CSE 410/565 Computer Security
Spring 2022

Lecture 4: Data Integrity and Hash Functions

Department of Computer Science and Engineering
University at Buffalo

Outline

- So far we discussed encryption as means to data confidentiality protection
- Next, we will talk about data integrity protection
  - this covers message authentication codes
  - we also discuss hash functions as a tool for integrity protection and other applications
- Everything we are discussing so far assumes a computationally limited adversary
  - doesn't have unlimited resources, can't search through the key space, etc.

Data Integrity

- Encryption protects data only from a passive attack
  - we often also want to protect message from active attacks (modification or falsification of data)
  - such protection is called message or data authentication
- Goals of message authentication
  - a message is authentic if it came from its alleged source in its genuine form
  - message authentication allows verification of message authenticity

Message Authentication

- How can message authentication be performed?
  - in addition to the message itself, another token that authenticates the message, often called a tag, is transmitted
  - the cryptographic primitive is called a Message Authentication Code (MAC)
- Message authentication is independent of encryption
  - it can be used with or without encryption
  - a number of applications benefit from message authentication alone

Message Authentication

- What do we want from a message authentication code?
  - a tag should be easy to compute by legitimate parties, but hard to forge by an adversary
- MAC constructions use a secret key
  - 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

- A MAC scheme is defined by three algorithms:
  - key generation: a randomized algorithm, which on input a security parameter n, produces key a k
  - MAC generation: a possibly randomized algorithm, which on input a message m and key k, produces a tag t
  - MAC verification: a deterministic algorithm, which on input a message m, tag t, and key k, outputs a bit b

Message Authentication Codes

- Properties of MAC algorithms
  - most fundamentally, we desire correctness and security
    - correctness requires that a correctly computed tag will always verify
    - security will be defined later and intuitively requires that it is hard to forge a tag on a new message without the key
  - from a performance point of view, we desire (and can achieve) tags of a fixed size (i.e., independent of the message length)

Message Authentication Codes

- 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
- Against which kind of attack do we want to be resilient?

Message Authentication Codes

- 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
- Resilience against which type would be preferred?
- And, as usual, key recovery is the most damaging attack on MAC

Message Authentication Codes

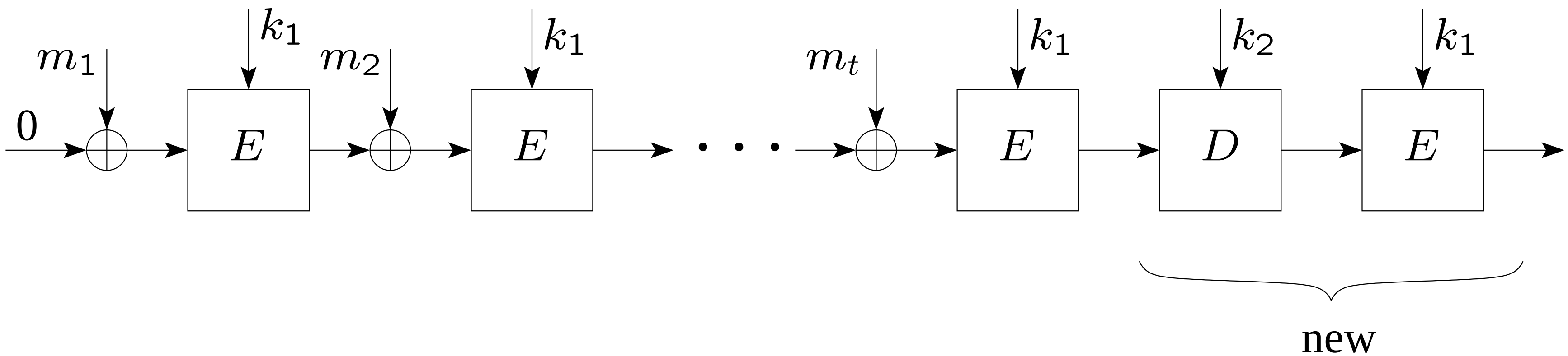
- We desire for a MAC to be existentially unforgeable under an adaptive chosen-message attack
  - an adversary is allowed to query tags on messages of its choice
  - at some point it outputs a pair (m, t)
  - the forgery is considered successful if m hasn't been queried before and t is a valid tag for it
  - as with encryption, security guarantees depend on the security parameter
- MACs do not prevent all traffic injections
  - a replayed message will pass verification process
  - it is left to the application to make each message unique

