Chapter 4 | Message Authentication Codes

Thursday, May 25, 2023 10:20 AM

§ 4.1 Secure Communication and Message Integrity

Story:

- One of the most basic goals of cryptography
  is to enable parties to communicate over an
  open communication channel
  in a secure way

- One immediate question that arises, however, is
  what do we mean by "secure communication"?

- In Chapter 3,
  we showed how it is possible to obtain
  private communication
  over an open channel

  i.e. we showed how encryption can be used to
  prevent an eavesdropper (or possibly a more active adversary)
  from learning anything about the
  content of the messages sent over an
  unprotected communication channel.

- However,
  not all security concerns are related to
  the ability or inability of an adversary to
  learn something about messages being sent

- Specifically,
  when two parties communicate,
  they have the implicit assumption that the
  message sent by one party is
  indeed received by the other party.

- This expectation of message integrity
  is the source of a critical security concern

- E.g.
  consider the case that a large supermarket chain
  sends an email request to purchase 10,000
  crates of soda from a supplier

  Upon receiving this request, the supplier must ensure the following:

  (a) Is the order authentic?
  That is, did the supermarket chain
  really issue the order
  or was it issued by an adversary who spoofed the email address
  (something that's remarkably easy to do!)

  (b) If the order was issued by the supermarket,
  then the supplier must still ask
  whether the details of the order
  that it receeived are exactly those
  sent by the supermarket
  or were these details somehow changed
  en route by an adversarial router?

  NB: The order itself is not secret
  and therefore the question of privacy does not arise at all!

  NB2: Rather, the problem is that of
  message integrity.

- Any unprotected online purchase order/banking operation/email/SMS
  cannot be trusted

  (however, in practice, people often take the caller ID, or
  email return address are taken to be "proofs of identity"
  in many cases)

  This leaves users exposed to adversarial attacks

- Reiteration:
  - Goals of privacy and message authentication
    are often confused and unnecssarily intertwined (as mentioned before)
  - At the end of the chapter
    we show how to combine the two goals (privacy and integrity) simultaneously

§ 4.2 Encryption and Message Authentication

Story:

- We have already stressed that
  the problems of privacy and message authentication are distinct
- However, this does not necessarily mean that
  their solutions are distinct
- Specifically, at first sight
  it may seem that encryption should immediately solve
  the problem of message authentication as well
- This is because
  a ciphertext completely hides the contents of the message
  thus, it seems that an adversary cannot possibly modify
  an encrypted message en route
  i.e. all that it sees is "random garbage"

- Despite its intuitive appeal
  the claim that encryption solves the
  problem of message authentication is
  completely false.

Stream ciphers and message authentication.

Story:

- First, consider the case that
  a message m is encrypted using a stream cipher
  i.e. $E_k(m) = G(k) \oplus m$
  where G is a pseudorandom generator.
- Such ciphertexts are
  very easy to manipulate.
  E.g. flipping any bit in c results in
  the same bit being flipped in m upon decryption.

  Thus, given a ciphertext c that encrypts a message m
  it is possible to modify c to c' such that
  $D_k(c')$ equals $D_k(c)$ except
  for the least significant (or any other) bit that was flipped.

- Note that such a modification may be very useful
  E.g.
  most electronic messages consist of a header and a body

Further,
headers have a fixed format and
contain a number of flags.

Using the strategy defined here,
it is straightforward to modify flags
in the header in an encrypted message
(using the fixed format, an adversary can easily know which bits to flip).

Needless to say, such flags can have significant meaning
(for example, whether to buy or sell a stock).

Furthermore, if the message m represents a financial transaction
where the *amount* appears in a fixed place
then an adversary can easily modify this amount.

NB: even if the modification is oblivious
(meaning that the adversary does not know what the amount is changed to)
the result may still be very damaging

Block ciphers and message authentication.

Story:

- The aforementioned attacks
  utilise the fact that flipping a single bit
  in a ciphertext generated via a stream cipher
  results in the flipping of the same bit in the decrypted plaintext
- In contrast,
  block ciphers seem to be significantly harder to attack

- [unclear to me]
  This is because [in some cases]
  a block cipher is pseudorandm function and so
  flipping a single bit in the ciphertext of a block cipher
  results in the entire block becoming scrambled upon decryption
  Yet, this does not ensure protection against message tampering because
  the recipient must be able to (at the very least)
  detect that one of the blocks
  has ecome scrambled.
  But this ability is application dependent
  so cannot be a general solution.

- Further, ability to tamper with a message depends on
  the mode of operation being used.

  - Electronic Code Book (ECB)

    Encryption:
    - Given $m = m_1 \dots m_\ell$
      $c = \langle F_k(m_1) \dots F_k(m_\ell) \rangle$
      where F is a pseudorandom permutation

    NB: The order of the blocks can be changed
    (and the new ciphertext stays a legal ciphertext)

  - Cipher block chaining (CBC) mode

    Mode 2 — Cipher Block Chaining (CBC) mode.

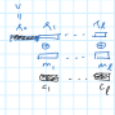
    Encryption:
    - $v \leftarrow \{0,1\}^n$
    - $c = \langle v, F_k(v \oplus m_1), F_k(c_1 \oplus m_2), \dots F_k(c_{\ell-1} \oplus m_\ell) \rangle$

    NB: By flipping any bit of v, one can flip the bit of the first block
    often the first part of the message contains the "header"
    that contains sensitive information

The book does not comment on the following:

- Output feedback (OFB) mode

Encryption:
- $v \leftarrow \{0,1\}^n$
- generate a stream v as follows
  - $r_0 := v$
  - $r_i := F_k(r_{i-1})$
- $c_i := m_i \oplus r_i$



- Counter Mode (CTR) mode

Encryption:
- $\text{ctr} \leftarrow \{0,1\}^n$
- $r_i := F_k(\text{ctr} + i)$ (addition is modulo 2^n)
- $c_i := r_i \oplus m_i$

- Thus, **message authentication codes** are needed—
  i.e. additional mechanisms are needed to ensure that
  communicating parties can detect if
  the messages being exchanged have been
  tampered with

§ 4.3 Message Authentication Codes — Definitions

Story:
- The aim of a message authentication code
  is to prevent an adversary from
  modifying a message
  sent from one party to another
  without the parties detecting that
  a modification has been made.

- As in the case of encryption,
  such a task is only possible
  if the communicating parties have some secret
  that the adversary does not know
  (else nothing can prevent an adversary from
  impersonating the party sending the message)

- We therefore assume the parties share the same secret key—thus
  the notion of message authentication code
  belongs to the world of *private-key cryptography*

Informal definition: Message Authenticat Code (MAC)

It is an algorithm that acts on a message
The output is called a MAC tag
(this is sent, along with the message).

Security: Requires no adversary can generate a valid MAC tag
(on any message that was not sent by the
legitimate communicating parties)

Story: We first define the syntax of what a MAC is and then show the security.

Definition 4.1 (**MAC—syntax**)
A message authentication code or MAC is a tuple of
probabilistic time algorithms: $(\text{Gen}, \text{Mac}, \text{Vrfy})$ fulfilling the following:

1. Gen: on input 1^n, outputs $k \leftarrow \{0,1\}^n$
2. Mac: (generates MAC tags)
   Input: (key and a message) $k \in \{0,1\}^n$, $m \in \{0,1\}^*$
   Output: (tag) $t \in \{0,1\}^*$

The value t is called the **MAC tag**.
(i.e. maps a key and a message (k, m) to a tag t).

3. Vrfy:
   Input: (key, message and tag) $k \in \{0,1\}^n, m \in \{0,1\}^*$ and $t \in \{0,1\}^*$
   Output: $b \in \{0,1\}$

   (1 means valid and 0 means invalid)

4. Require that $\mathrm{Vrfy}_k\big(m, \mathrm{Mac}_k(m)\big) = 1$
   for all $m \in \{0,1\}^*, k \in \{0,1\}^n, n \in \mathbb{N}$

If there exists a function $\ell(\cdot)$ s.t. $\mathrm{Mac}_k(\cdot)$ is defined
only over messages of length $\ell(n)$ and $\mathrm{Vrfy}_k(m, t)$ outputs 0 for
every m that is not of length $\ell(n)$
then we say that $(\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy})$ is
a fixed length MAC with length parameter ℓ.

Remark:
The fourth point in Definition 4.1 can be relaxed so that
$\Pr\big[\mathrm{Vrfy}_k\big(m, \mathrm{Mac}_k(m)\big) = 1\big] > 1 - \mathrm{negl}(n)$

where negl is a negligible function and
the probabilities are taken over
the choice of k and any internal coin tosses of Mac and Vrfy.

Security of message authentication codes.

Story:
- Definition 4.1 says nothing about the security

Intuition for security
- No poly time adversary should be able to generate a
  valid MAC tag on any "new" message
  (i.e. a message not sent by the communicating parties)

Story:
- As with any definition
  we have to define the adversary's power
  and to define
  what is considered a "break" of the scheme

- An assistant to party (only the party has the secret key)
  may be able to do the following:
  - influence the messages being tagged
  - read the messages being tagged
  - have access to a list of messages and their tags

- The following captures all such scenarios

Informal Definition: Existential unforgeability against a chosen message attack
- The adversary has access to a "MAC tag oracle"
  that produces the tag corresponding to the message inputted
  using the secret key shared by the parties
- To break the scheme, it is only required that
  the adversary produces a tag for
  a "new message" (i.e. one not yet queried to the MAC tag oracle)

Story:

- Existential unforgeability refers to the fact that the scheme
  works for "any message"—does not assume any distribution over messages
- Chosen message attack:
  Because the adversary can choose the messages who's tags
  it would like to see to carry out its attack

Definition: Security game $\text{Mac-forge}_{\mathcal{A},\Pi}(n)$

1. $k \leftarrow \{0,1\}^n$
2. $(m,t) \leftarrow \mathcal{A}^{\text{Mac}_k(\cdot)}(1^n)$ &
   let Q denote the queries asked by $\mathcal{A}$
   during the execution.
3. $\text{Out} = 1$ if $m \notin Q$
   $\text{Vrfy}_k(m,t) = 1$.
   0 else

Question: Shouldn't we give access to the verification oracle as well?

Definition 4.2

A message authentication code $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$
  is existentially unforgeable under an adaptive chosen-message attack
    (or just secure)
  if for all poly time adversaries $\mathcal{A}$
    there is a negligible function negl s.t.
      $\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n)$

NB:

- a message authentication code can always be broken
  with negligible probability
    (thus, there's no hope of ensuring the adversary wins with prob 0)

- To see this, let $q(\cdot)$ be a polynomial denoting
  the length of the MAC tags for the scheme
    (i.e. for a key of length n and message m
      the output tag t is of length at most $q(|m| + n)$.

- Then, a naïve attack that works for any scheme
  is to randomly guess and this works with probability $2^{-q(|m|+n)}$

- While the attack is trivial, it gives us a
  lower bound on the required length of the tag
  In particular, it should be super-logarithmic; otherwise
  $q = \mathcal{O}(\log n)$ would mean $2^{-q} = 1/\text{poly}$
    and this probability of success can be amplified to be constant
      (repeat and union bound).

Replay attacks and message authentication codes

Story:

- User Alice, sends her bank an order to
  transfer $1,000 from her account to Bob's account.

- Alice is the legitimate user
  and so she also applies a MAC code
    (so the bank knows it is authentic)

- Bob is unable to intercept the message and
  change the sum to $10,000
    because this would involve forging the MAC scheme

- However, Bob could simply send the same message
  ten times repeatedly!
  If the bank accepts all these messages,
  then $10,000 will be transferred to Bob's account.

- Such an attack is called a **replay attack**
  and the MAC mechanism by itself
  does not prevent it

- Rather, the applicatino using the MAC is responsible for preventing replays
  - This is because the legitimacy or illegitimacy of replays
    depends on the application
  - Further, this cannot be solved by considering an
    isolated message—rather the context and history must be accounted for
  - Thus, it is left to higher-level applications

- Two possible techniques for preventing replays are
  using (a) unique sequence numbers in transactions
  (apply the MAC tag to both the message and the sequence number)
  and
  (b) timestamps.

§ 4.4 Constructing Secure Message Authentication Codes

Story:

- A natural tool for constructing MACs is
  a pseudorandom function

- Intuitively,
  if the MAC tag t is obtained by
  applying a pesudorandm function to the message m
  then forging a MAC involves guessing the input/output behaviour
  of a pseudorandom function

- More formally,
  we know that the probability of guessing the value of a random function
  on an unobserved point is 2^{-n} (when the output length is n).
- Therefore,
  the probability of guessing such a value from a pseudorandom function
  can only be negligibly different.

- Technicality: recall our pseudorandom functions only work with
  inputs of fixed length n (matching the length of the key)
  - We therefore first construct fixed-length MAC with length parameter $\ell(n) = n$.

Construction 4.3 | Fixed-length MAC

Let $F: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be a function such that for every k
$F_k(\cdot)$ maps n-bit strings to n-bit strings.

Define a fixed-length MAC as follows:

- $\mathrm{Gen}(1^n)$: upon input 1^n, choose $k \leftarrow \{0,1\}^n$

- $\mathrm{Mac_k(m)}$: upon input $k \in \{0,1\}^n$ and message $m \in \{0,1\}^n$
  compute $t = F_k(m)$. (If $|m| \neq |k|$ then output $\perp$)

- $\mathrm{Vrfy}_k(m,t)$: upon input key $k \in \{0,1\}^n$, message $m \in \{0,1\}^n$ and a tag $t \in \{0,1\}^n$
  output 1 iff $t = F_k(m)$
  (if the lengths are incorrect, then output 0).

Theorem 4.4

Assume that the function F used in Construction 4.3 (above) is a pseudorandom function.
Then, Construction 4.3 is a fixed-length message authentication code
with length parameter $\ell(n) = n$ that is
secure (existentially unforgeably under chosen message attacks).

Proof

Strategy:

- First analyse by assuming a truly random function
- then show that replacing random with pseudorandom has no effect.

Let:

- $\mathcal{A}$ be a PPT adversary and
- $\epsilon(\cdot)$ be a function so that

$$\Pr\left[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1\right] = \epsilon(n) \qquad [4.1]$$

Goal: We show that this implies the existence of a
PPT alogirthm that can distinguish pseudorandom fn from a random fn
with advantage $\epsilon(n)$

NB: Thus, $\epsilon(n)$ must be negligible.

Consider: Message authentication code $\widetilde{\Pi}$
that is the same as Π (in Construction 4.3)
except
a truly random function f_n is used
instead of a pseudorandom function F
(As before, NB, this is not a "legal MAC" because it is
not efficient. Neverthless, this is used for the sake of the proof only.)

NB: It is easy to see that

$$\Pr\left[\text{Mac-forge}_{\mathcal{A},\widetilde{\Pi}}(n) = 1\right] \leq 1/2^n \qquad [4.2]$$

because for any $m \notin Q$, the value $t = f_n(m)$ is
uniformly distributed in $\{0,1\}^n$
from the point of view of the adversary $\mathcal{A}$.

Story:

- We now construct a PPT distinguisher D that is
given an oracle
(viz. either of a pseudorandom function or
a truly random function)
and works as follows.

Construction: Distinguisher $D^{\mathcal{O}(\cdot)}(1^n)$ (oracle)

(a) $(m, t) \leftarrow A^{\mathcal{O}}(1^n)$

(b) output = 1 if $m \notin Q$ & t is valid, i.e. $\mathcal{O}(m) = t$.

0 else.

Claim: It follows that (using (4.1) & (4.2))

$$\Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] = \Pr\left[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1\right] = \epsilon(n)$$

&

$$\Pr\left[D^{f_n(\cdot)}(1^n) = 1\right] = \Pr\left[\text{Mac-forge}_{\mathcal{A},\widetilde{\Pi}}(n) = 1\right] \leq \frac{1}{2^n}$$

Therefore,

$$\left|\Pr\left[D^{F_k(\cdot)}(1^n)=1\right] - \Pr\left[D^{f_n(\cdot)}(1^n)=1\right]\right| \geq \epsilon(n) - \frac{1}{2^n}.$$

NB: Since F is a pseudorandom function
$\epsilon(n) - 2^{-n}$ must be neglible and thus
ϵ must be negligible (as stated in the goal above)

Conclusion: $\mathcal{A}$ succeeds in Mac-forge with at
most negligible probability
and therefore Construction 4.3 is
existentially unforgeable under chosen message attacks.

