CSE5014 CRYPTOGRAPHY AND NETWORK SECURITY

Dr. Qi Wang

Department of Computer Science and Engineering
Office: Room413, CoE South Tower
Email: wangqi@sustech.edu.cn

Formal definition

- Fix A, Π. Define a randomized experiment $Forge_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *oracle* $Mac_k(\cdot)$; let M be the set of messages submitted to this oracle
  3. A outputs (m, t)
  4. A *succeeds*, and the experiment evaluates to 1, if $Vrfy_k(m, t) = 1$ and $m \notin M$

Formal definition

- Fix A, Π. Define a randomized experiment $Forge_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *oracle* $Mac_k(\cdot)$; let M be the set of messages submitted to this oracle
  3. A outputs (m, t)
  4. A *succeeds*, and the experiment evaluates to 1, if $Vrfy_k(m, t) = 1$ and $m \notin M$

Definition 6.2 Π is *secure* if for all PPT attackers A, there is a *negligible* function ϵ such that

$$\Pr[Forge_{A,\Pi}(n) = 1] \leq \epsilon(n)$$

Formal definition

- Fix A, Π. Define a randomized experiment $Forge_{A,\Pi}(n)$:
  1. $k \leftarrow Gen(1^n)$
  2. $A(1^n)$ interacts with an *oracle* $Mac_k(\cdot)$; let M be the set of messages submitted to this oracle
  3. A outputs (m, t)
  4. A *succeeds*, and the experiment evaluates to 1, if $Vrfy_k(m, t) = 1$ and $m \notin M$

Definition 6.2 Π is *secure* if for all PPT attackers A, there is a *negligible* function ϵ such that

$$\Pr[Forge_{A,\Pi}(n) = 1] \leq \epsilon(n)$$

Note: A CPA-secure encryption scheme is not a secure MAC.

Construction

- Let F be a length-preserving PRF (aka block cipher)

Construction

- Let F be a length-preserving PRF (aka block cipher)

- Construct the following MAC Π:
  - Gen: choose a uniform key k for F
  - $Mac_k(m)$: output $F_k(m)$
  - $Vrfy_k(m, t)$: output 1 iff $F_k(m) = t$

Construction

- Let F be a length-preserving PRF (aka block cipher)

- Construct the following MAC Π:
  - Gen: choose a uniform key k for F
  - $Mac_k(m)$: output $F_k(m)$
  - $Vrfy_k(m, t)$: output 1 iff $F_k(m) = t$

- **Theorem 6.3** Π is a *secure* MAC

Construction

- Let F be a length-preserving PRF (aka block cipher)

- Construct the following MAC Π:
  - *Gen*: choose a uniform key k for F
  - $Mac_k(m)$: output $F_k(m)$
  - $Vrfy_k(m,t)$: output 1 iff $F_k(m) = t$

- **Theorem 6.3** Π is a *secure* MAC

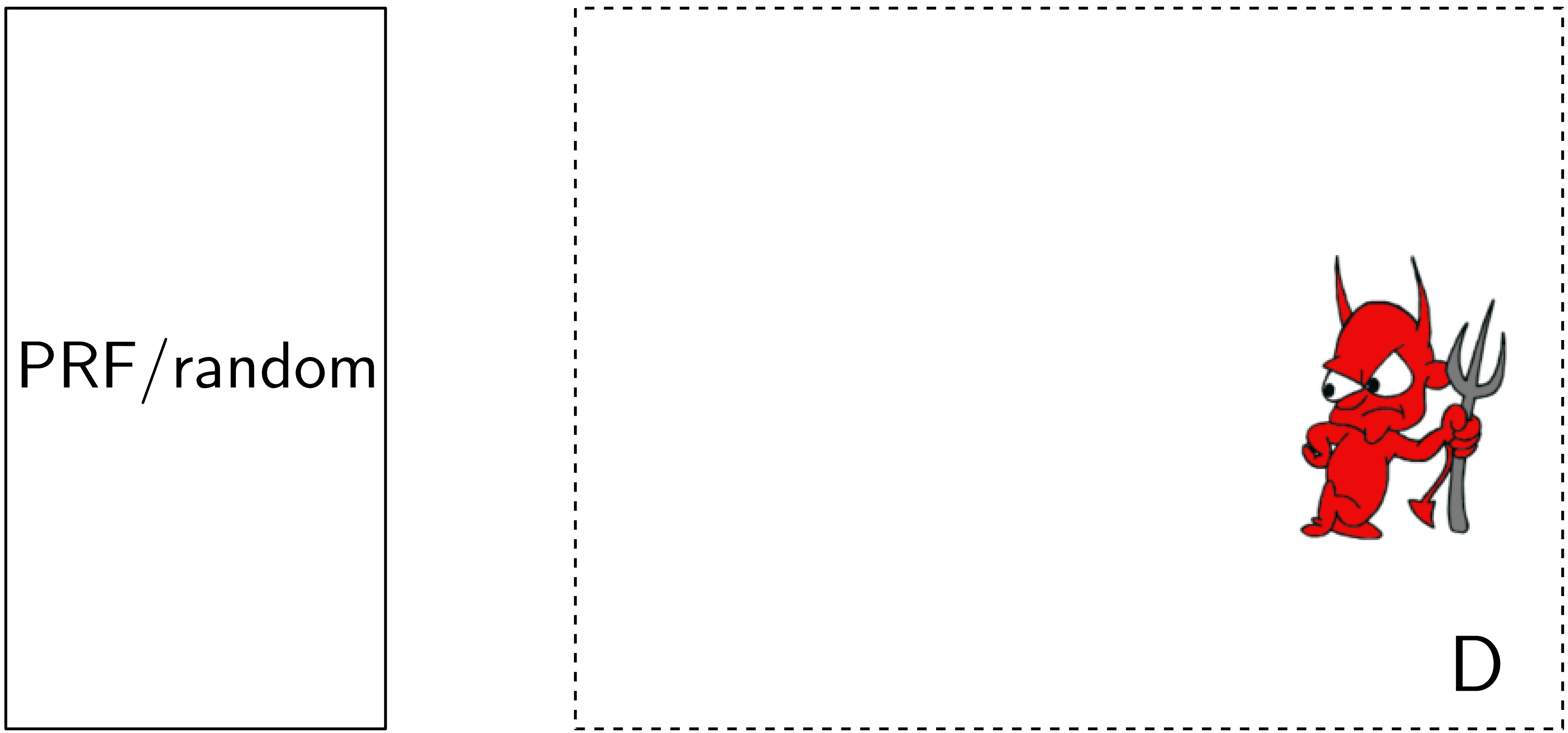
Theorem 4.6 in Textbook

Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC

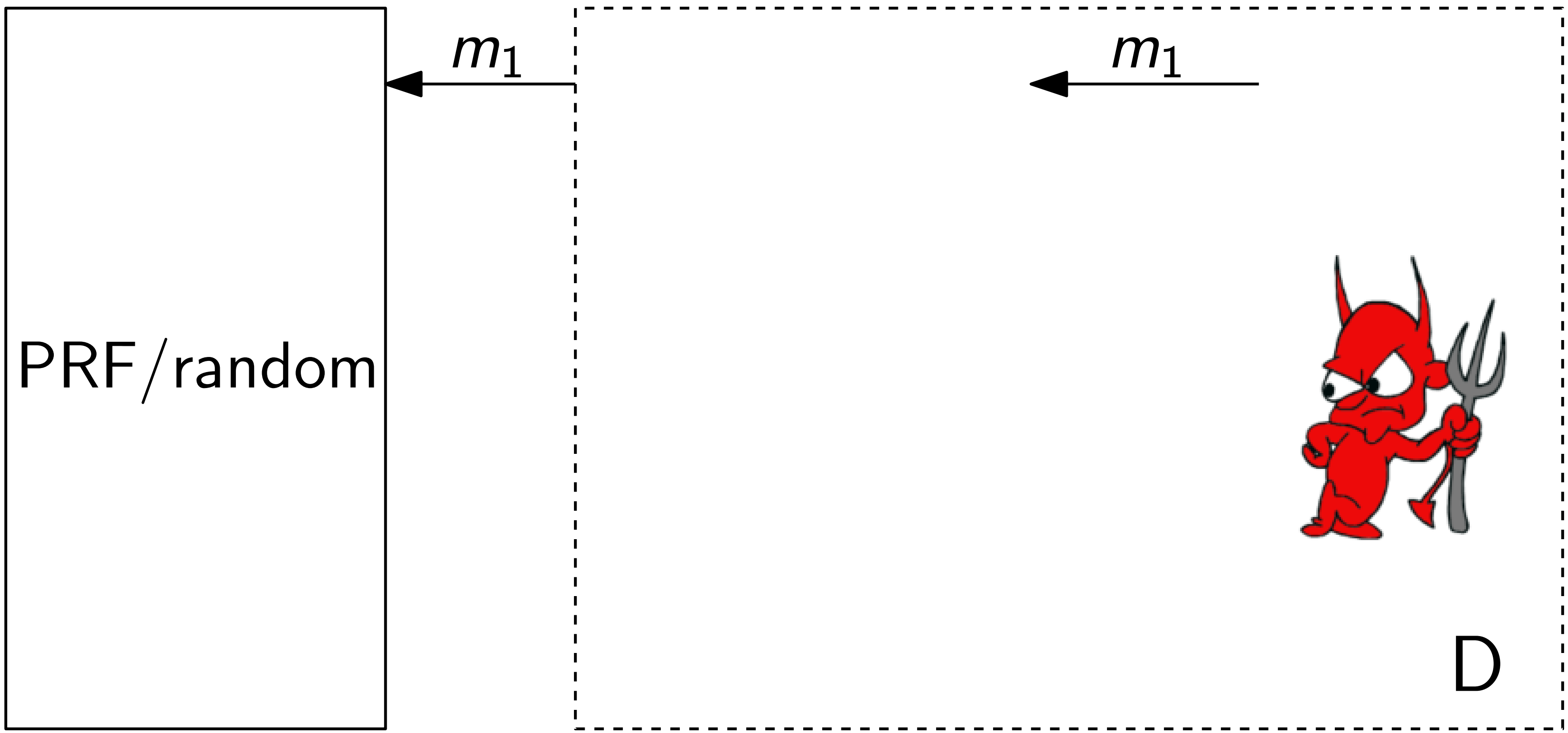
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



