Chapter 11

Message Integrity and Message Authentication
Objectives

- To define message integrity
- To define message authentication
- To define criteria for a cryptographic hash function
- To define the Random Oracle Model and its role in evaluating the security of cryptographic hash functions
- To distinguish between an MDC and a MAC
- To discuss some common MACs
The cryptography systems that we have studied so far provide secrecy, or confidentiality, but not integrity. However, there are occasions where we may not even need secrecy but instead must have integrity.

Topics discussed in this section:
11.1 Document and Fingerprint
11.2 Message and Message Digest
11.3 Difference
11.4 Checking Integrity
11.5 Cryptographic Hash Function Criteria
One way to preserve the integrity of a document is through the use of a fingerprint. If Alice needs to be sure that the contents of her document will not be changed, she can put her fingerprint at the bottom of the document.
The electronic equivalent of the document and fingerprint pair is the message and digest pair.

**Figure 11.1 Message and digest**

- **Message** (document)
- **Hash function**
- **Message digest** (fingerprint)
11.1.3 Difference

The two pairs (document / fingerprint) and (message / message digest) are similar, with some differences. The document and fingerprint are physically linked together. The message and message digest can be unlinked separately, and, most importantly, the message digest needs to be safe from change.

Note

The message digest needs to be safe from change.
11.1.4 Checking Integrity

Figure 11.2 Checking integrity
A cryptographic hash function must satisfy three criteria: preimage resistance, second preimage resistance, and collision resistance.

**Figure 11.3** Criteria of a cryptographic hash function

- Preimage resistance
- Second preimage resistance
- Collision resistance
11.1.5 Continued

**Preimage Resistance**

*Preimage Attack*

Given: \( y = h(M) \)

Find: \( M' \) such that \( y = h(M') \)

---

**Figure 11.4 Preimage**

M: Message  
Hash: Hash function  
h(M): Digest

---

Given: \( y \)

Find: any \( M' \) such that \( y = h(M') \)

---

Eve

To Bob
Can we use a conventional lossless compression method such as StuffIt as a cryptographic hash function?

Solution
We cannot. A lossless compression method creates a compressed message that is reversible.

Can we use a checksum function as a cryptographic hash function?

Solution
We cannot. A checksum function is not preimage resistant, Eve may find several messages whose checksum matches the given one.
11.1.5 Continued

Second Preimage Resistance

Second Preimage Attack

Given: M and h(M)

Find: M’ ≠ M such that h(M) = h(M’)

Figure 11.5 Second preimage

M: Message
Hash: Hash function
h(M): Digest

Eve

Given: M and h(M)
Find: M’ such that M ≠ M’, but h(M) = h(M’)

Alice

To Bob
Collision Resistance

Collision Attack

Given: none
Find: $M' \neq M$ such that $h(M) = h(M')$

Figure 11.6 Collision

$M$: Message
Hash: Hash function
$h(M)$: Digest

Find: $M$ and $M'$ such that $M \neq M'$, but $h(M) = h(M')$
The Random Oracle Model, which was introduced in 1993 by Bellare and Rogaway, is an ideal mathematical model for a hash function.

Topics discussed in this section:

11.2.1 Pigeonhole Principle
11.2.2 Birthday Problems
11.2.3 Attacks on Random Oracle Model
11.2.4 Attacks on the Structure
Example 11.3

Assume an oracle with a table and a fair coin. The table has two columns.

Table 11.1  *Oracle table after issuing the first three digests*

<table>
<thead>
<tr>
<th>Message</th>
<th>Message Digest</th>
</tr>
</thead>
<tbody>
<tr>
<td>4523AB1352CDEF45126</td>
<td>13AB</td>
</tr>
<tr>
<td>723BAE38F2AB3457AC</td>
<td>02CA</td>
</tr>
<tr>
<td>AB45CD1048765412AAAB6662BE</td>
<td>A38B</td>
</tr>
</tbody>
</table>

a. The message AB1234CD8765BDAD is given for digest calculation. The oracle checks its table.
b. The message 4523AB1352CDEF45126 is given for digest calculation. The oracle checks its table and finds that there is a digest for this message in the table (first row). The oracle simply gives the corresponding digest (13AB).
Example 11.4

The oracle in Example 11.3 cannot use a formula or algorithm to create the digest for a message. For example, imagine the oracle uses the formula \( h(M) = M \mod n \). Now suppose that the oracle has already given \( h(M_1) \) and \( h(M_2) \). If a new message is presented as \( M_3 = M_1 + M_2 \), the oracle does not have to calculate the \( h(M_3) \). The new digest is just \( [h(M_1) + h(M_2)] \mod n \) since

\[
    h(M_3) = (M_1 + M_2) \mod n = M_1 \mod n + M_2 \mod n = [h(M_1) + h(M_2)] \mod n
\]