□

Variable-length message authentication codes.

domain extension

Story:

- Construction 4.3 is important in that
  it shows a general paradigm for constructing
  secure message authentication codes

- That is, it demonstrates that
  any pseudorandom function suffices.

- However, in its current form,
  this construction is only capable of dealing with messages of
  fixed length—a limitation that is unacceptable
  in many if not most
  applications.

- Thus,
  we show how a general (variable length) MAC
  can be constructed from a fixed-length one.

Remark:
The construction is not very efficient
(and unlikely to be used in practice—there are other more efficient schemes)
nonetheless,
we include the construction for its simplicity.

Story:

- Practical schemes appear later in § 4.5 and 4.7

- We first chalk out some simple ideas

- In all subsequent constructions,
  the idea is to break the message into blocks and
  apply a pseudorandom function to the blocks
  in some way

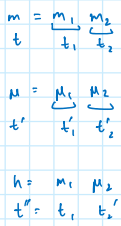
1. Apply a pseudorandom function to the first block:
   This clearly is not a secure MAC because
   nothing prevents an adversary from
   changing all the other blocks apart from the list

2. Exclusive-OR all of the blocks and apply a pseudorandom function to the result:
   In this case, all an adversary needs to do is to
   change the message such that
   the XOR of the blocks does not change
   (thus, implying that the MAC tag remains the same)

3. Apply a pseudorandom function to each block separately and output the result
   This is similar to ECB mode in § 3.6.4
   no blocks can be easily modified but
   blocks can be removed/repeated and their order interchanged
   Thus, this method is not secure.
   We also note that
   blocks from different messages can be
   combined into a new message

$m = m_1 m_2$, $t = t_1 t_2$

Thus, this method is not secure.
We also note that
blocks from different messages can be
combined into a new message.



Story:

- The actual construction also
  breaks the message into blocks
  and applies the pseudorandom function
  to each block
- However, this must be done carefully to ensure
  that the blocks cannot be rearranged and
  that the blocks from different tags
  cannot be intertwined
- This is achieved by including additional information in every block
  i.e. (in addition to part of the message)
  each block contains
  - an index of its position in the series
    (prevents rearrangement of the blocks)
  - the same random identifier
    (prevents blocks from different signatures from being combined)
  - total number of blocks
    (so that blocks cannot be dropped from the end of the message)
- Here's the construction

Construction 4.5 Variable-length MAC

F has the syntax of keyed pseudorandom function, i.e.
Let $F: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be a function such that
for every $k \in \{0,1\}^n$, $F_k: \{0,1\}^n \to \{0,1\}^n$.

Define a variable length MAC as follows:

- $\mathrm{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathrm{Mac}_k(m)$:
  Input: $k \in \{0,1\}^n$, $m \in \{0,1\}^*$ of length at most $2^{\frac{n}{4}-1}$
  Parse: m into d blocks of $m_1 \dots m_d$ of length $n/4$ each
  (In order to ensure unique encoding, the last block is padded with 10^*)
  Choose: $r \leftarrow \{0,1\}^{n/4}$
  Compute: (for each $i \in \{1 \dots d\}$)
  $t_i = F_k(r||d||i||m_i)$
  where i and d are uniquely encoded into
  strings of length $n/4$ and $||$ denotes concatenation
  (NB: $i \le d \le 2^{\frac{n}{4}}$ so both i and d need at most $n/4$ bits)
  Finally, output the tag $t = (r, t_1 \dots t_d)$.
- $\mathrm{Vrfy}_k(m,t)$:
  Input: k, m, t (key, message and tag)
  Run: The Mac_k algorithm, except that instead of using a random r
  use the r that appears in the tag t.
  Output: 1 if the tags match

Theorem 4.6

Assume that the function F used in Construction 4.5 is a pseudorandom function.
Then, Construction 4.5 is a message authentication code
that is secure
(i.e. existentially unforgeable under chosen message attacks).

Proof:

Story:

- Intuition
  - If the random identifier r is different
    in every signature that the adversary recieves from the oracle
    then the forgery must either contain a
    new identifier
    or
    it must somehow maniulate the blocks of a signed message
  - In both cases the adversary must guess the output of the
    pseudorandom function at a "new point".

Notation:

- Let $\mathcal{A}$ be a PPT adversary and let $\epsilon(\cdot)$ be a function s.t.
  $\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] = \epsilon(n)$

Strategy:

- We show that this implies the existence of a PPT algorithm
  that can distinguish the pseudorandom function
  from a random one
  with advantage at least $\epsilon(n) - \text{negl}(n)$ for a negligible function $\text{negl}(\cdot)$
- Thus, this would imply that $\epsilon(\cdot)$ must be negligible (as required).

Notation:

- Let a message authentication code $\widetilde{\Pi}$ be the same as
  Π (in Construction 4.5) except that
  a truly random function f_n is used
  instead of the pseudorandom function F.

Goal 1:

Show that
$\Pr[\text{Mac-forge}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \leq \text{negl}(n)$
for a negligible function $\text{negl}(\cdot)$.

Notation:

- Let Q be the set of queries made by $\mathcal{A}$ in Mac-forge with
  $\widetilde{\Pi}$ and let (m, t) be its output.

Story:

- We analyse the probability that $m \notin Q$ and yet
  t is a valid MAC tag for m.

Notation:

- Parse $t = (r, t_1 \dots t_d)$.

We have the following cases:

1. The identifier r appearing in the tag t output by $\mathcal{A}$
   is different from all identifiers obtained by $\mathcal{A}$
   from its MAC oracle during the experiment.

   NB1: This implies the funciton f_n was
   never applied to a block of the form
   $(r,\star,\star,\star)$ during Mac-forge with $\widetilde{\Pi}$.

   NB2: Since f_n is truly random
   it follows that the probability that $\mathcal{A}$ succeeds
   in guessing any single t_i is at most 2^{-n}.

   (It actually needs to successfully guess all the $t_1 \dots t_d$ values
   because the new identifier r must appear in all the blocks
   Nonetheless, it suffices to bound its success by 2^{-n})

2. The identifier r appearing in the tag t output by $\mathcal{A}$
   appears in exactly one of the MAC tags obtained by $\mathcal{A}$
   from its MAC oracle during the experiment.

   Notation: Denote by m' the message that $\mathcal{A}$ queried to its oracle
   for which the reply t' contained the identifier r.

   NB: Since $m \notin Q$, it holds that $m \neq m'$ where
   m is the message output by $\mathcal{A}$.

   Notation: Let d and d' be the
   two number of blocks in the parsing of m and m', respectively.

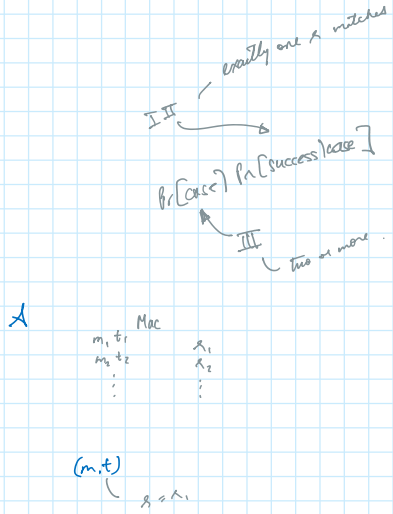
   There are two subcases

   (a) Case 1: $d = d'$.
   NB: The message content of one of the blocks must be different
   (i.e. for some i it must hold that $(m_i, i) \neq (m'_i, i)$
   where i denotes the index of the different block)

   NB2: As in the previous case, this means
   the random function f_n was never applied to a block with
   content (r, d, i, m_i) during Mac-forge with $\widetilde{\Pi}$

   and so $\mathcal{A}$ can succeed in guessing t_i with
   probability at most 2^{-n}.

   (b) Case 2: $d \neq d'$.
   NB: In this case, each block in the parsed message m
   is of the form $(r, d, \star, \star)$.
   NB2: However, r was only used in generating the MAC for
   m' of length d'.
   NB3: Thus, f_n was never applied to a
   block of the form $(r, d, \star, \star)$ during the experiment
   (The function f_n was only applied to blocks with a different r'
   or of the form $(\mathrm{r}, \mathrm{d}', \star, \star)$)
   Conclusion: As above, this means that $\mathcal{A}$ can succeed with
   probability at most 2^{-n}.



3. The identifier r appearing in the tag t
   output by $\mathcal{A}$ appears in
   two or more of the MAC tags
   (obtained by $\mathcal{A}$ from its MAC oracle during the game)

   This case appears with negligible probability—
   we show two MAC tags (generated legally)
   have the same identifier with at most negligibly probability

   NB1: Length of a random identifier is $n/4$.
   NB2: For N messages, probability at least two MAC tags
   have the same identifier is
   $C_2^N \cdot 2^{-\frac{n}{4}} = \mathcal{O}(N^2)/2^{n/4}$
   number of pairs times times the pair has the same identifier

   NB3: N is poly, thus the probability is negligible as required.

Conclusion
- The above analysis covers all possible cases
- Thus, $\mathcal{A}$ can succeed in Mac-forge with $\widetilde{\Pi}$
  with at most negligible probability
  (proving Eq. 4.4)

Goal 2: Construct a distinguisher (that wins with probability $\epsilon(\cdot) + \mathrm{negl}$)

Story:

- The rest of the proof
  builds a distinguisher (exactly as in the proof of Theorem 4.4)
- The only difference is that the distinguisher carries out the parsing first
  and then uses its oracle function.

Claim:
$|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f_n(\cdot)}(1^n) = 1]| \geq \epsilon(n) - \text{negl}$

and so $\epsilon(\cdot)$ must be negligible as required.

□

4.5 CBC–MAC

Story:

- Theorem 4.6 above provides a simple proof that
  it is possible to construct MACs for messages of any length
  (given only pseudorandom functions that work
  for a fixed input/output length)

- Thus, for example,
  it demonstrates that it is possible to use
  block ciphers (i.e. pseudorandom functions)
  as a basis for constructing secure MACs.

- The problem, however, with the construction is that
  in order to compute a MAC tag on
  a message of length $\ell \cdot n$ (ℓ is the number of blocks; n is the size of block)
  (the scheme parses the message as $4\ell \cdot n/4$)
  it is necessary to apply the block cipher 4ℓ times.

  - More seriously, the size of the MAC tag is $4\ell n$.

- The CBC-MAC construction is based on the CBC mode of encryption
  and is widely used in practice.

  - Construction: similar to Construction 4.3
    (in that the message is broken up into blocks
    and a block cipher is then applied)

  - However,
    to compute a tag on a message of length $\ell \cdot n$
    where n is the size of the block
    the block cipher is applied only ℓ times.

  - Further,
    the size of the MAC tag is only
    n bits (i.e. a single block).

  - CAVEAT: This basic scheme is **not secure** in the **general case**.
    (Recall: CBC private key encryption is secure)

Construction 4.7 Basic CBC-MAC (fixed length)

The basic CBC-MAC construction is as follows:
Let $F: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be a function
(supposed to be pseudorandom function).

- $\text{Gen}(1^n)$: choose a uniformly distributed $k \leftarrow \{0,1\}^n$
- $\text{Mac}_k(m)$:
  Input: $k \in \{0,1\}^n$ (key) and $m \in \{0,1\}^{\ell \cdot n}$ (message)

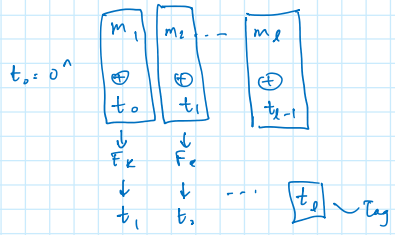
Procedure:

(1) Let $m = m_1 \dots m_\ell$ where m_i is of length n
and set $t_0 = 0^n$

(2) For $i = 1$ to ℓ
set $t_i \leftarrow F_k(t_{i-1} \oplus m_i)$ using F

(3) Output t_ℓ

- $\text{Vrfy}_k(m, t)$:
  Input: $k \in \{0,1\}^n$ and $m \in \{0,1\}^{\ell \cdot n}$ and tag t of length n
  Output: 1 if $t = \text{Mac}_k(m)$ and 0 otherwise.



Theorem 4.8
[Me: It should be ok to have ℓ should be a function of n]

Let ℓ *be any fixed value*.
If F is a pseudorandom function s.t.
for every $k \in \{0,1\}^n$
the function $F_k: \{0,1\}^n \to \{0,1\}^n$
then **Construction 4.7** is a
fixed-length MAC with
length parameter $\ell \cdot n$
that is secure (viz. existentially unforgeable under a chosen-message attack).

Story:

- The proof of Therome 4.8 is very involved
  and thus omitted.
- Stress: Construction 4.7 is secure only when
  the length of the message is fixed.
- The advantage of this construction over Construction 4.3 (worked with only length n)
  is that any length can be chosen
  (as long as it is fixed (i.e. the function $\ell(\cdot)$)

  Thus, we are not limited by the input/output length of the
  pseudorandom function.

- NB: It is not really necessary to take the length
  to be a multiple of n as long as padding is used

CBC-MAC vs CBC encryption

There are two differences b/w the basic CBC-MAC and the CBC mode of encryption

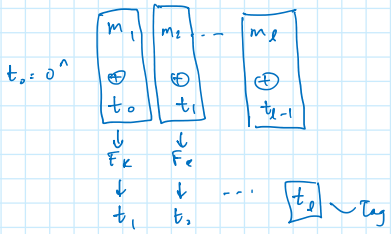
Me: This is very interesting!

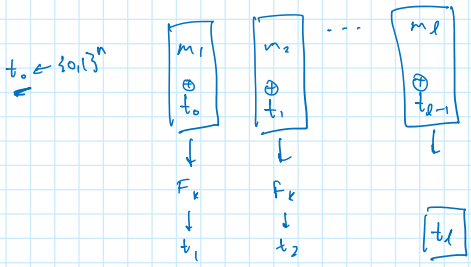
1. CBC encryption uses a random initial vector (r) and this was
   crucial for obtaining security.
   In contrast,
   CBC-MAC uses no initial vector (r) and this is
   also crucial for obtaining security
   (i.e. a CBC MAC with a random initial vector (r),
   is not secure MAC).

2. In CBC encryption
   all blocks are output by the encryption algorithm
   whereas in CBC MAC
   only the last block is output.