Message Authentication Codes

- There are two most common (standardized) implementations of MAC functions
  - CBC-MAC: based on a symmetric encryption (e.g., AES) in Cipher Block Chaining (CBC) mode with some modifications
    - varying IV is not permitted
    - only a single block is produced
    - additional security measures are in place to support variable-length messages
  - HMAC: based on a hash function
- We'll discuss the latter and need to look at hash functions first

Message Authentication Codes

- A CBC-MAC variant secure in the presence of variable-length messages:



Hash Functions

- A hash function h is an efficiently-computable function that maps an input x of an arbitrary length to a (short) fixed-length output $h(x)$
  - hash functions have many uses including hash tables
- We are interested in cryptographic hash functions that must satisfy certain security properties
  - it is computationally hard to invert $h(x)$
  - it is computationally hard to find collisions in h
- Other uses of hash functions include
  - password hashing
  - in digital signatures
  - in intrusion detection and forensics

Hash Functions

- h must satisfy the following security properties:
  - Preimage resistance (one-way): given $h(x)$, it is difficult to find x
  - Second preimage resistance (weak collision resistance): given x, it is difficult to find x' such that $x' \neq x$ and $h(x') = h(x)$
  - Collision resistance (strong collision resistance): it is difficult to find any x, x' such that $x' \neq x$ and $h(x') = h(x)$
- Additional properties normally present in cryptographic hash functions:
  - input bits and output bits should not be correlated
  - it should be hard to find any two inputs x and x' such that $h(x)$ and $h(x')$ differ only in a small number of bits
  - given $h(x)$, it should be difficult to recover any substring of the input

Attacks on Hash Functions

- Brute force search attack
  - success solely depends on the length of the hash n
  - difficulty of finding a preimage or a second preimage is 2^n
  - difficulty of finding a collision with probability 0.5 is about $2^{n/2}$
    - this is due to so-called birthday attack that computes hashes of $2^{n/2}$ versions of a message (discussed in CSE 664)
    - collision resistance is desired for a general-use hash function
- Cryptanalysis attacks are specific to hash function algorithms

Hash Functions

- Well known hash function algorithms:
  - MD5
  - SHA-1
  - SHA-2 family (SHA-256, SHA-384, and others)
  - new SHA-3
- Normally hash function algorithms are iterated
  - they use a compression function
  - the input is partitioned into blocks
  - a compression function is used on the current block m_i and the previous output h_{i-1} to compute

$$h_i = f(m_i, h_{i-1})$$

Hash Function Algorithms

- Families of customized hash functions
  - MD2, MD4, MD5 (MD = message digest)
    - all have 128-bit output
    - MD4 and MD5 were specified as internet standards in RFC 1320 and 1321
    - MD5 was designed as a strengthened version of MD4 before weaknesses in MD4 were found
    - collisions have been found for MD4 in 2^{20} compression function computations (90s)
    - in 2004 collisions for many MD5 configurations were found
    - MD5 (and all preceding versions) are now too weak and not to be used

Hash Function Algorithms

- Secure Hash Algorithm (SHA)
  - SHA was designed by NIST and published in FIPS 180 in 1993
  - In 1995 a revision, known as SHA-1, was specified in FIPS 180-1
    - it is also specified in RFC 3174
  - SHA-0 and SHA-1 have 160 bit output and MD4-based design
  - In 2002 NIST produced a revision of the standard in FIPS 180-2
  - SHA-2 hash functions have length 256, 384, and 512 to be compatible with the increased security of AES
    - they are known as SHA-256, SHA-384, and SHA-512
  - Also, SHA-224 was added to compatibility with 3DES

Hash Function Algorithms

- Security of SHA
  - brute force attack is harder than in MD5 (160 bits vs. 128 bits)
  - SHA performs more complex transformations that MD5
    - it makes finding collisions more difficult
  - in 2004 collisions in SHA-0 were found in $< 2^{40}$
  - in 2005 collisions have been found in “reduced” SHA-1 (2^{33} work)
  - finding collisions in the full version of SHA-1 is estimated at $< 2^{69}$
  - several other attacks followed and SHA-1 is considered too weak
  - SHA-2 is a viable option, but has the same structure as in SHA-1 (security weaknesses may follow)

