

암호프로그래밍

CRYPTOGRAPHY PROGRAMMING

4. 해시함수

정보보호학과
이병천 교수

차례

2

- 1. 강의 개요
- 2. 암호와 정보보호
- 3. 프로그래밍 환경 구축 - 웹, 파이썬
- **4. 해시함수**
- 5. 메시지인증코드
- 6. 비밀번호 기반 키생성
- 7. 대칭키 암호
- 8. 공개키 암호
- 9. 전자서명
- 10. 인증서와 공개키기반구조(PKI)

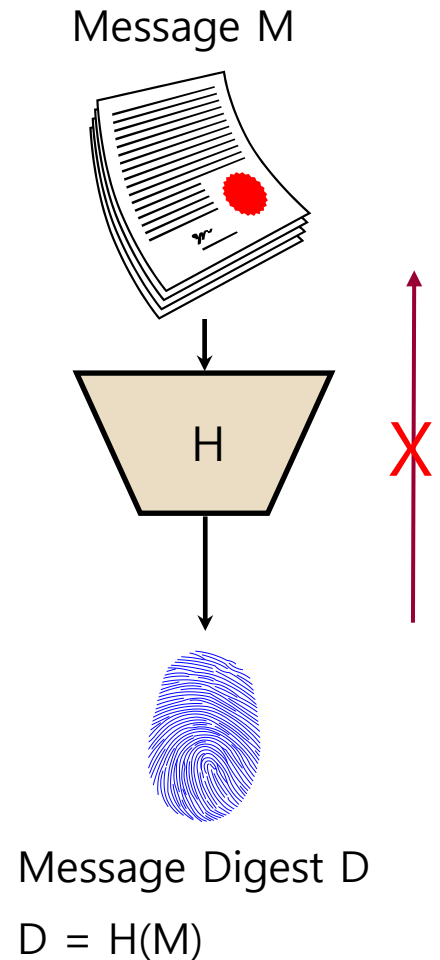
4. 해시함수

1. 해시함수
2. node-forge 패키지
3. 웹 – 서버측
4. 웹 – 클라이언트측
5. 파이썬

1. 해시함수

4

- 해시함수(hash function)란?
 - ▣ 임의의 길이의 데이터를 입력받아서 고정된 길이의 특징값을 출력하는 함수
 - ▣ 메시지 다이제스트(message digest)라고도 부름
 - ▣ 암호키를 사용하지 않는 공개된 함수.
 - ▣ 동일한 입력값에 대해 항상 동일한 해시값을 출력함
 - ▣ 입력값으로부터 해시값을 계산하는 것은 쉽지만 해시값으로부터 그것을 출력하는 입력값을 찾는 것은 어려움

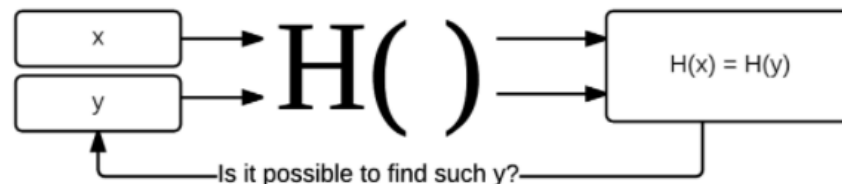


해시함수

5

□ 해시함수의 요구조건

- 역상 저항성(*Pre-image resistance*): 주어진 출력 y 에 대하여 그것의 입력값을 구하는 것이 계산상 불가능하다. $y = h(M)$ 를 만족하는 메시지 M 을 찾는 것이 어려움
- 제2 역상 저항성(*Second pre-image resistance*): 주어진 입력에 대하여 같은 출력을 내는 또 다른 입력을 찾아내는 것이 계산상 불가능하다. $h(M)=h(M')$, $M \neq M'$ 을 만족하는 다른 입력값 M' 을 찾는 것이 어려움
- 충돌저항성(*Collision resistance*): 같은 출력을 내는 임의의 서로 다른 두 입력 메시지를 찾는 것이 계산상 불가능하다.
 $h(M)=h(M')$