   Not just a technical point:
   outputting all blocks is needed to enable decryption (for CBC encryption)
   while for a MAC, doing this makes the MAC insecure!

working out the attacks when there's a random initial vector





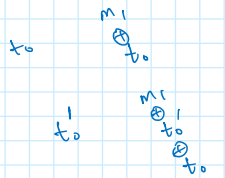
$(t_0, t_\ell) \leftarrow Enc(m)$

$(t_0', t_\ell') \leftarrow Enc(m')$

$t_\ell \oplus t_\ell' = t_{\ell-1}$

$t_\ell = F_k(m_\ell \oplus t_{\ell-1})$

$t_\ell' = F_k(m_\ell \oplus t_{\ell-1}')$



$(t_0, t_\ell) \leftarrow Enc(m_1 \cdots m_\ell)$

t_0' t_ℓ' m'

general ℓ attack

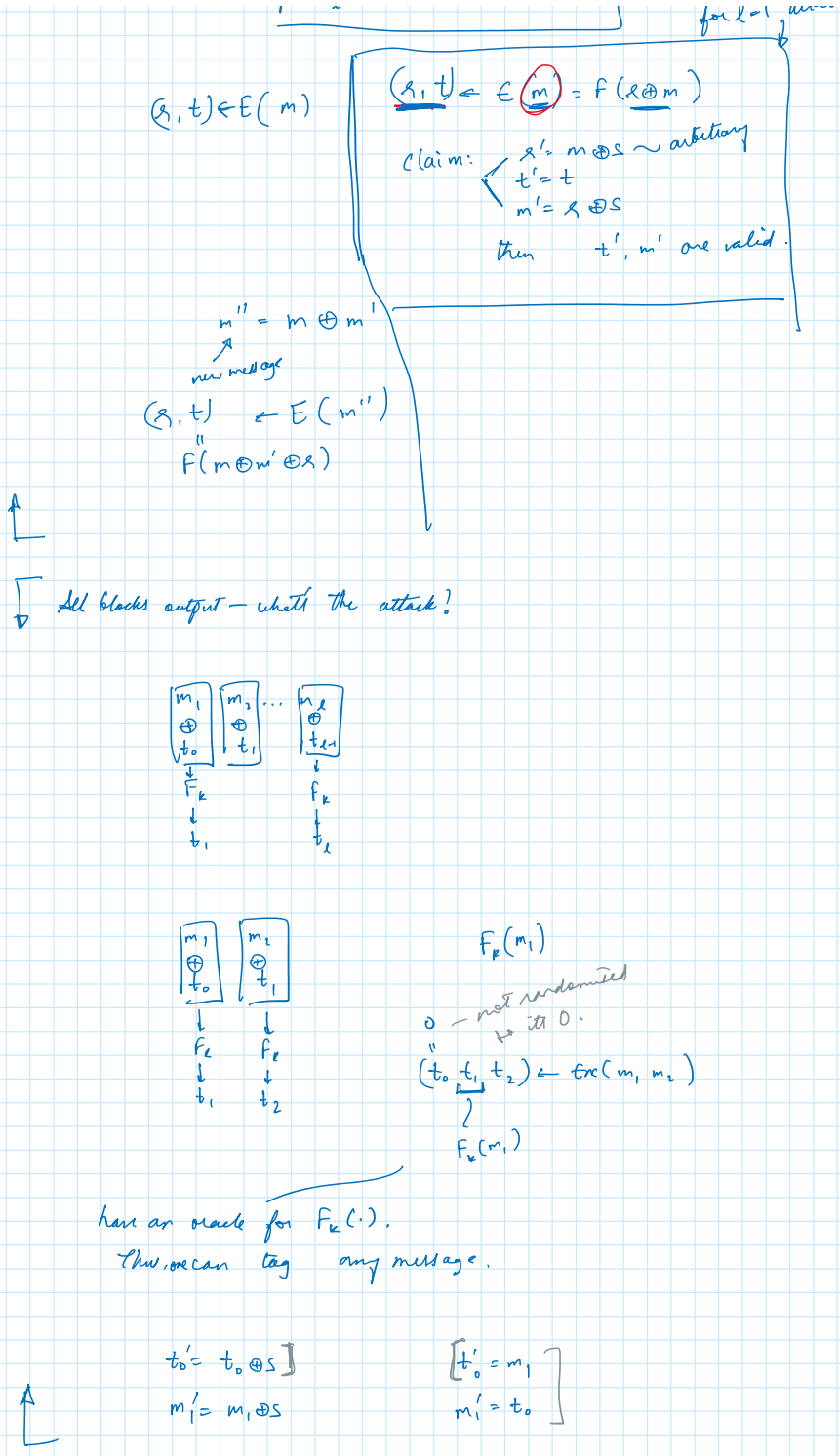
$m' := m_1 \oplus s, m_2 \cdots m_\ell$

$t_0' = t_0 \oplus s$

$t_\ell' = t_\ell$

for $\ell-1$ attack

$(r, t) \leftarrow F(m') = F(r \oplus m)$



Story:

- This is a good example of the fact that
  harmless-looking modifications to
  cryptographic constructions can render them insecure
  - Crucial to always implement the
    exact construction and
    not some slight variant (unless there's a proof)
- Further
  it is crucial to understand the cosntructions.

  E.g.
  In many cases a crypto library

provides a programmer with a
"CBC function"
However, it does not distinguish b/w use of this function
b/w authentication and encryption.

Construction 4.7 and variable-length messages

Story:

- Theorem 4.8 states that
  the basic CBC-MAC construction is
  only secure for fixed-length messages

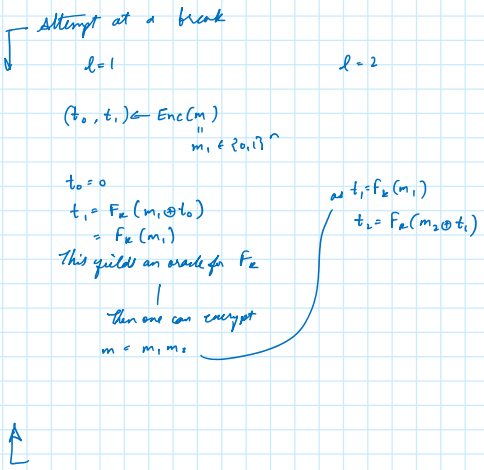
- However, in the general case of variable-length messages
  it is easy to generate forgery for the basic CBC construction

  [Left as an exercise—find an attack; TODO]

  - Remark: the attack that does exist,
    does not provide much control
    to the adversary over the
    content of the forged message.

  - Babble: Neverthless, It's crucial that
    cryptographic constructions be secure for all applications
    and we have no way of knowing
    that the attack described above will not harm any appliaciton.

    We also don't know if more devastating attacks exist—
    when one attack is known,
    often many more are possible.



Secure CBC-MAC for variable-length messages

Story:

- To obtain a secure MAC via CBC construction for variable messages,
  Construction 4.7 must be modified

- This can be done in a number of ways

- The following have been proven secure:

1. Apply the pseudorandom function (block cipher) to the block length ℓ
   of the input message (in order to obtain a key k_ℓ).
   Then compute the basic CBC-MAC using the key k_ℓ
   and send the resulting tag along with the block length.

2. *Prepend* the message with its block length ℓ <This is later
   and then compute the CBC-MAC referred to as 1
   (the first block contains the number of blocks to follow).
   We stress that *appending* the message with its block length
   is *not secure*!

3. Choose two different keys $k_1 \leftarrow \{0,1\}^n$ and $k_2 \leftarrow \{0,1\}^n$ < and this as 2
   Then compute the basic CBC-MAC using k_1
   let t_1 be the result.
   The output MAC-tag is defined to be $t = F_{k_2}(t_1)$.

   [Me: not really clear what is going on in this case;
   but my simple attack does not work]

NB1: The third option has the advantage that it is
not necessary to know the message length
before starting to compute the MAC.

NB2: Disadvantage is that it requires two keys

However, at the expense of two additional applications
of the pseudorandom function
it is possible to store a single key k and
then derive keys
$k_1 = F_k(1)$ and $k_2 = F_k(2)$
at the beginning of the computation.

★ [Book changed—second edition, 2014 version]

These were missing in our previous edition.

Strong MACs. As defined, a secure MAC ensures that an adversary cannot generate a valid tag on a new message that was never previously authenticated. But it does not rule out the possibility that an attacker might be able to generate a new *tag* on a previously authenticated message. That is, a MAC guarantees that if an attacker learns tags $t_1, \ldots$ on messages $m_1, \ldots$, then it will not be able to forge a valid tag t on any message $m \notin \{m_1, \ldots\}$. However, it may be possible for an adversary to "forge" a *different* valid tag $t'_1 \neq t_1$ on the message m_1. In general, this type of adversarial behavior is not a concern. Nevertheless, in some settings it is useful to consider a stronger definition of security for MACs where such behavior is ruled out.

Formally, we consider a modified experiment Mac-sforge that is defined in exactly the same way as Mac-forge, except that now the set $\mathcal{Q}$ contains *pairs* of oracle queries and their associated responses. (That is, $(m,t) \in \mathcal{Q}$ if $\mathcal{A}$ queried $\mathsf{Mac}_k(m)$ and received in response the tag t.) The adversary $\mathcal{A}$ succeeds (and experiment Mac-sforge evaluates to 1) if and only if $\mathcal{A}$ outputs (m,t) such that $\mathsf{Vrfy}_k(m,t) = 1$ and $(m,t) \notin \mathcal{Q}$.

Canonical verification. For deterministic message authentication codes (that is, where Mac is a deterministic algorithm), the canonical way to perform verification is to simply re-compute the tag and check for equality. In other words, $\mathsf{Vrfy}_k(m,t)$ first computes $\tilde{t} := \mathsf{Mac}_k(m)$ and then outputs 1 if and only if $\tilde{t} = t$. Even for deterministic MACs, however, it is useful to define a separate Vrfy algorithm in order to explicitly distinguish the semantics of *authenticating* a message vs. *verifying* its authenticity.

Formally, we consider a modified experiment Mac-sforge that is defined in exactly the same way as Mac-forge, except that now the set $\mathcal{Q}$ contains *pairs* of oracle queries and their associated responses. (That is, $(m,t) \in \mathcal{Q}$ if $\mathcal{A}$ queried $\mathsf{Mac}_k(m)$ and received in response the tag t.) The adversary $\mathcal{A}$ succeeds (and experiment Mac-sforge evaluates to 1) if and only if $\mathcal{A}$ outputs (m,t) such that $\mathsf{Vrfy}_k(m,t) = 1$ and $(m,t) \notin \mathcal{Q}$.

DEFINITION 4.3 *A message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is* strongly secure, *or a* strong MAC, *if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there is a negligible function* negl *such that:*

$$\Pr[\mathsf{Mac\text{-}sforge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

PROPOSITION 4.4 *Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be a secure MAC that uses canonical verification. Then Π is a strong MAC.*

[trivial because canonical verification works only for deterministic schemes]

§ 4.4.2 * Proof of Security

Story:

- We prove security of different variants of CBC-MAC
- We begin by summarising the result
  and then give details of the proof
- The proof is quite involved
  intended for advanced readers [Like us □]

Convention:

Fix a keyed function F
that for security parameter n
maps $\mathcal{K} \times \mathcal{M} \to \{0,1\}^n$ where $\mathcal{K} = \mathcal{M} = \{0,1\}^n$

Definition:

Keyed function CBC:
for security parameter n
input: $\{0,1\}^n$ (key) and
$(\{0,1\}^n)^*$ (inputs i.e. strings whose length is a multiple of n)

Definition:

$$CBC_k(x_1 \dots x_\ell) := F_k(F_k(\dots F_k(F_k(x_1) \oplus x_2) \oplus \dots) \oplus x_\ell)$$

where $|x_1| = \dots = |x_\ell| = |k| = n$

(We leave CBC undefined on the empty string)

nb: note that CBC is exactly the same as
CBC-MAC, although here we consider
Inputs of different lengths

Def^n. Prefix free:

A set $P \subset (\{0,1\}^n)^*$ is Prefix Free if

def: Prefix Free

A set $P \subset (\{0,1\}^n)^*$ is Prefix Free if

(a) it does not contain the empty string &

(b) no string $x \in P$ is a prefix of any other string $x' \in P$.

Story: Will prove the following.

Theorem 4.13

Suppose: F is a pseudorandom f^n
restrict inputs to being prefix-free.

Then: CBC is also a pseudorandom f^n.

More formally, for all PPT distinguishers D
that query their oracle on a
prefix free set of inputs,
there exists a negligible f^n negl st.

$$\left| Pr[D^{CBC_k(\cdot)}(1^n) = 1] - Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq negl(n)$$

where k is chosen uniformly from $\{0,1\}^n$ &
f is chosen uniformly from
the set of f^ns mapping
$(\{0,1\}^n)^* \to \{0,1\}^n$

(i.e. the value of f at each point is
uniform & independent of the
values of f at all other points)

Story:

Story:

- Thus, we can convert a pseudorandom function *F*
  for fixed-ength inputs into a
  pseudorandom function CBC
  for arbitrary length inputs
  (subject to a constraint on which inputs can be queried!)

- To use this for message authentication
  we adapt the idea of Construction 4.5 as follows:

CONSTRUCTION 4.5

Let F be a pseudorandom function. Define a fixed-length MAC for messages of length n as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, output the tag $t := F_k(m)$. (If $|m| \neq |k|$ then output nothing.)
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^n$, and a tag $t \in \{0,1\}^n$, output 1 if and only if $t \stackrel{?}{=} F_k(m)$. (If $|m| \neq |k|$, then output 0.)

A fixed-length MAC from any pseudorandom function.

Construction [MAC for arbitrary length messages]:

- to authenticate a message m
- first apply some encoding function
  encode

encode
to obtain a non-empty string
$\text{encode}(m) \in (\{0,1\}^n)^*$
then output the tag
$\text{CBC}_k(\text{encode}(m))$

Remarks

- For this to be secure (cf. the proof of Theorem 4.6)
  the encoding needs to be prefix-free
  namely
  to have the property that for
  any distinct (legal) messages m_1, m_2
  the string $\text{encode}(m_1)$ is not a prefix of $\text{encode}(m_2)$.

- This implies that
  for any set of (legal) messages $\{m_1 \dots\}$
  the set of encoded messages
  $\{\text{encode}(m_1), \dots\}$ is prefix-free.

Remarks 2:

We now examine two concrete applications of this idea

- Fix ℓ and let
  the set of legal messages be $\{0,1\}^{\ell(n)\cdot n}$
  Then, one can take the encoding
  $\text{encode}(m) = m$
  and observe that it is prefix-free
  (since one string cannot be a prefix of another string
  of the same length)
  This is exactly (basic) CBC-MAC
  and the above property implies that
  basic CBC-MAC is secure for messages of
  any fixed length (cf Theorem 4.12).

- One way of handling arbitrary-length (non-empty) messages
  (technically, messages of length at most 2^n) is
  to encode a string $m \in \{0,1\}^*$ by
  prepending its length $|m|$ (encoded as an n-bit string)
  (and then appending 0s to make the length a multiple of n)

  This encoding is prefix-free
  and we therefore obtain a secure MAC for
  arbitrary length messages

Story:

- The rest of this section is devoted to proving Theroem 4.13

Proof strategy:
- Replace the pseudorandom function F_k with a random function g
- Prove the result in this "ideal setting"
- A standard argument (as before) shows
  the result holds even with pseudorandom functions.

Definition: CBC_g
In proving the theorem,
we analyse CBC when it is "keyed" with a random function g, i.e.

$$\text{CBC}_g(x_1 \dots x_\ell) \coloneqq g(g(\dots g(g(x_1) \oplus x_2) \oplus \cdots) \oplus x_\ell).$$

NB: CBC_g as defined here is not efficient
(since the representation of g requires space exponential in n)

Proof strategy (resumed):

- We show that if g_n is chose uniformly from Func_n
  then CBC_g is indistinguishable from a function mapping
  $(\{0,1\}^n)^*$ to n-bit strings
  as long as a prefix-free set of inputs is queried

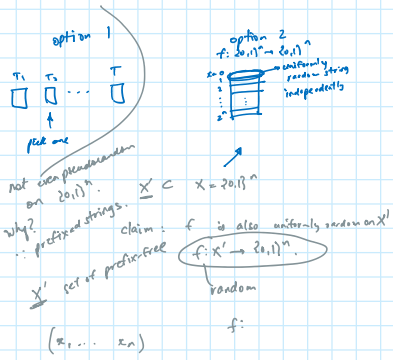
- More precisely, we show the following.

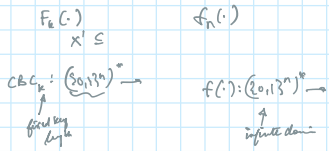
Claim 4.14

Fix any $n \geq 1$. For all distinguishers D that query their oracle
on a prefix-free set of q inputs
where the longest such input contains ℓ blocks
it holds that

$$\left|\Pr\left[D^{\mathrm{CBC}_{g_n}(\cdot)}(1^n)=1\right]-\Pr\left[D^{f(\cdot)}(1^n)=1\right]\right| \leq \frac{q^2\ell^2}{2^n},$$

where g_n is chose uniformly from Func_n, and
f is chose uniformly from the set of functions mapping
$(\{0,1\}^n)^*$ to $\{0,1\}^n$.



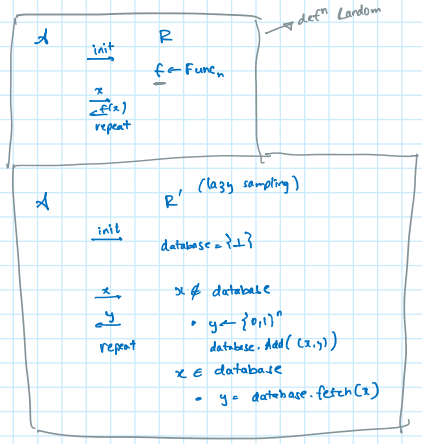


Me: One might wonder why we are using a random function g to create
another random function CBC?
The point is that for a fixed block length n
we only use a random function, g_n, from n bits to n bits
to create a new function CBC that is random
on all prefix-free strings.