This violates the third requirement that each digest must be randomly chosen based on the message given to the oracle.
If \( n \) pigeonholes are occupied by \( n + 1 \) pigeons, then at least one pigeonhole is occupied by two pigeons. The generalized version of the pigeonhole principle is that if \( n \) pigeonholes are occupied by \( kn + 1 \) pigeons, then at least one pigeonhole is occupied by \( k + 1 \) pigeons.
Example 11.5

Assume that the messages in a hash function are 6 bits long and the digests are only 4 bits long. Then the possible number of digests (pigeonholes) is $2^4 = 16$, and the possible number of messages (pigeons) is $2^6 = 64$. This means $n = 16$ and $kn + 1 = 64$, so $k$ is larger than 3. The conclusion is that at least one digest corresponds to four $(k + 1)$ messages.
11.2.2 Birthday Problems

Figure 11.7 Four birthday problems

a. First problem

b. Second problem

c. Third problem

d. Fourth problem

Equal with $P \geq 1/2$
11.2.2 Continued

Summary of Solutions

Solutions to these problems are given in Appendix E for interested readers; The results are summarized in Table 11.3.

Table 11.3  Summarized solutions to four birthday problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Probability</th>
<th>General value for $k$</th>
<th>Value of $k$ with $P = 1/2$</th>
<th>Number of students $(N = 365)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \approx 1 - e^{-k/N}$</td>
<td>$k \approx \ln[1/(1 - P)] \times N$</td>
<td>$k \approx 0.69 \times N$</td>
<td>253</td>
</tr>
<tr>
<td>2</td>
<td>$P \approx 1 - e^{-(k - 1)/N}$</td>
<td>$k \approx \ln[1/(1 - P)] \times N + 1$</td>
<td>$k \approx 0.69 \times N + 1$</td>
<td>254</td>
</tr>
<tr>
<td>3</td>
<td>$P \approx 1 - e^{k(k - 1)/2N}$</td>
<td>$k \approx {2 \ln [1/(1 - P)]}^{1/2} \times N^{1/2}$</td>
<td>$k \approx 1.18 \times N^{1/2}$</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>$P \approx 1 - e^{-k^2/2N}$</td>
<td>$k \approx {\ln [1/(1 - P)]}^{1/2} \times N^{1/2}$</td>
<td>$k \approx 0.83 \times N^{1/2}$</td>
<td>16</td>
</tr>
</tbody>
</table>
Comparison

Figure 11.8 Graph of four birthday problem
11.2.3 Attacks on Random Oracle Model

Preimage Attack

Algorithm 11.1 Preimage attack

Preimage_Attack (D)
{
    for (i = 1 to k)
    {
        create (M[i])
        T ← h(M[i])      // T is a temporary digest
        if (T = D) return M[i]
    }
    return failure
}

The difficulty of a preimage attack is proportional to $2^n$. 
11.2.3 Continued

Example 11.6

A cryptographic hash function uses a digest of 64 bits. How many digests does Eve need to create to find the original message with the probability more than 0.5?

Solution

The number of digests to be created is $k \approx 0.69 \times 2^n \approx 0.69 \times 2^{64}$. This is a large number. Even if Eve can create $2^{30}$ (almost one billion) messages per second, it takes $0.69 \times 2^{34}$ seconds or more than 500 years. This means that a message digest of size 64 bits is secure with respect to preimage attack, but, as we will see shortly, is not secured to collision attack.
11.2.3 Continued

Second Preimage Attack.

Algorithm 11.2 Second preimage attack

Second_Preimage_Attack (D, M)
{
    for (i = 1 to k −1)
    {
        create (M [i])
        T ← h (M [i]) // T is a temporary digest
        if (T = D) return M [i]
    }
    return failure
}

The difficulty of a second preimage attack is proportional to $2^n$. 

11.24
Collision Attack

Algorithm 11.3 Collision attack

Collision_Attack
{
    for (i = 1 to k)
    {
        create (M[i])
        D[i] ← h (M[i])  // D[i] is a list of created digests
        for (j = 1 to i - 1)
        {
            if (D[i] = D[j]) return (M[i] and M[j])
        }
    }
    return failure
}

The difficulty of a collision attack is proportional to $2^{n/2}$.
A cryptographic hash function uses a digest of 64 bits. How many digests does Eve need to create to find two messages with the same digest with the probability more than 0.5?

