COMP412 Computer Security

Lec 08 Cryptographic Hash Functions

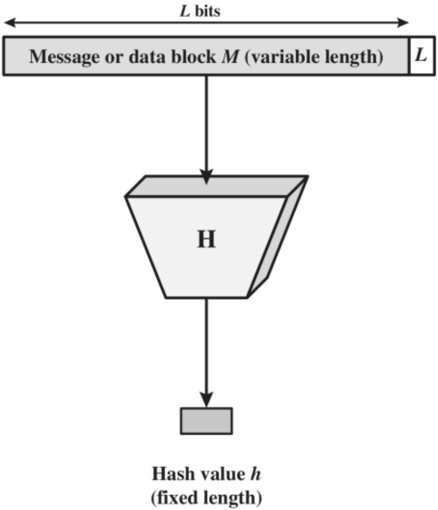
Dr. Xiaochen Yuan
2021/2022

Contents

Hash Functions

- Hash function $\mathcal{H}$: variable-length block of data M input; fixed-size hash value $h = \mathcal{H}(M)$ output
- Applying $\mathcal{H}$ to large set of inputs should produce evenly distributed and random looking outputs
- Cryptographic hash function: computationally **infeasible** to find:
  1. M that maps to known h (**one-way property**)
  2. M_1 and M_2 that produce same h (**collision-free property**)
- Used to determine whether or not data has changed
- Examples: message **authentication**, **digital signatures**, one-way password file, intrusion/virus detection, PRNG

Cryptographic Hash Function



Credit: Figure 11.1 in Stallings, *Cryptography and Network Security*, 6th Ed.

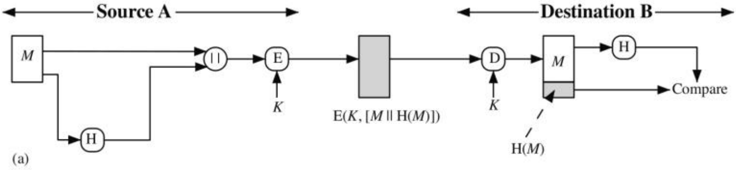
Contents

Message Authentication

- Verify the integrity of a message
  - Ensure data received are exactly as sent
  - Assure identity of the sender is valid
- Hash function used to provide message authentication called message digest

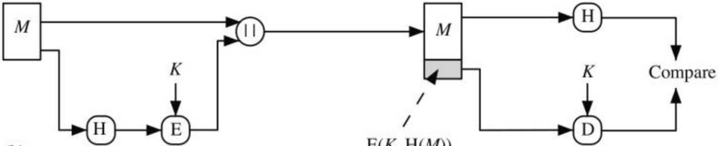
Message Authentication Example (a)

- Encrypt the message and hash code using **symmetric encryption**



Message Authentication Example (b)

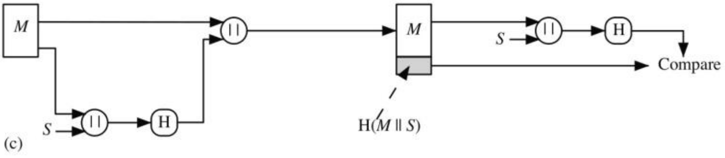
- Encrypt only hash code
- Reduces computation overhead when confidentiality not required



(b)

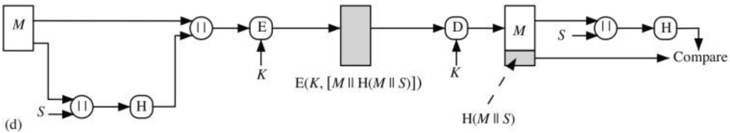
Message Authentication Example (c)

- Shared secret S is hashed
- **No encryption** needed



Message Authentication Example (d)

- Shared secret combined with confidentiality



Authentication and Encryption

- Sometimes desirable to avoid encryption when performing authentication
  - Encryption in software can be slow
  - Encryption in hardware has financial costs
  - Encryption hardware can be inefficient for small amounts of data
  - Encryption algorithms may be patented, increasing costs to use
- Message Authentication Codes (or keyed hash function)
  - Take secret key K and message M as input; produce hash (or MAC) as output
  - Combining hash function and encryption produces same result as MAC; but MAC algorithms can be more efficient than encryption algorithms
  - MAC covered in next topic

Contents

Digital Signatures

- Aim of a signature: prove to anyone that a message originated at (or is approved by) a particular user
- Symmetric key cryptography
  - Two users, A and B, share a secret key K
  - Receiver of message (user A) can verify that message came from the other user (B)
  - User C *cannot* prove that the message came from B (it may also have came from A)
- Public key cryptography can provide signature: only one user has the private key

Digital Signature Operations (Concept)

Signing

- User signs a message by encrypting with own **private key**

$$S = E(PR_A, M)$$

- User attaches signature to message

Verification

- User verifies a message by decrypting signature with signer's **public key**

$$M^t = D(PU_A, S)$$

- User then compares received message M with decrypted M^t; if identical, signature is verified

