CSE 3400/CSE 5850 - Introduction to Cryptography & Cybersecurity
/ Introduction to Cybersecurity)

Lecture 5
Message Authentication Codes

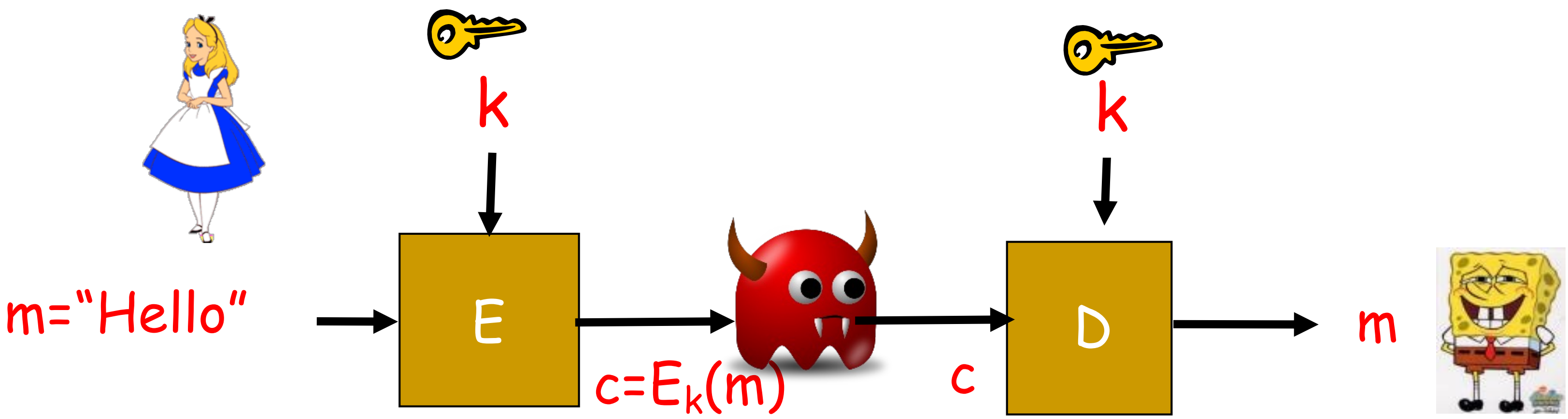
Ghada Almashaqbeh
UConn

Adapted from the textbook slides

Outline

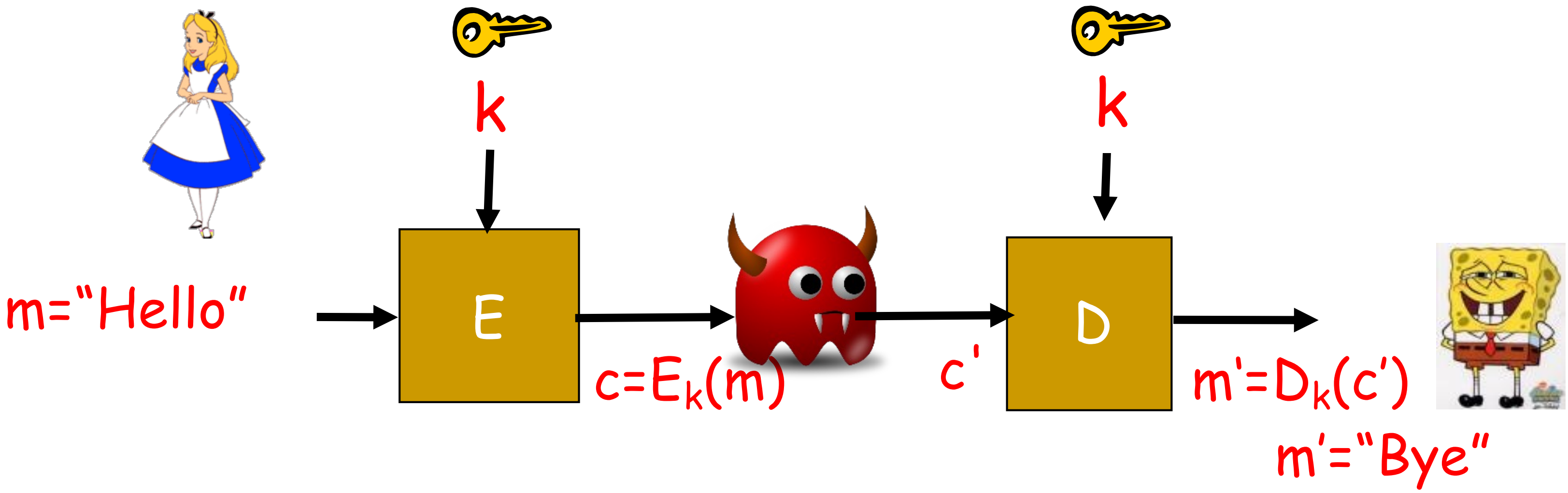
- Motivation.
- Message authentication codes (MACs) definition.
- MAC security definition.
- MAC constructions.
- Combining message authentication and encryption.