Hash Function Algorithms

- SHA-3
  - search for SHA-3 family was announced by NIST in 2007
    - it was required to support digests of 224, 256, 384, and 512 bits and messages of at least $2^{64} - 1$ bits
  - the winner, Keccak, was announced in 2012 and the SHA-3 standard was released in 2015 as NIST's FIPS 202
  - Keccak is a family of sponge functions
    - it is a mode of operation that builds a function mapping variable-length input to variable-length output using a fixed-length permutation and a padding rule
    - SHA-3 can be used with one of seven Keccak permutations
    - the design is distinct from other widely used techniques

Back to Message Authentication

- How do we construct a MAC from a hash function h and key k?
  - consider defining $\mathsf{Mac}_k(m) = h(k||m)$
    - knowledge of the key is required for efficient computation and verification
    - one-way property of h makes key recovery difficult
  - unfortunately, this construction is not as secure as we would like
    - iterative nature of hash function computation gives room for easy forgeries
- HMAC is a more complex construction with provable security

MAC Algorithms

- Hash-Based MAC – HMAC
- Goals:
  - use available hash functions without modifications
  - preserve the original performance of the hash function
  - use and handle keys in a simple way
  - allow replacement of the underlying hash function
  - have a well-understood cryptographic analysis of its strength

HMAC

- HMAC
  - $\mathsf{HMAC}_k(x) = h((K \oplus opad)||h((K \oplus ipad)||x))$
  - K is the key k padded to a full block ($\geq \texttt{512}$ depending on hash function)
  - $ipad = \texttt{0x3636...36}$ and $opad = \texttt{0x5C5C...5C}$ are fixed padding constants
- HMAC is efficient to compute
  - the entire message is hashed only once
  - the second time h is called on only two blocks

HMAC

- HMAC Security
  - security is related to that of the underlying hash function
    - we want $k_1 = h(K \oplus opad)$ and $k_2 = h(K \oplus ipad)$ to be rather independent and close to random
    - then HMAC is existentially unforgeable under an adaptive chosen-message attack for messages of any length
  - HMAC provides greater security than the security of the underlying hash function
  - no known practical attacks if a secure hash function is used according to the specifications

Confidentiality + Integrity

- How do we use a MAC in combination with encryption?
  - message authentication
    - $A \xrightarrow{m, \mathsf{Mac}_k(m)} B$
  - encrypt and authenticate
    - $A \xrightarrow{\mathsf{Enc}_{k_1}(m), \mathsf{Mac}_{k_2}(m)} B$
  - authenticate then encrypt
    - $A \xrightarrow{\mathsf{Enc}_{k_1}(m, \mathsf{Mac}_{k_2}(m))} B$
  - encrypt then authenticate
    - $A \xrightarrow{\mathsf{Enc}_{k_1}(m), \mathsf{Mac}_{k_2}(\mathsf{Enc}_{k_1}(m))} B$

Confidentiality + Integrity

- Analysis of prior constructions:
  - encrypt and authenticate
    - transmitting $\mathsf{Mac}_{k_2}(m)$ may leak information about m
  - authenticate then encrypt
    - has a chosen-ciphertext attack against the general version, which has been successfully applied in practice
  - encrypt then authenticate
    - satisfies the definition of authenticated encryption and is CCA-secure
- The keys k_1 and k_2 must be different!

Authenticated Encryption

- Do I have to use encryption and MAC separately or are there authenticated encryption modes?
  - recently, authenticated encryption modes have been proposed
- Some good reads:
  - https://blog.cryptographyengineering.com/2012/05/19/how-to-choose-authenticated-encryption/
  - https://stackoverflow.com/questions/1220751/how-to-choose-an-aes-encryption-mode-cbc-ecb-ctr-ocb-cfb

Authenticated Encryption

- Good options to consider:
  - Offset Codebook (OCB) mode
    - state of the art in authenticated encryption
    - proposed internet standard
    - has licensing restrictions
    - see http://web.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm for more information
  - Galois/Counter Mode (GCM)
    - does not have licensing restrictions
    - can be used as an alternative for commercial software

Summary

- We so far covered
  - symmetric encryption, block ciphers
  - encryption standards (DES, AES)
  - message authentication codes
  - hash functions (MD5, SHA-1, SHA-2, SHA-3)
- More to come
  - public key cryptography
  - pseudo-random number generators