해시함수의 용도

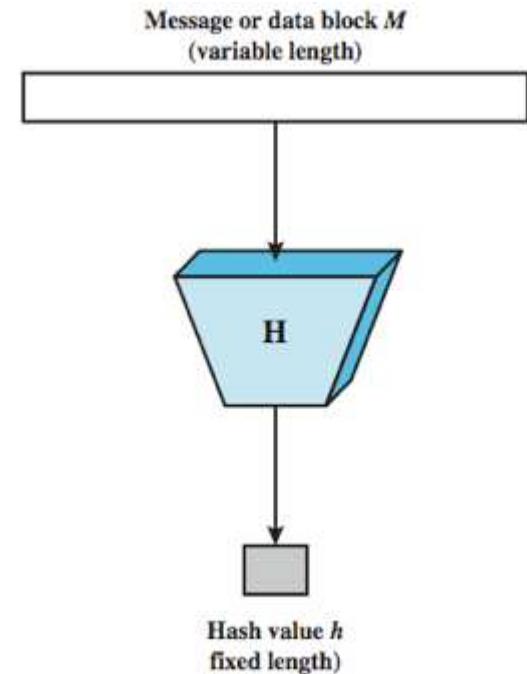
6

- 메시지 다이제스트: 문서의 위조 방지
 - ▣ 해시값을 생성하여 함께 제공
 - ▣ 전자서명과 함께 사용
- 안전한 난수생성
 - ▣ 난수생성함수에서 해시함수를 이용
- 패스워드의 안전한 저장
 - ▣ 사용자의 패스워드를 서버에서는 암호화된 해시값으로 저장
- 안전한 암호화키 생성
 - ▣ 패스워드 기반 키생성 알고리즘 (PBKDF2)
- 체크섬 생성 및 검증
 - ▣ 인터넷으로 배포되는 소프트웨어의 원본 보증

해시함수

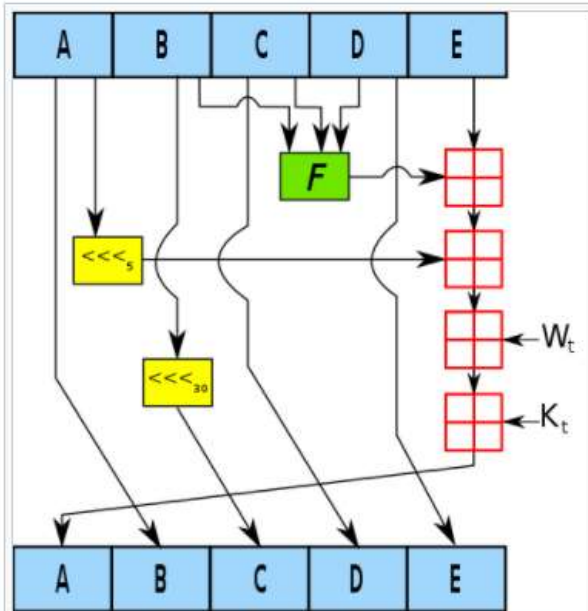
7

- 해시 알고리즘
 - ▣ Md5 – 128비트 출력
 - ▣ Sha1 – 160비트 출력
 - ▣ Sha256 – 256비트 출력
 - ▣ Sha384 – 384비트 출력
 - ▣ Sha512 – 512비트 출력
- 해시 알고리즘의 보안성
 - ▣ 256비트 이상의 출력비트 사용
 - ▣ Md5, Sha1 알고리즘 사용 금지