Encryption Ensures Confidentiality



- Man-in-the-Middle attacker 'learns nothing' about message

Integrity and Authentication?



- How can the recipient know that the message was not tampered with and it is the original one sent by the sender?

Does Encryption Prevent Forgery?

- Cannot be guaranteed.
  - Several secure encryption schemes are malleable (an attacker might be able to alter the ciphertext, and hence, the decrypted plaintext will be different).
- Clearly not for bitwise stream ciphers (& OTP).
  - Given $c=m\oplus k$, attacker can send $c\oplus mask$, to invert any bit in decrypted message.
- Example, send "Pay Bob $100" encrypted using OTP.
  - Eve can change it to "Pay Eve $100" (note that this is a KPA attacker). How?
    - Take the ciphertext of the letter "B" above, denote it as c[4].
    - Note that $c[4] = k[4] \oplus$ "B" (note that we do know the key!)
    - Compute a mask that does the following: $c[4] \oplus mask = k[4] \oplus$ "E" (this boils down to computing "B" $\oplus$ mask = "E")
    - Repeat that for the rest of the letters.

Message Authentication Codes (MACs)

- A MAC allows a recipient to ***validate*** that a message was ***not tampered*** with and that it was sent by a ***key holder***

It is a symmetric key setup!

Valid MAC → Only Sponge and I know *k*. So he sent *m*.

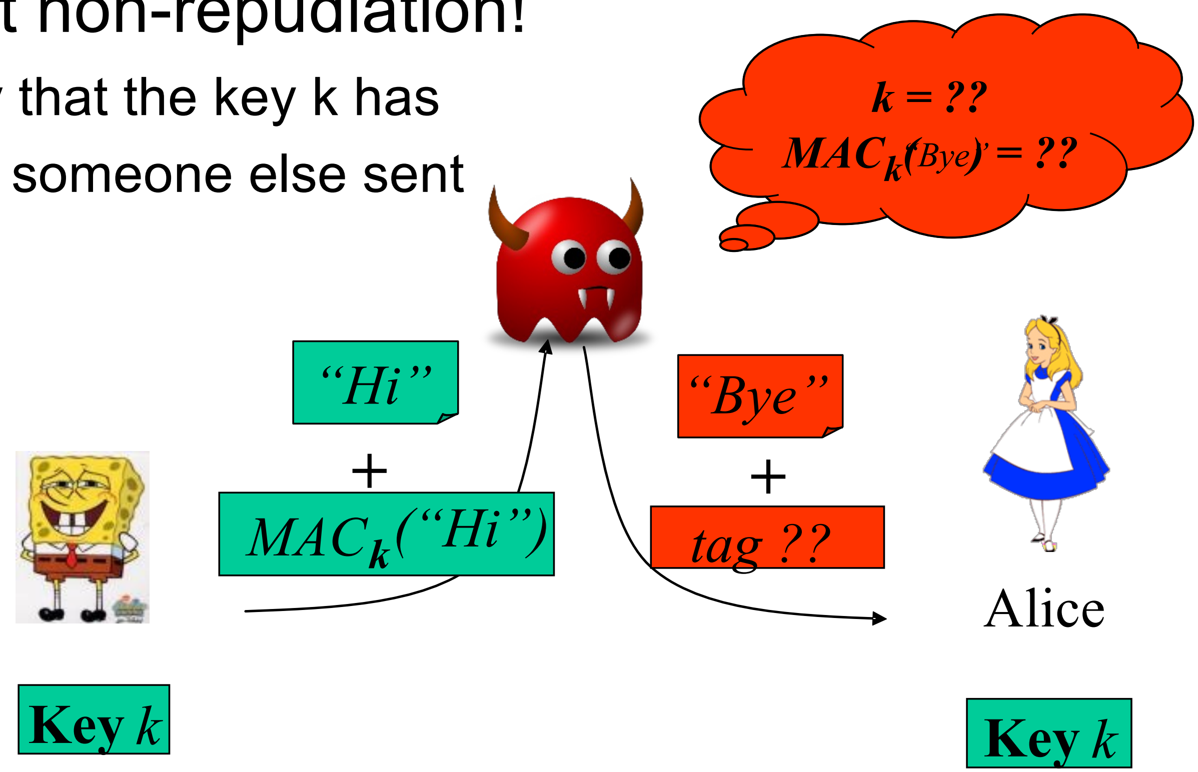
m="Hi", $MAC_k(m)$ →

Key *k*

Key *k*

Message Authentication Codes (MACs)

