Message integrity

Secrecy vs. Integrity

- How encryption can be used to prevent a passive eavesdropper from learning anything about messages sent over an open channel.
- However, not all security concerns are related to secrecy, and not all adversaries are limited to passive eavesdropping.
- In many cases, it is of equal or greater importance to guarantee message integrity (or message authentication) against an active adversary who can inject messages on the channel or modify messages in transit
- E.g. Bank transactions

Secrecy vs. integrity

- So far we have been concerned with ensuring *secrecy* of communication

- What about *integrity*?
  - I.e., ensuring that a received message originated from the intended party, and was not modified
    - Even if an attacker controls the channel!
  - Standard error-correction techniques not enough!
    - The right tool is a *message authentication code*

Secrecy vs. integrity

- Secrecy and integrity are *orthogonal* concerns
  - Possible to have either one without the other
  - Sometimes you might want one without the other
  - Most often, both are needed

- Encryption does not (in general) provide *any* integrity
  - Integrity is even stronger than non-malleability
  - None of the schemes we have seen so far provide any integrity

Encryption vs. Message Authentication

- The goals of secrecy and message integrity are different, so are the techniques and tools for achieving them
- Encryption does not (in general) provide any integrity, and encryption should not be assumed to ensure message authentication unless it is specifically designed with that purpose in mind

Message Authentication Codes (MACs)

- The aim of a message authentication code is
  - to prevent an adversary from modifying a message sent by one party to another, or
  - from injecting a new message, without the receiver detecting that the message did not originate from the intended party.

MAC

- Two users who wish to communicate in an authenticated manner begin by generating and **sharing a secret key k** in advance of their communication.
- When one party wants to send a message m to the other, she computes a **tag** t based on the message and the shared key, and sends the message **m along with t** to the other party.
- The tag is computed using a tag-generation algorithm Mac;
- The sender of a message m computes ***t ← Mack(m)*** and transmits **(m,t)** to the receiver.
- Upon receiving (m,t), the second party verifies **whether t is a valid tag on the message m** (with respect to the shared key) or not.

MAC

- A verification algorithm Vrfy that takes as input the shared key as well a message m and a tag t, and indicates whether the given tag is valid.

Message authentication code (MAC)

- A *message authentication code* is defined by three PPT algorithms (Gen, Mac, Vrfy):
  - Gen: takes as input 1^n; outputs k. (Assume $|k| \geq n$.)
  - Mac: takes as input key k and message $m \in \{0,1\}^*$; outputs tag t

    $$t := \mathrm{Mac}_k(m)$$

  - Vrfy: takes key k, message m, and tag t as input; outputs 1 ("accept") or 0 ("reject")

For all m and all k output by Gen,
$\mathrm{Vrfy}_k(m, \mathrm{Mac}_k(m)) = 1$

Fixed-length MAC

If there is a function ℓ such that for every k output by $\mathsf{Gen}(1^n)$, *algorithm* Mac_k *is only defined for messages $m \in \{0,1\}^{\ell(n)}$, then we call the scheme a* fixed-length MAC for messages of length $\ell(n)$.

Security of Message Authentication Codes

- No efficient adversary should be able to generate a valid tag on any "new" message that was not previously sent (and authenticated) by one of the communicating parties.
- Consider only probabilistic polynomial-time adversaries

Adversary

- An attacker “breaks” the scheme if it succeeds in outputting a forgery, i.e., if it outputs a message m along with a tag t such that