So we are somehow extending the domain of the random function
to an infinite set, while still maintaining
randomness on a subset.

NB:
The claim is unconditional
and does not impose any constraints on the
running time of D.
Thus, we may take D to be deterministic.

NB2:
The above implies Theorem 4.13 using
standard technques (we have already seen).
In particular, for any D running in poly time
we must have $q(n), \ell(n) = \mathrm{poly}(n)$
and so $q(n)^2\ell(n)^2 2^{-n}$ is negligible.



Proof (of Claim 4.14):

Fix some $n \geq 1$.

Strategy: We proceed in three steps
(1) We define a notion of smoothness and
(2) Prove that CBC is smoth
(3) Then we show that smoothness implies the claim
(i.e. one can replace CBC_g with a random function
with negligible effect)

STEP 1 | NOTION OF SMOOTHNESS

Let: $P = \{X_1, \ldots X_q\}$ be a prefix-free set of
q inputs

where

< NB: we are now considering

q inputs

where

- each $X_i \in (\{0,1\}^n)^*$
- the longest string in P contains ℓ blocks.
  (i.e. each $X_i \in P$ contains at most ℓ blocks of length n).

< NB: we are now considering
q many inputs

each input has $\leq \ell$ blocks
each block of length n

NB: For any $t_1 \dots t_q \in \{0,1\}^n$

it holds that

$$\Pr\left[\bigwedge_{i=1}^{q} f(X_i) = t_i\right] = 2^{-nq}$$

recall: f arbitrary $(\{0,1\}^n)^* \to \{0,1\}^n$.

< Applying f to Xs and requiring
the resulting string matches
$t_1, \dots t_q$
(each block is of length n)

Recall:
where the prob. is over uniform choice of the f^n
f from the set of f's
$(\{0,1\}^n)^* \to \{0,1\}^n$.

Definition:

(q, ℓ, δ)-smooth CBC: if for every prefix-free set $P = \{X_1, \dots X_q\}$ (as above),

recall: ℓ constrains P as $\max_i \{\text{blocks in } X_i\} \leq \ell$

& every $t_1 \dots t_q \in \{0,1\}^n$, it holds that

$$\Pr\left[\bigwedge_{i=1}^{q} \mathrm{CBC}_g(X_i) = t_i\right] \geq (1-\delta) \cdot 2^{-nq}$$

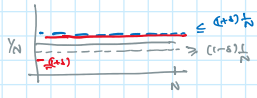
recall:
where the probability is over uniform choice of $g \in \mathrm{Func}_n$.

< For q many inputs
each input of at most ℓ blocks

encoding them with CBC
is (δ) almost the same as
encoding them with
the random function f
(as described above)

Story:

In words, CBC is smooth if
for every fixed set of input/output pairs $\{(X_i, t_i)\}$
(where the $\{X_i\}$ form a prefix- free set)
the probability that $\mathrm{CBC}_g(X_i) = t_i$ for all i
is δ-close to the probability that
$f(X_i) = t_i$ for all i
(recall: g is a random function from $\{0,1\}^n \to \{0,1\}^n$)`



STEP 2 | CBC SATISFIES THIS NOTION OF SMOOTHNESS

Claim 4.15:
CBC_g is (q, ℓ, δ)-smooth, for $\delta = q^2\ell^2 \cdot 2^{-n}$.

Proof of Claim 4.15

Rough:

Def^n: $C_g(X)$ ← set of g inputs on which g is evaluated

Definition: $C_g(X)$
Let $X \in (\{0,1\}^n)^*$ be a string of arbitrary many blocks with each block of size n.
Parse $X = x_1 \dots$ where $x_i \in \{0,1\}^n$.

Definition: $\mathcal{C}_g(X)$

Let $X \in (\{0,1\}^n)^*$ be a string of arbitrary many blocks with each block of size n.
Parse $X = x_1 \ldots$ where $x_i \in \{0,1\}^n$.

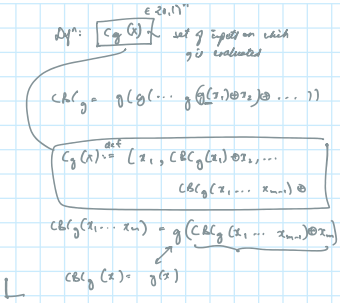
Then, $\mathcal{C}_g(X)$ denotes the
set of inputs on which g is evaluated durng
the evaluation of $\mathrm{CBC}_g(X)$,

i.e.

if $X = x_1 \ldots x_m$ (where $m \le \ell$ later)

$$\mathcal{C}_g(X) := \big(x_1, \mathrm{CBC}_g(x_1) \oplus x_2, \ldots \mathrm{CBC}_g(x_1 \ldots x_{m-1}) \oplus x_m\big)$$

I_1 I_2 ... I_m



Notn: $X \in (\{0,1\}^n)^m$

$X' \in (\{0,1\}^n)^{m'}$

$x_1 \; x_2 \ldots \; x_m$

$\mathcal{C}_g(X) = (I_1 \ldots I_m)$

$I_1 \; I_2 \; I_3 \; (I_m)$

$\mathcal{C}_g(X') = (I_1' \ldots I_{m'}')$

dependence on g

$g(I_1) \; g(I_2) \ldots g(I_{m-1})$

Def^n: Non-trivial collision in X

if $I_i = I_j$ for some $i \neq j$.

< NB: suppose $m = m'$;
collisions
do not depend on $g(I_m)$

: Non-trivial collision b/w X & X'

if (a) $I_i = I_j'$

(b) $X \neq X'$

$\Updownarrow$

$(x_1 \ldots x_i) \neq (x_1' \ldots x_j')$

(even if $i = j$) ← even one element distinct should suffice

: Non-trivial collision in P

if at least one of the following hold

(a) $\exists \; X \in P$ s.t. X has non-trivial collision

(b) $\exists \; X, X' \in P$ s.t. they have a non-trivial collision

: Coll — Event that there is a non-trivial collision in P.

Story: We show the following.

Claim: $\Pr[\, CBC_g(x_i) = t_i \mid \neg \mathrm{Coll}\,] = 2^{-n} \quad [*]$

$\Pr[\mathrm{Coll}] = \delta \le q^2 \ell^2 \cdot 2^{-n} \quad [**]$

Proving [*]

Consider: Choosing a uniform g by
choosing, one-by-one
uniform values for the
outputs of g on different inputs.

NB: Determining whether there's a
non-trivial collision b/w two strings
$X, X' \in P$ can be done as follows:

non-trivial collision b/w two strings

$X, X' \in P$ can be done as follows:

(a) choose the values for $g(I_1)$
$g(I_1')$

(when $I_1 = I_1'$, these values are the same)

(b) choose the values for $g(I_2)$
$g(I_2')$

(Recall: $I_2 = g(I_1) \oplus x_2$
$I_2' = g(I_1') \oplus x_2'$
these are fixed once I_1, I_1' are fixed)

⋮ (and so on)

until the values for $g(I_{m-1})$ & $g(I_{m-1}')$

have been chosen.

NB: (by the definition) one needn't evaluate

$g(I_m)$ & $g(I_m')$

to determine whether there're non-trivial

collisions b/w X & X'. [See NB above]

Story: One can continue this way to determine

whether Coll happened or not

by choosing values of g on all

(but the final entries)

(recall: final entry of $C_g(X)$ is I_m)

for each $C_g(X_1) \cdots C_g(X_q)$

Rough

Recall: $P = (X_1 \cdots X_q)$

$X \in P$

$X = (x_1, \ldots x_m)$ with each x_i of length n.

$C(X) = x_1, g(x_1) \oplus x_2, \ldots, g(\cdots g(g(x_1) \oplus x_2) \oplus \ldots) \oplus x_\ell$

Summary:

$g(x_1)$ | $g(I_1)$
$g(g(x_1) \oplus x_2)$ | $g(I_2)$
$g(g(g(x_1) \oplus x_2) \oplus x_3))$ | ⋮
$g(g(\cdots g(x_{\cdots}) \oplus x_m)\ldots)$ | $g(I_m)$

Mc: they argue about the prefix free requirement.

Assume: Coll has not occurred after

fixing the values of g on

various inputs as described above.

Consider: Final entries in each of $C_g(X_1) \cdots C_g(X_q)$.

(recall: $C_g(X) = (I_1 \cdots I_m)$)

This entry.

NB: These entries are all distinct

(otherwise there would have been a collision
recall: no Is can match)

Claim: Value of g at the final points had

not been evaluated (by the sampling procedure described above)

[Justif^n: This could only happen

if e.g. for $X, X' \in P$

Rough/Skip

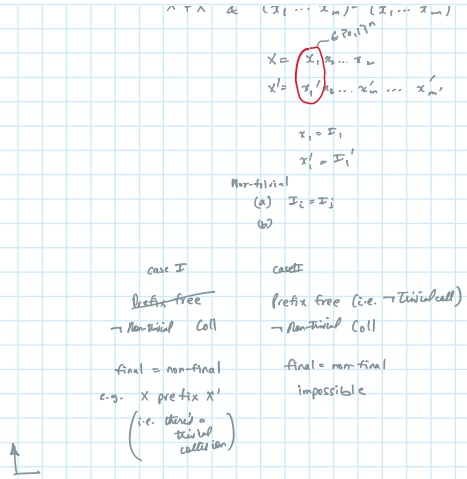
Thus, the only way this can happen is if
(last entry matching non-last entry)

$X \neq X'$ & $(x_1' \cdots x_m') = (x_1 \cdots x_m)$

$X = x_1 x_2 \ldots x_m$

$X' = x_1' x_2 \ldots x_m' \ldots x_{m'}'$

[Justif^n: This could only happen
if e.g. for $X, X' \in P$
if the last entry In of $C_g(X)$ (say)
equals some entry I_j of $C_g(X')$ (for some j).
But we assumed Coll doesn't happen.
(NB: We assumed the strings are prefix-free
so "trivial collisions" are already excluded.)



NB: Since g is a random function
the above means that
$\mathrm{CBC}_g(X_1) \dots \mathrm{CBC}_g(X_q)$
are uniform and independent of each other
(as well as other values of g that have been fixed)

Justification: This is because $\mathrm{CBC}_g(X_i)$ depends
on g evaluated at the last entry of
$C(X_i)$ which has not yet (by the sampling procedure)
been queried.
And g is a random function.

Consequence:

$\forall\ t_1 \dots t_q \in \{0,1\}^n$ it holds
$\Pr[\forall i: \mathrm{CBC}_g(X_i) = t_i \mid \neg \mathrm{coll}] = 2^{-nq}$.

Proving (**) of the claim

(Recall Claim: $\Pr[\mathrm{CBC}_g(X_i) = t_i \mid \neg \mathrm{Coll}] = 2^{-nq}$ [*]
$\Pr[\mathrm{Coll}] = \delta \leq q^2 \ell^2 \cdot 2^{-n}$ [**])

Defn: $\mathrm{Coll}_{i,j}$:
For distinct $X_i, X_j \in P$
$\mathrm{Coll}_{i,j}$ denotes the event that
there's a non-trivial collision in
X_i or X_j
OR
a non-trivial collision
b/w X_i and X_j.

NB: $\mathrm{Coll} = \bigvee_{i,j} \mathrm{Coll}_{i,j}$ — $i \neq j$

NB2: By a union bound, one has

$$P_A[\text{Coll}] \le \sum_{i,j:\, i<j} P_A[\text{Coll}_{ij}]$$

$$\le \binom{q}{2} \cdot \max_{ij} P_A[\text{Coll}_{ij}] \quad \text{(choose 2 elements)}$$

$$\le \frac{q^2}{2} \max_{ij} P_A[\text{Coll}_{ij}].$$

$A = \bigcup_j B_j$

$P_r[A] \le \sum_j P_A[B_j]$

$\frac{q \cdot (q-1)}{2} \le \frac{q^2}{2}$

Goal: Upper Bound Coll_{ij}.

NB: the probability is maximised when x & x' are both as long as possible.

∴ We assume both have l blocks.

i.e. $x = (x_1, \ldots x_l)$

$x' = (x'_1, \ldots x'_l)$

Rough:

t: largest int s.t. $(x_1, \ldots x_t) = (x'_1, \ldots x'_t)$.

NB: $t < l$ else $x = x'$.

Assume: $t > 0$ (analysis also works when $t = 0$)

Let: $(I_1, I_2, \ldots)$ $= \ell(x)$; $(I'_1, I'_2, \ldots)$ $= \ell(x')$

NB: $(I'_1, \ldots I'_t) = (I_1, \ldots I_t)$

Consider: choosing g by choosing uniform values for the outputs of g, one-by-one.

(as follows, in $2l-2$ steps).

Steps 1 Through $t-1$ (if $t > 1$):

On each step i, choose a uniform value for $g(I_i)$ thus defining I_{i+1} & I'_{i+1} (that are equal).

$g(I_{t-1}) \oplus x_t = I_t$

$I_1\ I_2 \ldots I_{t-1}\ I_t \mid I_{t+1} \ldots$

$I'_1\ I'_2 \ldots I'_{t-1}\ I'_t \mid I'_{t+1} \ldots$

(same)

$g(I_t) \oplus x_{t+1} = I_{t+1}$

$g(I'_t) \oplus x'_{t+1} = I'_{t+1}$

thus defining I_{t+1} & I'_{t+1} (that are equal).

Same

$g(I_t) \oplus x_{t+1} = I_{t+1}$

$g(I'_t) \oplus x'_{t+1} = I'_{t+1}$

$g(I_t)$

Step t: Choose a uniform value for $g(I_t)$. Thus defining I_{t+1} & I'_{t+1}.

$g(I_t) \oplus x_{t+1} = I_{t+1}$

$g(I_t) = I'_{t+1}$

Steps $t+1$ to $\ell-1$ (if $t < \ell-1$):

Choose, sequentially, uniform values for each of $g(I_{t+1}), \ldots g(I_{\ell-1})$

thus defining $I_{t+2}, \ldots I_\ell$.

Steps ℓ to $2\ell-2$ (if $t < \ell-1$):

Choose, sequentially, uniform values for each of $g(I'_{t+1}), \ldots g(I'_{\ell-1})$

thus defining $I'_{t+2}, \ldots I'_\ell$.

M/Question:
- here there may be collisions right?
- Don't we need to be consistent?

Def^n: $Coll(k)$ — event that a non-trivial collision occurs by step k.

Then: $Pr[Coll_{ij}] = Pr[\bigvee_k Coll(k)]$

$\leq Pr[Coll(1)] + \sum_{k=2}^{2\ell-2} Pr[Coll(k) \mid \overline{Coll(k-1)}]$

Prop A.9

$Pr[\bigvee_i A_i] \leq Pr[A_1] + \sum_{i=2}^{k} Pr[A_i \mid \neg A_1 \wedge \cdots \neg A_i]$

$Pr(A_1 \vee A_2) \leq Pr(A_1 \vee A_2 \mid A_1)\, Pr(A_1) + Pr[A_1 \vee A_2 \mid \neg A_1)\, Pr(\neg A_1)$

$\leq Pr(A_1) + Pr[A_2 \mid \neg A_1)$

claim: For $k < t$, $Pr[Coll(k) \mid \overline{Coll(k-1)}] = k/2^n$.

Justif^n:
- if no no non-trivial collision by step $k-1$, the value of $g(I_k)$ is chosen uniformly at random in step k.
- A non-trivial collision only happens if $I_{k+1} = g(I_k) \oplus x_{k+1}$ matches one of $\{I_1 \ldots I_k\}$

matches one of

$\{I_1 \dots I_k\}$

(which are all distinct ∵ coll(k-1) has not happened)

claim: $Pr[\text{Coll}(t) \mid \overline{\text{Coll}(t-1)}] \le 2t/2^n$

Justif^n: By similar reasoning, here there're two values I_{t+1}, I'_{t+1} to consider & we note that they cannot be equal to each other.

