Cryptology / part II Keränen /Teeriaho (Ramk 2006)

7. Hash functions and MAC:s

MAC = Message Authetication Codes

7.1 Hash functions

Definition:

Hash -function is a *one way function* , which produces from the message a fixed length **hash value** (message digest). Hash -function is used to secure the **integrity** of the message.

Uses of hash functions

1. Message digest is created from the message using a hash function. The receiver of the message compares the message digest received with the message with the hash value he calculated directly from the message. It both are equal, the message is not changed.

2. Password hash.
For example in Unix operating system (also in Windows), the users passwords are not stored in the password file. Instead the hash values of passwords are stored. In Unix the hash - function is DES - block cipher.

Requirements of a godd hash function h(m)

1) Hash function h(m) must be *one way function.*. It should not be possible to calculate the message from its hash.

2) Knowing message M and its hash h(M), it should be impossible to generate another message M' , with the same hash.

3) It should be impossible to generate two messages M and M', with the same hash value

Security of hash - values

The most used hash functions SHA-1 and MD5 have been broken by the Chinese woman: professor Xiaoyun Wang during 2004 - 2005.

These hash functions are still used as if nothing happened. Some cryptographers recommend that MD5 and SHA-1 should not be used anymore. SHA-2 is still regarded more reliable.

(In password files it is catastrophic if a hacker is able to create an own valid password, that has the same hash value than your password)

Finding reliable hash functiont is one of the biggest problems in recent cryptography

■ 7.2 Some hash - functions

1) MD2, MD4 and MD5 are hash functions made by Ronald Rivest
The length of the hash in MD5 is 128 bits = 16 bytes.

2) SHA-1 (Secure Hash Algorithm) is NIST:s standard.
The length in SHA-1 is 160 bits = 20 bytes.

3) DES produces 64 bit password hash in CBC -mode

4) Rivest's newest Hash (6/2005): "Rivest - Abelian Square Free Dithering for Iterated Hash Functions" uses Veikko Keränen's Abelian square free 4 alphabet string"
C++ -code and description is found at Rivest's web site.

Idea is to prevent attacks by "salting" the message with parts of Keränen's stirng.

Ronald Rivest (1947 -) is one of the leading cryptographers . He is the R in RSA.
He has created among other achievements encryption algorithms RC2, RC4, RC5 and RC6 , and hash functions MD2, MD4 and MD6 .
He a professor in MIT in Boston, USA
Web-site : http://theory.lcs.mit.edu/~rivest/

For example Finnish state recommends SHA-1 and MD5 to be used as hash functions.
Minimum hash length is 128 bits.

■ 7.3 MAC functions

Definition

MAC creates a digest from message M and symmetric key K.

MAC provides both **interity and authentication.**

HMAC - functions are most used MAC's

Definition:

HMAC uses a traditional hash function and symmetric key to create a message digest.

HMAC formula:

$$HMAC(m , K) = h (K \oplus opad) \| h (K \oplus ipad) \| m))$$

m = message
K = symmetric key
h = hash algorithm (SHA or MD5)
opad = constant = 5C5C5C...5C (hex number 5C repeated 16 times)
ipad = constant = 363636..36 (hex number 36 repeated 16 times)
$\|$ means concatenation of strings
$\oplus$ means XOR addition

In HMAC the key K and ipad are XOR:red and the result is placed before the message . Then you calculate the hash value.
A block obtained from opad and K is placed before the result of the previous step and hash is calculated again.

8 Digital signature

■ 8.1 Goals of digital signature

Definition:

Digital signature is an algoritm which provides **the integrity and authentity** of the message

Digitally signed document is legally as valid as a document signed in the presence of witnesses.

■ 8.2 Principle of digital signature

Sender A:

1. **A calculate a hash value** (message digest) from the message .