- Use shared key k to authenticate messages
- Pair *(tag , m)* is <u>valid</u> ***iff*** $tag=MAC_k(m)$
- Very efficient
- Does not support non-repudiation!
  - Sponge may say that the key k has been stolen, and so someone else sent the message.



Defining MAC Security

- Following the `conservative design principle':
- Consider most powerful attacker
  - Let attacker receive tag for any message it wants (so it has an oracle access to MAC_k).
- And `easiest' attacker-success criteria
  - Attacker wins if it can produce a valid tag for any message
    - Except for these that the attacker asked to authenticate

MAC Security Definition

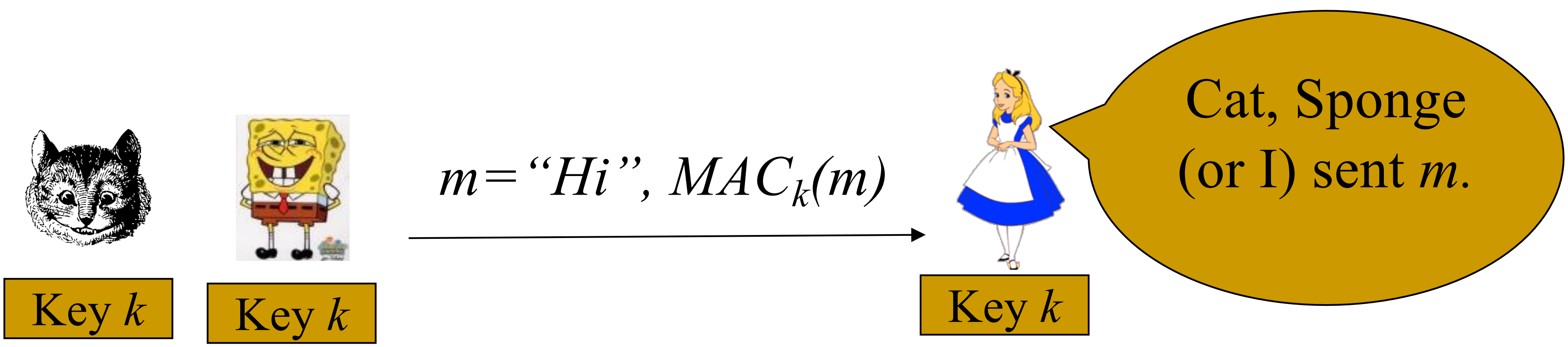
Definition 4.1 (MAC)**.** *For any givne integer l and domain D, a function $f^{MAC} : \{0,1\}^* \times D \rightarrow \{0,1\}^l$ is called a* (secure) MAC *if for all efficient algorithms $\mathcal{A}$, the* advantage function $\varepsilon^{MAC}_{f^{MAC},\mathcal{A}}(n)$ *is* negligible *in n ($\varepsilon^{MAC}_{f^{MAC},\mathcal{A}}(n) \in NEGL(n)$), i.e., smaller than any positive polynomial for sufficiently large n (as $n \rightarrow \infty$), where:*

$$\varepsilon^{MAC}_{f^{MAC},\mathcal{A}}(n) \equiv \Pr_{k \xleftarrow{\$} \{0,1\}^n} \left[(m, f^{MAC}_k(m)) \leftarrow \mathcal{A}^{f^{MAC}_k(\cdot|\text{except } m)}(1^n) \right] - \frac{1}{2^l} \quad (4.1)$$

The probability is taken over the uniformly-random choice of an n bit key, $k \xleftarrow{\$} \{0,1\}^n$, as well as over the coin tosses of $\mathcal{A}$.

On the Use of MACs

- $MAC_k(m)$ may expose information about m!
  - Example: Let *MAC* be any secure MAC. Define $MAC'_k(m)=MAC_k(m)||Lsb(m)$, where Lsb is least significant bit.
- MAC shows a key-holder computed it
  - Could be any key holder (even recipient)…
- Replay attacks: an old message (and its tag) is being resent.
  - Need to Ensure freshness (more about this later).