(1) t is a valid tag on the message m (i.e., Vrfyk(m,t) = 1), and

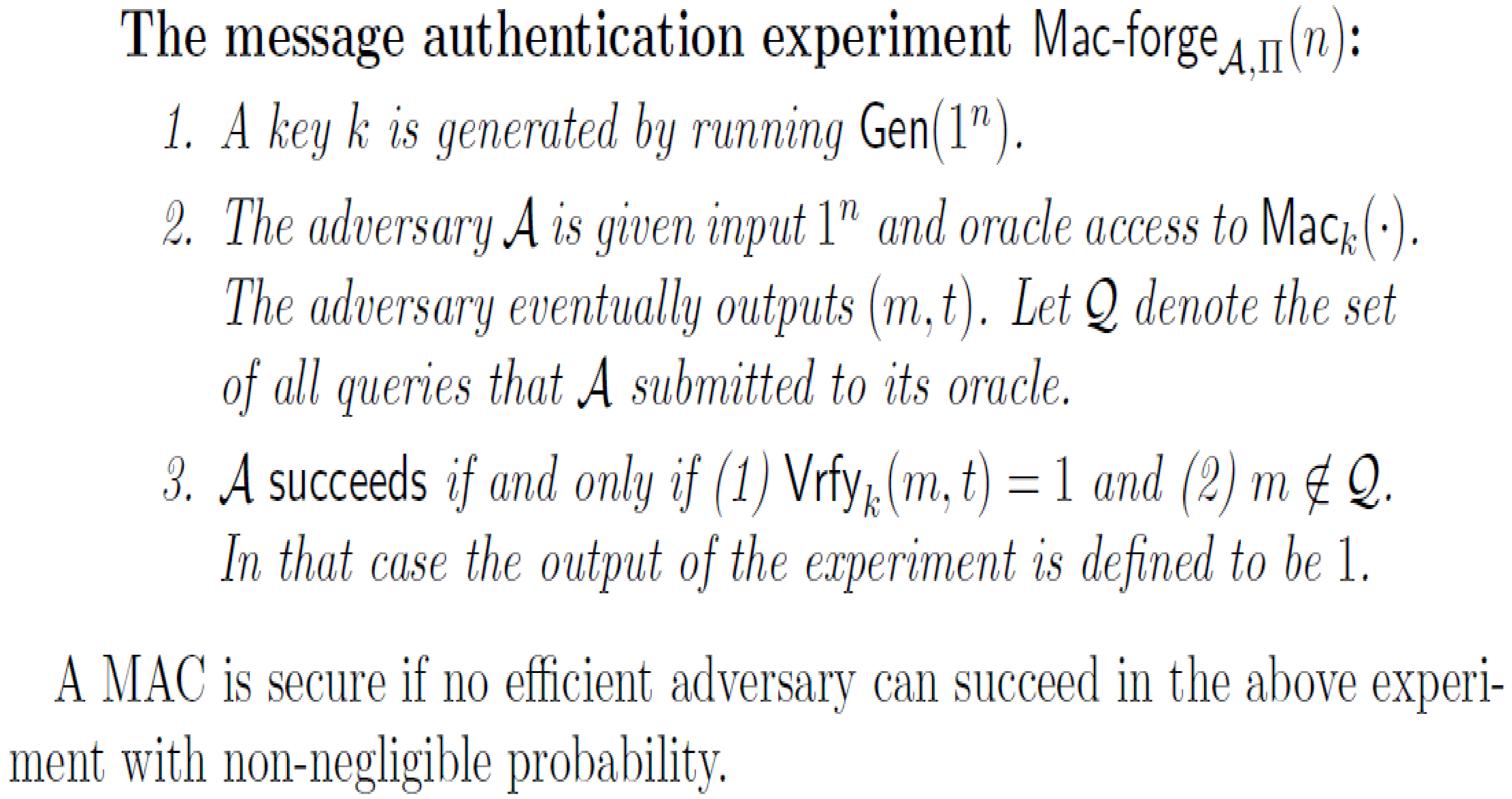
(2) the honest parties had not previously authenticated m (i.e., the adversary had not previously requested a tag on the message m from its oracle).

These conditions imply that if the adversary were to send (m,t) to one of the honest parties, then that party would be mistakenly fooled into thinking that m originated from the other legitimate party (since Vrfyk(m,t) = 1) even though it did not

Existentially unforgeable

- A MAC that cannot be broken in the above sense is said to be existentially unforgeable under an adaptive chosen-message attack. “Existentially unforgeable” refers to the fact that the adversary is unable to forge a valid tag on any message

Message authentication experiment Mac-forgeA,Π(n):



The message authentication experiment $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n)$:

1. *A key k is generated by running* $\mathsf{Gen}(1^n)$.
2. *The adversary $\mathcal{A}$ is given input 1^n and oracle access to* $\mathsf{Mac}_k(\cdot)$. *The adversary eventually outputs (m,t). Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ submitted to its oracle.*
3. $\mathcal{A}$ succeeds *if and only if (1)* $\mathsf{Vrfy}_k(m,t) = 1$ *and (2) $m \notin \mathcal{Q}$. In that case the output of the experiment is defined to be* 1.

