Applied Cryptography and Computer Security
CSE 664 Spring 2020

Lecture 8: Data Integrity

Department of Computer Science and Engineering
University at Buffalo

Overview

- Going back to the security objectives cryptography helps to achieve:
  - confidentiality
  - integrity
  - authentication
    - entity authentication
    - data authentication
  - access control
  - non-repudiability
- We'll discuss the integrity objective next (in the symmetric key setting)

Data Integrity

- Encryption does not protect data from modification by another party
  - recall the modes of encryption we talked about
- We normally want to ensure that the data arrives in its original form
  - i.e., we want data integrity
- How can we do that?
  - attach a verification tag?
  - how can we make sure that an adversary cannot compute the tag for messages of its choice?

Data Integrity

- This means that we also want to ensure that data comes from an authenticated source
  - i.e., we want data origin authentication
- We'll use message authentication codes (MAC)
  - a secret key is shared by two communicating parties
  - a MAC cannot be computed (or verified) without the key
- To achieve source authentication and message integrity:
  - the sender computes $t = \mathsf{MAC}_k(m)$ and sends (m, t)
  - the receiver recomputes $t' = \mathsf{MAC}_k(m)$ for received m and compares it to t

Message Authentication Codes

- Formally, a message authentication code is composed of PPT algorithms $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ s.t.:

  1. key generation algorithm Gen, on input a security parameter 1^n, outputs a key k, where $|k| \geq n$.
  2. tag generation algorithm Mac, on input a key k and message $m \in \{0, 1\}^*$, outputs a tag t, i.e., $t \leftarrow \mathsf{Mac}_k(m)$
  3. verification algorithm Vrfy, on input a key k, a message m, and a tag t, outputs a bit b, where $b = 1$ means the tag is valid and $b = 0$ means it is invalid, i.e., $b := \mathsf{Vrfy}_k(m, t)$

MAC

- What properties do we want?
  - correctness
    - ?
  - security
    - someone without the key shouldn't be able to forge a MAC on a message
    - given pairs $(m_i, \mathsf{Mac}_k(m_i))$, computing a new pair $(m, \mathsf{Mac}_k(m))$ such that $m \neq m_i$ should be hard

MAC

- Classification of attacks on MACs:
  - known-text attack: one or more pairs $(m_i, \mathsf{Mac}_k(m_i))$ are available
  - chosen-text attack: one of more pairs $(m_i, \mathsf{Mac}_k(m_i))$ are available for m_i's chosen by the adversary
  - adaptive chosen-text attack: the m_i's are chosen by the adversary, where successive choices can be based on the results of prior queries
- Which one do we want?

MAC

- Classification of forgeries:
  - selective forgery: an adversary is able to produce a new MAC pair for a message under her control
  - existential forgery: an adversary is able to produce a new MAC pair but with no control of the value of the message
- Which would we prefer??
- And, as usual, key recovery is the most damaging attack on MAC

MAC Security

- We construct an experiment for MACs existentially unforgeable under an adaptive chosen-message attack
- Let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be a message authentication code
- Message authentication experiment $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n)$:
  1. generate $k \leftarrow \mathsf{Gen}(1^n)$
  2. adversary $\mathcal{A}$ is given 1^n and oracle access to $\mathsf{Mac}_k(\cdot)$; let Q denote the set of queries $\mathcal{A}$ makes to the oracle
  3. $\mathcal{A}$ eventually outputs a pair (m, t)
  4. output 1 ($\mathcal{A}$ wins) iff (a) $\mathsf{Vrfy}_k(m, t) = 1$ and (b) $m \notin Q$

MAC Security

- Definition: A message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is secure if any PPT adversary $\mathcal{A}$ has at most negligible probability of suceeding in the above experiment, i.e.,

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

- Important: MACs do not prevent all traffic injections (e.g., replay attacks)
  - a replayed message will pass verification process
  - addressing this problem by MACs only cannot be done and is left to the application
    - use sequence numbers or time-stamps to make each message unique

Constructing Message Authentication Codes

- We can use pseudo-random functions for constructing fixed-length MACs
  - let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudo-random function
- MAC construction (for security parameter n):
  - Gen: on input 1^n, choose $k \overset{R}{\leftarrow} \{0,1\}^n$
  - Mac: on input key $k \in \{0,1\}^n$ and message $m \in \{0,1\}^n$, output tag $t := F_k(m)$
  - Vrfy: on input key $k \in \{0,1\}^n$, message $m \in \{0,1\}^n$, and tag $t \in \{0,1\}^n$, output 1 if and only if $t = F_k(m)$; and output 0 otherwise

Constructing Message Authentication Codes

- Security of our MAC construction:
  - Theorem: assuming that F is a pseudo-random function, the above fixed-length MAC construction is secure (existentially unforgeable under an adaptive chosen-message attack)
  - Proof intuition
    - as before, first substitute the pseudo-random object with a truly random
    - what is the probability that the output of random function can be predicted on a "new point"?
    - what is the "difference" between pseudo-random and random functions?

