Computer and network security

Beau De Clercq

2020-2021

Contents

1 Introduction

2 Symmetric ciphers

3 Message authentication

3.1 Hash functions

A hash function H is a function that takes input data blocks of length M and returns a hash value of fixed size R.
A cryptographic hash function that also satisfies following conditions:

- One way property: it should be infeasible to find a data object that maps to a predefined hash value.
- Collision free property: it should be infeasible to find 2 data objects that map to the same hash value.
- Use padding to pad up input to fixed length and add the length l of the block in bits.

By satisfying the first two properties, hash functions can be used to determine if data has been altered.

3.1.1 Applications

Hash functions can be used in an number of applications:

- Message authentication: to ensure a message hasn't been altered.
- Digital signatures: ensure the authenticity of messages and identity of the sender.
- One-way password file: store hash value of password in plain text file.
- Intrusion/virus detection: store H(f) for each file to determine if files have been modified.
- Pseudorandom function: use H to generate pseudorandom private key.

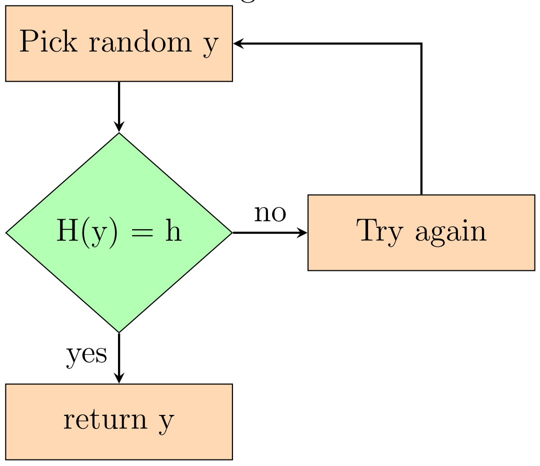
3.1.2 Security requirements

Cryptographic hash functions must adhere to following security requirements:

- Basic:
  - Input data can be of any size
  - Output is of fixed length
  - H(x) is easy to compute
- Advanced:
  - Given h, it is hard to find y: H(y) = h
  - It is hard to find y: y≠x & H(y)=H(x)
  - It is hard to find (x, y): H(x)=H(y)

3.1.3 Attacks

- Brute force preimage attack
  The goal of this attack is to find a y such that H(y) = h for a given hash value h.
  The attack itself goes as follows:



  On average this attack need 2^{m-1} attempts for an m-bit hash value.

- Brute force collision resistance attack
  The goal of this attack is to find two values x and y such that H(x) =

H(y). This attack need $2^{\frac{m}{2}}$ attempts for an m-bit hash value.

Add image of attack from slides

3.2 Secure Hash Algorithm (SHA)

Add SHA overview from slides

3.2.1 SHA512

Add image from slides

3.2.2 SHA512 block processing

Add image from slides

Add image of round function from slides

In this 80 round process, a message of 1024 bits is transformed into 80 values of 64 bits each which are then individually processed in sequence by means of a round function.
After the final round a word-by-word addition 64-bit words $\mathrm{mod} 2^{64}$ is performed to obtain the final hashed value.

3.2.3 Message schedule

The first 16 words $W_0..W_{15}$ are derived directly from the input block M_i. All other words are derived as follows:

$$\begin{aligned}
W_t &= W_{t-16} \boxplus \delta_0(W_{t-15}) \boxplus W_{t-7} \boxplus \delta_1(W_{t-2}) \\
\delta_0(x) &= ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x) \\
\delta_1(x) &= ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x) \\
ROTR^n(x) &= \text{circular right bit-shift by n bits} \\
SHR^n(x) &= \text{right bit-shift of x by n bits with} \\
&\quad \text{padding by n zeros on the left}
\end{aligned}$$

3.3 Length extension attack and SHA3

3.3.1 Length extension attack

Type of attack where an attacker can use H(M_1) and the length of M_1 to calculate H($M_1|M_2$) for an attack controlled message M_2 without needing to know the content of M_1. This is done by taking the old message and using it as 'intermediary' input in the hashing algorithm.

3.3.2 SHA3

Add image from slides

3.4 Message authentication

There are 8 types of attacks:

1. Disclosure of message content
2. Traffic analysis
3. Masquerade
4. Content modification
5. Sequence modification
6. Timing modification
7. Source repudiation
8. Destination repudiation

Items 1 and 2 can be countered using confidentiality mechanisms, eg symmetric encryption.
Items 3 to 6 can be countered by using message authentication to verify that the received message hasn't been altered.
Digital signatures can be seen as an extension to message authentication as it is an authentication technique that also includes measures to counter repudiation by the source.

A message authentication function is a function that produces an authenticator (a value to be used to authenticate a message). There are 3 classes of message authentication functions:

- Hash functions:
  - Concatenate message M with secret key S and hash the stream

  - Send message M together with hash value: if it is intercepted an attacker can't create a new hash value since the key is unknown
  - Check authentication: combine M with S, hash the stream and compare with received hash value

- Message encryption, eg using symmetric encryption, provides both confidentiality and authentication if key K is secret. If the decrypted message M is not meaningful then the message can be considered altered.

- Message authentication code (MAC): generates fixed-size cryptographic checksum based on message and secret key
  - Does not need to be reversible
  - send message with checksum: if the checksum calculated by the receiver doesn't match then the message has been altered
  - Cannot be used to provide digital signatures: a signature need to be verifiable by anyone and MAC requires the use of a secret key

3.4.1 Security requirements for MAC

- If an opponent observes M and $MAC(K, M)$, it should be infeasable to construct M' such that $MAC(K, M') == MAC(K, M)$.