A MAC is secure if no efficient adversary can succeed in the above experiment with non-negligible probability.

message authentication code

DEFINITION 4.2 A message authentication code Π = (Gen,Mac,Vrfy) *is* existentially unforgeable under an adaptive chosen-message attack*, or just* secure*, if for all probabilistic polynomial-time adversaries* A*, there is a negligible function* negl *such that:*

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

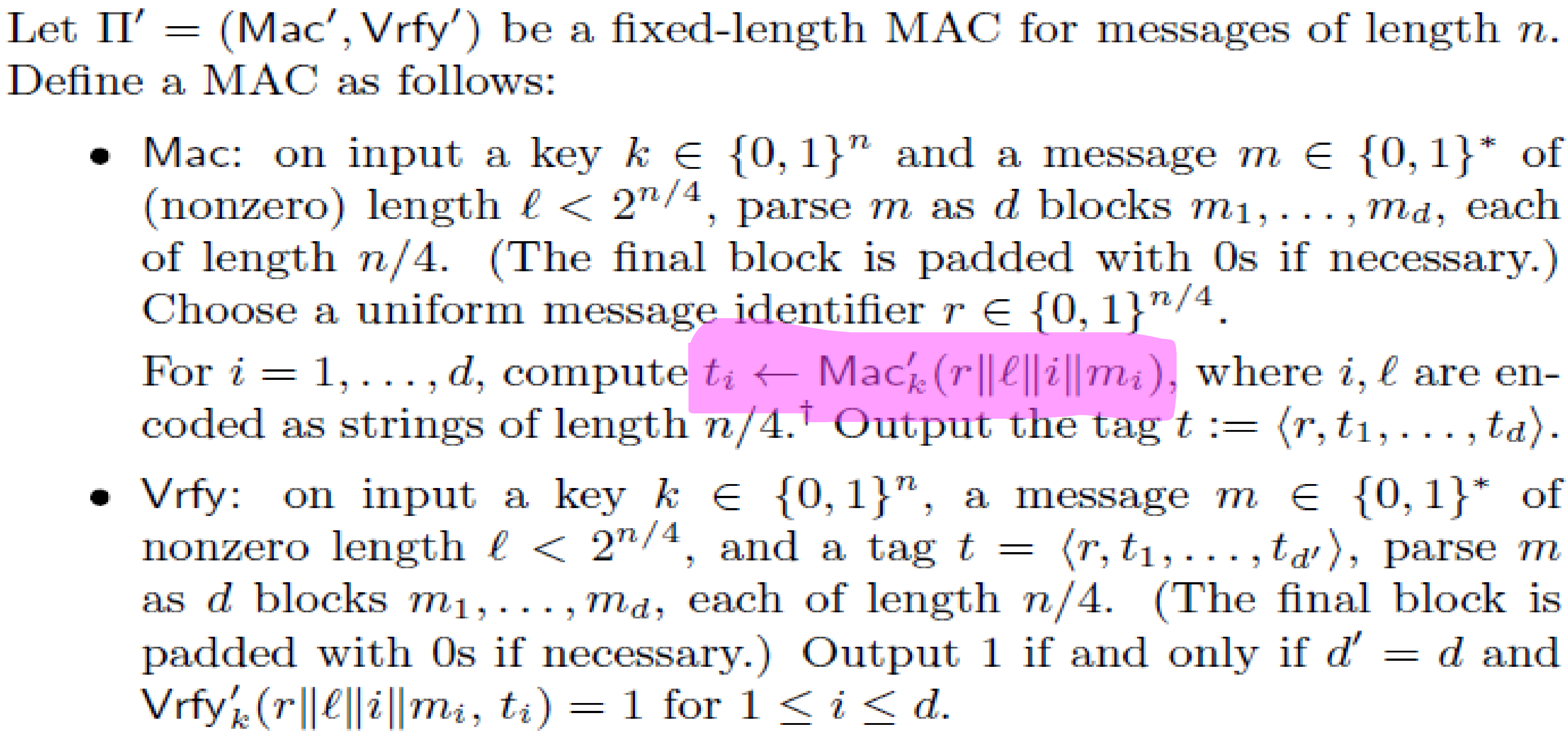
For arbitrary-length messages

- Parse the message m as a sequence of n-bit blocks $m_1,\ldots,m_d$ and authenticate each block separately - block re-ordering attack can happen
- Authenticating a block index along with each block. That is, we now compute $t_i = \mathsf{Mac}'_k(i \| m_i)$

- Truncation attack can happen
- Can be thwarted by additionally authenticating the message length along with each block.