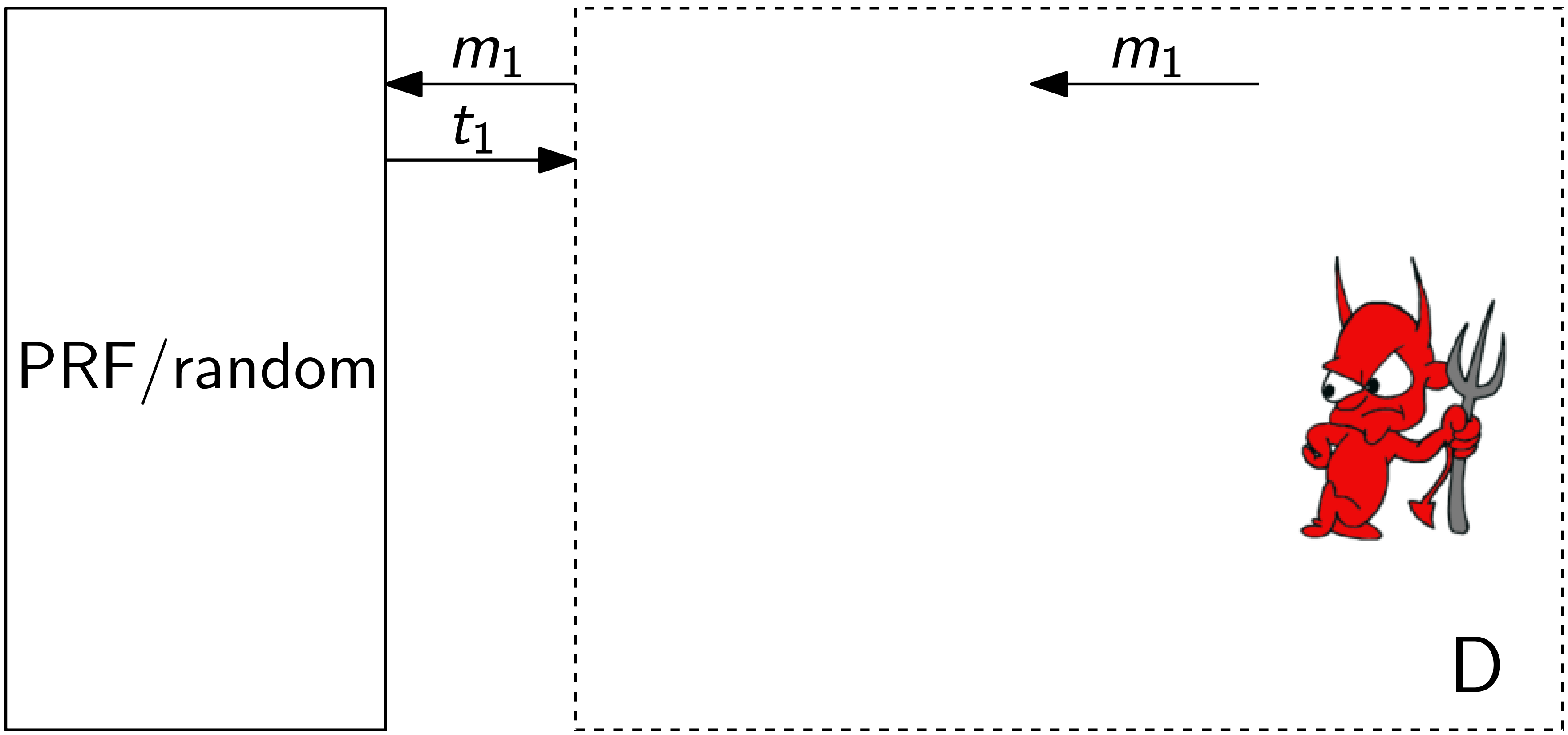
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



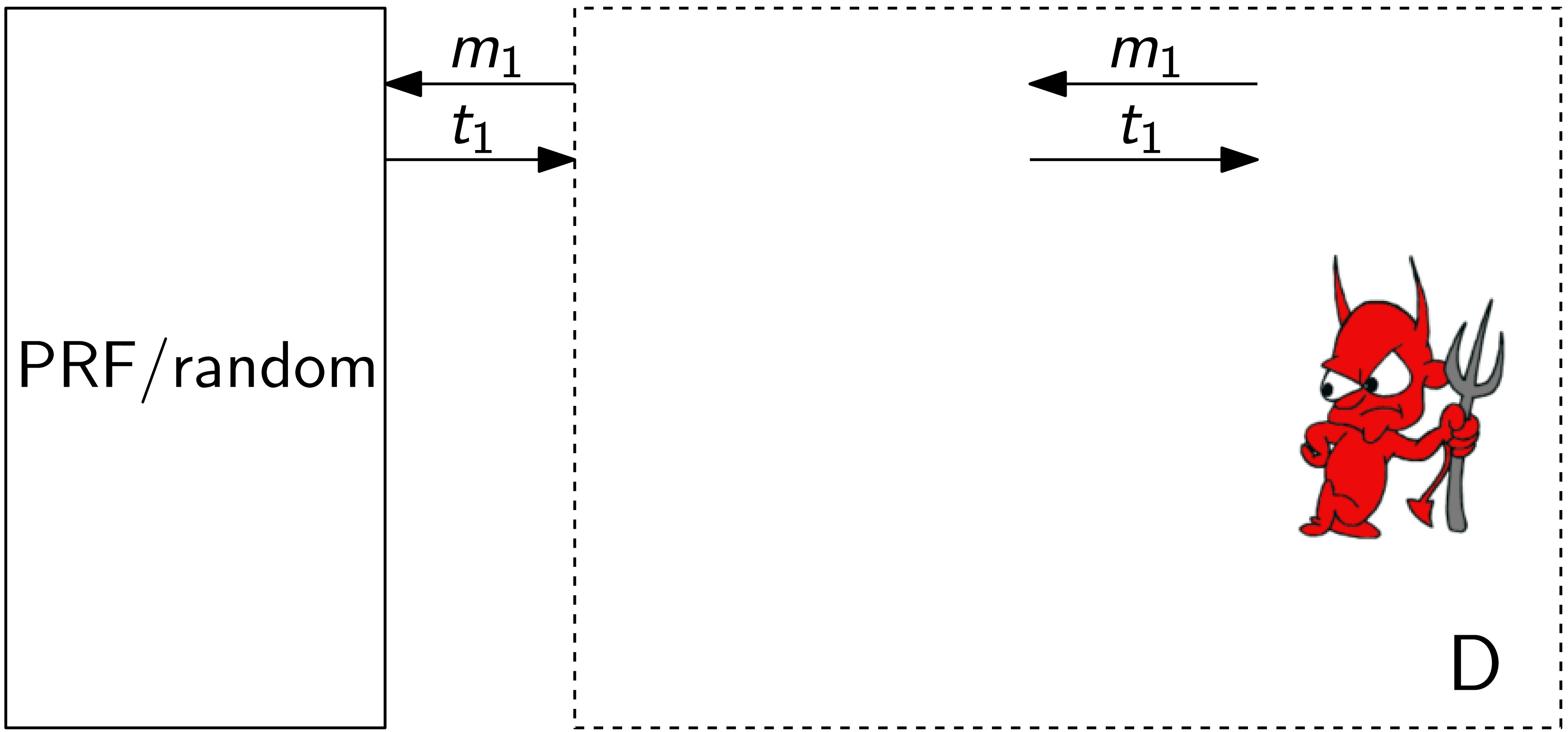
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



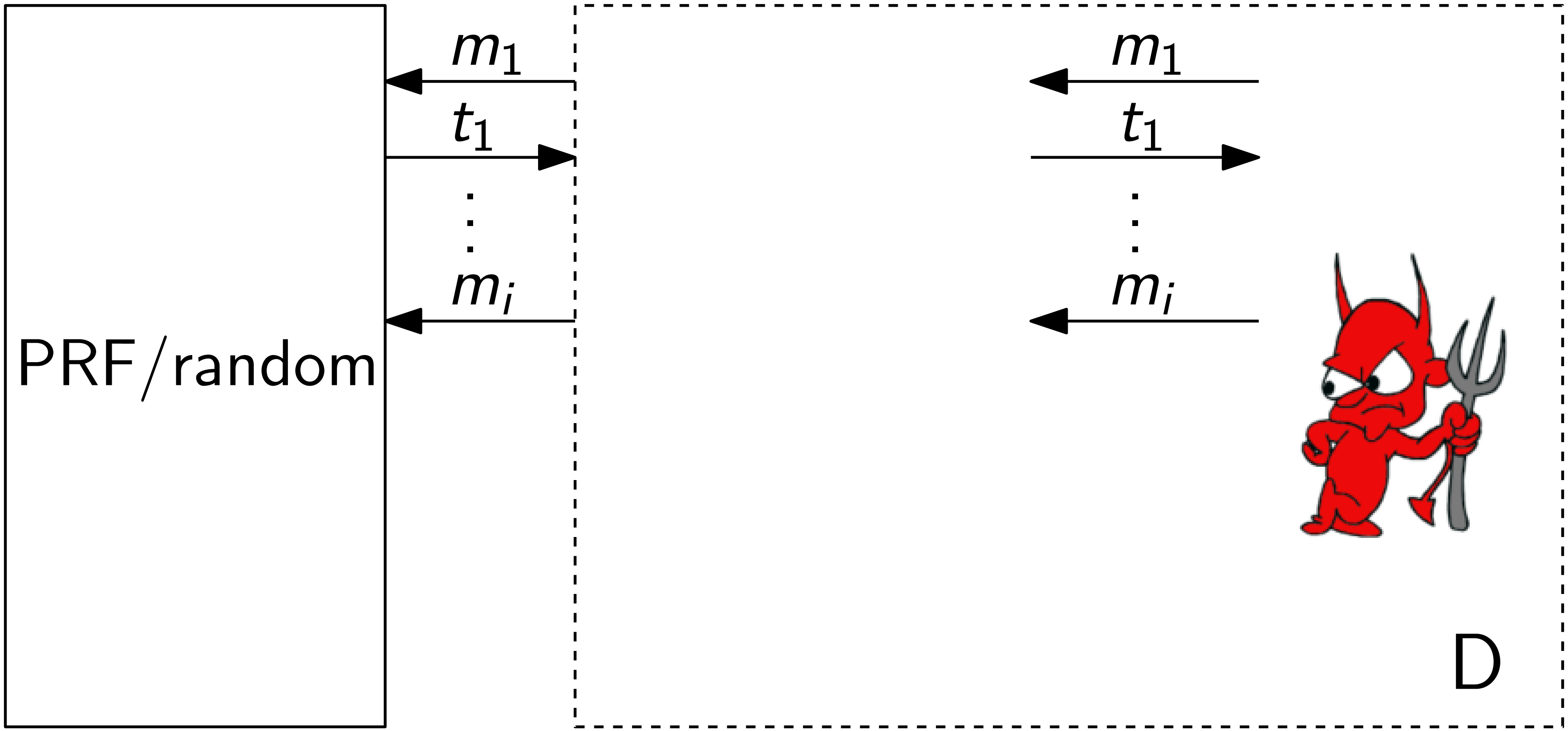
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