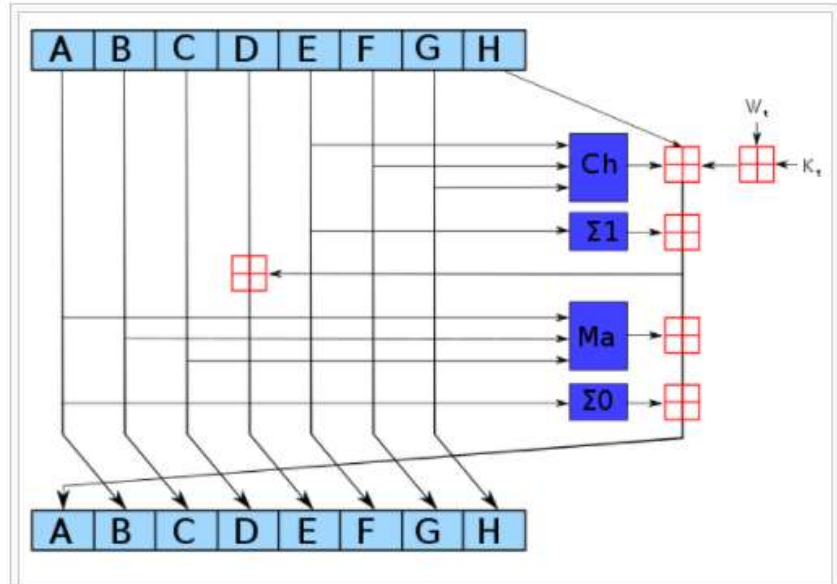
해시함수

8



One iteration within the SHA-1 compression function:
 A, B, C, D and E are 32-bit words of the state;
 F is a nonlinear function that varies;
 \lll_n denotes a left bit rotation by n places;
 n varies for each operation;
 W_t is the expanded message word of round t ;
 K_t is the round constant of round t ;
 \boxplus denotes addition modulo 2^{32} .

SHA-1



One iteration in a SHA-2 family compression function. The blue components perform the following operations:

$$\text{Ch}(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G)$$

$$\text{Ma}(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$

$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$

$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

The bitwise rotation uses different constants for SHA-512. The given numbers are for SHA-256.

The red \boxplus is addition modulo 2^{32} for SHA-256, or 2^{64} for SHA-512.