$$t_i = \mathsf{Mac}'_k(\ell \| i \| m_i)$$

For arbitrary-length messages

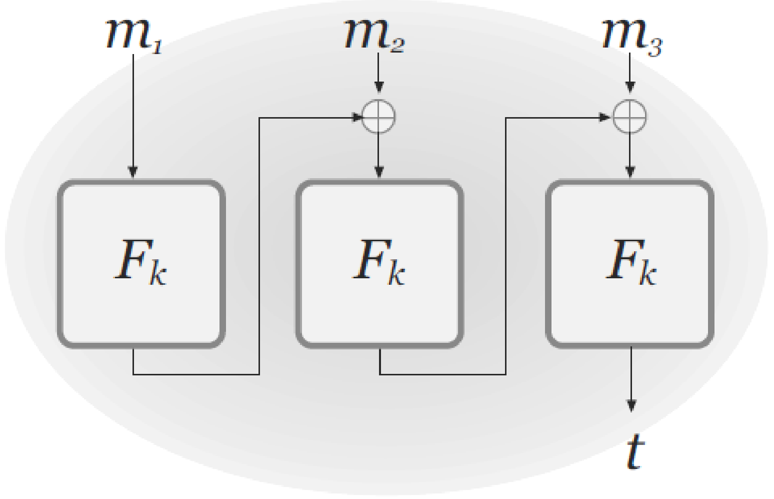


Let $\Pi' = (\mathsf{Mac}', \mathsf{Vrfy}')$ be a fixed-length MAC for messages of length n. Define a MAC as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$, parse m as d blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Choose a uniform message identifier $r \in \{0,1\}^{n/4}$.

  For $i = 1, \ldots, d$, compute $t_i \leftarrow \mathsf{Mac}'_k(r \| \ell \| i \| m_i)$, where i, ℓ are encoded as strings of length $n/4$.[†] Output the tag $t := \langle r, t_1, \ldots, t_d \rangle$.
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of nonzero length $\ell < 2^{n/4}$, and a tag $t = \langle r, t_1, \ldots, t_{d'} \rangle$, parse m as d blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Output 1 if and only if $d' = d$ and $\mathsf{Vrfy}'_k(r \| \ell \| i \| m_i, t_i) = 1$ for $1 \leq i \leq d$.

CBC-MAC



THEOREM 4.10 *Let ℓ be a polynomial. If F is a pseudorandom function, then Construction 4.9 is a secure MAC for messages of length $\ell(n) \cdot n$.*

Let F be a pseudorandom function, and fix a length function $\ell(n) > 0$. The basic CBC-MAC construction is as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message m of length $\ell(n) \cdot n$, do the following (set $\ell = \ell(n)$ in what follows):

  1. Parse m as $m = m_1, \ldots, m_\ell$ where each m_i is of length n.
  2. Set $t_0 := 0^n$. Then, for $i = 1$ to ℓ, set $t_i := F_k(t_{i-1} \oplus m_i)$.

  Output t_ℓ as the tag.

- Vrfy: on input a key $k \in \{0,1\}^n$, a message m, and a tag t, do: If m is not of length $\ell(n) \cdot n$ then output 0. Otherwise, output 1 if and only if $t \overset{?}{=} \mathsf{Mac}_k(m)$.

CBC-MAC vs. CBC-mode encryption

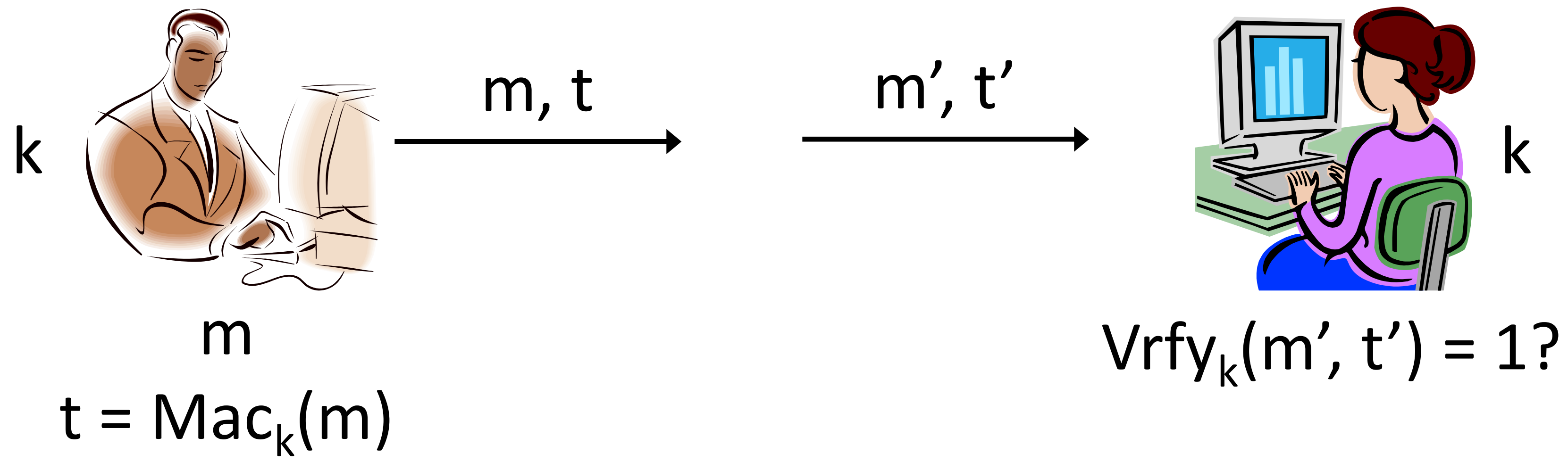
1. CBC-mode encryption uses a random IV and this is crucial for security. In contrast, CBC-MAC uses no IV
2. 2. In CBC-mode encryption all intermediate values ti are output by the encryption algorithm as part of the ciphertext,