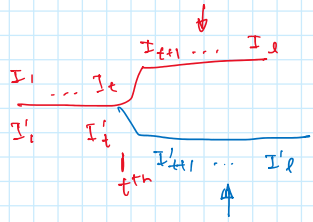
to have a collision coll(t), need to ensure I_{t+1} matches either $I_1 \dots I_t$

$g(I_t) \oplus x_{t+1} = I_{t+1}$

$g(I_t) \oplus x'_{t+1} = I'_{t+1}$ OR I'_{t+1} matches $I_1 \dots I_t$

i.e. $g(I_t) \in \{I_1 \oplus x_{t+1}, I_1 \oplus x_{t+1}, \dots I_t \oplus x_{t+1}, I_1 \oplus x'_{t+1}, \dots I_t \oplus x'_{t+1}\}$

$2t$ values



claim: for $k > t$,

$Pr[\text{Coll}(k) \mid \overline{\text{Coll}(k-1)}] = (k+1)/2^n$.

Justif^n: "arguing as before".

A | $t+1 < k < \ell - 1$ (i.e. $t < \ell - 1$)

$g(I_k) \oplus x_{k+1} = I_{k+1}$

for collision

$g(I_k) \in \{I_1 \oplus x_{k+1}, I_2 \oplus x_{k+1} \dots I_k \oplus x_{k+1}, I'_{t+1} \oplus x_{k+1}\}$

$I_1 I_2 \dots I_t \; I_{t+1} \dots I_{k-1} I_k \dots I_\ell$

$I'_{t+1} \dots I'_{k-1} I'_k$

no collision

$\therefore Pr[\text{Coll}(k) \mid \neg\text{coll}(k-1)] = \frac{k+1}{2^n}$ (these are $k+1$ elements)

B | $\ell < k < 2\ell - 2$

$g(I_k) \oplus x_{k+1} = I_{k+1}$

for collision

$g(I_k) \in \{I_1 \oplus x_{k+1}, \dots I_\ell \oplus x_{k+1}, I'_{t+1} \oplus x_{k+1} \dots I'_k \oplus x_{k+1}\}$

$Pr[\text{Coll}(k) \mid \neg\text{coll}(k-1)] = \frac{\ell + (k-t)}{2^n}$

$\le \frac{2\ell}{2^n}$

$(\because k \le \ell$, ℓ drop $t)$

Using eq^n (4.7), one has

$$\max_{i,j} Pr[\text{Coll}_{i,j}] \le Pr[\text{coll}(1)] + \sum_{k=2}^{2\ell-2} Pr[\text{coll}(k) \mid \neg\text{Coll}(k-1)]$$

$$= \frac{1}{2^n} + \sum_{k=2}^{t-1} \frac{k}{2^n} + \frac{2t}{2^n} + \sum_{k=t+1}^{\ell-1} \frac{(k+1)}{2^n} + \sum_{k=\ell}^{2\ell-2} \frac{2\ell}{2^n}$$

$I_1 = I_2 \in \{I_1\}$ ~ t^{th} collision

collision $g(I_1) \oplus x_1 = I_2$

$Pr[\text{Coll}(1)] = \frac{1}{2^n}$

$$\le \frac{1}{2^n}\left(kt^2 + 2t + \ell^2 + 2.4\ell^2\right)$$

$V_s = a_1 + \dots a_s$

$V_{s+d} = a_2 + \dots + a_s + a_{s+1}$

$= V_s + a_{s+1} - a_1$

$V_s = a_1 + a_1 + d + a_1 + 2d \dots a_1 + (s-1)d$

$= sa_1 + d\left(\sum_{i=1}^{s-1} i\right)$

$$\le \frac{12\ell^2}{2^n}$$

From eq^n (4.6), one has

$$Pr[\text{Coll}] \le \frac{q^2}{2} \max_{i,j} Pr[\text{Coll}_{i,j}]$$

$S_n := \underbrace{1+2+\dots}_{n+1}\ n$ $\quad n+1$

$$\le 6\,\frac{q^2\ell^2}{2^n} = \delta \qquad n+1 \cdot \frac{n}{2}$$

our δ is slightly different

The book is $q^2\ell^2/2^n$

$\Rightarrow Pr[\neg\text{Coll}] \ge 1 - \delta$

Recall: Eq (4.5) said

$Pr[\forall i : CBC_A(x_i) = t_i \mid \neg\text{Coll}] = 2^{-n}q$

Recall: Eq (4.5) said

$$\Pr[\forall i : CBC_g(x_i) = t_i \mid \neg coll] = 2^{-n}q$$

$\Pr[A] \geq \Pr[A|B] \cdot \Pr[B]$ + Pr...

Therefore: $\Pr[\forall i : CBC_g(x_i) = t_i] \geq \Pr[\forall i : CBC_g(x_i) = t_i \mid \neg coll] \cdot \Pr[\neg coll]$

$$\geq 2^{-n}q \cdot (1-\delta)$$

Thus, CBC_g is (q, ℓ, δ)-smooth for $\delta = 6\frac{q^2\ell^2}{2^n}$

└

Step 3 | One can replace CBC_g with a random function (with almost no effect)

Step: We now show that

smoothness implies the theorem.

Assume: • w.l.g. $\mathcal{D}$ always makes q (distinct) queries each containing at most ℓ blocks.

is it clear that the # queries cannot depend on the random f?

• $\mathcal{D}$ may choose its queries adaptively (i.e. depending on the answers to previous queries) but the set of $\mathcal{D}$'s queries must be prefix free.

Notation: • Let $X_1 \ldots X_q \in (\{0,1\}^n)^*$

$t_1 \ldots t_q \in \{0,1\}^n$.

• Then, define

$\alpha(X_1 \ldots X_q ; t_1 \ldots t_q) = 1$

$\Leftrightarrow$

$\mathcal{D}$ outputs 1 when making queries $X_1 \ldots X_q$ & getting $t_1 \ldots t_q$.

NB: α depends on $\mathcal{D}$'s output (in addition to the values it queried)

(e.g. if $\mathcal{D}$ does not make X_1 as its first query, then $\alpha(X_1, \ldots ; \ldots) = 0$.)

• Let $\vec{X} = (X_1 \ldots X_q)$

$\vec{t} = (t_1 \ldots t_q)$

Then, one has

$$\Pr\left[D^{CBC_g(\cdot)}(1^n)=1\right] = \sum_{\vec{x}\text{ prefix free};\ \vec{t}} \alpha(\vec{x},\vec{t}) \cdot \Pr\left[\forall i: CBC_g(x_i)=t_i\right]$$

$$\geq \sum_{\vec{x}\text{ prefix free};\ \vec{t}} \alpha(\vec{x},\vec{t}) \cdot (1-\delta) \cdot \Pr\left[\forall i: f(x_i)=t_i\right]$$

$$= (1-\delta) \cdot \Pr\left[D^{f(\cdot)}(1^n)=1\right]$$

(Recall: where, above, $g \leftarrow \text{Func}_n$ & $f \leftarrow \text{Func}\left[(\{0,1\}^n)^{\leq \ell} \to \{0,1\}^n\right]$)

NB: $\Pr\left[D^{f(\cdot)}(1^n)=1\right] - \Pr\left[D^{CBC_g(\cdot)}(1^n)=1\right]$

$\leq \delta \cdot \Pr\left[D^{f(\cdot)}(1^n)=1\right]$

$\leq \delta$

NB2: A symmetric argument for when D outputs 0, completes the proof.

$$\left|\Pr\left[D^A=1\right] - \Pr\left[D^B=1\right]\right| \overset{\text{goal}}{\leq} \delta$$

$$\Pr\left[D^A=1\right] - \Pr\left[D^B=1\right] \leq \delta \quad \checkmark$$

$$\Pr\left[D^B=1\right] - \underset{\text{NB1}}{=} \Pr\left[D^A=1\right] \overset{\text{to show}}{\leq \delta}$$

$$\left(1-\Pr\left[D^A=1\right]\right) - \left(1-\Pr\left[D^B=1\right]\right)$$

$$\overset{(1)}{=} \Pr\left[D^A=0\right] - \Pr\left[D^B=0\right] \leq \delta$$

(1) use B like A except D outputs 0

□

4.5 Authenticated Encryption

Story:

- In Ch 3,
  we studied how it is possible to
  obtain secrecy in the private-key setting
  using encryption.

- In this Chapter,
  we showed how to ensure
  integrity
  using message authentication codes.

- One might naturally want to achieve
  both goals simultaneously
  and
  this is now our focus

- It is best practice to always
  ensure secrecy and integrity by default
  in the private-key setting.

- Indeed, in many applications
  where screcy is required
  it turns out integrity is essential also.

- Moreover, a lack of integrity
  can sometimes lead to a
  breach of secrecy

4.5.1 Definitions

Story:

- We begin, as usual,
  by defining precisely what we wish to achieve

- At an abstract level
  our goal is to realise
  an "ideally secure" communication channel
  that provides both secrecy and integrity.

- Pursuing a definition of this sort
  is beyond the scope of this book.

- Instead, we provide a simpler set of definitions
  that treat secrecy and integrity separetly.

- These definitions
  and our subsequent analysis
  suffice for understanding the key issues at hand

(CAUTION: In contrast to encryption and MACs,
the field has not yet settled on
standard terminology and
definitions for authenticated encryption)

Intuitive definition

- Let $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a private-key encryption scheme
- We define security by
  separately defining secrecy and integrity (as already mentioned).

- The notion of screcy we consider
  is one we have seen before:

  we require that Π be secure against
  chosen-ciphertext attacks
  i.e. it be CCA-secure

  - We are concerned about chosen-ciphertext attacks here because
    we are explicitly considering an active adversary
    who can modify the data
    sent from one honest party to the other.

- Our notion of integrity will be
  essentially that of existential unforgeability under
  an adaptive chosen-message attack.

  - Since Π does not satisfy the syntax of a message authentication code
    we introduce a definition specific to this case.

Formal Definition: Unforgeable encryption game/experiment
Consider the following security game for
a private-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$
an adversary $\mathcal{A}$, and value n for the security parameter.

$\text{Enc-Forge}_{\mathcal{A},\Pi}(n)$

1. $k \leftarrow \{0,1\}^n$
2. $c \leftarrow \mathcal{A}^{\text{Enc}_k(\cdot)}(1^n)$

   let Q denote the queries asked by $\mathcal{A}$ during the execution
3. Out = 1 if $\text{Dec}_k(c) \neq \perp \wedge \text{Dec}_k(c) \notin Q$

   0 else.

< couldn't we allow
the adversary to output
m instead of c?
[stupid question!]
There's no explicit tag
we want to ensure that the
adversary cannot produce a
new valid ciphertext, that it has
not observed/requested

Definition 4.16

A private-key encryption scheme Π is **unforgeable** if
for all probabilistic poly-time adversaries $\mathcal{A}$
there is a negligible function negl such that

$$\Pr[\text{Enc-Forge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

Story:

- Paralleling our discussion about verification queries
  following Definition 4.2
  here one could also consider a stronger definition in which
  $\mathcal{A}$ is additionally given access to a decryption oracle.

Inf. Claim: The secure cosntruction we present below,
also satisfies that stronger definition.

Define a (secure) *authenticated encryption* scheme.

Definition 4.17
A private-key encryption scheme is an **authenticated encryption scheme**
if it is *CCA secure* and *unforgeable*.

4.5.2 Generic Constructions

Story:
- It may be tempting to think that
  any reasonable combination of
  a secure encryption scheme and
  a secure message authentication code
  should result in an authenticated encryption scheme.

- In this section, we show that this is not the case!

- This demonstrates that
  even secure cryptographic tools can be
  combined in such a way that

the result is insecure
and highlights once again the
importance of definitions and proofs of security.

- On the positive side
  we show how encryption and message authentication
  can be combined properly to achivee
  joint secrecy and integrity.

Notation:

- In the following, we let
  $\Pi_E = (\text{Enc}, \text{Dec})$ be a CPA secure encryption scheme and let
  $\Pi_M = (\text{Mac}, \text{Vrfy})$ denote a message authentication code
  where key generation in both schemes
  simply involves choosing a uniform n bit key.

aeou< I would have
thought they would take
Π_E
to be CCA
secure...strange

Story/Notation:

- There are three natural approaches to combining
  encryption and message authentication
  using independent keys k_E and k_M
  for Π_E and Π_M respectively.

Story:

1. **Encrypt-and-authenticate**
   In this method, encryption and message authentication are
   computed independently in parallel
   i.e. given a plaintext message m
   the sender transmits the ciphertext (c, t) where
   $c \leftarrow \text{Enc}_{k_E}(m)$ and $t \leftarrow \text{Mac}_{k_M}(m)$

   The receiver decrypts c to recover m
   assuming no error occured, it then verifies the tag t.

   If $\text{Vrfy}_{k_M}(m, t) = 1$,
   the receiver outputs m
   otherwise
   outputs an error

2. **Authenticate-then-encrypt**
   Here a MAC tag t is first computed
   and then the message and tag are encrypted together.

   That is, given a message m,
   the sender transmits the ciphertext c computed as

   $t \leftarrow \text{Mac}_{k_M}(m)$ and $c \leftarrow \text{Enc}_{k_E}(m||t)$.

   The receiver decrypts c to obtain $m||t$;
   assuming no error occurs,
   it then verifies the tag t.

   As before, if $\text{Vrfy}_{k_M}(m, t) = 1$
   then the reciever outputs m.
   otherwise it outputs an error.

3. **Encrypt-then-authenticate:**
   In this case,
   the message m is first encrypted and then
   a MAC tag is computed over the result

   That is, the ciphertext is the pair (c, t) where
   $c \leftarrow \text{Enc}_{k_E}(m)$ and $t \leftarrow \text{Mac}_{k_M}(c)$

   (see also Construction 4.18)

If $\text{Vrfy}_{k_M}(c, t) = 1$ then
the receiver decrypts c and outputs the result
otherwise, it outputs an error.

Story:

- We analyse each of the above approaches when
  they are instantiated with "generic" secure components
  i.e. an arbitrary CPA secure encryption scheme
  and an arbitrary
  (strongly) secure MAC. < Recall, this menas that
  the adversary is given a "decryption" oracle
  i.e. the verification

- We want an approach that
  provides joint secrecy and integrity
  when using any (secure) components
  and we will therefore reject as "unsafe" any approach for which

  there exists even a single
  counterexample of a secure encryption scheme/MAC
  for which the combination is insecure.

- This "all-or-nothing" approach
  reduces the likelihood of implmentation flaws

  Specifically, an authenticated encryptoun scheme
  might be implemented by making calls to an "encryption subroutine"
  and a "message authentication subroutine"
  and the implementation of those subroutines may be
  changed at some later point in time.
  (This commonly occurs when cryptographic libraries are updated
  or when standards are modified)
  Implementing an approach whose security depends on
  how its components are implemented
  (rather than the security they provide)
  is therefore dangerous

- We stress that
  if an approach is rejected, this does not mean that
  it is insecure for all possible instantiations of the compenents—
  it does, however, mean that
  any instantiation of the approach must be analysed
  and proven secure before it is used.

Analysis:

Encrypt-and-authenticate.

- Recall that in this approach
  encryption and message authentication are
  carried out independently.

- Given a message m,
  the transmitted value is (c, t)
  where $c \leftarrow \text{Enc}_{k_E}(m)$ and $t \leftarrow \text{Mac}_{k_M}(m)$

- This approach may not achieve even
  the most basic level of secrecy

- To see this
  note that a secure MAC does not guarantee *any* secrecy
  and so it is possible for tha tag $\text{Mac}_{k_M}(m)$ to leak
  information about m to an eavesdropper.

  (As a trivial example,
  consider a secure MAC where

the first bit of the tag is always
equal to the first bit of the message)

So, the encrypt-and-authenticate approach
may yield a scheme that does not even have
indistinguishable encryptions in the presence of
an eavesdropper.

In fact, the encrypt-and-authenticate approach is likely
to be insecure against chosen-plaintext attacks
even when instantiated with standard components
(unlike the contrived counterexample in teh previous paragraph).

[general argument]
In particular, if a *deterministic* MAC like
CBC-MAC is used, then
the tag computed on a message (for some fixed key k_M) is
the same every time.

This allows an eavesdropper to identify
when the same message is sent twice
and so the scheme is not CPA secure.

Authenticate-then-encrypt.