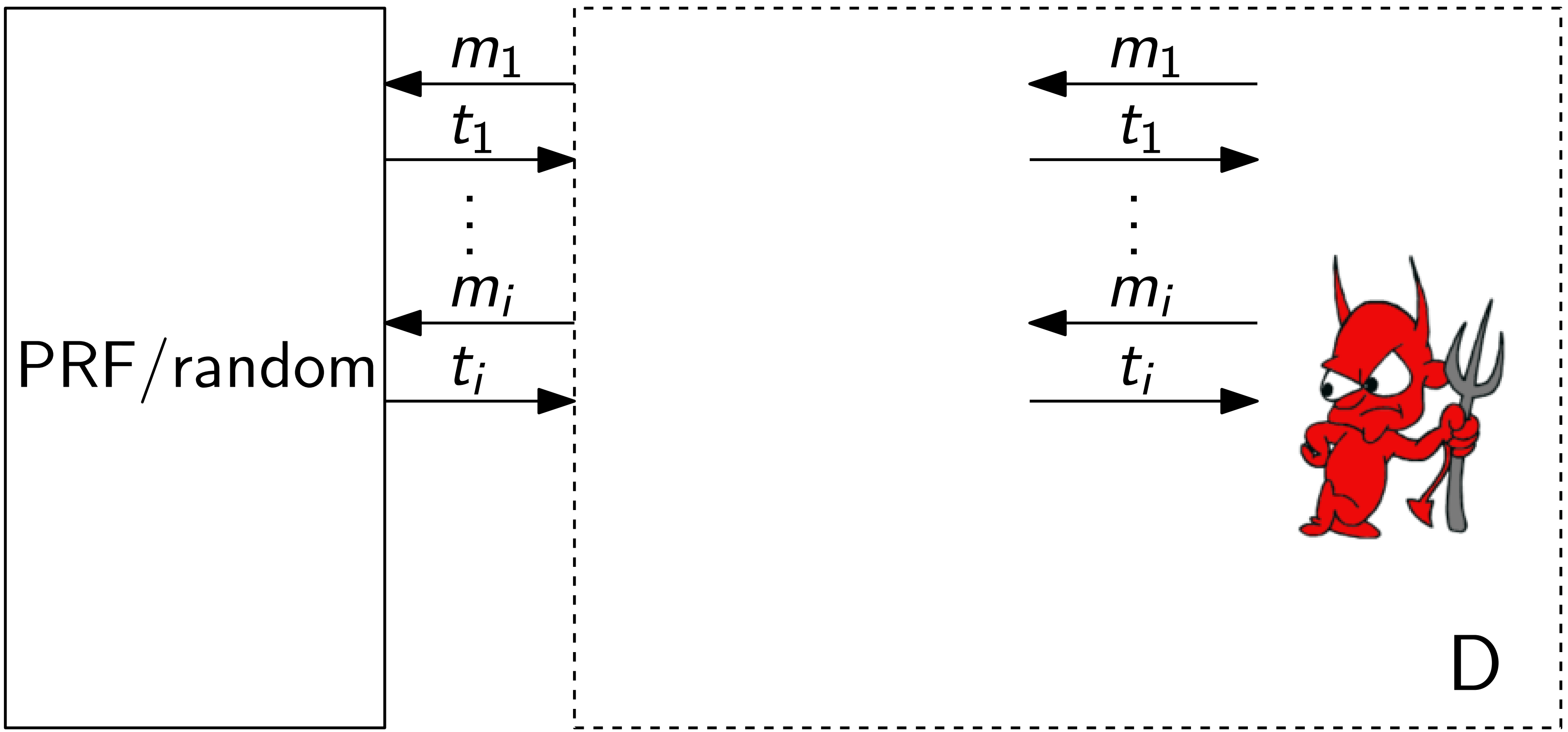
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



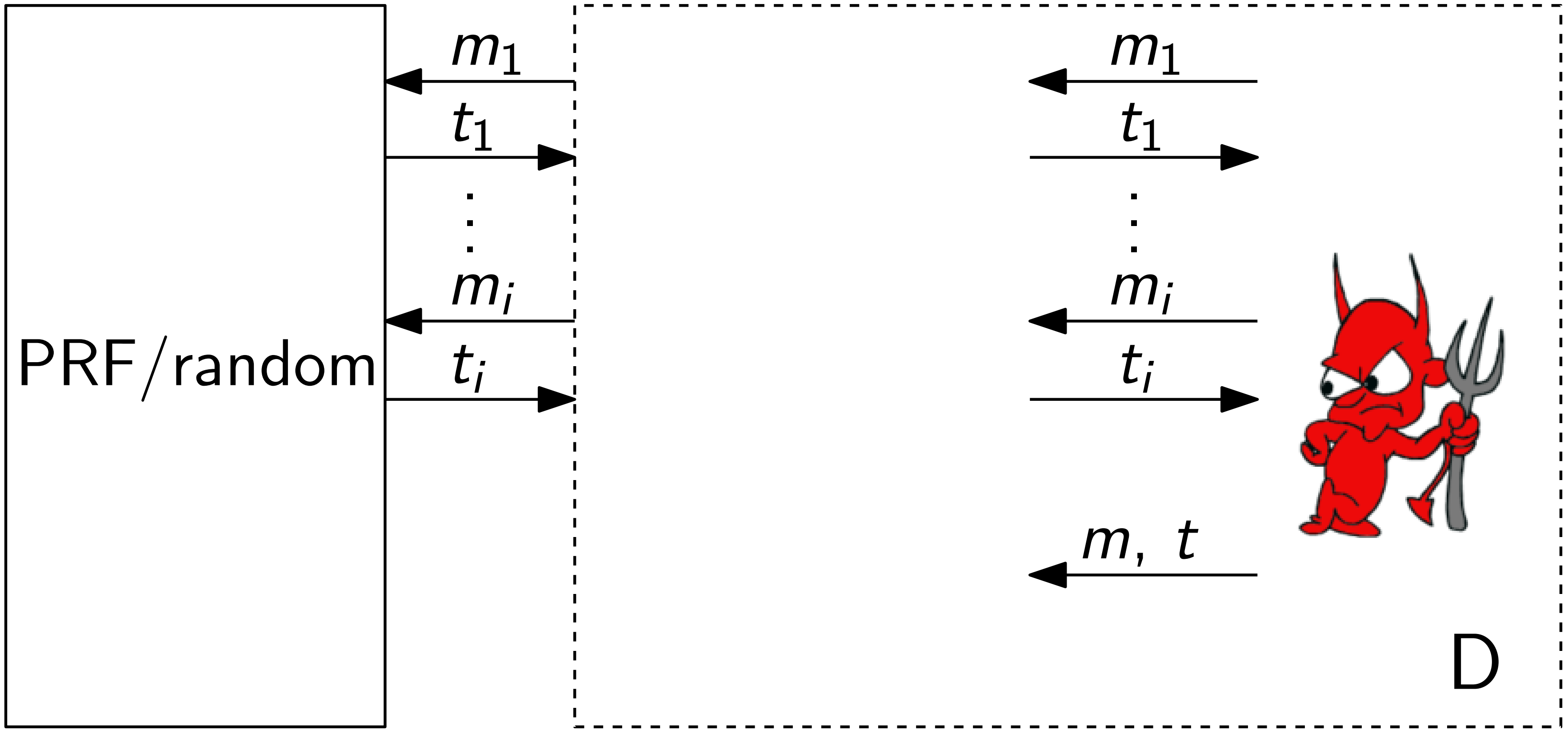
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



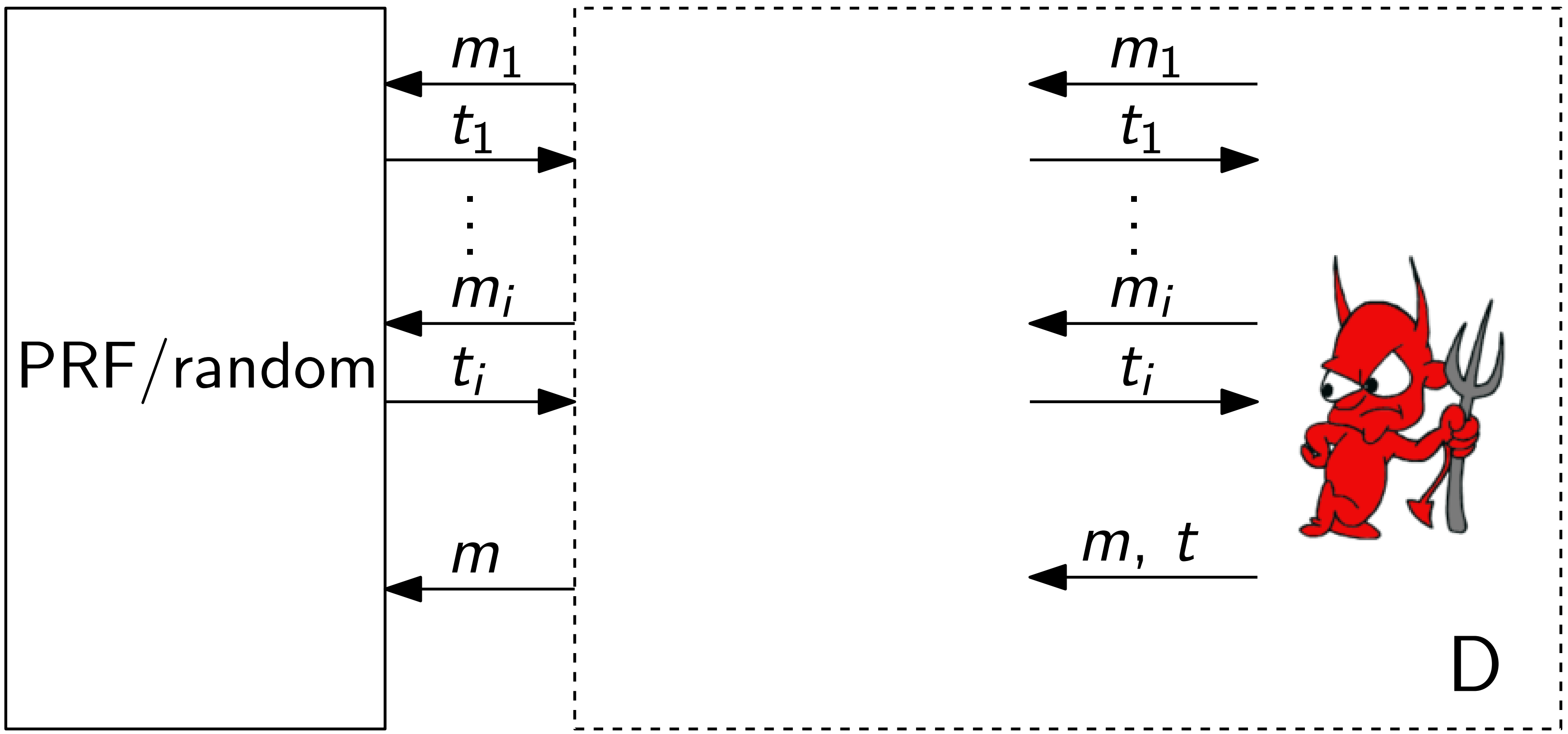
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