2. The hash value is send with the message encrypted with the **sender A'a private key.**

Receiver B:

3. B **uses sender's** public key to open the signature and obtains the hash value.
4. B **calculates** a hash value directly from the message with the same hash function.
5. B **compares** both **hash values**

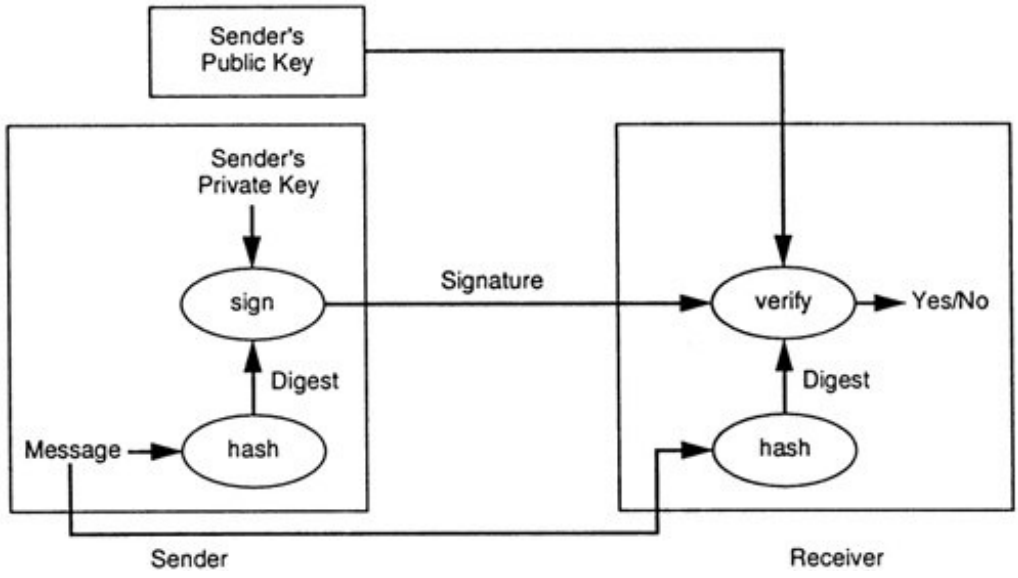
If the hahs values match , B is sure that

A) message is unchanged

B) message is sent by A, because it was opened with A's public key.

Integrity and authentity are confirmed.

Figure



* For digital signatures we need some hash function (SHA, RD5) + a public key cipher RSA , Elgamal or ECC.

* In USA one standard signature algorithm is DSA

* When using digital signature, the message and the signature can be encrypted , but this is not necessary

■ 8.3 Digital RSA - signature without encryption

Sender:	
1. Calculate the message digest	S = hash(m)
2. Calculate the signature	$S' = S^d \mod n$
Receiver:	
3. Receiver opens the signature	$S = S'^e \mod n$

Here d is sender's private key , e and n are his public keys.

■ 8.4 Digital RSA - signature with encryption

1. Sender A calculate the hash	S = hash(m)
2. Calculation of signature + encryption	$S' = (S^{d_A} \mod n_A)^{e_B} \mod n_B$
3. B opens the signature	$S = (S'^{d_B} \mod n_B)^{e_A} \mod n_A$

Note! If $n_A > n_B$, the keys are used in reverse order:

$S' = (S^{e_B} \mod n_B)^{d_A} \mod n_A$ (order reversed)

$S = (S'^{e_A} \mod n_A)^{d_B} \mod n_B$

■ Example of a digital signature with RSA *(no encryption)*

Here we create a simple hash which uses rotation of bits and calculation a remainder mod $2^{16}-1$.

```
myhash[bits_] := Module[{n},
  n = 2^16 - 1;
  Mod[FromDigits[RotateLeft[bits, 2], 2], n]
 ]
```

Let us try this hash for a 100 bit message

```
bits = Table[Random[Integer, 1], {i, 1, 100}]

{0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0}
```

```
myhash[bits]

45663
```

Assume the Alice has following RSA keys.

```
p = 1987; q = 2309;
{n = p * q, e = 33227}  ;
d = PowerMod[e, -1, (p - 1) * (q - 1)];
```

Alice calculate the digital signature:

```
sig = PowerMod[myhash[bits], d, n]

3357217
```

Bob opens the signature using Alice's public exponent e

```
opened = PowerMod[sig, e, n]

45663
```

Bob has received the message and calculates the hash directly from the message and compares the two hash values:

```
myhash[bits] == opened

True
```

Message is not changed and it is from Alice.

■ 8.5 Digital signature standard DSS

digital signature standard

NIST took in 1991 as a US digital signature standard an algorithm called DSS which uses SHA-1 hash function.

In this presentation we are not going into details of DSS.

Actually the algorithm of the signature is called DSA. Combination of DSA and SHA-1 is called DSS.