Digital Signature Operations (Practice)

No need to encrypt entire message; encrypt **hash of message**

Signing

- User signs a message by encrypting hash of message with own **private key**

$$S = E(PR_A, H(M))$$

- User attaches signature to message

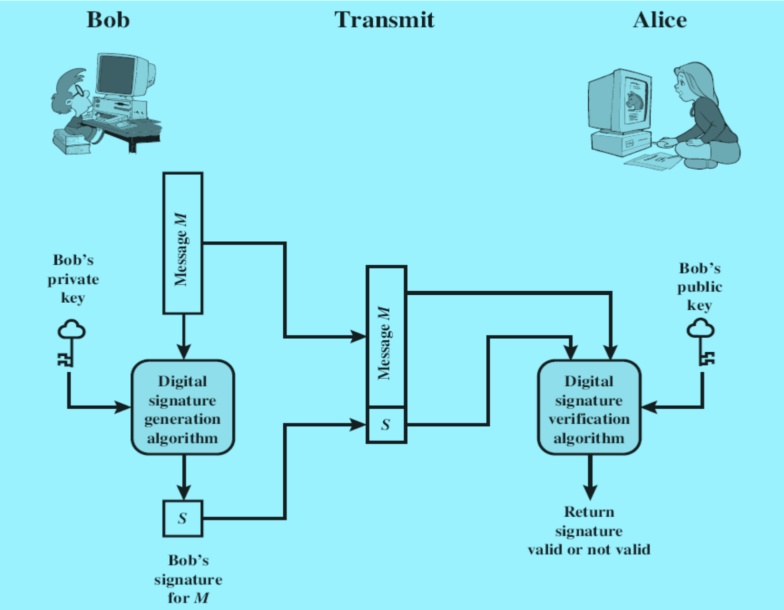
Verification

- User verifies a message by decrypting signature with signer's **public key**

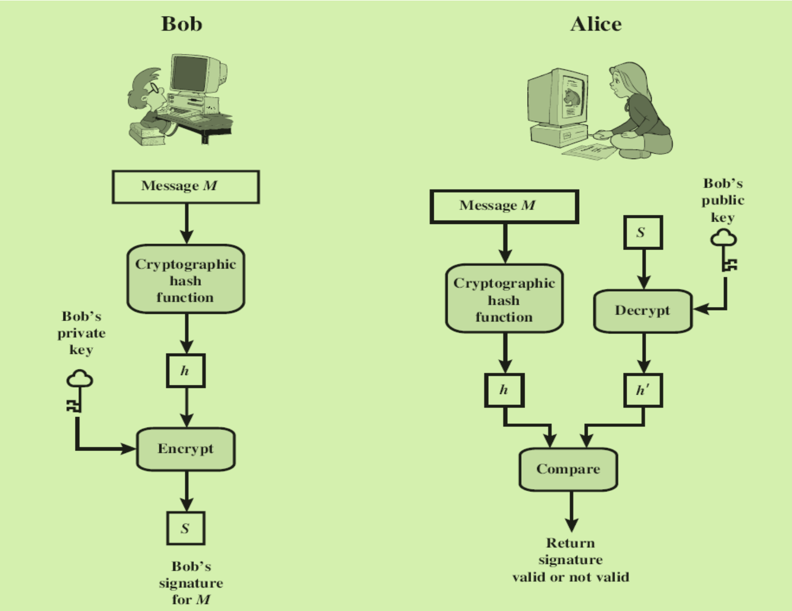
$$h = D(PU_A, S)$$

- User then compares hash of received message, $H(M)$, with decrypted h; if identical, signature is verified

Digital Signatures
Generic Model



Digital Signatures
Essential Elements



Digital Signatures
Schemes

ElGamal Digital Signature Scheme

*The **ElGamal digital signature scheme** stems from the **ElGamal cryptosystem** based upon the security of the one-way function of exponentiation in modular rings and the difficulty of solving the discrete logarithm problem.*

Schnorr Digital Signature Scheme

- "Claus Peter Schnorr"

Digital Signatures
ElGamal Scheme

Let q is prime number & α is a primitive root of q

Generate the private/ public keys

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.

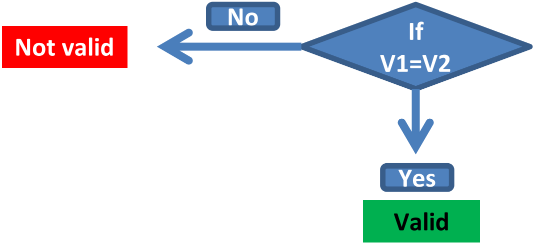
Sign the message

1. Choose a random integer K such that $1 \leq K \leq q - 1$ and $\gcd(K, q - 1) = 1$. That is, K is relatively prime to $q - 1$.
2. Compute $S_1 = \alpha^K \bmod q$. Note that this is the same as the computation of C_1 for ElGamal encryption.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo $q - 1$.
4. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
5. The signature consists of the pair (S_1, S_2).