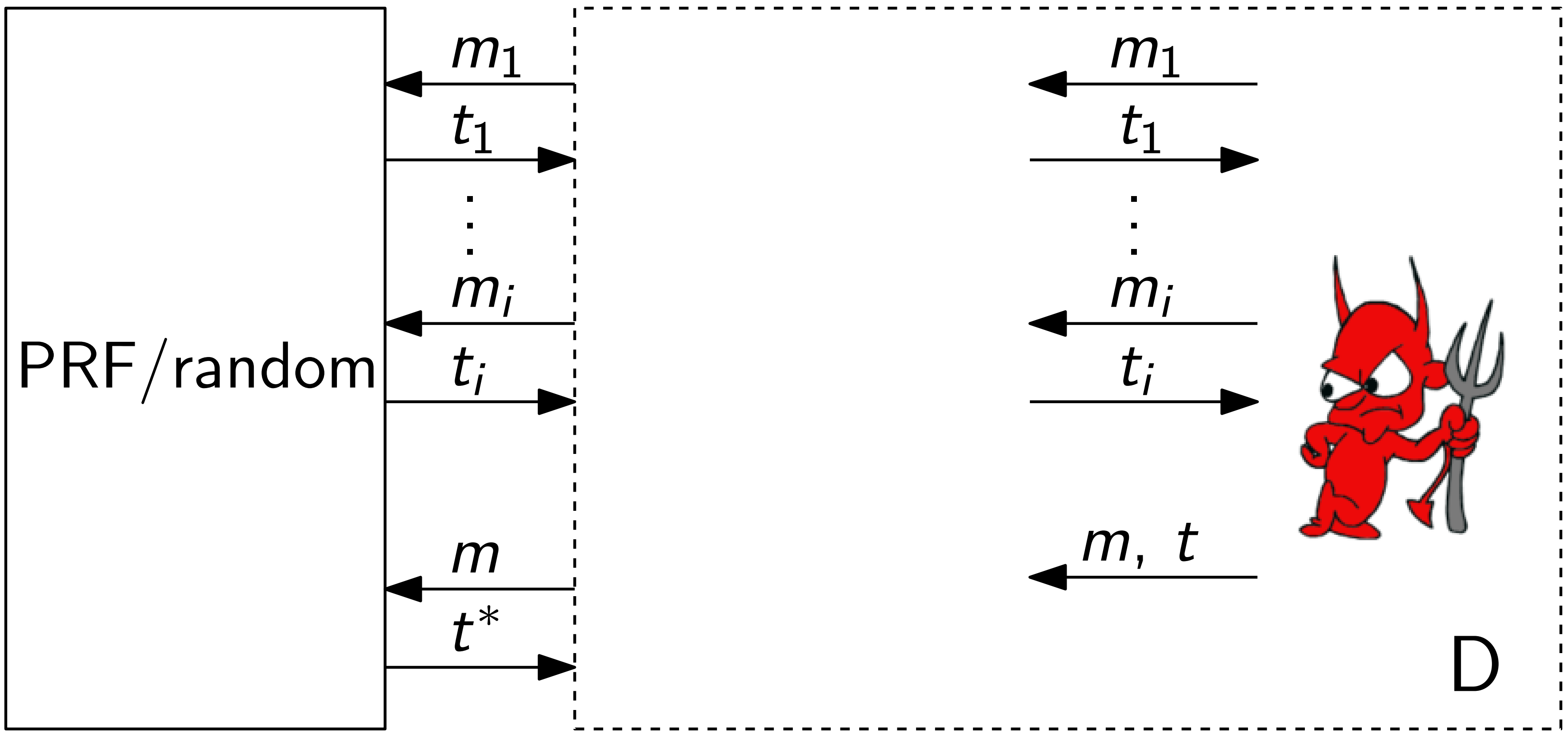
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



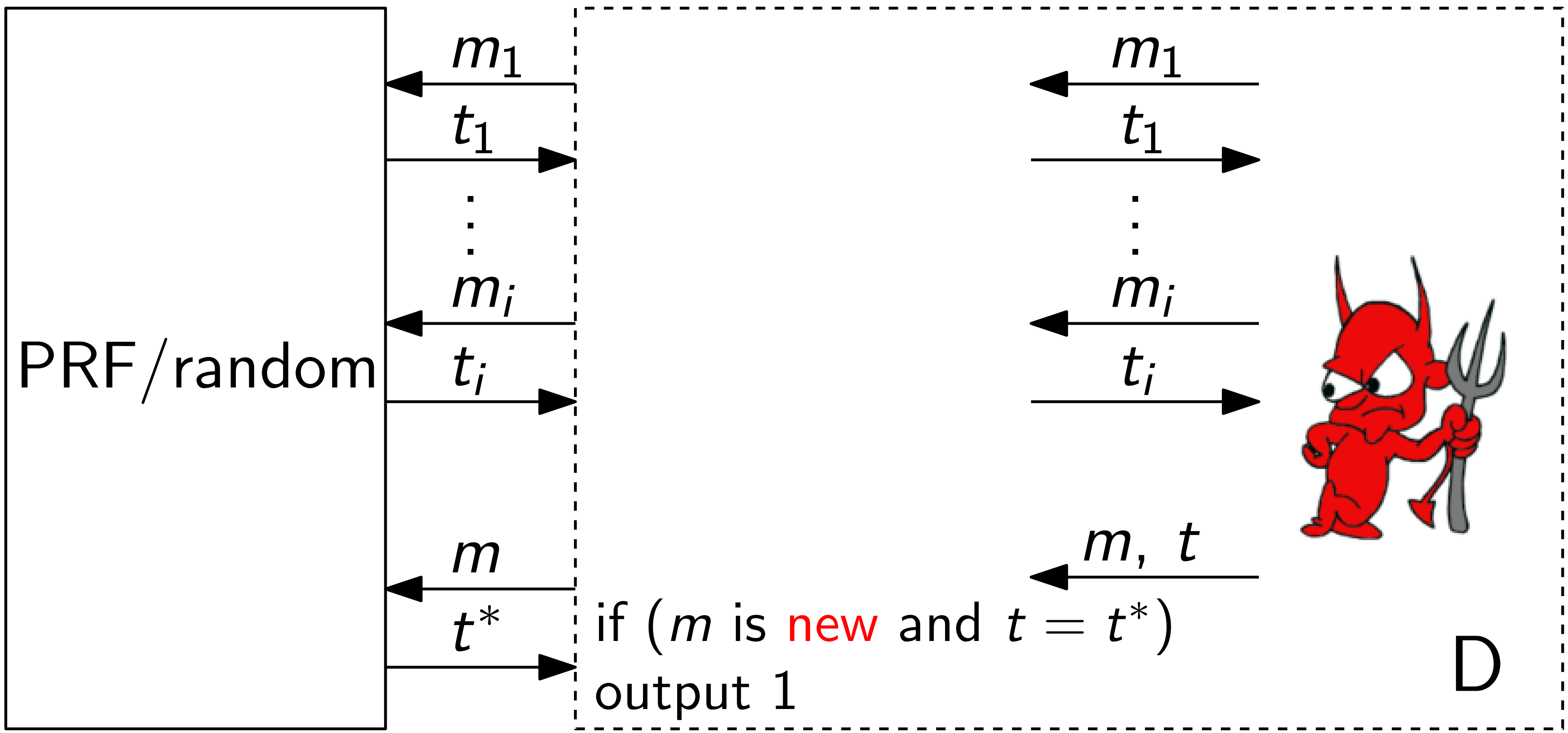
Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



Proof by reduction

- **Theorem 6.3** Π is a *secure* MAC



Analysis

- When D interacts with F_k for uniform k, the view of the adversary is *identical* to its view in the real MAC experiment
  - $\Pr[D^{F_k}$ outputs $1] = \Pr[Forge_{Adv,\Pi}(n) = 1]$

Analysis

- When D interacts with F_k for uniform k, the view of the adversary is *identical* to its view in the real MAC experiment
  - $\Pr[D^{F_k} \text{ outputs } 1] = \Pr[Forge_{Adv,\Pi}(n) = 1]$
- When D interacts with uniform f, then seeing $f(m_1), \ldots, f(m_i)$ does not help predict $f(m)$ for any $m \notin \{m_1, \ldots, m_i\}$
  - $\Pr[D^{f} \text{ outputs } 1] \leq 2^{-n}$

Analysis

- When D interacts with F_k for uniform k, the view of the adversary is *identical* to its view in the real MAC experiment
  - $\Pr[D^{F_k} \text{ outputs } 1] = \Pr[Forge_{Adv,\Pi}(n) = 1]$
- When D interacts with uniform f, then seeing $f(m_1), \ldots, f(m_i)$ does not help predict $f(m)$ for any $m \notin \{m_1, \ldots, m_i\}$
  - $\Pr[D^{f} \text{ outputs } 1] \leq 2^{-n}$
- Since F is a *PRF*, $|\Pr[D^{F_k} \text{ outputs } 1] - \Pr[D^{f} \text{ outputs } 1]| < negl(n)$

  $\Rightarrow \Pr[Forge_{Adv,\Pi}(n) = 1] = \Pr[D^{F_k} \text{ outputs } 1] \leq 2^{-n} + negl(n)$

Drawbacks

- This only works for *fixed-length* messages
- This only works for *short* messages
  - E.g., AES has a 128-bit block size (shorter than a tweet!)

Drawbacks

- This only works for *fixed-length* messages
- This only works for *short* messages
  - E.g., AES has a 128-bit block size (shorter than a tweet!)
- So, the previous construction is limited to authenticating *short, fixed-length* messages

Drawbacks

- This only works for *fixed-length* messages
- This only works for *short* messages
  - E.g., AES has a 128-bit block size (shorter than a tweet!)
- So, the previous construction is limited to authenticating *short, fixed-length* messages
- One natural idea:
  - $Mac'_k(m_1, \ldots, m_\ell) = Mac_k(m_1), \ldots, Mac_k(m_\ell)$
  - $Vrfy'_k(m_1, \ldots, m_\ell, t_1, \ldots, t_\ell) = 1$ iff
    $Vrfy_k(m_i, t_i) = 1$ for all i

Drawbacks

- This only works for *fixed-length* messages
- This only works for *short* messages
  - E.g., AES has a 128-bit block size (shorter than a tweet!)
- So, the previous construction is limited to authenticating *short, fixed-length* messages
- One natural idea:
  - $Mac'_k(m_1, \ldots, m_\ell) = Mac_k(m_1), \ldots, Mac_k(m_\ell)$
  - $Vrfy'_k(m_1, \ldots, m_\ell, t_1, \ldots, t_\ell) = 1$ iff $Vrfy_k(m_i, t_i) = 1$ for all i
  - Is this secure?