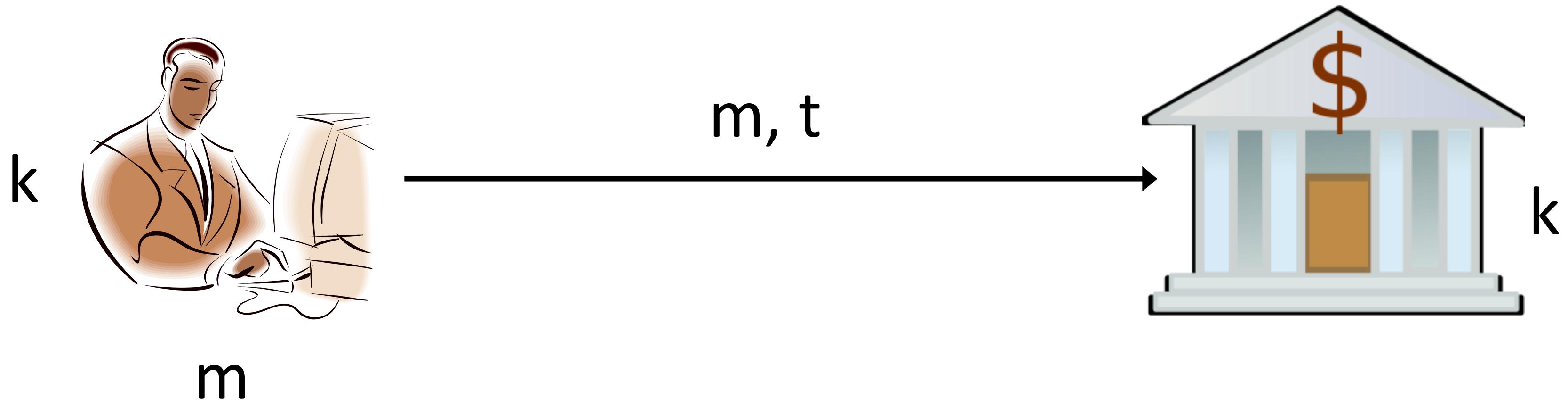
- whereas in CBC-MAC only the final block is output as the tag.
- If CBC-MAC is modified to output all the {ti} obtained during the course of the computation then it is no longer secure.