Constructing MAC: Three Approaches

1. Design `from scratch`, validate security by failure to cryptanalyze
   - Huge effort, risk → do only for few `building blocks`
   - Maybe from EDC (Error Detection Code), but it is not secure for every EDC.
2. Robust combiner of (two) MAC candidates:
   - $MAC_{k,k'}(m)=f_k(m)||f'_{k'}(m)$, $MAC_{k,k'}(m)=f_k(m)\oplus f'_{k'}(m)$ are secure MAC, if *either* f or f' is a secure MAC.
3. Provable-secure constructions from:
   - PRF/PRP/Block ciphers (next)
     - First: PRF/PRP → Fixed-Input-Length (FIL) MAC
   - Hash functions (later) – even more efficient.

Theorem: every PRF is also a MAC

Let F be a PRF from domain D to range $\{0,1\}^l$.
Then F is also an l-bit MAC for D.

- Proof sketch: construct an attacker against PRF using the attacker against the MAC.
  - For a random function, the outcome of any `new' value is random.
    - So, probability of guessing is 2^{-l}.
  - If a `new' outcome of a PRF can be guessed with significantly higher probability (which is the MAC over a new message), then we can distinguish between it and a random function! $\blacksquare$

Every PRF is also a MAC

- A PRF is a MAC for *l*-bit messages.
- *(l.n)*-bit FIL MAC from n-bit PRP (block cipher): use CBC-MAC – a variant of CBC
  - What standard crypto function can we use as a PRF?
  - A block cipher ? But …

Using a Block Cipher for MAC

- **Problem 1:** block cipher is PRP, not PRF
  - Solution: the switching lemma says that a PRP is also a PRF!

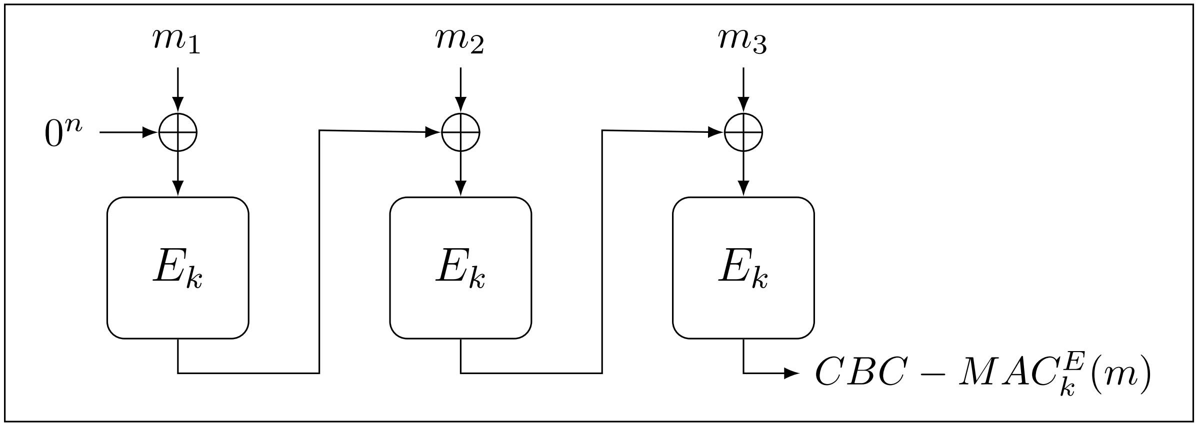
Using a Block Cipher for MAC

- **Problem 2:** block ciphers are defined only for (short) fixed input length (FIL)
  - Ideally a MAC should work for any input string (Variable Input Length – VIL)
  - We already had a similar problem… where?
    - Block ciphers.
  - We solved by using various encryption modes of operation.
  - A solution for MACs: the CBC-MAC mode of operation!

Cipher Block Chaining MAC: CBC-MAC

Split plaintext *m into* blocks

Fixed, known (zero) Initialization Vector (*IV*)



The tag is the cipher of the last block

$$CBC\text{-}MAC^E{}_k(m_1||m_2||..||m_l) = E_k(m_l \oplus E_k(\ldots E_k(m_1))))$$

Recall: MACs are deterministic functions

CBC-MAC

- Widely deployed standard
- More efficient 'modes' exist
  - E.g., allow for parallel computation.
- It is also provably secure.

Theorem [BKR94]: if E is a FIL-PRF for domain $\{0,1\}^n$, then $CBC\text{-}MAC^E$ is a PRF for domain $\{0,1\}^{ln}$ (for $l>1$).