- Here, a MAC tag $t \leftarrow \mathrm{Mac}_{k_M}(m)$ is first computed
then (m, t) is encrypted and
the resulting value $\mathrm{Enc}_{k_E}(m, t)$ is transmitted.

- We show that this combination also does not necessarily yield an
authenticated encryption scheme

- Actually, we have already encountered a CPA secure encryption scheme
for which this approach is insecure (recall § 3.7.2)

  - Recall,
  this scheme works by first padding the plaintext
  (which in our case will be $m||t$)
  in a specific way so the result is
  a multiple of block length
  and then encrypting the result using CBC mode

  - During decryption
  if an error in the padding is detected after
  performing the CBC mode decryption
  then a "bad padding" error is returned.

  - With regard to authenticate-then-encrypt,
  this means there are now
  two sources of potential decryption failure
  (a) the padding may be incorrect or
  (b) the MAC tag may not verify.

  - Schematically, Dec' in the combined scheme
  works as follows:

  $\mathrm{Dec}'_{k_E,k_M}(c)$:

  1. Compute $\tilde{m} := \mathrm{Dec}_{k_E}(c)$.

  If an error in the padding is detected,
  return "bad padding".

  2. Parse $\tilde{m}$ as $m||t$.

2. Parse $\tilde{m}$ as $m \| t$.

If $Vrfy_{k_m}(m, t) = 1$ return m;

else output "authentication failure".

- Assuming the attacker can distinguish b/w
  two error messages, the attacker can apply the
  same CPA attack
  [as in § 3.7.2 to the above scheme
  to recover the entire original plaintext from
  a given ciphertext]
  to recover the entire original plaintext from a given ciphertext.

  [This is due to the fact that
  padding-oracle attack shown in § 3.7.2
  relies only on the ability to learn
  whether or not there was a padding error
  something that is revealed by this scheme.]

- This type of attack has been carried out successfully in
  the real world in various settings
  e.g. in configurations of IPsec that use
  authenticate-then-encrypt.

- One way to fix the above scheme would be to ensure
  only a *single* error message is returned
  regardless of the source of decryption failure.

  This is an unsatisfying solution for several reasons
  (1) There may be legitimate reaons
  (e.g. usability, debugging) to have multiple error messages

  (2) forcing the error messages to be the same means
  that the combination is no longer truly generic
  i.e. it requires the implementer of the
  authenticate-then-encrypt approach
  to be aware of what error messages are returned by the
  underlying CPA secure encryption scheme

  (3) most of all, its extraordinarily hard to ensure
  that different erros cannot be
  distinguished since, e.g.
  even a difference in the time to return each of these errors
  may be used by an adversary to distinguish them
  (see § 4.2 [this we didn't have in our older book, TODO: add])

  Some versions of SSL tried using only a single error message
  in conjunction with an authenticate-then-encrypt appoarch
  but a padding-oracle attack was still successfully carried out
  using timing information of this sort.

  Conclusion:
  Authenticate-then-encrypt does not provide
  authenticated encryption in general
  and should not be used.

Encrypt-then-authenticate

- In this approach,
  - the message is first encrypted and then
    a MAC is computed over the result, i.e.

(c, t) where
$c \leftarrow \mathrm{Enc}_{k_E}(m)$ and $t \leftarrow \mathrm{Mac}_{k_M}(c)$

- Decryption of (c, t) is done as $\mathrm{Dec}_{k_E}(c)$, assuming $\mathrm{Vrfy}_{k_M}(c, t) = 1$
  otherwise output $\perp$

CONSTRUCTION 4.18

Let $\Pi_E = (\mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme and let $\Pi_M = (\mathsf{Mac}, \mathsf{Vrfy})$ be a message authentication code, where in each case key generation is done by simply choosing a uniform n-bit key. Define a private-key encryption scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- Gen': on input 1^n, choose independent, uniform $k_E, k_M \in \{0,1\}^n$ and output the key (k_E, k_M).
- Enc': on input a key (k_E, k_M) and a plaintext message m, compute $c \leftarrow \mathsf{Enc}_{k_E}(m)$ and $t \leftarrow \mathsf{Mac}_{k_M}(c)$. Output the ciphertext $\langle c, t \rangle$.
- Dec': on input a key (k_E, k_M) and a ciphertext $\langle c, t \rangle$, first check whether $\mathsf{Vrfy}_{k_M}(c, t) \stackrel{?}{=} 1$. If yes, then output $\mathsf{Dec}_{k_E}(c)$; if no, then output $\perp$.

A generic construction of an authenticated encryption scheme.

- This approach is sound
  as long as the MAC is strongly secure (as in Definition 4.3)
  [see the "pasted" material just after we started using the "new book"]

  - Strong security of the MAC ensures
    that an adversary will be unable to to
    generate *any* valid ciphertext
    that
    it did not recieve from its encryption oracle.

  - This immediately implies that 4.18 is unforgeable
    [Unclear: Why strong? Wouldn't a secure MAC also work?]

Strong MACs. As defined, a secure MAC ensures that an adversary cannot generate a valid tag on a new message that was never previously authenticated. But it does not rule out the possibility that an attacker might be able to generate a new *tag* on a previously authenticated message. That is, a MAC guarantees that if an attacker learns tags $t_1, \ldots$ on messages $m_1, \ldots$, then it will not be able to forge a valid tag t on any message $m \notin \{m_1, \ldots\}$. However, it may be possible for an adversary to "forge" a *different* valid tag $t_1' \neq t_1$ on the message m_1. In general, this type of adversarial behavior is not a concern. Nevertheless, in some settings it is useful to consider a stronger definition of security for MACs where such behavior is ruled out.

Formally, we consider a modified experiment Mac-sforge that is defined in exactly the same way as Mac-forge, except that now the set $\mathcal{Q}$ contains *pairs* of oracle queries and their associated responses. (That is, $(m, t) \in \mathcal{Q}$ if $\mathcal{A}$ queried $\mathsf{Mac}_k(m)$ and received in response the tag t.) The adversary $\mathcal{A}$ succeeds (and experiment Mac-sforge evaluates to 1) if and only if $\mathcal{A}$ outputs (m, t) such that $\mathsf{Vrfy}_k(m, t) = 1$ and $(m, t) \notin \mathcal{Q}$.

You're right, we should look at how they definedd "unforgeable"
in the definition of secure authenticated encryption

Recall, the unforgeable encryption experiment
does not care if you produce another tag for an existing message
Right?
So what you're saying likely applies to the security part, not unforgeability
I was asking about unforgeability

The unforgeable encryption experiment $\mathsf{Enc\text{-}Forge}_{\mathcal{A},\Pi}(n)$**:**

1. *Run* $\mathsf{Gen}(1^n)$ *to obtain a key* k.
2. *The adversary* $\mathcal{A}$ *is given input* 1^n *and access to an encryption oracle* $\mathsf{Enc}_k(\cdot)$. *The adversary outputs a ciphertext* c.
3. *Let* $m := \mathsf{Dec}_k(c)$, *and let* $\mathcal{Q}$ *denote the set of all queries that* $\mathcal{A}$ *asked its encryption oracle. The output of the experiment is* 1 *if and only if (1)* $m \neq \perp$ *and (2)* $m \notin \mathcal{Q}$.

  - As for CCA security
    the MAC computed over the ciphrtext
    has the effect of making the decryption oracle useless
    Since,
    for every ciphertext (c, t) the adversary submits to its decryption oracle
    either the adversary already knows the decryption
    (if it received (c, t) from its own encryption oracle) or
    else can expect the result to be an error
    (since the adversary cannot generate any new valid ciphertexts).

    This means the CCA security of the combined scheme
    reduces to the CPA security of Π_E.

    Observe that
    the MAC is verified before the decryption takes place

    Thus,
    MAC verification cannot leak anything
    about the plaintext
    (contrast to authenticate-then-encrypt approach—
    recall the padding oracle attack)

«Just to understand this last line, we spent over four hours
understanding the padding oracle attack »

- We now formalise the above arguments

Theorem 4.19
Let Π_E be a CPA secure private-key encryption scheme, and let
Π_M be a strongly secure message authentication code.
Then Construction 4.18 is an authenticated encryption scheme.

Proof.

Notation: Let Π' denote the scheme resulting from Construction 4.18

Goal: We need to show that Π' is unforgeable, and
that it is CCA secure.

[ME: the book repeats a bit but it keeps things clear...]

Story:
Following the intuition above
say a ciphertext (c, t) is valid
(wrt some fixed secret key (k_E, k_M))
if $\text{Vrfy}_{k_M}(c, t) = 1$.

We show that strong security of Π_M implies that
(except with negligible probability)
any "new" ciphertexts the adversary submits to
the decryption oracle will be invalid.

As discussed already, this immediately implies unforgeability
(in fact stronger than unforgeability).

This fact, also renders the decryption oracle useless
and means that CCA security of Π' reduces to
the CPA security of Π_E.

Notation:
- $\mathcal{A}$:
Let $\mathcal{A}$ be a probabilistic poly-time adversary
attacking Construction 4.18 in a CCA attack (cf Definition 3.33)

< reference is likely wrong; book changed

- New ciphertext:
Say a ciphertext (c, t) is *new* if $\mathcal{A}$ did not receive (c, t) from its encryption oracle
or as the challenge ciphertext.

- Event ValidQuery:
Let ValidQuery be the event that $\mathcal{A}$ submits
a new ciphertext (c, t) to its decryption oracle
which is valid (i.e. for which $\text{Vrfy}_{k_M}(c, t) = 1$).

Story: We prove the following.

Claim 4.20
$\Pr[\text{ValidQuery}] \leq \text{negl}$

Proof.
Intuition:
- This is idue to the fact that if ValidQuery occurs then
the adversary has forged a new, valid pair (c, t) in the
Mac-sforge game/experiment.

Notation:
- $q(\cdot)$

Let q be a polynomial upper bound on the number of
decryption-oracle queries made by $\mathcal{A}$ [defined above Claim 4.20]
and consider the following adversary $\mathcal{A}_M$ attacking the
message authentication code Π_M
(i.e. $\mathcal{A}_M$ running in experiment $\text{Mac-sforge}_{A_M,\Pi_M}(n)$)

[Me: They will show that a CCA attacking adversary $\mathcal{A}$ on Π can be used to
construct an adversary $\mathcal{A}_M$ for strong mac forge game for the MAC used in Π]

[Me: (in case the following gets confusing; otherwise, feel free to skip)
The idea is that the CCA adversary $\mathcal{A}$ is going to query the
decryption oracle at a new "valid" encryption (c, t).
So, $\mathcal{A}_M$ is going to simply hope that some $i \leftarrow \{1 \dots q\}$ (where q bounds the number of
decryption queries $\mathcal{A}$ makes) is where this "new valid encryption" is queried to the
decryption oracle. Then, $\mathcal{A}_M$ simulates the interaction with $\mathcal{A}$ and
finally, at the ith query, simply outputs (c, t) that $\mathcal{A}$ asked to be decrypted
If indeed $\mathcal{A}_M$'s guess is correct
$\mathcal{A}$ would have handed $\mathcal{A}_M$ a new c with its tag t
and this should allow it to break the security for strong mac forge]

Adversary $\mathcal{A}_M$:

$\mathcal{A}_M$ is given input 1^n and has access to a MAC oracle $\text{Mac}_{K_M}(\cdot)$.

1. Choose uniform $k_E \in \{0,1\}^n$ and $i \in \{1 \dots q(n)\}$.

2. Run $\mathcal{A}$ on input 1^n.
When $\mathcal{A}$ makes an encryption oracle query for the message m,
answer it as follows:

(i) Compute $c \leftarrow \text{Enc}_{K_E}(m)$
(ii) Query c to the MAC oracle and receive t in response.
Return (c, t) to $\mathcal{A}$.

The challenge ciphertext is prepared as in the actual scheme
(with $b \leftarrow \{0,1\}$, select m_b that gets encrypted)

When $\mathcal{A}$ makes a decryption-oracle query for the ciphertext (c, t)
answer it as follows:

If this is the ith decryption-oracle query
output (c, t). [as in, this is the output of $\mathcal{A}_M$; not a response to $\mathcal{A}$]
Otherwise:
(i) If (c, t) was a response to a previous encryption-oracle query
for a message m, return m. [to $\mathcal{A}$]
(ii) Otherwise, return $\perp$. [to $\mathcal{A}$]

Story:
In essence, $\mathcal{A}_M$ is "guessing" that the ith decryption-oracle query of $\mathcal{A}$ will be
the first new, valid query $\mathcal{A}$ makes.

In that case, $\mathcal{A}_M$ outputs a valid forgery on a message c
that it had never previously submitted to its
own MAC oracle.

NB: Clearly, $\mathcal{A}_M$ runs in probabilistic poly time.

Analysis: What's the probability that $\mathcal{A}_M$ produces a good forgery?

NB1: View of $\mathcal{A}$ when run as a subroutine by $\mathcal{A}_M$ is
distributed identically to
the view of $\mathcal{A}$ in experiment $\text{PrivK}^{\text{CCA}}_{\mathcal{A},\Pi'}(n)$
until event ValidQuery occurs.

[Justification:
Note that the encryption-oracle queries of $\mathcal{A}$

(and also the computation of the challenge ciphertext)
is simulated perfectly by $\mathcal{A}_M$.

As for the decryption oracle queries of $\mathcal{A}$,
until ValidQuery occurs
these are all simulated properly

In case (i) this is obvious
and for case (ii), if the ciphertetx (c, t) submitted to
the decryption oracle is new
then as long as ValidQuery has not yet accurred
the answer to the decryption-oracle query is indeed $\perp$.

[NB-nested □: Case (i) is exactly the case that (c, t) is not new
and case (ii) is exactly the case that (c, t) is new]

NB/Recall: $\mathcal{A}$ is not allowed to submit the challenge ciphertext to the
decryption oracle.]

NB2: Because the view of $\mathcal{A}$ when urn as a subroutine by $\mathcal{A}_M$ is
distributed identically to
the view of $\mathcal{A}$ in $\text{PrivK}^{\text{CCA}}_{\mathcal{A},\Pi'}(n)$ until event ValidQuery occurs,
the probability of
event ValidQuery in experiment $\text{Mac-forge}_{\mathcal{A}_M,\Pi_M}(n)$
is the same as
the probability of that
event in experiment $\text{PrivK}^{\text{CCA}}_{\mathcal{A},\Pi'}(n)$.

NB3: If $\mathcal{A}_M$ correctly guesses the first index i when
ValidQuery occurs, then
$\mathcal{A}_M$ outputs (c, t)
for which $\text{Vrfy}_{k_M}(c, t) = 1$
(since (c, t) is valid) and
for which it was never given tag t in
response to the query $\text{Mac}_{k_M}(c)$
(since (c, t) is new).

In this case, $\mathcal{A}_M$ succeeds in experiment $\text{Mac-sforge}_{\mathcal{A}_M,\Pi_M}(n)$.

NB4: The probability that $\mathcal{A}_M$ guesses i correctly is $1/q(n)$. Therefore

$$\Pr\left[\text{Mac-sforge}_{\mathcal{A}_M,\Pi_M}(n) = 1\right] \geq \Pr[\text{ValidQuery}]/q(n).$$

Conclusion: Since Π_M is a strongly secure MAC and q is polynomial
we conclude that $\Pr[\text{ValidQuery}]$ is negligible.

□

★ « Left off here....

Story:

- We use Claim 4.20 to prove security of Π'.
  - The easier case is to prove Π' is unforgable
    Follows immediately because
    (a) The adversary $\mathcal{A}'$ in the unforgeable encryption experiment
    is at most as strong as the adversary in the
    chosen-ciphertext experiment
    (in the former, the adversary only has
    access to an encryption oracle)

    (b) When $\mathcal{A}'$ outputs a ciphertext (c, t) at the end
    it succeeds => (c, t) is valid and new
    (ME: ~~in fact, even this condition was weaker~~
    ~~the tagged message had to be new;~~

~~here, even if a new~~
the Claim 4.20 also only shows
that the probability of finding a new "tag" (by tag, I mean the full (c, t); because this is what is used in the Enc-Forge game)
is negligible at most
But note that we had to use a strong mac
to derive Claim 4.20).

