext 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.

A Padding Oracle Attack

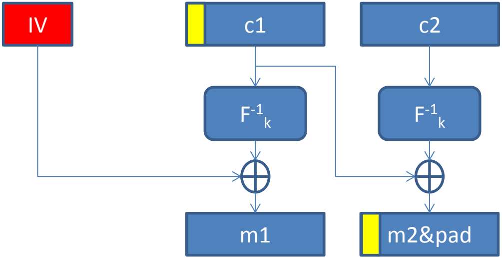
Step 1: learn b (number of padded bytes).



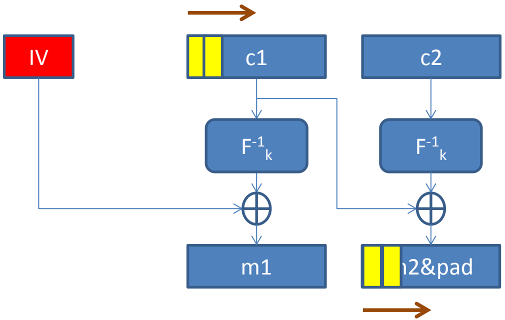
A Padding Oracle Attack



A Padding Oracle Attack

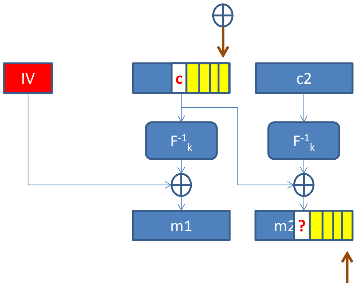


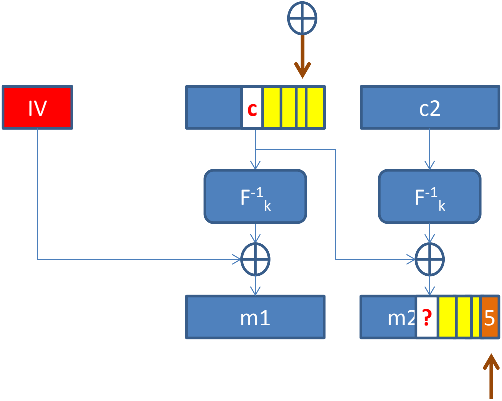
A Padding Oracle Attack

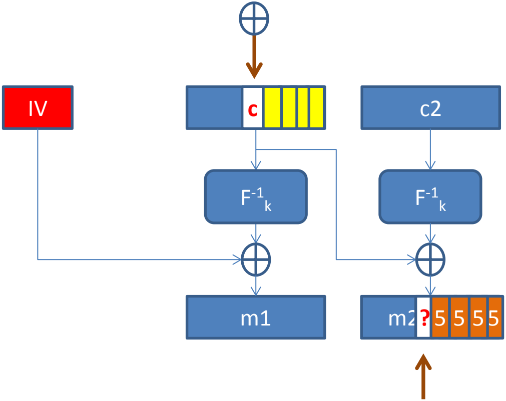


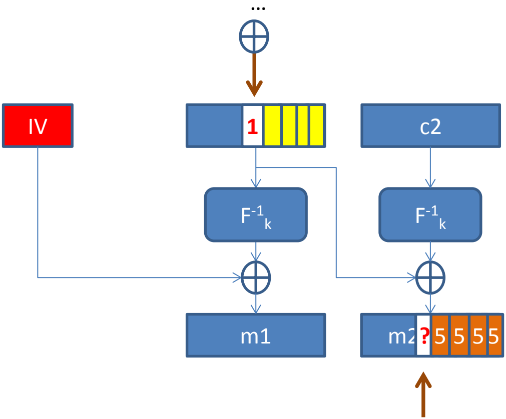
Second step, recover the plaintext byte by byte.
The adversary modifies c_1 with the perturbation