Construction

- Need to prevent (at least)
  - Block reordering
  - Truncation
  - "Mixing-and-matching" blocks from multiple messages

Construction

- Need to prevent (at least)
  - Block reordering
  - Truncation
  - "Mixing-and-matching" blocks from multiple messages

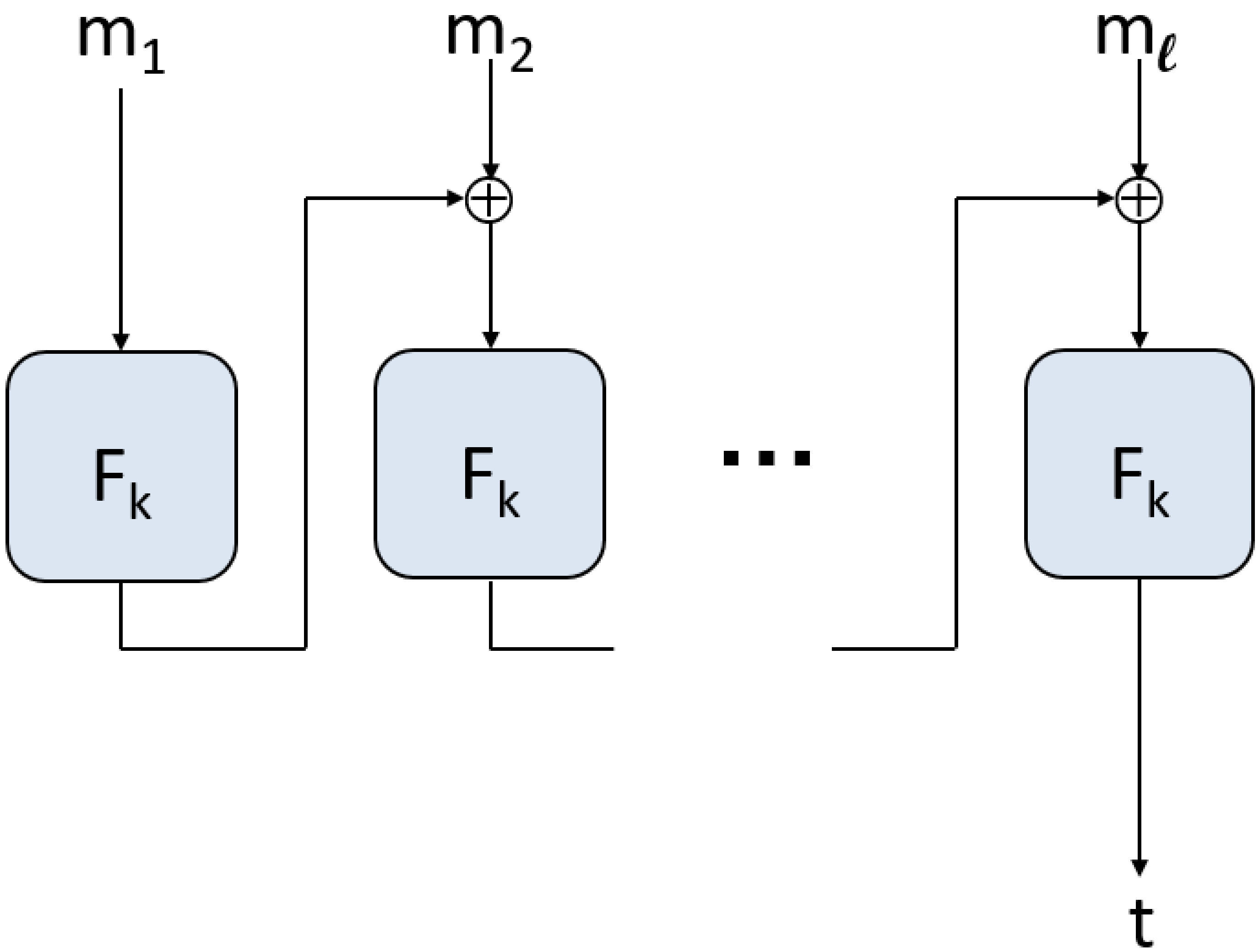
- One solution:
  - $Mac'_k(m_1, \ldots, m_\ell) =$
    $r, Mac_k(r||\ell||1||m_1), \ldots, Mac_k(r||\ell||\ell||m_\ell)$

Construction

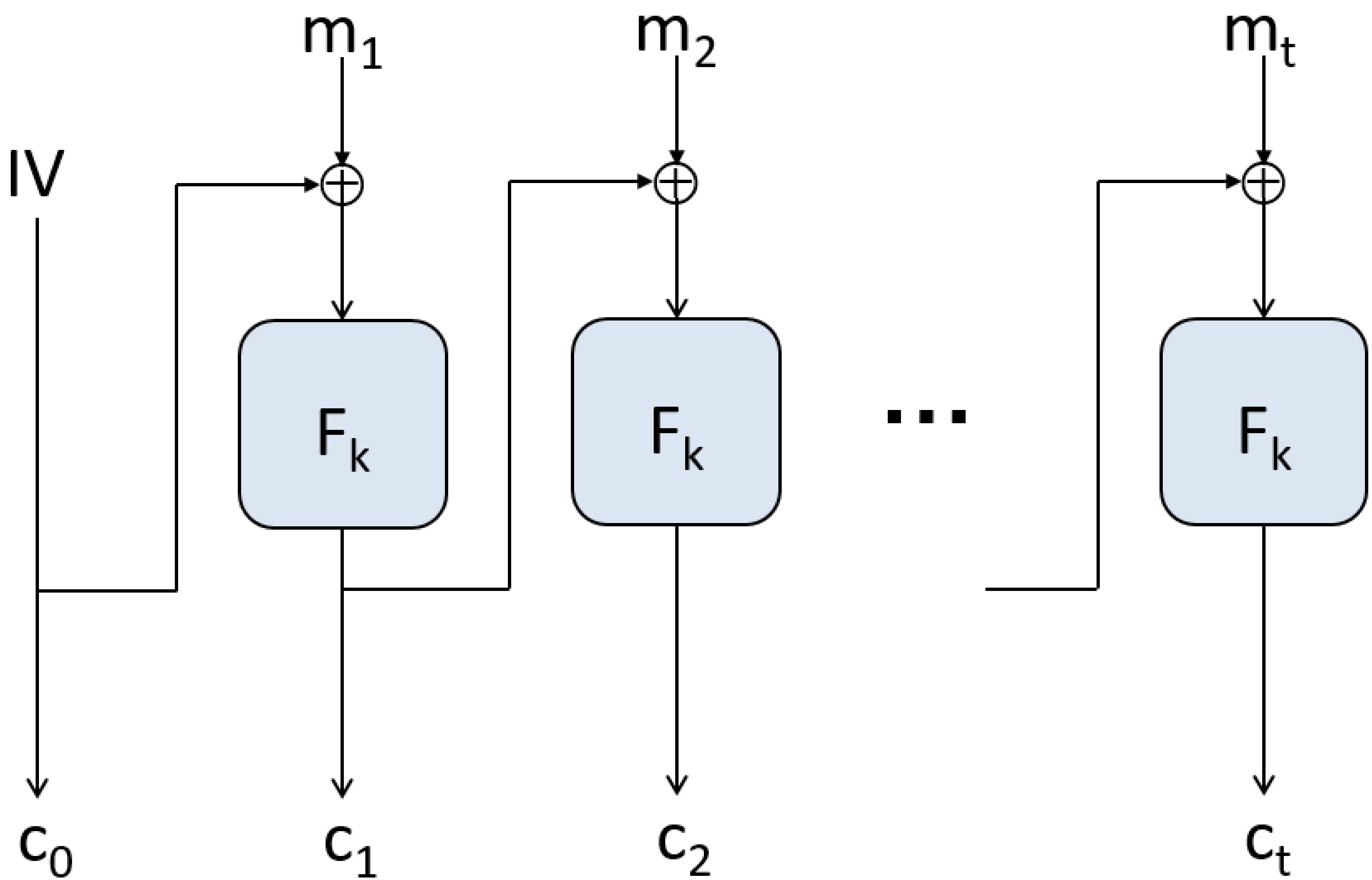
- Need to prevent (at least)
  - Block reordering
  - Truncation
  - "Mixing-and-matching" blocks from multiple messages

- One solution:
  - $Mac'_k(m_1, \ldots, m_\ell) = r, Mac_k(r||\ell||1||m_1), \ldots, Mac_k(r||\ell||\ell||m_\ell)$
  - See Construction 4.7 & Theorem 4.8
  - Not very efficient. Can we do better?

(Basic) CBC-MAC



CBC mode



CBC-MAC vs. CBC-mode

- CBC-MAC is *deterministic* (no IV)
  - MACs do not need to be randomized to be secure
  - Verification is done by re-computing the result
- In CBC-MAC, *only the final value* is output
- Both are essential for security

Security of (basic) CBC-MAC

- If F is a PRF with block length n, then for any fixed ℓ basic CBC-MAC is a secure MAC for messages of length $\ell \cdot n$
- The sender and receiver must agree on the length parameters ℓ in advance
  - Basic CBC-MAC is not secure if this is not done! (Attacks?)

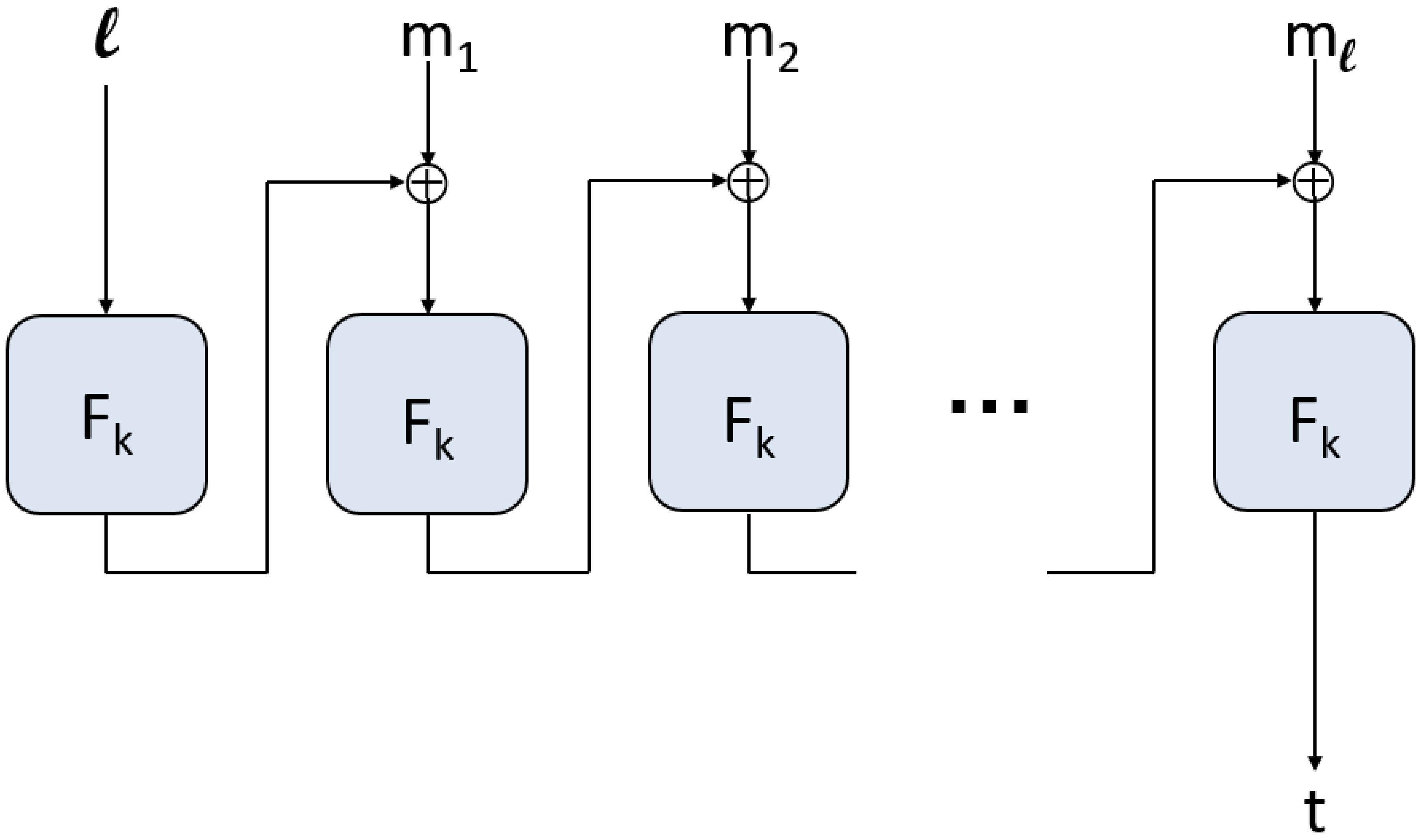
Security of (basic) CBC-MAC

- If F is a PRF with block length n, then for any fixed ℓ basic CBC-MAC is a secure MAC for messages of length $\ell \cdot n$

- The sender and receiver must agree on the length parameters ℓ in advance
  - Basic CBC-MAC is not secure if this is not done! (Attacks?)