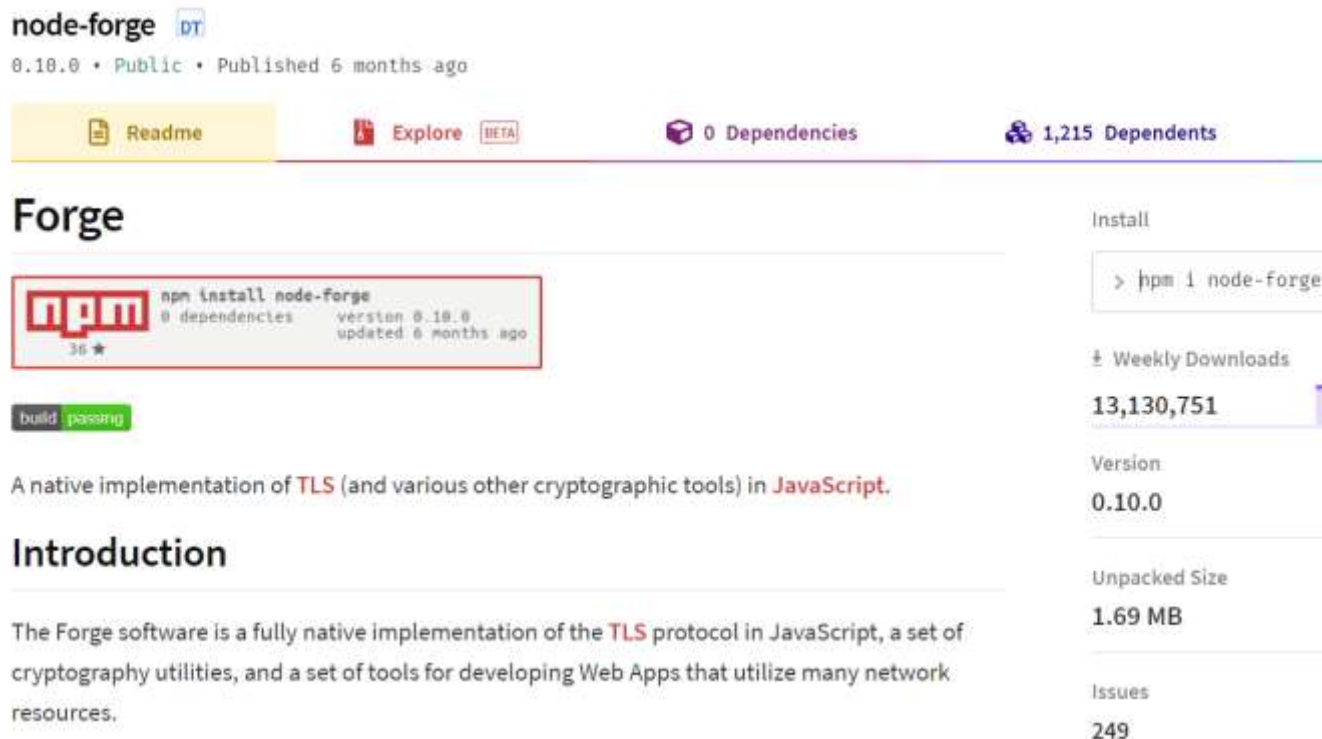
SHA-2

SHA-224, SHA-256, SHA-384, SHA-512,
 SHA-512/224, SHA-512/256

2. node-forge 패키지

9

- Node-forge 패키지 소개
 - ▣ TLS, PKI와 여러 도구들을 포함한 자바스크립트 암호 라이브러리
 - ▣ <https://www.npmjs.com/package/node-forge> API 문서 참조



The screenshot shows the npm package page for 'node-forge'. At the top, it displays the package name 'node-forge' with a 'DT' badge, version '0.10.0', and 'Public' status. Below this are buttons for 'Readme', 'Explore', and '0 Dependencies', along with '1,215 Dependents'. The main section features the 'Forge' title and a code block for installation: `npm install node-forge`. A green 'build passing' badge is visible. The description states it's a native implementation of TLS in JavaScript. The 'Introduction' section begins with 'The Forge software is a fully native implementation of the TLS protocol in JavaScript...'. On the right sidebar, there's an 'Install' button with the command `> npm i node-forge`, 'Weekly Downloads' of 13,130,751, 'Version' 0.10.0, 'Unpacked Size' of 1.69 MB, and 'Issues' of 249.

node-forge DT
0.10.0 • Public • Published 6 months ago

[Readme](#) [Explore](#) BETA [0 Dependencies](#) [1,215 Dependents](#)

Forge

```
npm install node-forge
0 dependencies version 0.10.0
36 stars updated 6 months ago
```

build passing

A native implementation of **TLS** (and various other cryptographic tools) in **JavaScript**.

Introduction

The Forge software is a fully native implementation of the **TLS** protocol in JavaScript, a set of cryptography utilities, and a set of tools for developing Web Apps that utilize many network resources.

Install
`> npm i node-forge`

Weekly Downloads
13,130,751

Version
0.10.0

Unpacked Size
1.69 MB

Issues
249

Node-Forge에 구현된 내용

10

□ Transports

- [TLS](#)
- [HTTP](#)
- [SSH](#)
- [XHR](#)
- [Sockets](#)

□ Ciphers

- [CIPHER](#)
- [AES](#)
- [DES](#)
- [RC2](#)

□ PKI

- [RSA](#)
- [RSA-KEM](#)
- [X.509](#)
- [PKCS#5](#)
- [PKCS#7](#)
- [PKCS#8](#)
- [PKCS#10](#)
- [PKCS#12](#)
- [ASN.1](#)