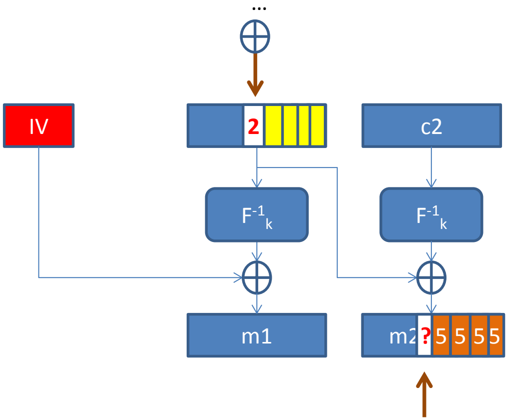
$$\Delta_n = 0x0 \cdots 0x00xn \underbrace{0\text{x}b + 1 + b \cdots 0\text{x}b + 1 + b}_{b \text{ times}}$$

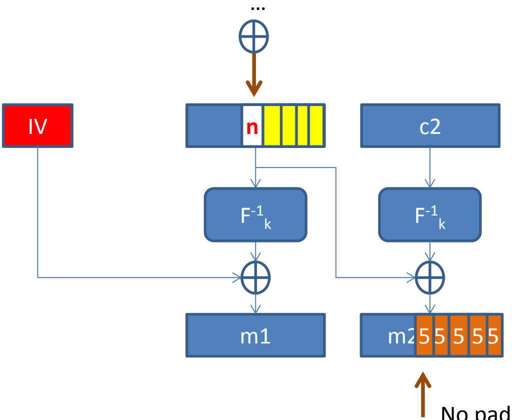



IV
c
c2
F-1k
F-1k
m1
m2
?
5

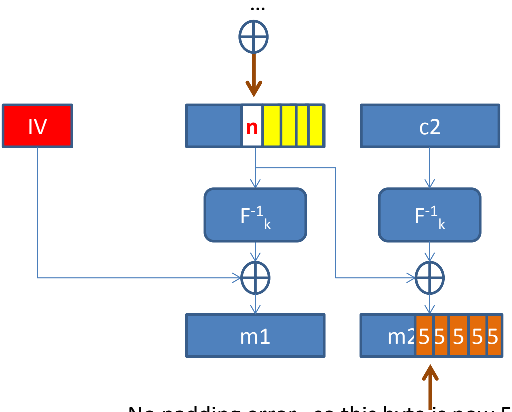

IV
c
c2
F-1k
F-1k
m1
m2
?
5
5
5
5


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


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

MAC then Encrypt

- The decryption may fail for two different reasons: incorrect padding or invalid tag!
- 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!)

Outline

Information Theoretic MACs

- All the MACs we have talked about so far have computational security, i.e. the adversary's running time is *bounded*
- Can we build a MAC that is secure even in the presence of *unbounded* adversaries?
- We cannot get a perfectly secure MAC since adversaries can guess a valid tag with probability $1/2^t$, if t is the length of the tags.
- Information theoretic MACs: success probability cannot be better than $1/2^t$. Are they achievable?
- Yes, BUT with a bound on the number of messages that can be authenticated!

Information Theoretic MACs

Most basic case: *only one message* can be authenticated.

Definition (One-time message authentication experiment)

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

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

We drop the security parameter n, as we are dealing with unbounded adversaries!

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*).

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 $h_k(m)$, we have that $\forall m \neq m'$, and $\forall t, t' \in \mathcal{T}$ it holds

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

  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 : outputs a uniformly chosen key $k \leftarrow \mathcal{K}$
- Mac(k, m): outputs the tag $h_k(m)$
- Verify(k, m, t): outputs 1 iff $m \in \mathcal{M}$ and $t = h_k(m)$, otherwise outputs 0

Information Theoretic MAC: a construction from a strongly universal function

Theorem

Given a strongly universal function $h : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$, a message authentication code that is based on h is one-time $1/|\mathcal{T}|$-secure.

Proof.

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

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

□

Strongly Universal Function: a Concrete Construction

Example

Consider $\mathbb{Z}_p$ 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

$$|k| \geq 2n.$$

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.