- Several ways to handle variable-length messages
  - One of the simplest: *prepend* the message length before applying (basic) CBC-MAC

CBC-MAC



Authenticated encryption (secrecy + integrity)

- We have shown primitives for achieving *secrecy* and *integrity* in the private-key setting
  What if we want to achieve both?

Authenticated encryption (secrecy + integrity)

- We have shown primitives for achieving *secrecy* and *integrity* in the private-key setting
  What if we want to achieve both?

- Secrecy notion: CCA-security

- Integrity notion: unforgeability
  - Adversary cannot generate ciphertext that decrypts to a previously unencrypted message

Constructions

- There are three natural generic constructions:
  - Encrypt and Authenticate (E&A): Compute $c = Enc_{k_1}(m)$ and $t = Mac_{k_2}(m)$ and send (c, t) (SSH style)
  - Authenticate and then Encrypt (AtE): Compute $t = Mac_{k_2}(m)$ and then $Enc_{k_1}(t)$ (SSL style)
  - Encrypt and then Authentication (EtA): Compute $c = Enc_{k_1}(m)$ and $t = Mac_{k_1}(c)$ and send (c, t) (IPSec style)

Constructions

- There are three natural generic constructions:
  - Encrypt and Authenticate (E&A): Compute $c = Enc_{k_1}(m)$ and $t = Mac_{k_2}(m)$ and send (c, t) (SSH style)
  - Authenticate and then Encrypt (AtE): Compute $t = Mac_{k_2}(m)$ and then $Enc_{k_1}(t)$ (SSL style)
  - Encrypt and then Authentication (EtA): Compute $c = Enc_{k_1}(m)$ and $t = Mac_{k_1}(c)$ and send (c, t) (IPSec style)

Note: In all these methods, we use independent keys (k_1, k_2) for encryption and authentication

Constructions

- Th
  - –
  - –
  - –

The order of encryption and authentication for protecting communications (Or: how secure is SSL?)*

Hugo Krawczyk†

June 6, 2001

Abstract

We study the question of how to generically compose *symmetric* encryption and authentication when building "secure channels" for the protection of communications over insecure networks. We show that any secure channels protocol designed to work with any combination of secure encryption (against chosen plaintext attacks) and secure MAC must use the encrypt-then-authenticate method. We demonstrate this by showing that the other common methods of composing encryption and authentication, including the authenticate-then-encrypt method used in SSL, are not generically secure. We show an example of an encryption function that provides (Shannon's) perfect secrecy but when combined with any MAC function under the authenticate-then-encrypt method yields a totally insecure protocol (for example, finding passwords or credit card numbers transmitted under the protection of such protocol becomes an easy task for an active attacker). The same applies to the encrypt-and-authenticate method used in SSH.

On the positive side we show that the authenticate-then-encrypt method is secure if the encryption method in use is either CBC mode (with an underlying secure block cipher) or a stream cipher (that xor the data with a random or pseudorandom pad). Thus, while we show the generic security of SSL to be broken, the current standard implementations of the protocol that use the above modes of encryption are safe.

Nc

$(k_1$

14 -

Generic constructions

- Generically combine an *encryption scheme* and a *MAC*

Generic constructions

- Generically combine an *encryption scheme* and a *MAC*
- **Goal**: the combination should be an authenticated encryption scheme when instantiated with any *CPA-secure* encryption scheme and any *secure* MAC

Generic constructions

- Generically combine an *encryption scheme* and a *MAC*
- **Goal**: the combination should be an authenticated encryption scheme when instantiated with any *CPA-secure* encryption scheme and any *secure* MAC

- Encrypt and authenticate (E&A)

k1, k2

m

$c \leftarrow \mathrm{Enc}_{k1}(m)$

$t = \mathrm{Mac}_{k2}(m)$

$\xrightarrow{\text{c, t}}$

k1, k2

$m = \mathrm{Dec}_{k1}(c)$

$\mathrm{Vrfy}_{k2}(m, t) = 1?$

Problems

- The *tag* t might leak information about m!
  - Nothing in the definition of security for a MAC implies that it hides information about m
  - So, the combination may not even be *EAV-secure*

Problems

- The *tag* t might leak information about m!
  - Nothing in the definition of security for a MAC implies that it hides information about m
  - So, the combination may not even be *EAV-secure*

- If the MAC is deterministic (as is CBC-MAC), then the tag leaks whether the same message is encrypted twice
  - I.e., the combination will not be *CPA-secure*

Authenticate then encrypt (AtE)

k1, k2  $\xrightarrow{\quad c \quad}$  k1, k2

m

$t = \mathrm{Mac}_{k2}(m)$

$c \leftarrow \mathrm{Enc}_{k1}(m \mid t)$

$m \mid t = \mathrm{Dec}_{k1}(c)$

$\mathrm{Vrfy}_{k2}(m, t) = 1?$

Authenticate then encrypt (AtE)

c

k1, k2 — k1, k2

m

$t = \mathrm{Mac}_{k2}(m)$

$c \leftarrow \mathrm{Enc}_{k1}(m \mid t)$

$m \mid t = \mathrm{Dec}_{k1}(c)$

$\mathrm{Vrfy}_{k2}(m, t) = 1?$

- Problems
  - Padding-oracle attack still works
  - Other counterexamples are also possible
  - The combination may not be *CCA-secure*

AtE is not secure in general

- Idea: consider the *CPA-secure* entryption scheme in **Theorem 5.1**, if combined with every secure MAC in the form of AtE, by proving it is malleable, we show that it is not *CCA-secure*.

AtE is not secure in general

- Idea: consider the *CPA-secure* entryption scheme in **Theorem 5.1**, if combined with every secure MAC in the form of AtE, by proving it is malleable, we show that it is not *CCA-secure*.

  $Gen(1^n)$: choose a uniform key $k \in \{0,1\}^n$
  $Enc_k(m)$, for $|m| = |k|$
  – Choose uniform $r \in \{0,1\}^n$ (*nonce/ initialization vector*)
  – Output ciphertext $\langle r, F_k(r) \oplus m \rangle$
  $Dec_k(c_1, c_2)$: output $c_2 \oplus F_k(c_1)$

  Theorem 5.1 If F is a pseudorandom function, then this scheme is *CPA-secure*.

AtE is not secure in general

- Given such a CPA-secure encryption schemed (Gen, Enc, Dec), we build a new encryption scheme (Gen', Enc', Dec'):

AtE is not secure in general

- Given such a CPA-secure encryption schemed (Gen, Enc, Dec), we build a new encryption scheme (Gen', Enc', Dec'):

  For an n-bit plaintext m, first apply an encoding of m into a $2n$-bit string m' by representing each bit m_i, $i = 1, \ldots, n$, in m with two bits in m':
  1. if bit $m_i = 0$, then $(m'_{2i-1}, m'_{2i}) = (0, 0)$
  2. if bit $m_i = 1$, then $(m'_{2i-1}, m'_{2i}) = (0, 1)$ or $(1, 0)$

  The encryption Enc is then applied to m'.

AtE is not secure in general

- Given such a CPA-secure encryption schemed (*Gen*, *Enc*, *Dec*), we build a new encryption scheme (*Gen*′, *Enc*′, *Dec*′):

  For an n-bit plaintext m, first apply an encoding of m into a $2n$-bit string m' by representing each bit m_i, $i = 1, \ldots, n$, in m with two bits in m':
  1. if bit $m_i = 0$, then $(m'_{2i-1}, m'_{2i}) = (0, 0)$
  2. if bit $m_i = 1$, then $(m'_{2i-1}, m'_{2i}) = (0, 1)$ or $(1, 0)$

  The encryption *Enc* is then applied to m'.

  For decrypting $c = Enc'_k(m')$, one first applies the decryption *Dec* to obtain m', which is then decoded into m by mapping
  $(0, 0) \mapsto 0$, $(0, 1)$ or $(1, 0) \mapsto 1$
  If m' contains a pair $(m'_{2i-1}, m'_{2i}) = (1, 1)$, the decoding outputs the invalidity sign $\perp$.

AtE is not secure in general

- We consider an active attack:
  When an attacker Eve sees a transmitted ciphertext $c = Enc'_k(m)$, she can learn the first bit m_1 of m as follows:
  She intercepts c, flips the first two bits (c_1, c_2) of c, and sends the modified ciphertext c' to its destination. If she can obtain the information of whether the decryption output a valid or invalid plaintext then Eve learns the first bit of m. This is so since the modified c' is valid if and only if $m_1 = 1$.

AtE is not secure in general

- We consider an active attack:
  When an attacker Eve sees a transmitted ciphertext $c = Enc'_k(m)$, she can learn the first bit m_1 of m as follows:
  She intercepts c, flips the first two bits (c_1, c_2) of c, and sends the modified ciphertext c' to its destination. If she can obtain the information of whether the decryption output a valid or invalid plaintext then Eve learns the first bit of m.
  This is so since the modified c' is valid if and only if $m_1 = 1$.
  Recall that AtE means compute $t = Mac_{k_2}(m)$ and then send $Enc_{k_1}(m||t)$. The MAC is applied to the data before encoding and encryption.

AtE is not secure in general

- We consider an active attack:
  When an attacker Eve sees a transmitted ciphertext $c = Enc'_k(m)$, she can learn the first bit m_1 of m as follows: She intercepts c, flips the first two bits (c_1, c_2) of c, and sends the modified ciphertext c' to its destination. If she can obtain the information of whether the decryption output a valid or invalid plaintext then Eve learns the first bit of m. This is so since the modified c' is valid if and only if $m_1 = 1$. If the original bit is 1, the change in ciphertext will result in the same decrypted plaintext and then the MAC check will succeed.