□ Message Digests

- [SHA1](#)
- [SHA256](#)
- [SHA384](#)
- [SHA512](#)
- [MD5](#)
- [HMAC](#)

□ Utilities

- [Prime](#)
- [PRNG](#)
- [Tasks](#)
- [Utilities](#)
- [Logging](#)
- [Debugging](#)
- [Flash Networking Support](#)

API 문서의 사용법 참조

<https://www.npmjs.com/package/node-forge>

Node-Forge 설치

11

- 프로젝트 폴더 생성
 - ▣ > md forge
 - ▣ > cd forge
- 프로젝트 초기화
 - ▣ > npm init
- 서버측 패키지 설치
 - ▣ > npm install node-forge
 - ▣ node_modules 폴더에 설치됨
 - ▣ 서버측 프로그램에서 다음과 같이 불러서 사용
 - var forge = require('node-forge');

Node-Forge 설치

12

- 클라이언트측 패키지 설치
 - ▣ > bower install forge
 - ▣ 패키지 이름이 forge로 다름에 주의
 - ▣ bower_components 폴더에 설치됨
 - ▣ 클라이언트 html 파일에서 forge.min.js 파일을 불러서 사용
 - <script src=bower_components\forge\dist\forge.min.js></script>

3. 웹 - 서버측

13

□ API

- ▣ Message digest(Hash)를 위한 객체는 forge.md
- ▣ 구현된 해시 알고리즘: Sha1, Sha256, Sha384, Sha512, Md5

□ 사용 방법

- ▣ Create → update → digest

```
var md = forge.md.md5.create();  
md.update('The quick brown fox jumps over the lazy dog');  
console.log(md.digest().toHex());  
// output: 9e107d9d372bb6826bd81d3542a419d6
```

해시함수의 결과는 난수처럼 보이는 ByteStringBuffer 객체
이것을 Hex로 출력하기 위해서는 toHex() 함수 이용

해시함수 - Message Digest

14

- 해시함수의 결과는 난수처럼 보이는 ByteStringBuffer
 - ▣ 이것을 Hex로 출력하기 위해서는 toHex() 함수 이용
 - ▣ 또는 forge.util.bytesToHex 함수 이용

```
var forge = require('node-forge');  
var inputText = 'The quick brown fox jumps over the lazy dog';  
console.log('Input Text: '+inputText);
```

```
var md = forge.md.md5.create();  
md.update(inputText);  
console.log('MD5: '+md.digest().toHex());
```

```
var md = forge.md.md5.create();  
md.update(inputText);  
var result = md.digest();  
console.log(result);  
console.log('MD5: '+forge.util.bytesToHex(result));
```

> Node digest.js

```
F:\AppliedCrypto\forge>node digest.js  
Input Text: The quick brown fox jumps over the lazy dog  
MD5: 9e107d9d372bb6826bd81d3542a419d6  
ByteStringBuffer {  
  data: '□\u0010}□7+□ □kØ\u001d5B □\u0019Ö',  
  read: 0,  
  _constructedStringLength: 16 }  
MD5: 9e107d9d372bb6826bd81d3542a419d6
```