Solution
The number of digests to be created is $k \approx 1.18 \times 2^{n/2} \approx 1.18 \times 232$. If Eve can test $2^{20}$ (almost one million) messages per second, it takes $1.18 \times 2^{12}$ seconds, or less than two hours. This means that a message digest of size 64 bits is not secure against the collision attack.
11.2.3 Continued

Alternate Collision Attack

Algorithm 11.4  Alternate collision attack

Allocate Collision Attack (M[k], M′[k])
{
    for (i = 1 to k)
    {
        D[i] ← h(M[i])
        D′[i] ← h(M′[i])
        if (D[i] = D′[j]) return (M[i], M′[j])
    }
    return failure
}

The difficulty of an alternative collision attack is proportional to $2^{n/2}$. 
Summary of Attacks

Table 11.4 shows the level of difficulty for each attack if the digest is $n$ bits.

Table 11.4  Levels of difficulties for each type of attack

<table>
<thead>
<tr>
<th>Attack</th>
<th>Value of $k$ with $P=1/2$</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preimage</td>
<td>$k \approx 0.69 \times 2^n$</td>
<td>$2^n$</td>
</tr>
<tr>
<td>Second preimage</td>
<td>$k \approx 0.69 \times 2^n + 1$</td>
<td>$2^n$</td>
</tr>
<tr>
<td>Collision</td>
<td>$k \approx 1.18 \times 2^{n/2}$</td>
<td>$2^{n/2}$</td>
</tr>
<tr>
<td>Alternate collision</td>
<td>$k \approx 0.83 \times 2^{n/2}$</td>
<td>$2^{n/2}$</td>
</tr>
</tbody>
</table>
Originally hash functions with a 64-bit digest were believed to be immune to collision attacks. But with the increase in the processing speed, today everyone agrees that these hash functions are no longer secure. Eve needs only $2^{64/2} = 2^{32}$ tests to launch an attack with probability $1/2$ or more. Assume she can perform $2^{20}$ (one million) tests per second. She can launch an attack in $2^{32}/2^{20} = 2^{12}$ seconds (almost an hour).
MD5 (see Chapter 12), which was one of the standard hash functions for a long time, creates digests of 128 bits. To launch a collision attack, the adversary needs to test $2^{64}$ ($2^{128}/2$) tests in the collision algorithm. Even if the adversary can perform $2^{30}$ (more than one billion) tests in a second, it takes $2^{34}$ seconds (more than 500 years) to launch an attack. This type of attack is based on the Random Oracle Model. It has been proved that MD5 can be attacked on less than $2^{64}$ tests because of the structure of the algorithm.
SHA-1 (see Chapter 12), a standard hash function developed by NIST, creates digests of 160 bits. The function is attacks. To launch a collision attack, the adversary needs to test $2^{160/2} = 2^{80}$ tests in the collision algorithm. Even if the adversary can perform $2^{30}$ (more than one billion) tests in a second, it takes $2^{50}$ seconds (more than ten thousand years) to launch an attack. However, researchers have discovered some features of the function that allow it to be attacked in less time than calculated above.
The new hash function, that is likely to become NIST standard, is SHA-512 (see Chapter 12), which has a 512-bit digest. This function is definitely resistant to collision attacks based on the Random Oracle Model. It needs $2^{512/2} = 2^{256}$ tests to find a collision with the probability of 1/2.
The adversary may have other tools to attack hash function. One of these tools, for example, is the meet-in-the-middle attack that we discussed in Chapter 6 for double DES.
A message digest does not authenticate the sender of the message. To provide message authentication, Alice needs to provide proof that it is Alice sending the message and not an impostor. The digest created by a cryptographic hash function is normally called a modification detection code (MDC). What we need for message authentication is a message authentication code (MAC).

**Topics discussed in this section:**
11.3.1 Modification Detection Code (MDC)
11.3.2 Message Authentication Code (MAC)
A modification detection code (MDC) is a message digest that can prove the integrity of the message: that message has not been changed. If Alice needs to send a message to Bob and be sure that the message will not change during transmission, Alice can create a message digest, MDC, and send both the message and the MDC to Bob. Bob can create a new MDC from the message and compare the received MDC and the new MDC. If they are the same, the message has not been changed.
11.3.1 Continued

Figure 11.9 Modification detection code (MDC)

M: Message
Hash: Cryptographic hash function
MDC: Modification detection code
11.3.2 Message Authentication Code (MAC)

**Figure 11.10** Message authentication code

M: Message
MAC: Message authentication code
K: A shared secret key
11.3.2 Continued

The security of a MAC depends on the security of the underlying hash algorithm.

Note
11.3.2 Continued

**Nested MAC**

**Figure 11.11 Nested MAC**

- Message
- Hash
- MAC
- Hash
- MAC
11.3.2 Continued

**HMAC**

Figure 11.12
Details of HMAC
Figure 11.13 CMAC

Mᵢ: Message block i

Encryption algorithm

Encryption algorithm

Encryption algorithm

Select n leftmost bits

CMAC function

m 0-bits

K

Encryption algorithm

Multiply by x or x²

k

Key-generator for last step

n-bit CMAC