AtE is not secure in general

- We consider an active attack:
  When an attacker Eve sees a transmitted ciphertext $c = Enc'_k(m)$, she can learn the first bit m_1 of m as follows: She intercepts c, flips the first two bits (c_1, c_2) of c, and sends the modified ciphertext c' to its destination. If she can obtain the information of whether the decryption output a valid or invalid plaintext then Eve learns the first bit of m. This is so since the modified c' is valid if and only if $m_1 = 1$. If the original bit is 1, the change in ciphertext will result in the same decrypted plaintext and then the MAC check will succeed.
  In a sense, the MAC just makes things worse since a failure of authentication is easier to be learnt by the attacker.

AtE is not secure in general

- Instead of this CPA-secure encryption scheme, even we use a perfect secure one-time pad, AtE is not generally secure.

AtE is not secure in general

- Instead of this CPA-secure encryption scheme, even we use a perfect secure one-time pad, AtE is not generally secure.

  Note:

  1. The application of an encoding to a plaintext before encryption is used commonly for padding and other purposes.
  2. Encoding of this type can be motivated by stronger security requirements: e.g., to prevent an attacker from learning the exact length of transmitted messages or other traffic analysis information.

AtE is not secure in general

- Instead of this CPA-secure encryption scheme, even we use a perfect secure one-time pad, AtE is not generally secure.

  Note:

  1. The application of an encoding to a plaintext before encryption is used commonly for padding and other purposes.
  2. Encoding of this type can be motivated by stronger security requirements: e.g., to prevent an attacker from learning the exact length of transmitted messages or other traffic analysis information.

  If one wants to **claim** the security of AtE, one needs to analyze the combination as a whole or use stronger or specific properties of the encryption function.

AtE is not secure in general

- Instead of this CPA-secure encryption scheme, even we use a perfect secure one-time pad, AtE is not generally secure.

  Note:

  1. The application of an encoding to a plaintext before encryption is used commonly for padding and other purposes.
  2. Encoding of this type can be motivated by stronger security requirements: e.g., to prevent an attacker from learning the exact length of transmitted messages or other traffic analysis information.

  If one wants to **claim** the security of AtE, one needs to analyze the combination as a whole or use stronger or specific properties of the encryption function.

  Read the paper [Krawczyk 2001] & the textbook

  Note: This does not mean that SSL is not secure, but does mean that it is not *generically secure*.

Encrypt then authenticate (EtA)

k1, k2

m

$c \leftarrow \mathrm{Enc}_{k1}(m)$

$t = \mathrm{Mac}_{k2}(c)$

$\xrightarrow{c,\ t}$

k1, k2

$\mathrm{Vrfy}_{k2}(c, t) = 1?$

$m = \mathrm{Dec}_{k1}(c)$

Encrypt then authenticate (EtA)

k1, k2 $\qquad$ c, t $\longrightarrow$ $\qquad$ k1, k2

m

$c \leftarrow \mathrm{Enc}_{k1}(m)$

$t = \mathrm{Mac}_{k2}(c)$

$\mathrm{Vrfy}_{k2}(c, t) = 1?$

$m = \mathrm{Dec}_{k1}(c)$

- Security
  - If the encryption scheme is *CPA-secure* and the MAC is *secure* (with unique tags), then this is an *authenticated encryption scheme*
  - It achieves something even stronger: Given ciphertexts corresponding to (chosen) plaintexts $m_1, \ldots, m_k$, it is infeasible for an attacker to generate any new, valid ciphertext!

Encrypt then authenticate (EtA)

k1, k2

m

$c \leftarrow \mathrm{Enc}_{k1}(m)$

$t = \mathrm{Mac}_{k2}(c)$

$\xrightarrow{c,\ t}$

k1, k2

$\mathrm{Vrfy}_{k2}(c, t) = 1?$

$m = \mathrm{Dec}_{k1}(c)$

- Encrypt-then-authenticate (with independent keys) is the recommended generic approach for constructing *authenticated encryption*

Encrypt then authenticate (EtA)

k1, k2 $\xrightarrow{c,\ t}$ k1, k2

m

$c \leftarrow \mathrm{Enc}_{k1}(m)$

$t = \mathrm{Mac}_{k2}(c)$

$\mathrm{Vrfy}_{k2}(c, t) = 1?$

$m = \mathrm{Dec}_{k1}(c)$

- Encrypt-then-authenticate (with independent keys) is the recommended generic approach for constructing *authenticated encryption*
- Other, more efficient constructions have been proposed and are an active area of research and standardization
  https://competitions.cr.yp.to/caesar.html

Encrypt then authenticate (EtA)

CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness

Timeline

- M-20, 2012.07.05–06: DIAC: Directions in Authenticated Ciphers. Stockholm.
- M-14, 2013.01.15: Competition announced at the Early Symmetric Crypto workshop in Mondorf-les-Bains; also announced online.
- M-7, 2013.08.11–13: DIAC 2013: Directions in Authenticated Ciphers 2013. Chicago.
- M0, 2014.03.15: Deadline for first-round submissions.
- M2, 2014.05.15: Deadline for first-round software.
- M5, 2014.08.23–24: DIAC 2014: Directions in Authenticated Ciphers 2014. Santa Barbara.
- M16, 2015.07.07: Announcement of second-round candidates.
- M17, 2015.08.29: Deadline for second-round tweaks.
- M18, 2015.09.15: Deadline for second-round software.
- M18, 2015.09.28–29: DIAC 2015: Directions in Authenticated Ciphers 2015. Singapore.
- M27, 2016.06.30: Deadline for Verilog/VHDL.
- M29, 2016.08.15: Announcement of third-round candidates.
- M30, 2016.09.15: Deadline for third-round tweaks.
- M30, 2016.09.26–27: DIAC 2016. Nagoya, Japan.
- M31, 2016.10.15: Deadline for third-round software.
- M40, 2017.07.15: Deadline for third-round Verilog/VHDL.
- M40, 2017.07.15: Deadline for optimized third-round software.
- M48, 2018.03.05: Announcement of finalists.
- M59: 2019.02.20: Announcement of final portfolio.

authenticated encryption

- Other, more efficient constructions have been proposed and are an active area of research and standardization

https://competitions.cr.yp.to/caesar.html

Secure sessions

- Consider parties who wish to communicate securely over the course of a session
  - "*Securely*" = secrecy and integrity
  - "*Session*" = period of time over which the parties are willing to maintain state

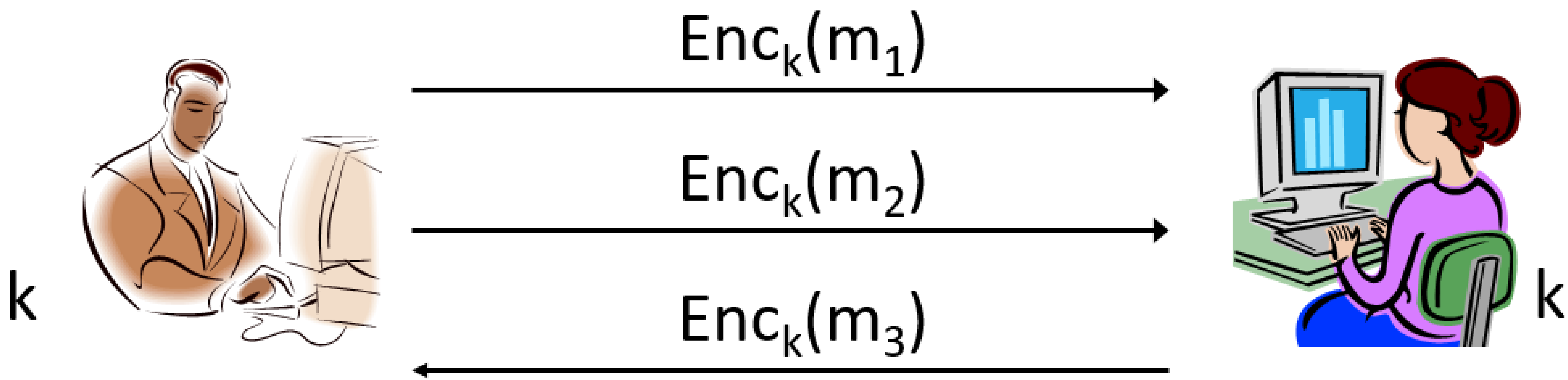
Secure sessions

- Consider parties who wish to communicate securely over the course of a session
  - "*Securely*" = secrecy and integrity
  - "*Session*" = period of time over which the parties are willing to maintain state
- Can use *authenticated encryption*

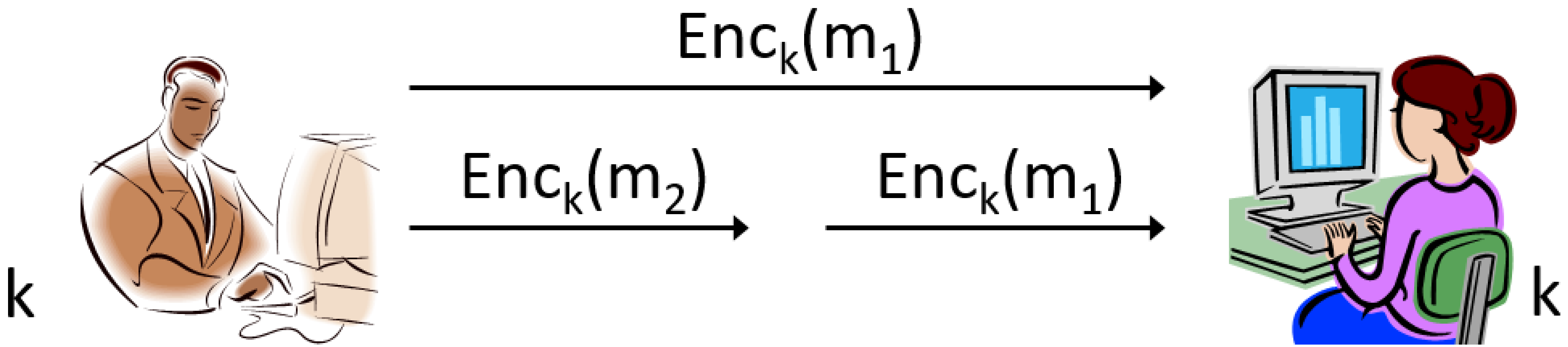
Secure sessions

- Consider parties who wish to communicate securely over the course of a session
  - "*Securely*" = secrecy and integrity
  - "*Session*" = period of time over which the parties are willing to maintain state