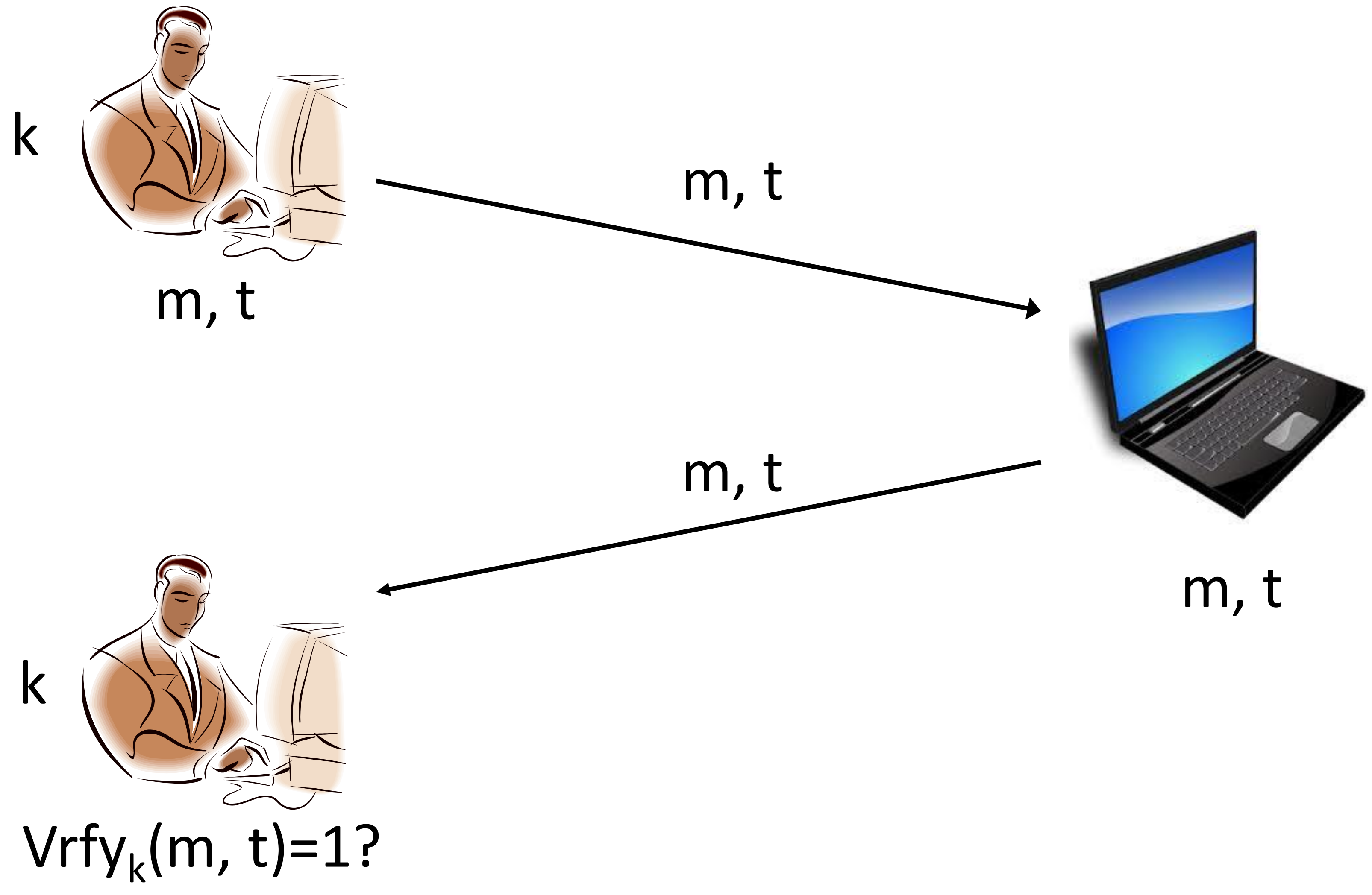
Secure CBC-MAC for arbitrary-length messages.