해시함수 - Message Digest

15

digest.js

```
var forge = require('node-forge');  
var inputText = 'The quick brown fox jumps over the lazy dog';  
console.log('Input Text: '+inputText);
```

```
var md = forge.md.md5.create();  
md.update(inputText);  
console.log('MD5: '+md.digest().toHex());
```

```
var md = forge.md.sha1.create();  
md.update(inputText);  
console.log('SHA1: '+md.digest().toHex());
```

```
var md = forge.md.sha256.create();  
md.update(inputText);  
console.log('SHA256: '+md.digest().toHex());
```

```
var md = forge.md.sha384.create();  
md.update(inputText);  
console.log('SHA384: '+md.digest().toHex());
```

```
var md = forge.md.sha512.create();  
md.update(inputText);  
console.log('SHA512: '+md.digest().toHex());
```

```
var md = forge.md.sha512.sha256.create();  
md.update(inputText);  
console.log('SHA512.SHA256: '+md.digest().toHex());
```

```
F:\AppliedCrypto\forge>node digest.js  
Input Text: The quick brown fox jumps over the lazy dog  
MD5: 9e107d9d372bb6826bd81d3542a419d6  
SHA1: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12  
SHA256: d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592  
SHA384: ca737f1014a48f4c0b6dd43cb177b0afd9e5169367544c494011e3317dbf9a509c  
SHA512: 07e547d9586f6a73f73fbac0435ed76951218fb7d0c8d788a309d785436bbb642e  
a0538f3db854fee6  
SHA512.SHA256: dd9d67b371519c339ed8dbd25af90e976a1eeefd4ad3d889005e532fc5b
```

4. 웹 - 클라이언트측

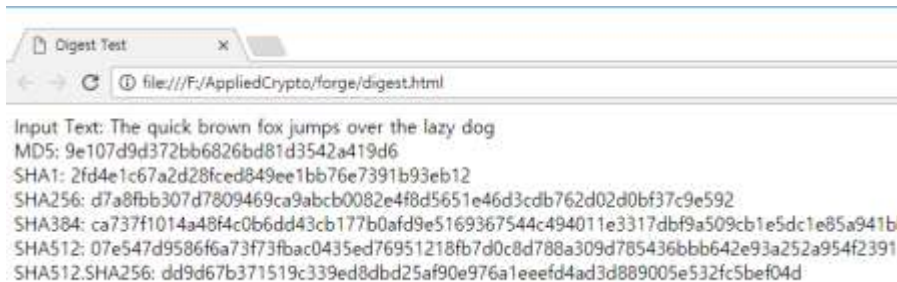
16

클라이언트측(브라우저)에서 실행

digest.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Digest Test</title>
    <script src='bower_components/forge/dist/forge.min.js'>
    </script>
    <script src='digest-c.js'> </script>
  </head>
  <body>
  </body>
</html>
```

console.log → document.write
개행명령
 추가



digest-c.js

```
var inputText = 'The quick brown fox jumps over the lazy dog';  
document.write('Input Text: '+inputText+'<br>');
```

```
var md = forge.md.md5.create();  
md.update(inputText);  
document.write('MD5: '+md.digest().toHex()+'<br>');
```

```
var md = forge.md.sha1.create();  
md.update(inputText);  
document.write('SHA1: '+md.digest().toHex()+'<br>');
```

```
var md = forge.md.sha256.create();  
md.update(inputText);  
document.write('SHA256: '+md.digest().toHex()+'<br>');
```

```
var md = forge.md.sha384.create();  
md.update(inputText);  
document.write('SHA384: '+md.digest().toHex()+'<br>');
```

```
var md = forge.md.sha512.create();  
md.update(inputText);  
document.write('SHA512: '+md.digest().toHex()+'<br>');
```