But Claim 4.20 shows that
the probability of this event is negligible

- It is only slightly more involved to prove that Π' is CCA secure.

$\Pr[E] = \Pr[E \wedge A] + \Pr[E \wedge \neg A]$

$\Pr[E \wedge A] \leq \Pr[A]$

$\Pr[E] \leq \Pr[A] + \Pr[E \wedge \neg A]$

Argument:

Let $\mathcal{A}$ again be a PPT adversary, attacking Π' in a CCA attack.

It holds that

$$\Pr\left[\mathrm{PrivK}^{\mathrm{CCA}}_{\mathcal{A},\Pi'}(n) = 1\right] \leq \Pr[\mathrm{ValidQuery}] + \Pr[\mathrm{PrivK}^{\mathrm{CCA}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathrm{ValidQuery}}]$$

We already have that the first term is negligible. The following bounds the second term.

Claim 4.21

There exists a negligible function negl such that

$$\Pr\left[\mathrm{PrivK}^{\mathrm{CCA}}{}_{\mathcal{A},\Pi'}(n) = 1 \wedge \overline{\mathrm{ValidQuery}}\right] \leq \frac{1}{2} + \mathrm{negl}(n).$$

Proof:

To prove this claim, we rely on CPA-security of Π_E.

Consider the following adversary $\mathcal{A}_E$ attacking Π_E in a chosen-plaintext attack:

(Me: recall that $\mathcal{A}$ is the adversary that breaks Π' (i.e. in CCA))

Adversary $\mathcal{A}_E$:

$\mathcal{A}_E$ is given input 1^n and has access to $\mathrm{Enc}_{k_E}(\cdot)$.

1. Choose uniform $k_M \in \{0,1\}^n$.
2. Run $\mathcal{A}$ on input 1^n.
   When $\mathcal{A}$ makes an encryption-oracle query
   for the message m, answer it as follows:
   (i) Query m to $\mathrm{Enc}_{k_E}(\cdot)$ and receive c in response.
   (ii) Compute $t \leftarrow \mathrm{Mac}_{k_M}(c)$ and return (c, t) to $\mathcal{A}$.

   When $\mathcal{A}$ makes a decryption-oracle query for the
   ciphertext (c, t), answer it as follows:

   if (c, t) was a response to a previous encryption-oracle query
   for a message m, return m.
   Else, return $\perp$.
3. When $\mathcal{A}$ outputs messages (m_0, m_1),
   output these same messages and
   receive a challenge ciphertext c in response.
   Compute $t \leftarrow \mathrm{Mac}_{k_M}(c)$ and
   return (c, t) as the challenge ciphertext for $\mathcal{A}$.
   Continue answering $\mathcal{A}$'s oracle queries as above.
4. Output if the same bit b' that is output by $\mathcal{A}$.

NB: $\mathcal{A}_E$ does not need a decryption oracle

because it simply assumes that any decryption query
by $\mathcal{A}$ that was not a result of a
previous encryption-oracle query
is invalid.

[trivial obseravtions]
NB2: $\mathcal{A}_E$ runs in PPT.
Furthermore, the view of $\mathcal{A}$ when run
as a subroutine by $\mathcal{A}_E$ is distributed identically
to the view of $\mathcal{A}$ in experiment $\text{PrivK}^{\text{CCA}}_{\mathcal{A},\Pi'}(n)$
as long as event ValidQuery *never occurs*.

Thus, the probability that $\mathcal{A}_E$ succeeds when ValidQuery does not occur
is the same as the probability that $\mathcal{A}$ succeeds
when ValidQuery does not occur, i.e.

$$\Pr\left[\text{PrivK}^{\text{CPA}}_{\mathcal{A}_E,\Pi_E}(n)=1 \wedge \overline{\text{ValidQuery}}\right] = \Pr\left[\text{PrivK}^{\text{CCA}}_{\mathcal{A},\Pi'}(n)=1 \wedge \overline{\text{ValidQuery}}\right]$$

$$\Rightarrow$$

$$\underbrace{\Pr\left[\text{PrivK}^{\text{CPA}}_{\mathcal{A}_E,\Pi_E}(n)=1\right]}_{\geq \frac{1}{2}+\text{negl}} \geq \Pr\left[\text{PrivK}^{\text{CPA}}_{\mathcal{A}_E,\Pi_E}(n)=1 \wedge \overline{\text{ValidQuery}}\right] = \Pr\left[\text{PrivK}^{\text{CCA}}_{\mathcal{A},\Pi'}(n)=1 \wedge \overline{\text{ValidQuery}}\right]$$

This completes the proof.

□

The need for independent keys.

- We conclude this section by stressing a basic principle of cryptography:
  - different instances of cryptographic primitives
    should always use independent keys.

To illustrate this here
consider what can happen to the encrypt-then-authenticate methodology
when the same key k is used for
both encryption and authentication

Let F be a strong pseudorandom permutation.

It follows that F^{-1} is a strong pseudorandom permutation also.

[You see where this going, don't you? □]

Define
$\text{Enc}_k(m) = F_k(m||r)$
for $m \in \{0,1\}^{n/2}$ and a uniform $r \in \{0,1\}^{n/2}$
and
$\text{Mac}_k(c) = F_k^{-1}(c)$.

It can be shown that the encryption is CPA secure (in fact it is even CCA secure)
and
we know the MAC is also secure.

However, clearly, encrypt-then-authenticate combination
using the same key k simply reveals the message in the clear!

i.e.
$\text{Enc}_k(m) = F_k(m||r)$
$\text{Mac}_k\big(\text{Enc}_k(m)\big) = F_k^{-1}(F_k(m||r)) = m||r$

NB: This does not contradict Theorem 4.19

since it expressly asks for independent keys k_M and k_E.

Book: The reader is encouraged to check where this independence is used in
the proof of Theorem 4.19
[Me: Oh, e.g. to be able to encrypt a MAC separately,
the adversary $\mathcal{A}'$ sampled fresh keys
for that part]

4.5.3 Secure Communication Sessions

Story:

- We briefly describe the application of authenticated encryption to the
setting of two parties who wish to
communicate "securely"
—with joint secrecy and integrity—
over the course of a communication session.

- (For the purposes of this section,
a comunication session is simply
a period of time during which
the communicating parties maintain state)

Disclaimer:

- In our treatement here
we deliberately informal
(a formal definition is quite involved and
this topic arguably lies more in
the area of network security than cryptography).

Story resumed:

- Let $\Pi = (\mathrm{Enc}, \mathrm{Dec})$ be an authenticated encryption scheme
- Consider two parties A and B
who share a key k and wish to use this key
to secure their communication over
the course of a session.

- The obvious thing to do here is to use Π
Whenever, say, A wants to transmit a message m to B
it computes $c \leftarrow \mathrm{Enc}_k(m)$ and sends c to B

In turn, B decrypts c to recover the result
(ignoring the result if decryption returns $\perp$).

Likewise, the same procedure is followed when B
wants to send a message to A

This simple approach, however, does not suffice—here are potential attacks

Reordering attack
An attacker can swap the order of messages.
For e.g.
if A transmits c_1 (an encryption of m_1) and
subsequently transmits c_2 (an encryption of m_2) then
an attacker who has some control over the network
can deliver c_2 before c_1
& cause B to output the messages in the wrong order.
(This causes a mismatch b/w the two parties' views
of their communication session.)

Replay attack
An attacker can *replay* a (valid) ciphertext c sent
previously by one of the parties.
Again, this causes a mismatch b/w what is sent by one party
and recieved by the other.

Reflection attack

An attacker can take a ciphertext c sent from A to B and
send it back to A.
This again can cause a mismatch b/w two parties' transcripts
their communication session:
A may output a message m even though B never sent such a message.

Story (cont.):

- Fortunately, the above attacks are easy to prevent using
counters to address the first two and
a directionality bit to prevent the third.

  [In practice, the issue of directionality is often solved
  by simply maintaining separate keys
  for each direction]

- We describe these in tandem:
Each party mantains two counters $\text{ctr}_{A,B}$ and $\text{ctr}_{B,A}$
keeping track of the number of messages from A to B
(resp. B to A) during the session.

- These counters are initialised to 0 and
incremented each time a party sends or
receives a (valid) message.

- The parties
also agree on a bit $b_{A,B}$ and define $b_{B,A}$ to be its complement.
(e.g. set $b_{A,B} = 0$ iff the identity of A is lexicographically smaller than the identity of B)
Me: Not fully clear what "identity of A/B"—perhaps it is Alice/Bob
and then lexicographically, Alice comes first

- When A wants to transmit a message m to B
she computes the ciphertext $c \leftarrow \text{Enc}_k(b_{A,B} \,||\, \text{ctr}_{A,B} \,||\, m)$
and sends c

- She then increments $\text{ctr}_{A,B}$.

- Upon receiving c
party B decrypts
if the result is $\perp$, B rejects
otherwise, B parses the decrypted message as
$b||\text{ctr}||m$

  if $b = b_{A,B}$ and $\text{ctr} = \text{ctr}_{A,B}$ then
  B outputs m and increments $\text{ctr}_{A,B}$

  otherwise, B rejects.

- The above steps,
mutatis mutandis (with appropriate changes)
are applied when B sends a message to A.

Remark: Since the parties anyway maintaining state (i.e. counters $\text{ctr}_{A,B}, \text{ctr}_{B,A}$)
the parties could easily use a stateful authenticated encryption scheme Π.

[Me: I don't think the book described stateful authenticated encryption]

§ 4.5.4 CCA Secure Encryption

Remark:

- It follows directly from the definition
  that any authenticated encryption scheme
  is also CCA secure.
  - Can there be CCA-secure private-key encryption schemes
    that are not unforgeable (i.e. that are forgeable)?
  - Indeed, there are (see Exercise 4.25).

4.25 Let F be a strong pseudorandom permutation, and define the following fixed-length encryption scheme: On input a message $m \in \{0,1\}^{n/2}$ and key $k \in \{0,1\}^n$, algorithm Enc chooses a uniform $r \in \{0,1\}^{n/2}$ and computes $c := F_k(m\|r)$. (See Exercise 3.18.) Prove that this scheme is CCA-secure, but is not an authenticated encryption scheme.

<TODO

Story:

- One can imagine applications where CCA security is needed
  but authenticated encryption is not.
  - One example:
    when private-key encryption is used for *key transport*.
  - A server gives a tamper-proof hardware token to a user
    where embedded in the token is a
    long-term key k.
  - The server can upload a fresh
    short-term key k' to this token
    by giving the user $\mathrm{Enc}_k(k')$
    i.e. the user is supposed to give this

Server

User

1. Tamper-proof hardware token (encoding k)
   is sent to the user.

Creates k' and sends $\mathrm{Enc}_k(k')$ to the user

uses

4.6* Information-Theoretic MACs

Sunday, July 2, 2023 8:09 PM

Story:

- Earlier, considered computationally bounded adversaries
- Now, as in Chapter 2, we ask about
  unbounded adverasiry

- We show under which conditions
  information-theoretic
  (as opposed to computational) security is attainable

  - A first observation:
    it is impossible to achieve "perfect" security in this context
    i.e. we cannot hope to have a MAC for which
    the prob. [adversary outputs a valid tag on previously
    unauthenticated message] = 0.

    The reason is that
    an adversary can simply guess a valid tag t
    on any message and the gues will be correct
    with probability at least $1/2^{|t|}$
    where $|t|$ denotes the tag of length of the scheme

    - The above example tells us
      what we can hope to achieve:
      a MAC with tags of length $|t|$ where the
      probability of forgery is at most $1/2^{|t|}$
      even for unbounded adversaries.
      We will see that this is achievable
      but only under restrictions on
      how many messages are authenticated by the
      honest parties.

- We first define information-theoretic security for
  message authentication codes.
  - A starting point
    is to take expremient $\text{Mac-forge}_{\mathcal{A},\Pi}(n)$ that is
    used to define the security for
    computationally secure MACs but
    to drop the security parameter n
    and require that $\Pr[\text{Mac-forge}_{\mathcal{A},\Pi} = 1]$
    should be "small" for all adversaries $\mathcal{A}$
    (and not just adversaries running in poly time).
    Me: Not very clear at the moment

  - As mentioned above
    (and as will be proved formally in Section 4.6.2)
    however, such a definition is impossible to achieve!

  - Rather, information-ethoretic security can be achieved
    only if we place some bound on the number of messages
    authenticated by the honest parties.

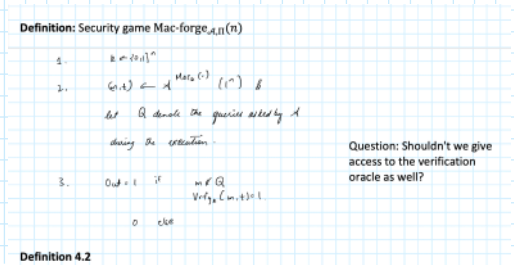
  - We look here at the most basic setting
    where the parties authenticate just a *single* message.

  - We refer to this as *one-time message authentication.*

The following expreiment modifies $\text{Mac-forge}_{\mathcal{A},\Pi}(n)$ following the above discussion.

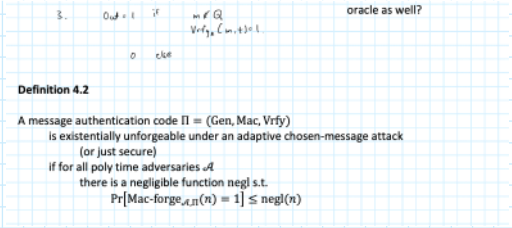
The one-time message authentication game $\textbf{Mac-forge}_{\mathcal{A},\Pi}^{\textbf{1-time}}$:

1. A key k is generated by running Gen
2. The adversary $\mathcal{A}$ queries
   a message m' and is given in return
   a tag $t' \leftarrow \text{Mac}_k(m')$
3. $\mathcal{A}$ outputs (m, t).
4. The output is 1 iff (a) $\text{Vrfy}_k(m, t) = 1$ (b) $m \neq m'$



3. $\mathcal{A}$ outputs (m, t).
4. The output is 1 iff (a) $\text{Vrfy}_k(m, t) = 1$ (b) $m \neq m'$

Definition 4.22
A message authentication code $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$
is one-time ϵ-secure,
or just ϵ-secure,
if for all (even unbounded) adversaries $\mathcal{A}$:
$\Pr[\text{Mac-forge}^{\text{1-time}}_{\mathcal{A},\Pi} = 1] \leq \epsilon$.



§ 4.6.1 Constructing Information-Theoretic MACs

Story:
- In this section
  we show how to build an ϵ-secure MAC based
  on any *strongly universal function*.

  [Footnote: "Strongly universal function"
  is sometimes called "Strongly universal hash function"]

  We then show a simple construction of the latter.

Definition: Strongly universal (or pairwise-independent)

- Let $h: \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ be a keyed function
  whose first input is a key $k \in \mathcal{K}$ and
  whose second input is
  taken from some domain $\mathcal{M}$.
  - Then,
    h is strongly universal if
    for any two distinct inputs
    m, m'
    the values $h_k(m)$ and $h_k(m')$ are
    uniformly and independently distributed in
    $\mathcal{T}$ when k is uniformly sampled.

Definition 4.23
A function $h: \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ is strongly universal if
for all distinct $m, m' \in \mathcal{M}$ and for all $t, t' \in \mathcal{T}$ it holds that
$\Pr[h_k(m) = t \wedge h_k(m') = t'] = 1/|\mathcal{T}|^2$

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

Story:
- The above should motivate the construction of a one-time message
  authentication code from any strongly universal function h.

- The tag t on a message m is obtained by
  computing $h_k(m)$ where
  the key k is uniform; See Construction 4.24 (below).

- Intuitively,
  even after an adversary observes the tag $t' \coloneqq h_k(m')$
  for any message m',
  the correct tag $h_k(m)$ for any other message m
  is still uniformly distributed in $\mathcal{T}$
  from the adversary's point of view.

- Thus, the adverasry can do nothing more than
  blindly guess the tag
  and thish guess will be correct only with
  probability $1/|\mathcal{T}|$.