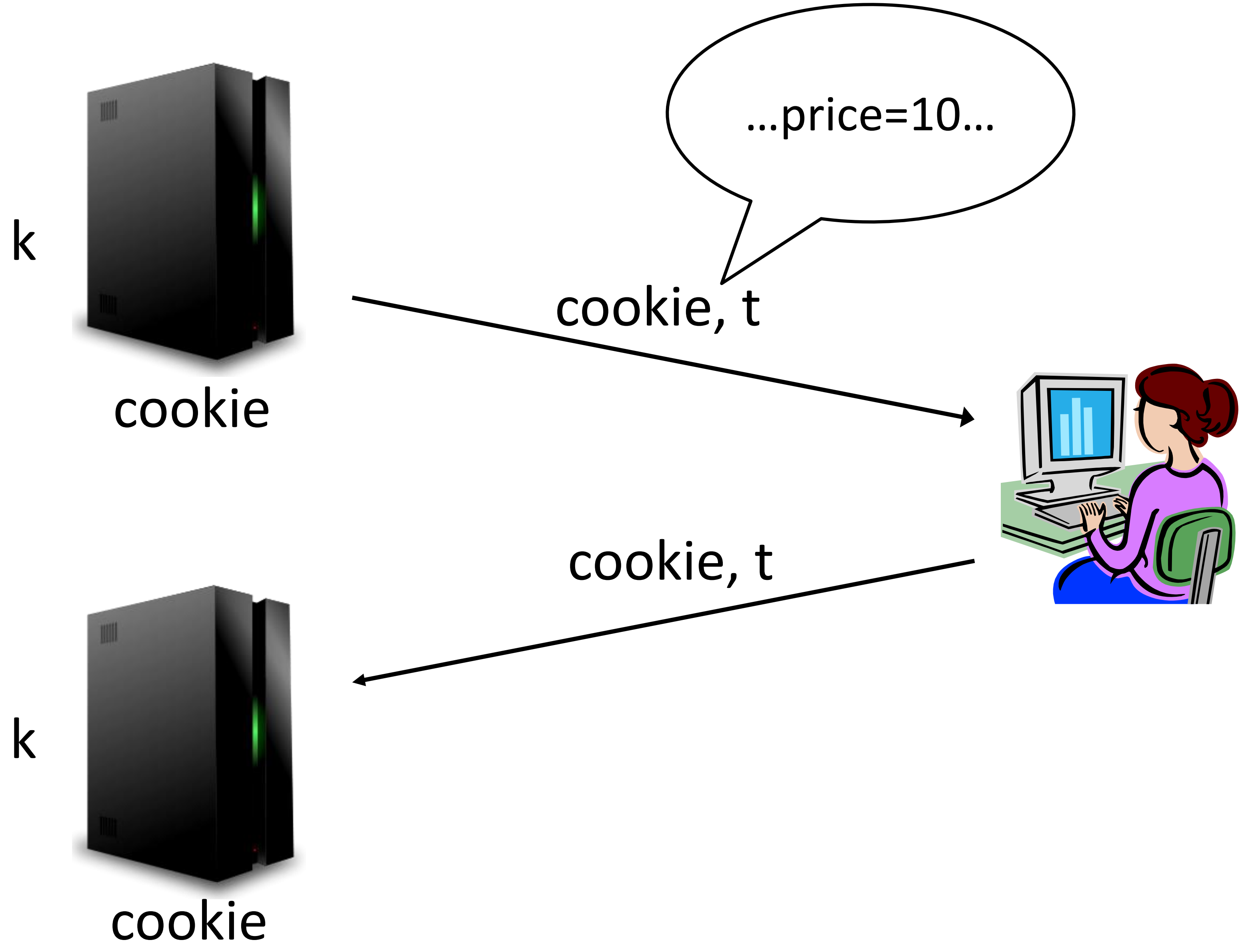
1. Prepend the message m with its length |m| (encoded as an n-bit string), and then compute basic CBC-MAC on the result
2. Key generation chooses two independent, uniform keys k1 ∈ $\{0,1\}^n$ and k2 ∈ $\{0,1\}^n$. Then to authenticate a message m, first compute the basic CBC-MAC of m using k1 and let t be the result; output the tag $\hat{t} := F_{k_2}(t)$.

$$\mathrm{CBC}_k(x_1, \ldots, x_\ell) \overset{\text{def}}{=} F_k\left(F_k\left(\cdots F_k(F_k(x_1) \oplus x_2) \oplus \cdots\right) \oplus x_\ell\right),$$