```
var md = forge.md.sha512.sha256.create();  
md.update(inputText);  
document.write('SHA512.SHA256: '+md.digest().toHex()+'<br>');
```


5. 해시 데모 페이지

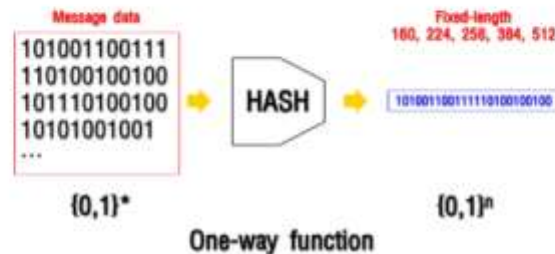
17

□ 실습

해시함수

해시함수는 임의의 길이의 입력메시지에 대하여 고정된 길이의 특징값(해시값)을 계산해주는 함수이다. 키가 사용되지 않으므로 입력메시지가 같으면 동일한 해시값을 출력한다. 해시함수는 다음과 같은 특성을 만족시켜야 한다.

1. 일방향성: 입력메시지로부터 해시값을 계산하는 것은 쉽지만 출력 해시값으로부터 그 해시값을 출력하는 입력메시지를 찾는 것은 어렵다.
2. 충돌회피성: 같은 해시값을 출력하는 두개의 입력메시지를 찾아내는 것은 어렵다.



해시알고리즘 선택

SHA256 ▼

메시지

해시함수 테스트 메시지. 이곳에 메시지를 넣어보세요...

해시값

해시함수 계산

해시함수 계산

초기화

5. 파이썬 – 해시

18

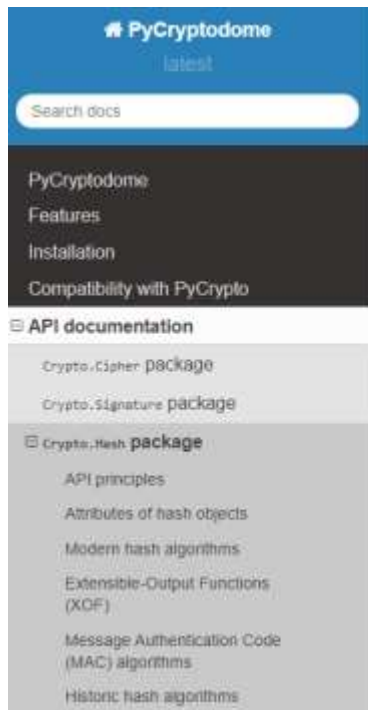
- 폴더 생성
 - ▣ > md python
 - ▣ > cd python
- Pycryptodome 패키지 설치 필요
 - ▣ > pip install pycryptodome

Package	Description
Crypto.Cipher	Modules for protecting confidentiality that is, for encrypting and decrypting data (example: AES).
Crypto.Signature	Modules for assuring authenticity , that is, for creating and verifying digital signatures of messages (example: PKCS#1 v1.5).
Crypto.Hash	Modules for creating cryptographic digests (example: SHA-256).
Crypto.PublicKey	Modules for generating, exporting or importing <i>public keys</i> (example: RSA or ECC).
Crypto.Protocol	Modules for facilitating secure communications between parties, in most cases by leveraging cryptographic primitives from other modules (example: Shamir's Secret Sharing scheme).
Crypto.IO	Modules for dealing with encodings commonly used for cryptographic data (example: PEM).
Crypto.Random	Modules for generating random data.
Crypto.Util	General purpose routines (example: XOR for byte strings).

파이썬 – 해시

19

- Crypto.hash 패키지 사용
 - ▣ From Crypto.Hash import SHA256



[Docs](#) » [API documentation](#) » [Crypto.Hash package](#)

[Edit on GitHub](#)

Crypto.Hash package

Cryptographic hash functions take arbitrary binary strings as input, and produce a random-like fixed-length output (called *digest* or *hash value*).

It is practically infeasible to derive the original input data from the digest. In other words, the cryptographic hash function is *one-way* (*pre-image resistance*).