- Corollary: ... then $CBC\text{-}MAC^E$ is a $\{0,1\}^{ln}$-MAC

But what of VIL (variable-length input) MAC?

CBC-MAC-based VIL-MAC

- Is CBC-MAC^E a VIL-MAC?
  - ***No!***
    - Ask for $b=CBC\text{-}MAC^E{}_k(a)=E_k(a)$;
    - then output (ac, b) so $m = ac$ with $tag = b$ where $c= a\oplus b$.
    - This is valid, since the attacker did not ask the oracle for a tag for *ac* and *b* for *ac* is a valid tag since $CBC\text{-}MAC^E{}_k(ac)=E_k(c\oplus E_k(a))=E_k(c\oplus b)= E_k(a\oplus b\oplus b)= E_k(a)= b$.
- Solution: prepend message length (called CMAC)
  - Let $CMAC^E{}_k(m)=CBC\text{-}MAC^E{}_k(L(m)||m)$
    - Where $L(m)$ is a 1-block encoding of $|m|$
  - CMAC is a secure VIL MAC construction!

Examples of MAC Constructions

- Are the following constructions a secure MAC:

1. Let E_k be a block cipher that takes input of length n bits. For a message m of length 2n bits, compute the tag as:

   $MAC_k(m) = E_k(m_L) \text{ xor } E_k(m_R)$

2. Let G be a secure PRG. For a message m of length n bits, compute the tag as:

   $MAC_k(m) = k \text{ xor } PRG(m)$

Combining Authentication and Encryption

- For confidentiality, use encryption
- For authentication, use MAC
- For both confidentiality and authentication?
  - Option 1: Combine MAC and encryption
    - Possible pitfalls (vulnerabilities)
  - Option 2: authenticated-encryption schemes (or modes)
    - Easier to deploy (securely)
    - Generic combination of MAC and Encryption schemes
    - Or direct combined constructions (can be more efficient)
      - Might be ad-hoc or rely on complex or less-tested security assumptions.

Generic MAC and Encryption Combinations

- Three standards, three ways…
  - Authenticate and encrypt (A&E):
    - $c = Enc(m)$, $tag = MAC(m)$, send (c, tag)
  - Authenticate then encrypt (AtE):
    - $tag = MAC(m)$, $c = Enc(m, tag)$, send c
  - Encrypt then authenticate (EtA):
    - $c = Enc(m)$, $tag = MAC(c)$, send (c, tag)
- Some of these may be vulnerable even when combining some <u>secure</u> encryption and MAC schemes!

Security of Generic MAC/Enc Combinations

- A&E may be vulnerable!
  - Example:
    - Let MAC be any secure MAC scheme
    - Let $MAC'_{k'}(m)=MAC_{k'}(m) || lsb(m)$
    - MAC' is a secure MAC.
    - But A&E(m) leaks least significant bit of m (even if the encryption scheme is secure!!!).
  - Recall that the security guarantee of a MAC is about integrity (or preventing forgery)!
    - It has nothing to do with confidentiality!
- What about AtE, EtA ?
  - AtE: also may be vulnerable (not IND-CPA)!

Security of Generic MAC/Enc Combinations

- How about EtA ? **Provably CCA-Secure** [CK01]!
  - ➔ Secure encryption; otherwise attack Enc(m) by appending MAC
  - ➔ Secure authentication, since any change in (c, MAC(c)) is detected
  - Also: reject fake messages w/o decryption ➔ efficiency and foil Denial of Service (DoS), CCA attacks
  - Note: using separate keys for Enc and MAC; what if we use same key?

Keys for MAC and Encryption?

Using same key for MAC+Encryption? Insecure

- Exercise: show (contrived) examples vulnerabilities:
  - A&E: both vulnerable…

$$E_{k',k''}(m) = E'_{k'}(m)||k''$$
$$MAC_{k',k''}(m) = MAC_{k''}(m)\ ||k'$$

  - (you can show other contrived examples for the other combinations.)
- So: should we use two independent keys?
  - Overhead: key generation, transmission, storage
- Secure enc+MAC – using a single key? Use PRFS

Solution: k_{mac}:= PRF_k(\`MAC'), k_{enc}:= PRF_k('Encrypt')

Covered Material From the Textbook

- Chapter 4
  - Sections 4.1 – 4.4 except sections 4.4.1, 4.4.5.
  - Section 4.5.3 (only what we covered in class) except sections 4.5.3.4, 4.5.4, 4.5.5

Thank You!