k
m, t
m′, t′
k
m
$t = \mathrm{Mac}_k(m)$
$\mathrm{Vrfy}_k(m', t') = 1?$

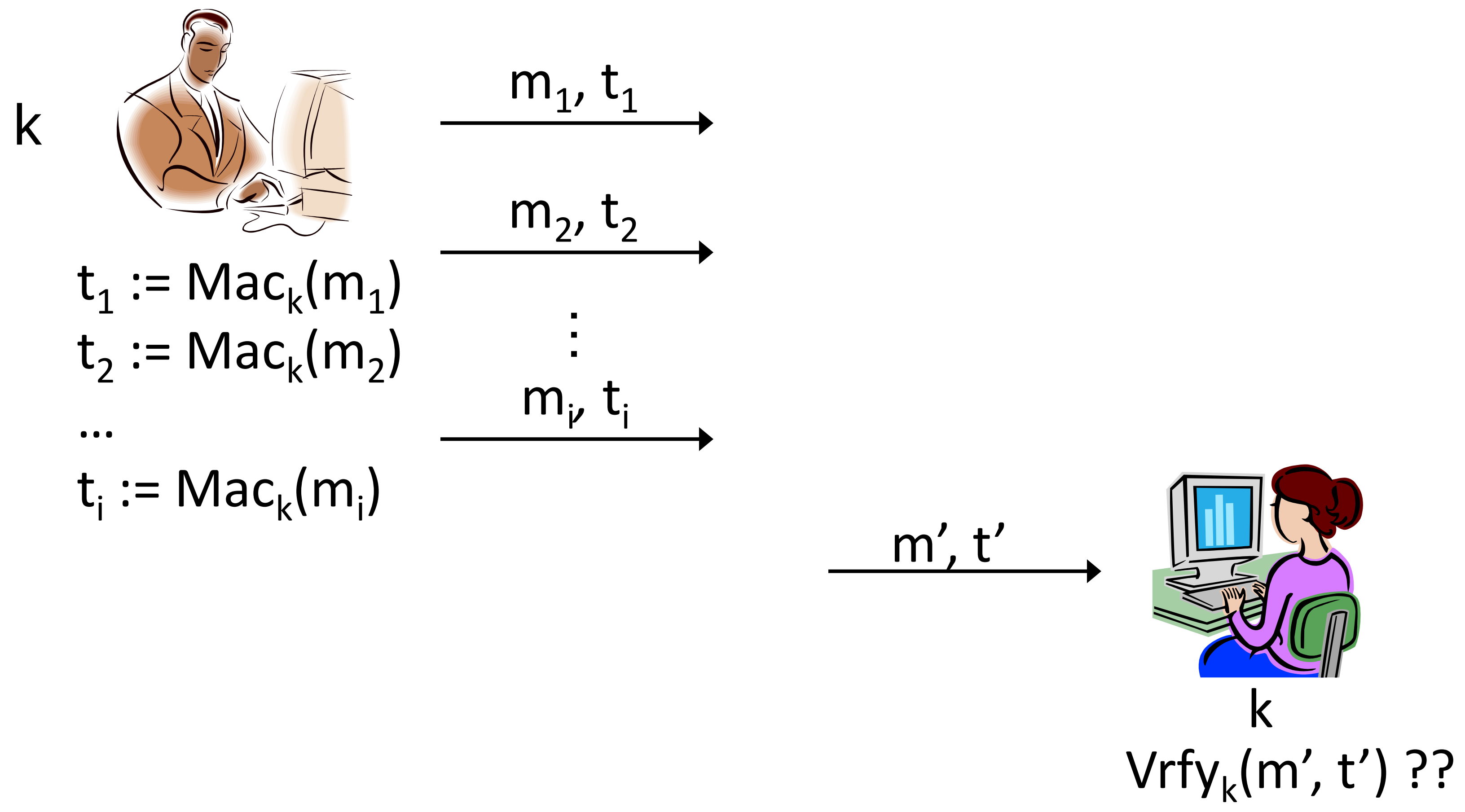

k
m
m, t
k


k
m, t
m, t
m, t
m, t
k
$\text{Vrfy}_k(m, t)=1$?


k
cookie
...price=10...
cookie, t
cookie, t
k
cookie

Security?

- Only one standard definition
- Threat model
  - "Adaptive chosen-message attack"
  - Assume the attacker can induce the sender to authenticate *messages of the attacker's choice*
- Security goal
  - "Existential unforgeability"
  - Attacker should be unable to forge a valid tag on *any* message not previously authenticated by the sender


k
$t_1 := \mathrm{Mac}_k(m_1)$
$t_2 := \mathrm{Mac}_k(m_2)$
…
$t_i := \mathrm{Mac}_k(m_i)$
m_1, t_1
m_2, t_2
⋮
m_i, t_i
m', t'
k
$\mathrm{Vrfy}_k(m', t')$??

Formal definition

- Fix A, Π
- Define randomized experiment $\text{Forge}_{A,\Pi}(n)$:
  1. $k \leftarrow \text{Gen}(1^n)$
  2. A interacts with an oracle $\text{Mac}_k(\cdot)$; 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 $\text{Vrfy}_k(m, t)=1$ and $m \notin M$

Security for MACs

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

$$\Pr[\mathrm{Forge}_{A,\Pi}(n) = 1] \leq \varepsilon(n)$$

Security?

- Is the definition too strong?
  - We don’t want to make any assumptions about what the sender might authenticate
  - We don’t want to make any assumptions about what forgeries are “meaningful”

- A MAC satisfying this definition can be used anywhere integrity is needed

Replay attacks

- Note that *replay attacks* are not prevented
  - No stateless mechanism can prevent them
- Replay attacks are often a significant real-world concern
- Need to protect against replay attacks at a higher level
  - Decision about what to do with a replayed message is application-dependent

Summary

Discussed about

- Secrecy vs. Integrity
- Encryption vs. Message Authentication
- MAC
- CBC MAC