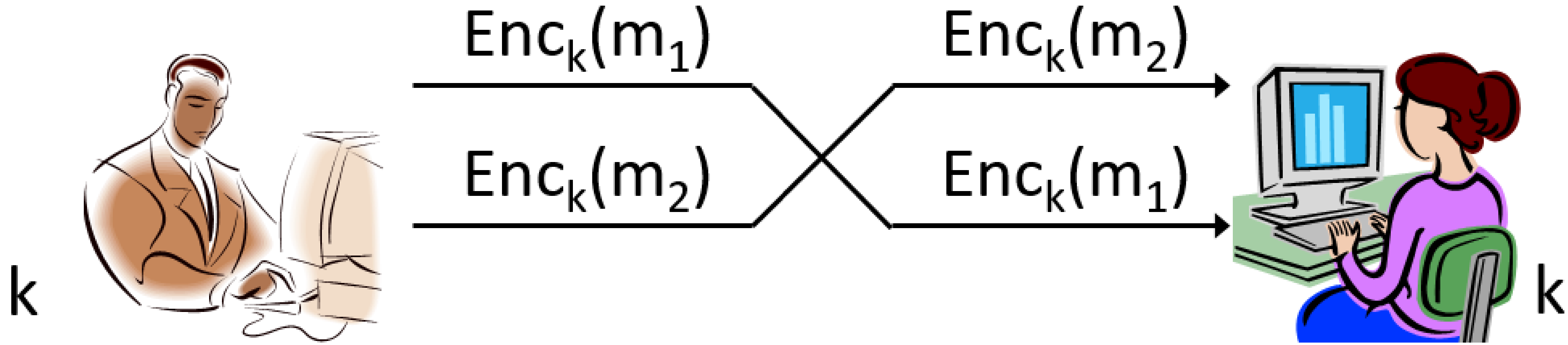
- Can use *authenticated encryption*



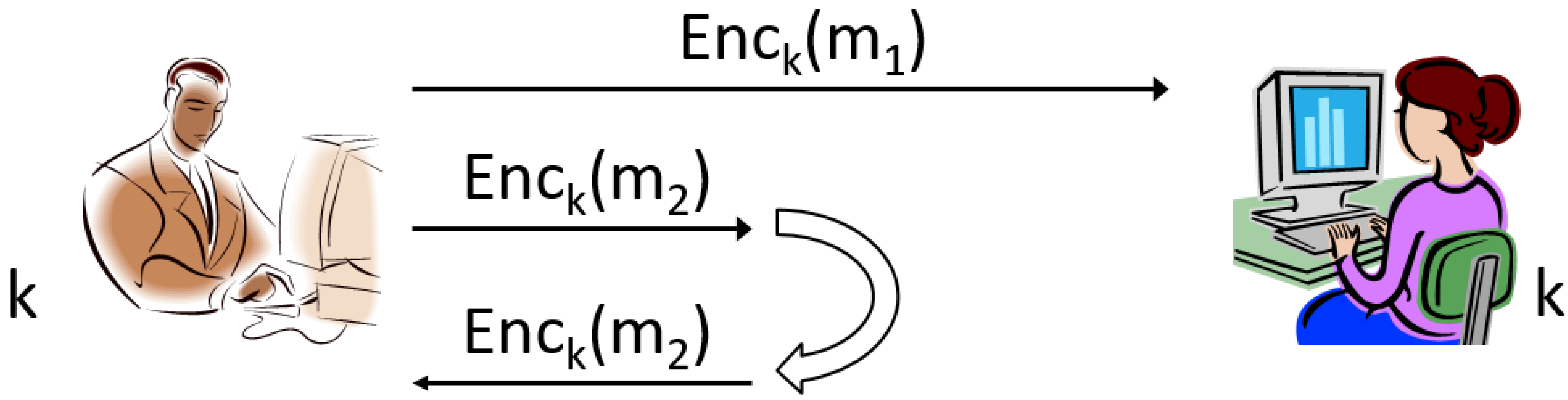
Replay attack



Re-ordering attack



Reflection attack

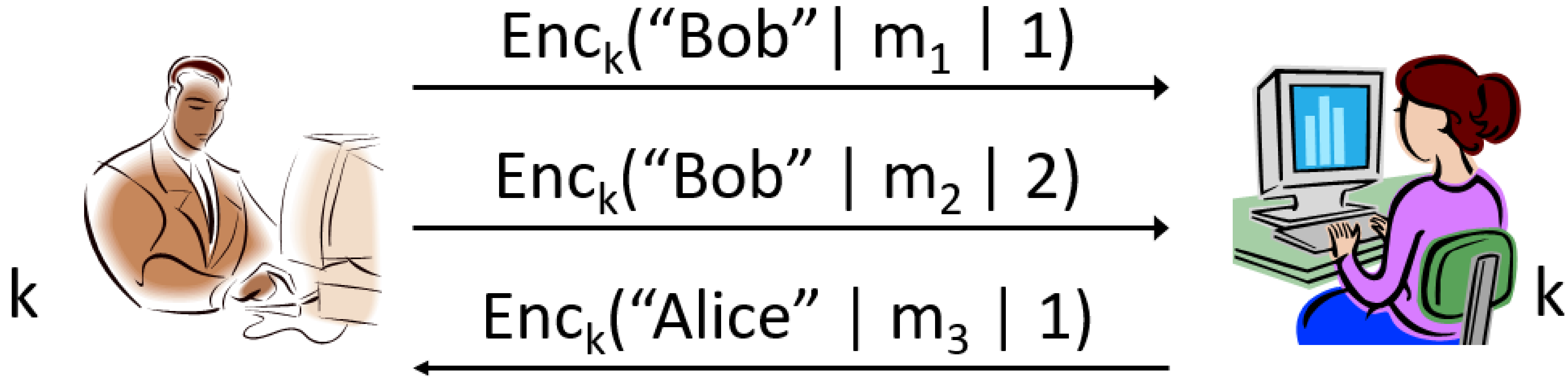


Secure sessions

- These attacks (and many others) can be prevented using *counters/sequence numbers* and *identifiers*

Secure sessions

- These attacks (and many others) can be prevented using *counters/sequence numbers* and *identifiers*



Secure sessions

- These attacks (and many others) can be prevented using *counters/sequence numbers* and *identifiers*
  - Can also use a *directionality bit* in place of identifiers

k

$\text{Enc}_k(\text{“Bob”} \mid m_1 \mid 1) \longrightarrow$

$\text{Enc}_k(\text{“Bob”} \mid m_2 \mid 2) \longrightarrow$

$\longleftarrow \text{Enc}_k(\text{“Alice”} \mid m_3 \mid 1)$

k

Hash functions

- (Cryptographic) *hash function*: deterministic function mapping arbitrary length inputs to a short, fixed-length output (sometimes called a *digest*)

Hash functions

- (Cryptographic) *hash function*: deterministic function mapping arbitrary length inputs to a short, fixed-length output (sometimes called a *digest*)

- Hash functions can be *keyed* or *unkeyed*
  – In practice, hash functions are unkeyed
  – We will assume unkeyed hash functions for simplicity

Collision-resistence

- Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a *hash function*
- A *collision* is a pair of distinct inputs x, x' such that $H(x) = H(x')$.

Collision-resistence

- Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a *hash function*
- A *collision* is a pair of distinct inputs x, x' such that $H(x) = H(x')$.
- **Theorem 7.1** H is *collision-resistant* if it is infeasible to find a collision in H.

Collision-resistence

- Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a *hash function*
- A *collision* is a pair of distinct inputs x, x' such that $H(x) = H(x')$.
- **Theorem 7.1** H is *collision-resistant* if it is infeasible to find a collision in H.
- What is the best "generic" collision attack on a hash function $H; \{0,1\}^* \to \{0,1\}^\ell$?
  – Note that collisions are guaranteed to exist!

Collision-resistence

- Let $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$ be a *hash function*
- A *collision* is a pair of distinct inputs x, x' such that $H(x) = H(x')$.
- **Theorem 7.1** H is *collision-resistant* if it is infeasible to find a collision in H.
- What is the best "generic" collision attack on a hash function $H; \{0,1\}^* \rightarrow \{0,1\}^\ell$?
  - Note that collisions are guaranteed to exist!
  - If we compute $H(x_1), \ldots, H(x_{2^\ell+1})$, we are guaranteed to find a collision.
  - Can we do better?

"Birthday" attacks

- Compute $H(x_1), \ldots, H(x_k)$
  - What is the probability of a collision?

"Birthday" attacks

- Compute $H(x_1), \ldots, H(x_k)$
  - What is the probability of a collision?

- Related to the so-called *birthday paradox*
  - How many people are needed to have a 50% chance that some two people share a birthday?

"Birthday" attacks

- Event A: at least two people in the room have the same birthday

  Event B: no two people in the room have the same birthday

  $\Pr[A] = 1 - \Pr[B]$

$$
\begin{aligned}
\Pr[B] &= \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdot \cdots \cdot \left(1 - \frac{n-1}{365}\right) \\
&= \prod_{i=1}^{n-1} \left(1 - \frac{i}{365}\right).
\end{aligned}
$$

  $\Pr[A] = 1 - \prod_{i=1}^{n-1} \left(1 - \frac{i}{365}\right)$

"Birthday" attacks

- Event A: at least two people in the room have the same birthday

  Event B: no two people in the room have the same birthday

  $\Pr[A] = 1 - \Pr[B]$

$$\begin{aligned} \Pr[B] &= \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdot \cdots \cdot \left(1 - \frac{n-1}{365}\right) \\ &= \prod_{i=1}^{n-1} \left(1 - \frac{i}{365}\right). \end{aligned}$$

  $\Pr[A] = 1 - \prod_{i=1}^{n-1} \left(1 - \frac{i}{365}\right)$

  $p(n; H) := 1 - \prod_{i=1}^{n-1}(1 - \frac{i}{H})$

“Birthday” attacks

- Since $e^x = 1 + x + \frac{x^2}{2!} + \cdots$, for $|x| \ll 1$, $e^x \approx 1 + x$

"Birthday" attacks

- Since $e^x = 1 + x + \frac{x^2}{2!} + \cdots$, for $|x| \ll 1$, $e^x \approx 1 + x$

  Thus, we have $e^{-i/H} \approx 1 - \frac{i}{H}$.

"Birthday" attacks

- Since $e^x = 1 + x + \frac{x^2}{2!} + \cdots$, for $|x| \ll 1$, $e^x \approx 1 + x$

  Thus, we have $e^{-i/H} \approx 1 - \frac{i}{H}$.

  Recall that $p(n; H) := 1 - \prod_{i=1}^{n-1}(1 - \frac{i}{H})$

  This probability can be approximated as

$$p(n; H) \approx 1 - e^{-n(n-1)/2H} \approx 1 - e^{-n^2/2H}.$$

"Birthday" attacks

- Since $e^x = 1 + x + \frac{x^2}{2!} + \cdots$, for $|x| \ll 1$, $e^x \approx 1 + x$

  Thus, we have $e^{-i/H} \approx 1 - \frac{i}{H}$.

  Recall that $p(n; H) := 1 - \prod_{i=1}^{n-1}(1 - \frac{i}{H})$

  This probability can be approximated as
  $$p(n; H) \approx 1 - e^{-n(n-1)/2H} \approx 1 - e^{-n^2/2H}.$$
  Let $n(p; H)$ be the smallest number of values we have to choose, such that the probability for finding a collision is at least p. By inverting the expression above, we have
  $$n(p; H) \approx \sqrt{2H \ln \frac{1}{1-p}}.$$

Next Lecture

- hash ...