Construction 4.24 | MAC from any strongly universal function

Let $h: \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ be a strongly universal function.
Define a MAC for messages in $\mathcal{M}$ as follows:

- Gen: choose uniform $k \in \mathcal{K}$ and output it as the key
- Mac: on input a key $k \in \mathcal{K}$
  a message $m \in \mathcal{M}$
  output the tag $t := h_k(m)$

- Vrfy: on input a key $k \in \mathcal{K}$
  a message $m \in \mathcal{M}$ and
  a tag $t \in \mathcal{T}$
  output 1 iff $t = h_k(m)$
  (if $m \notin \mathcal{M}$ then output 0).

Story:

- The above construction can be viewed as
  being analogous to Construction 4.5.

- This is because a strongly universal function h
  is identical to a random function
  as long as it is only evaluated twice.

Theorem 4.25

Let $h: \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ be a strongly universal function.

Then Construction 4.24 is a $1/|\mathcal{T}|$ secure MAC for messages in $\mathcal{M}$.

Proof.

Let $\mathcal{A}$ be an adversary

NB: As usual, in the information-theoretic setting
we may assume $\mathcal{A}$ is deterministic
without loss of generality

Consequence:
So the message m' that $\mathcal{A}$ "queries"
at the outset of the experiment is fixed.

Furthermore,
the pair (m, t) that $\mathcal{A}$ outputs at the end
is a deterministic function of the tag t'
on message m' that $\mathcal{A}$ recieves

Therefore, it holds that

$$\Pr\left[\text{Mac-forge}^{\text{1-time}}_{\mathcal{A},\Pi} = 1\right] = \sum_{t' \in \mathcal{T}} \Pr\left[\text{Mac-forge}^{\text{1-time}}_{\mathcal{A},\Pi} = 1 \wedge h_k(m') = t'\right]$$

$\Pr(A) = \sum_{B \in \mathcal{B} \text{ s.t. } UB = \text{mutually}} \Pr(A \wedge B)$

$$= \sum_{\substack{t \in \mathcal{T} \\ (m,t) := \mathcal{A}(t')}} \Pr\left[h_k(m) = t \wedge h_k(m') = t'\right]$$

(Adversary gets t' as a response from the "Mac oracle")

$$= \sum_{\substack{t' \in \mathcal{T} \\ (m,t) := \mathcal{A}(t')}} \frac{1}{|\mathcal{T}|^2} = \frac{1}{|\mathcal{T}|}.$$

□

Story:

- We now turn to a classical construction
  of a strongly universal function.

  - We assume some basic knowledge
    about arithmetic modulo a prime number
    readers may refer to Section 8.1.1 and 8.1.2 for the
    necessary background.

- Fix some prime p &
  let $\mathbb{Z}_p := \{0, \ldots, p-1\}$
  (field)

We take as our message space
$\mathcal{M} = \mathbb{Z}_p$
& the space of tags to also be
$\mathcal{T} = \mathbb{Z}_p$;
Thus, $\mathcal{K} = \mathbb{Z}_p \times \mathbb{Z}_p$.

Define: $h_{a,b}(m) := [a \cdot m + b \bmod p]$
where the notation $[x \bmod p]$ ...

Theorem 4.26
For any prime p, the function h is strongly universal.

Proof

- Fix any distinct $m, m' \in \mathbb{Z}_p$
  and
  any $t, t' \in \mathbb{Z}_p$.
- For which keys (a, b) does it hold that
  both $h_{a,b}(m) = t$ and $h_{a,b}(m') = t'$?
- This holds only if

  $a \cdot m + b = t \bmod p$ and $a \cdot m' + b = t' \bmod p$
- We thus have
  two linear equations in two unknowns a, b
- These two equations are
  both satisfied exactly when
  $a = [(t - t') \cdot (m - m')^{-1} \bmod p]$

  [just subtract the equations]
  and
  $b = [t - a \cdot m \bmod p]$

  [use a to find b]
- NB: $[(m - m')^{-1} \bmod p]$ exists because $m \neq m'$ and so
  $m - m' \neq 0 \bmod p$.
- Restated:
  for any m, m', t, t' as above
  there is a unique key (a, b)
  with $h_{a,b}(m) = t$ and $h_{a,b}(m') = t'$
- Since there are $|\mathcal{T}|^2$ keys
  we conclude that the probability (over choice of the key)
  that $h_{a,b}(m) = t$ and $h_{a,b}(m') = t'$
  is exactly
  $$\frac{1}{|\mathcal{K}|} = \frac{1}{|\mathcal{T}|^2}$$
  as required.

□

Parameters of Construction 4.24

- We briefly discuss the parameters of Construction 4.24
  when instantiated with the strongly universal function
  described above
  (ignoring the fact that p is not a power of 2).

- The construction is a $1/|\mathcal{T}|$ secure MAC with
  tags of length $\log |\mathcal{T}|$

  the tag length is optimal
  for the level of security achieved.

- Let $\mathcal{M}$ be some fixed message space for which
  we want to construct a one-time secure MAC

- The construction above
  gives a $1/|\mathcal{M}|$ secure MAC with
  keys that are twice the length of the messages.
  - NB: There are two problems (at opposite length of the spectrum)
    - First, if $|\mathcal{M}|$ is small
      then a $1/|\mathcal{M}|$ probability of forgery may be
      quite large
    - On the flip side
      if $|\mathcal{M}|$ is large then
      $1/|\mathcal{M}|$ probability of forgery
      may be an overkill
      (one might be willing to accept a
      (somewhat) larger prob. of forgery
      if that level of security can be achieved with
      shorter keys).
    - The first problem (when $|\mathcal{M}|$ is small)
      is easily addressed by simply embedding $\mathcal{M}$
      into a larger space (e.g. by padding)
    - The second can also be addressed [see references ...]

§ 4.6.2 Limitations on Information-Theoretic MACs

Story:

- In this section
  we explore limitations on information-theoretic MACs
  - We show that
    - any 2^{-n}-secure MAC must have
      keys of length at least $2n$
  - An extension of the proof shows that
    any ℓ-time 2^{-n}-secure MAC
    (where security is defined via
    a natural modification to Definition 4.23)
    requires keys of length at least $(\ell + 1) \cdot n$
  - A corollary is that
    no MAC with bounded-length keys
    can provide information theoretic
    security when authenticating an
    unbounded number of messages.

DEFINITION 4.23 *A function $h : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ is* strongly universal *if for all distinct $m, m' \in \mathcal{M}$ and all $t, t' \in \mathcal{T}$ it holds that*

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

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

In the following,
we assume the message space contains
at least two messages
(if not, there is no point in communicating; let alone authenticating)

Theorem 4.27
Let $\Pi = (\text{Gen}, \text{Mac}, \text{Vry})$ be a (one-time) 2^{-n}-secure MAC where

all keys output by Gen are of the same length.
Then the keys output by Gen must have
length at least $2n$.

Proof.

Story:

- Fix two distinct messages m_0, m_1 in the message space
- The intuition for the proof is that
  - there must be at least 2^n possibilities
    for the tag of m_0
    (or else the advesary could guess with prob better than 2^{-n});
  - further
    even conditioned on the value of the tag for m_0
    there must be 2^n possibilities for the tag m_1
    (or else the adversary colud forge
    a tag on m_1 with probability better than 2^{-n}).
  - Since
    each key defines tags for m_0 and m_1
    this means that there must be
    at least $2^n \times 2^n$ keys.
- We now make this formal.

Let: $\mathcal{K}$ denote the key space
(i.e. the set of all possible keys that
can be output by Gen)

Let: $\mathcal{K}(t_0)$ denote the set of keys for which
t_0 is a valid tag on m_0; i.e.

$$\mathcal{K}(t_0) \coloneqq \{k | \mathrm{Vrfy}_k(m_0, t_0) = 1\}.$$

NB: For any t_0 we must have

$$|\mathcal{K}(t_0)| \leq 2^{-n} \cdot |\mathcal{K}|. \qquad [*]$$

Otherwise the adversary could
simply output (m_0, t_0) as its forgery
this would be a valid f orgery with probability at least
$\frac{|\mathcal{K}(t_0)|}{|\mathcal{K}|} > 2^{-n}$.

Consider:
The Adversary $\mathcal{A}$ who
requests a tag on the message m_0
receives in return a tag t_0
chooses a uniform $k \in \mathcal{K}(t_0)$
and outputs $(\mathrm{m}_1, \mathrm{Mac}_k(\mathrm{m}_1))$ as its forgery.

NB: The probability that $\mathcal{A}$ outputs a valid forgery is
at least

$$\sum_{t_1} \Pr[\mathrm{Mac}_k(m_1) = t_1] \cdot \frac{1}{|\mathcal{K}(m_0, t_0)|} \geq \sum_{t_1} \Pr[\mathrm{Mac}_k(m_1) = t_1] \cdot \frac{2^n}{|\mathcal{K}|} = \frac{2^n}{|\mathcal{K}|}$$

where we used inequality [*] in the first step above.

NB2: By the claimed security of the scheme
the probability that the adversary can output
a valid forgery is at most 2^{-n}.

Thus: $|\mathcal{K}| \geq 2^{2n}$.
Since all keys have the same length
each key must have length at least $2n$.

References and Additional Reading

- Definition of MAC was adapted from [Bellare et al]
  digital signatures [Goldwasser]

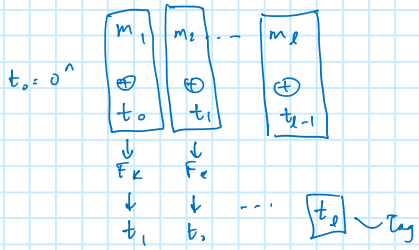
- CBC-MAC standardised in 1980s
  - CBC-MAC proved secure by Bellare
  - then by Bernstein direct but less intuitives)
- CMAC
  - new standard that handles unknown variable length MACs

[cheat-sheet] Ch 4

Friday, June 9, 2023 9:32 AM

- § 4.1 and 4.2
  - Secure can mean more than
    hiding messages from an adversary—e.g. bank, change amount, CCA also not enough
  - Message Integrity/Authentication
    is different from
    Encryption (as in Ch 3)
    - Showed ECB, CBC modes can be "tampered"
- § 4.3 MACs
  - Defined MAC
    - $\mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy}$
      $\mathrm{Mac}_k(m)$ produces a tag t
      $\mathrm{Vrfy}_k(m, \mathrm{Mac}_k(m)) = 1$
    - Security game (Mac-forge): give access to Mac_k(.) and ask for a new message to be tagged

    [strong MAC also defined in the other book]
  - Limitations: replay attack (repeeated money transfer)
- § 4.3 Fixed-length MAC
  - Construction: just apply a PRF on the message, $F_k(m)$
    - Thm 4.4: This construction is secure
      Proof (standard):
      replace PRF with a random function—show security
      then show if PRF vs random function can be distinguished, it breaks PRF security
- § 4.4 Variable length MACs [section number may be wrong]
  - Ideas on how to extend (and why the fail)
    - Apply PRF to first block (remaining message can be changed)
    - XOR PRF of all blocks (logical operation—red flag!)
    - Apply PRF to each block separately and output (blocks can be re-arranged)
  - Gives an "inefficient" construction [as in the tag is of length ~4 times that of the message]
    To encrypt $m_1 \dots m_d$ (a message with d blocks, each block of length n)
    Choose $r \leftarrow \{0,1\}^{n/4}$
    Compute the tag as $\mathrm{t_i} = F_k(r||d||i||m_i)$
    Output $r, t_1 \dots t_d$
  - Thm 4.6: The construction is secure
    - Proof (standard technique, replace with random first, and then elementary probability)
- § 4.5 CBC-MAC
  - Construction



    - Thm: The construction is secure when
      the function $\ell(\cdot)$ is fixed in advance
  - Surprising Observations
    - CBC-MAC vs CBC encryption

- □ if random r is used, then CBC-MAC becomes insecure
  if fixed r is used, then CBC encryption becomes insecure
- □ in CBC encryption, all blocks are output (otherwise the scheme is not secure)
  in CBC-MAC, only the last block is output (otherwise, the scheme is not secure)

- Variable length
  - Claim: When $\ell(\cdot)$ is not a fixed function, the scheme can be compromised
  - Fixes:
    (a) apply the PRF on ℓ to obtain a new key that is used in the subsequent PRF
    (b) Prepend ℓ to the message (NB: appending is not secure) [used later]
    (c) Use two keys (k_1, k_2),
    one for the CBC-MAC to get t and
    then re-encrypt the tag t using the second key to get the final tag $F_{k_2}(t)$

[book changed; 2014 version now]

- §4.4.2 Proof that arbitrary* length CBC-MAC is secure
  - Construction
    - □ Recall:
      prefix-free strings
      Encode: produces prefix-free strings
    - □ Define the function CBC (just as in CBC-MAC but for abritrary length)
      $\mathrm{CBC}_k(x_1 \dots x_\ell) \coloneqq F_k(\dots F_k(F_k(x_1) \oplus x_2) \dots) \oplus x_\ell)$
    - □ To tag m, apply CBC_k on $\mathrm{Encode}(m)$.

  - The proof that the construction is secure (i.e. unforgeable)
    is quite long
  - Recollection hints
    - □ Proof strategy:
      Use CBC_k with F_k replaced with a random function (call it CBC_g)
      and show
      CBC_g is indistinguishable from a random function f
      when restricted to prefix-free strings
    - □ Claim 4.14:
      Show CBC_g and f behave almost the same
      when quried on $\leq q$ prefix free strings of block length $\leq \ell$

      $$|\Pr\left[D^{\mathrm{CBC}_{g_n}(\cdot)}(1^n) = 1\right] - \Pr\left[D^{f(\cdot)}(1^n) = 1\right] \leq \frac{q^2\ell^2}{2^n}$$

      - A notion of smoothness
        Evaluations of CBC_g on q many prefix free strings, $X_1 \dots X_q$
        equals fixed strings $t_1 \dots t_q$ is at least $(1-\delta)\cdot 2^{-nq}$, i.e.
        $\Pr\left[\wedge_i \mathrm{CBC}_g(X_i) = t_i\right] \geq (1-\delta)\cdot 2^{-nq}$
      - Prove that CBC is smooth
        Elementary (but involved) probability analysis
      - Smoothness implies the claim
        [nice technique]
        Write out $\Pr\left[D^{\mathrm{CBC}_g}(1^n) = 1\right] = \sum \alpha(\vec{x}, \vec{t}) \cdot \Pr[\wedge_i \mathrm{CBC}_g(X_i) = t_i]$
        where the sum is over the xs and ts
        and α is the probability the distinguisher
        outputs 1 when its queries x are responded with ts
        Using the smoothness notion (first bullet point, above)
        one can relate it to $\Pr[D^f(1^n) = 1]$;
        Proceeding similarly for the zero output case,
        Claim 4.14 can be proved.
    - □ Given Claim 4.14, the security proof is essentially the same as that
      for the basic MAC which we saw

- § 4.5 Authenticated Encryption
  - Ensuring secrecy and integrity/no tampering
    - No consesus yet on terminology and definition
  - Definition (we use):
    CCA secure
    (for secrecy) [already defined]
    Unforgeable, $\text{Enc-Forge}_{\mathcal{A},\Pi}(n)$
    (for authentication) [like a MAC but the syntax is slightly different)

1. $k \leftarrow \{0,1\}^n$

2. $c \leftarrow \mathcal{A}^{Enc_k(\cdot)}(1^n)$

let Q denote the queries asked by $\mathcal{A}$ during the execution

3. Out = 1 if $Dec_k(c) \neq \perp \wedge Dec_k(c) \notin Q$

0 else.

- § 4.5.2 Generic Constructions
  - Encrypt and authenticate
    $c \leftarrow \text{Enc}_{k_E}(m)$ and $t \leftarrow \text{Mac}_{k_M}(m)$
    (insecure—the tag could be leaking the message)
  - Authenticate-then-encrypt
    $t \leftarrow \text{Mac}(m)$ then $c \leftarrow \text{Enc}(m||t)$
    (insecure but the counter-example is involved)
  - Encrypt-then-authenticate
    $c \leftarrow \text{Enc}(m)$ then $t \leftarrow (c)$

<
(secure; intuitively, encrypting the message as c ensures it is secret
(even if some other function is computed on it)
and then computing the tag on this c ensures
c was not tampered with en route
and the tagging algorithm never saw m directly
so no leakage)