- Given 2 randomly chosen messages M and M', the probability that $MAC(K, M) == MAC(K, M')$ should be 2^{-N} for an N-bit code. $-->$ codes should be distributed uniformly and random over the entire output space

- If M' is a known transformation of M, then $\mathbb{P}(MAC(K, M) = MAC(K, M') = 2^{-N}$.

3.5 Message authentication codes

3.5.1 Cipher-based MAC (CMAC)

- M is split into fixed size blocks $M_1, ..., M_N$.

- M_1 is encrypted with key K of k bits.

- Resulting block is then XORed with next input block, the result is then encrypted with the same key K. This process is repeated until block M_{N-1}.
- At the final block M_N: XOR M_N with M_{N-1} and b-bit constant K_1 derived from the original key.
- Encrypt the result using key K, then the leftmost significant bits represent the tag.
- If M is not a multiple of b: last block is padded with 10..0 and K_2 is used in stead of K_1.
- Determining K_1 and K_2:

  $$L = E(K, 0^b), K_1 = L * x, K_2 = L * x^2$$

  where multiplication happens in $GF(2^b)$

3.5.2 Hash-based MAC (HMAC)

- Advantages: hash functions execute faster and library code is widely available.
- Main issue: a secret key needs to be used when using a hash function.
- Objectives:
  - To use, without modification, available hash functions;
  - To allow for easy replaceability of the hash function;
  - To preserve the hash function's original performance;
  - To use and handle keys in a simple way;
  - To have a well understood cryptographic analysis.
- Structure:

  copy image from slide

- Cryptographic strengths:

  - HMAC can be proven secure provided that the embedded hash function has some reasonable cryptographic strengths.
  - A successful attack on HMAC is equivalent to one of the following:
    * The attacker is able to compute an output of the compression function.
    * The attacker finds collisions in the hash function.

4 Asymmetric encryption

4.1 Introduction

Asymmetric encryption uses both a public and private key, it's working is based on mathematical functions rather than substitution and permutations. AE addresses two concerns with symmetric encryption:

- Secret keys are distributed using a trusted key distribution center
- It does not enable digital signatures

4.1.1 Confidentiality using AE

Assume there is some source A that produces a message in plaintext X which is intended for destination B. B generates a related pair of keys: a public key, PU_b, and a private key, PR_b. PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.
With the message X and the encryption key PU_b as input, A forms the ciphertext $Y = E(PU_b, X)$. The intended receiver, in possession of the matching private key, is able to invert the transformation: $X = D(PR_b, Y)$. An adversary, observing Y and having access to PU_b, but not having access to PR_b or X, must attempt to recover X and/or PR_b. It is assumed that the adversary does have knowledge of the encryption (E) and decryption (D) algorithms. If the adversary is interested only in this particular message, then the focus of effort is to recover X by generating a plaintext estimate $\widehat{X}$. Often, however, the adversary is interested in being able to read future messages as well, in which case an attempt is made to recover PR_b by generating an estimate $\widehat{PR_b}$.

4.1.2 Authentication using AE

In this case, A prepares a message to B and encrypts it using As private key before transmitting it. B can decrypt the message using As public key. Because the message was encrypted using As private key, only A could have prepared the message.Therefore,the entire encrypted message serves as a digital signature. In addition, it is impossible to alter the message without access to As private key, so the message is authenticated both in terms of source and in terms of data integrity.
In this scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document must be kept in plaintext to be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the senders private key, it serves as a signature that verifies origin, content, and sequencing.

4.1.3 Combining confidentiality and authentication

It is possible to provide both the authentication function and confidentiality by a double use of the public-key scheme

- $Z = E(PU_b, E(PR_a, X))$
- $X = D(PU_a, D(PR_b, Z))$

where (PU_a, PR_a) is a key pair generated by the sender and is used for authentication and (PU_b, PR_b) is a key pair generated by the receiver and is used for confidentiality.
Encryption of a message M then goes as follows: $M \to E(M, PR_a) \to E(E(M, PR_a), PU_b) \to C$.
Decryption of a ciphertext C happens as follows: $C \to D(C, PR_b) \to D(D(C, PR_b), PU_a) \to M$.

4.1.4 Requirements for public-key cryptography

- It is computationally easy to generate the key pair (PU_b, PR_b).
- It is computationally easy, given the public key PU_b and a message M, to generate the corresponding ciphertext $C = E(PU_b, M)$.
- It is computationally easy, given the private key PR_b and a ciphertext C, to recover the plaintext $M = D(PR_b, C) = D(PR_b, E(PU_b, M))$.
- It is computationally infeasable, knowing PU_b or PR_b, to generate the other key.
- It is computationally infeasable, knowing PU_b and ciphertext C, to recover the original message M.
- Optional: the two keys can be applied in any order.

4.2 Rivest-Shamir-Adleman (RSA)

4.2.1 RSA basics

4.2.2 RSA ingredients

4.3 Efficient RSA operations

4.3.1 Efficient exponentiation in modular arithmetic

4.3.2 Efficient operation using public key

4.3.3 Efficient operation using private key

4.3.4 Efficient key generation

4.4 Attacks against RSA

4.4.1 Brute force attacks

4.4.2 Mathematical attacks

4.4.3 Timing attacks

4.4.4 Chosen ciphertext attacks

4.5 Digital signatures

4.5.1 Introduction

4.5.2 Required properties of a digital signature

4.5.3 RSA-PSS

4.5.4 RSA-PSS encoding

4.5.5 MGF

4.5.6 RSA-PSS signature verification

5 Key distribution