Digital Signatures
ElGamal Scheme (Cont.)

Verify the message

1. Compute $V_1 = \alpha^m \bmod q$.
2. Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$.



Digital Signatures
ElGamal Scheme (**Example**)

Let $q=19$;
Primitive roots of $q=\{2, 3, 10, 13, 14, 15\}$;
Choose $\alpha=10$

Alice choose $X_A = 16$ (Private Key)

- Help Alice to generate the Public Key?

- Help Alice to sign a message $m = 14$, what is the signature?

- Alice sends the signed message to Bob, help Bob to verify the message.

Digital Signatures
Schnorr Scheme

Generate the private/ public keys

1. Choose primes p and q, such that q is a prime factor of $p - 1$.
2. Choose an integer a, such that $a^q = 1 \bmod p$. The values a, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

Digital Signatures
Schnorr Scheme (Cont.)

Sign the message

1. Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$. This computation is a preprocessing stage independent of the message M to be signed.
2. Concatenate the message with x and hash the result to compute the value e:

$$e = \mathrm{H}(M \,\|\, x)$$

3. Compute $y = (r + se) \bmod q$. The signature consists of the pair (e, y).

Verify the message

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that $e = \mathrm{H}(M \,\|\, x')$.

Digital Signatures
Schnorr Scheme (**Example**)

Let $p = 23$, then $q = 11$,
Therefore $\alpha = 2$
Choose $s = 9$

- Generate the user's Public Key?

- Given the hash function $H(.) = 5$, generate the message signature. (Given $r = 3$)

- Verify the signed message.

Contents

Pre-images and Collisions

- For hash value $h = \mathcal{H}(x)$, x is pre-image of h
- $\mathcal{H}$ is a many-to-one mapping; h has multiple pre-images
- Collision occurs if $x \neq y$ and $\mathcal{H}(x) = \mathcal{H}(y)$
- Collisions are undesirable
- How many pre-images for given hash value?
  - If $\mathcal{H}$ takes b-bit input block, 2^b possible messages
  - For n-bit hash code, where $b > n$, 2^n possible hash codes
  - On average, if uniformly distributed hash values, then each hash value has 2^{b-n} pre-images

Requirements of Cryptographic Hash Function

- Variable input size: $\mathcal{H}$ can be applied to input block of <u>any size</u>
- Fixed output size: $\mathcal{H}$ produces <u>fixed length</u> output
- Efficiency: $\mathcal{H}(x)$ relatively easy to compute (practical implementations)
- Pre-image resistant: For any given h, computationally infeasible to find y such that $\mathcal{H}(y) = h$ (***one-way property***)
- Second pre-image resistant: For any given x, computationally infeasible to find $y \neq x$ with $\mathcal{H}(y) = \mathcal{H}(x)$ (***weak collision resistant***)
- Collision resistant: Computationally infeasible to find any pair (x, y) such that $\mathcal{H}(x) = \mathcal{H}(y)$ (***strong collision resistant***)
- Pseudo-randomness: Output of $\mathcal{H}$ meets standard tests for pseudo-randomness

Required Hash Properties for Different Applications

Weak hash function: Satisfies first 5 requirements (but not collision resistant)

Strong hash function: Also collision resistant

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

Credit: Table 11.2 in Stallings, *Cryptography and Network Security*, 6th Ed.

Brute Attacks on Hash Functions

Pre-image and Second Pre-image Attack

- Find a y that gives specific h; try all possible values of y
- With m-bit hash code, effort required proportional to 2^m

Collision Resistant Brute Attack

- Find any two messages that have same hash values
- Effort required is proportional to $2^{m/2}$
- Due to birthday paradox, easier than pre-image attacks

Practical Effort

- Cryptanalysis attacks possible in theory; complex
- Collision resistance desirable for general hash algorithms
- MD5 uses 128-bits: collision attacks possible (2^{60})
- SHA uses longer codes; collision attacks infeasible

Contents

MD5

- Message Digest algorithm 5, developed by Ron Rivest in 1991
- Standardised by IETF in RFC 1321
- Generates **128-bit** hash
- Was commonly used by applications, passwords, file integrity; no longer recommended
- Collision and other attacks possible; tools publicly available to attack MD5

SHA

- Secure Hash Algorithm, developed by NIST
- Standardised by NIST in FIPS 180 in 1993
- Improvements over time: SHA-0, SHA-1, SHA-2, SHA-3
- SHA-1 (and SHA-0) are considered insecure; no longer recommended
- SHA-3 in development, competition run by NIST

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Credit: Table 11.3 in Stallings, *Cryptography and Network Security*, 6th Ed.