Given the digest of one message, it is also practically infeasible to find another message (*second pre-image*) with the same digest (*weak collision resistance*).

Finally, it is infeasible to find two arbitrary messages with the same digest (*strong collision resistance*).

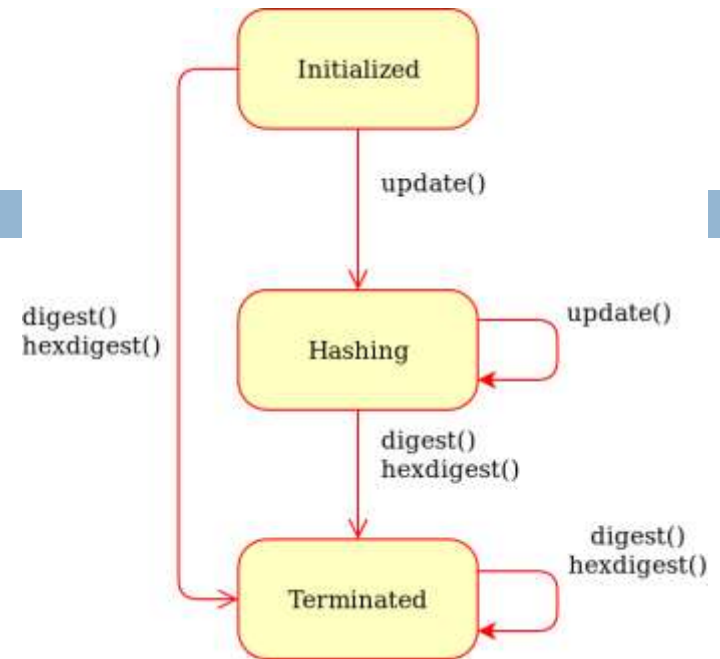
Regardless of the hash algorithm, an n bits long digest is at most as secure as a symmetric encryption algorithm keyed with $n/2$ bits (*birthday attack*).

Hash functions can be simply used as integrity checks. In combination with a public-key algorithm, you can implement a digital signature.

파이썬 – 해시

20

- 파일 생성
 - ▣ > hash.py
- 사용 순서
 - ▣ New() – 새로운 해시 객체 생성
 - ▣ Update() – 해시함수에 메시지를 입력
 - ▣ Digest(), hexdigest() – 해시값을 출력



파이썬 – 해시

21

□ 특징

- ▣ 해시함수에는 byte string 또는 byte array만 입력 가능
- ▣ Update() 메서드는 여러 번 적용할 수도 있음. 여기서 메시지 입력 순서에 따라 해시값 달라짐
- ▣ Digest() – 해시값을 byte string으로 출력
- ▣ Hexdigest() – 해시값을 hex string으로 출력

□ 한글 등 국제어 입력

- ▣ Text string을 byte array로 바꾸는 함수 이용
- ▣ `a_byte_array = bytearray(a_string, "utf8")`
- ▣ Encode('utf-8') 메서드 활용

□ 제공되는 해시 알고리즘

Modern hash algorithms

- SHA-2 family
 - SHA-224
 - SHA-256
 - SHA-384
 - SHA-512, SHA-512/224, SHA-512/256
- SHA-3 family
 - SHA3-224
 - SHA3-256
 - SHA3-384
 - SHA3-512
- BLAKE2
 - BLAKE2s
 - BLAKE2b

Extensible-Output Functions (XOF)

- SHAKE (in the SHA-3 family)
 - SHAKE128
 - SHAKE256

□ > hash.py

```
from Crypto.Hash import SHA256

message1 = b'Hash Test Message 1'
message2 = b'Hash Test Message 2'
message3 = b'Hash Test Message 3'
message4 = '해시함수 테스트 1'

h = SHA256.new()
h.update(message1)
print(h.hexdigest())

h = SHA256.new()
h.update(message2)
print(h.hexdigest())

h = SHA256.new()
h.update(message3)
print(h.hexdigest())

h = SHA256.new()
h.update(message4.encode('utf-8'))
print(h.hexdigest())
```

```
C:\Users\bclee\Desktop\cp2021-back\python>python hash.py
e861c346a4552910541650470331f0aebca094640b2bf38fb6745390f1bb01c2
8196a014576667c4182bddc072ce8e489d58e316d3a40f6fef17bada500d093c
9daaab1a3c888a5ab21306bceb136170b416bd7982e1f222cc753d3a4937b991
1b0412892117012032530bf124c46b2690d58834179dd8a01d12b6dfea83319f
```