Variable-Length MACs

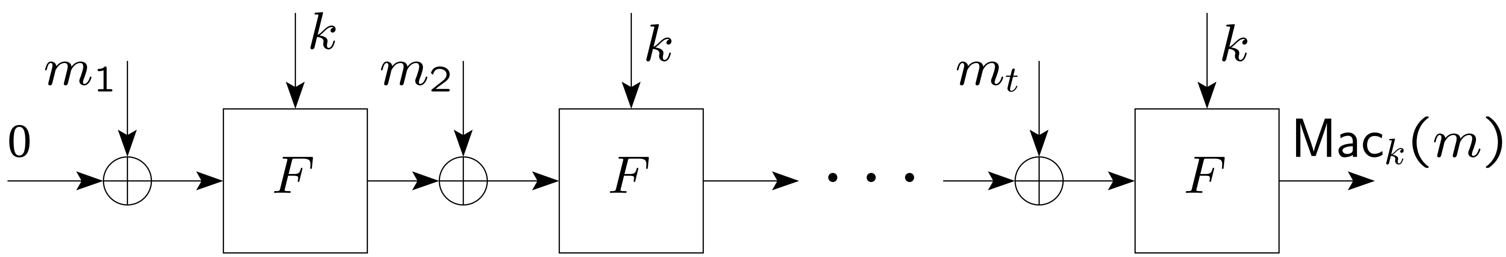
- Now how do we authenticate messages longer than n bits?
  - can partition a message into n-bit blocks
  - authenticate each block separately?
  - combine all messages into a single block?
- It is possible to construct secure MACs using only pseudo-random functions
  - must sequentially tie all blocks together
  - must ensure that tag forging based on message length is not possible

Variable-Length MACs

- MAC algorithms widely used in practice use chaining:
  - CBC-MAC (based on a block cipher)
  - HMAC (based on a hash function)
- They produce only one n-bit tag for messages of any length
  - specifically were designed to be efficient

MAC Algorithms

- CBC-MAC
  - DES in the cipher block chaining (CBC) mode has been a widely used MAC algorithm (FIPS 113 and ANSI standard X9.17)
  - uses the initialization vector 0
  - last block is used as the MAC



CBC-MAC

- Security of CBC-MAC
  - random IV is not used, it is set to constant 0^n
  - CBC-MAC is secure for messages of a fixed number of t blocks
- Compare this with CBC mode of encryption
  - random IV was necessary in encryption to prevent a codebook attack
  - random IV in a MAC construction gives room to tampering
  - all ciphertext blocks are necessary for decryption
  - using all ciphertext blocks as a MAC tag results in an insecure construction

CBC-MAC

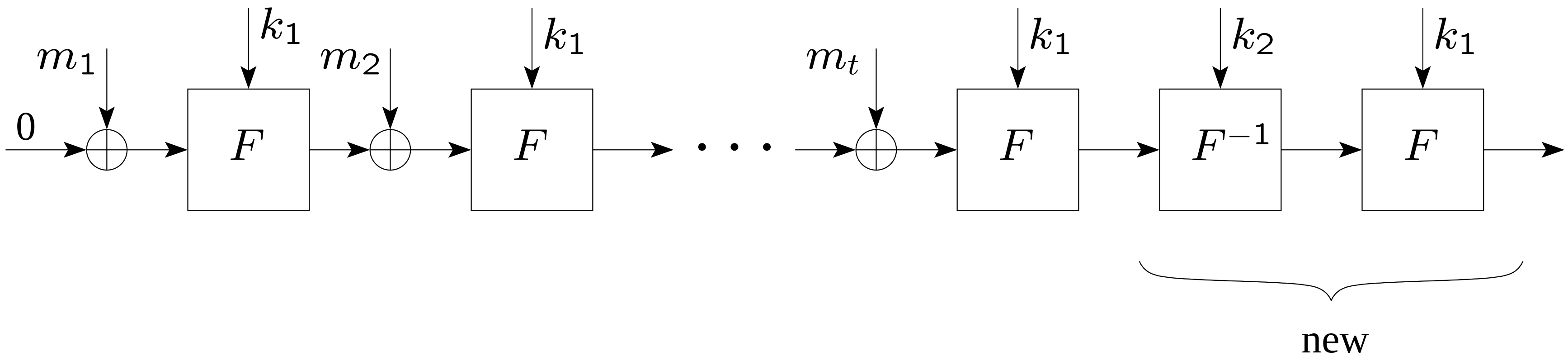
- If the number of blocks can vary, (adaptive chosen-text) existential forgery is possible
  - assume the adversary obtains a message-MAC pair (m_1, t_1)
  - the adversary queries a MAC for $m_2 = t_1$ and obtains (m_2, t_2)
  - then $t_2 = F_k(F_k(m_1))$ and is the MAC for the 2-block message $(m_1||0)$

CBC-MAC

- Another example of forgery in CBC-MAC
  - assume we have two pairs (m_1, t_1) and (m_2, t_2) for one-block messages m_1 and m_2
  - we request the MAC on a 2-block third message $m_3 = (m_1||z)$ and obtain $((m_1||z), t_3)$
  - then $t_1 = F_k(m_1)$, $t_2 = F_k(m_2)$, and $t_3 = F_k(t_1 \oplus z)$
  - we are able to construct the MAC for the new 2-block message $m_4 = m_2||(t_1 \oplus z \oplus t_2)$; it is also t_3
- The fix: do MAC strengthening

CBC-MAC

- One possibility of CBC-MAC strengthening:



  - this prevents the forgery without impacting the intermediate stages
  - (it also reduces the threat of exhaustive key search)
  - we can derive k_1 and k_2 from k as $k_1 = F_k(1)$ and $k_2 = F_k(2)$

CBC-MAC

- Other solutions are possible as well:

  1. prepend the input with a length block before the MAC computation
     - it is important that this block is not at the end
  2. create a length-dependent key from k
     - if ℓ is the number of blocks, first compute a new key as $k_\ell = F_k(\ell)$
     - use k_ℓ to produce the authentication tag
  3. $\ldots$

MAC Algorithms

- The next construction is HMAC
  - requires knowledge of hash functions
  - we'll look at cryptographic hash functions next
- To summarize what we've learned so far:
  - integrity is a separate security goal that requires tools designed for it
  - integrity or message authentication can be achieved using pseudo-random functions
  - CBC-MAC and HMAC are used in practice
- The key used for integrity protection must differ from the key used for confidentiality protection