CSCI 7000-001 — Lecture n. 10

October 2, 2001

1 Outline

- non-repudiation
- pretty good privacy with PGP (part 2)
- key management

2 An additional benefit of public-key cryptography

Consider these two types of MAC:

hash based Alice and Bob share a secret key K. Alice writes a message *msg* and attaches a MAC computed as follows:

$$\text{MAC} = \mathit{Alice}, H(\mathit{msg}, K)$$

Bob can verify that the message is authentic (*integrity*) by computing a MAC using the same formula and comparing it with Alice's MAC.

public-key based Bob knows Alice's public key P_A. Alice writes a message *msg* and attaches a MAC computed as follows:

$$\text{MAC} = \mathit{Alice}, S_A(H(\mathit{msg}))$$

Bob (and anyone having Alice's public key) can verify that the message is authentic by extracting the hash value H' using Alice's public key, $H' = P_A(\text{MAC})$, and comparing H' with the hash of the message $H(msg)$.

Now, why would you want to use a public-key based MAC, given that it is computationally more expensive?

Because public-key based MACs have a desirable advantage over MACs based on shared secret keys (hash- or CBC-based)...

3 Non repudiation

The advantage is called *non repudiation*: Alice can not deny that she created and signed the message.

How can she argue that she didn't do that, in the case of a hash-based MAC?

She can argue that Bob did it! After all, they share the same secret key K, and in fact Bob verifies the MAC using the exact same algorithm that Alice uses to create the MAC in the first place.

4 We have seen how to use PGP to...

- encrypt/decrypt data using a symmetric cipher

  ```
  pgp -c secret_info.txt
  ...
  pgp secret_info.txt.pgp
  ```

- create a key pair

  ```
  pgp -kg
  ...
  pgp secret_info.txt.pgp
  ```

- sign and verify messages

  ```
  pgp -s message.txt
  ...
  pgp message.txt.pgp
  ```

- attach a MAC (essentially by signing a message in plaintext)

```
pgp -st message.txt
pgp -sta message.txt
```

- encrypt a message with Paolo's public key

```
pgp -e message.txt Paolo
pgp -ea message.txt Paolo
```

- encrypt and sign a message

```
pgp -es message.txt Paolo
pgp -esa message.txt Paolo
```

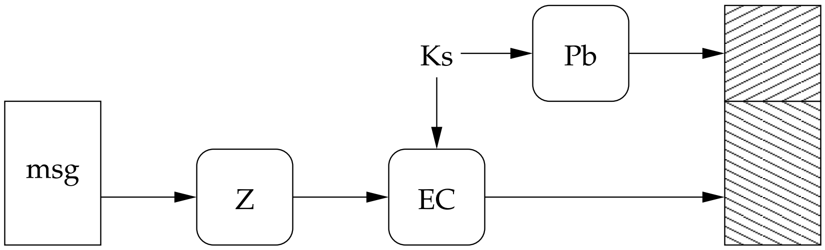
Now we will look a bit closer at PGP, and how it actually works.

5 Ingredients of PGP

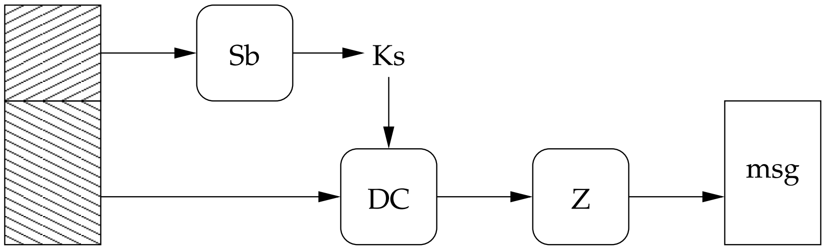
- state-of-the-art cryptographic functions as building blocks:
  - *hash:* SHA1
  - *symmetric:* IDEA, CAST-128, or 3DES
  - *public-key:* RSA, DSS (ElGamal)
- good combination of other functions:
  - compression (ZIP)
  - e-mail friendly encoding output (radix-64)
- ported on several platforms
- freely available
- source code also available
- commercial version, low-cost and fully compatible, for whoever wants support (provided by Viacryps, now Network Associates)
- developed independently of any governmental or standards organization

6 Encryption for confidentiality

PGP uses conventional ciphers to encrypt messages, using a one-time session key. The session key is then encrypted with the recipient's public-key:

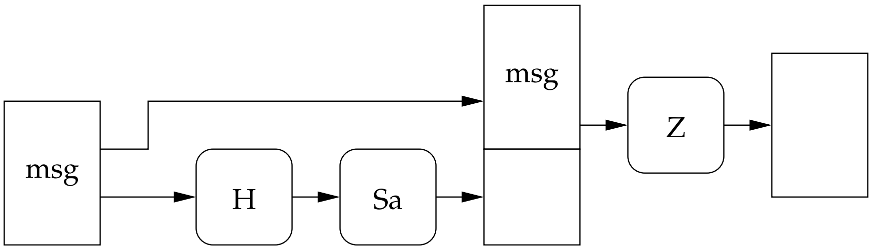


7 Decrypting messages



8 Public-key signature for integrity and authentication

PGP signatures are essentially public-key MACs:



Notice the position of the compression function.

9 Key ID

As seen above, a session key is encrypted with the recipient's public-key. Since every party can have multiple key pairs, this raises the issue of identifying keys.

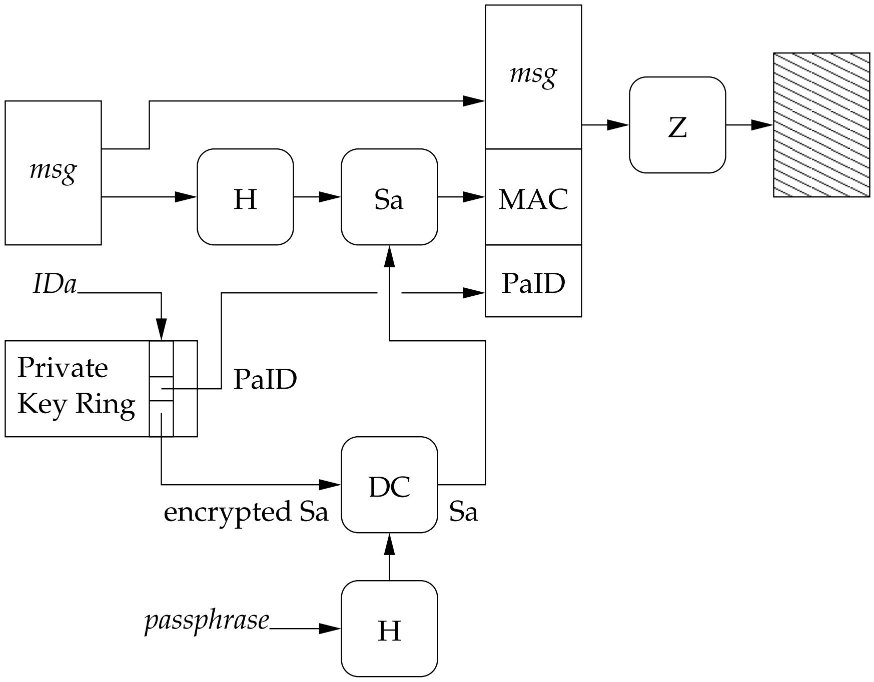
In other words, Bob may have more than one public/private key pair, therefore he needs to know which public key Alice used to encrypt.

One solution would be to attach that public key to the message, however public keys are quite long, so PGP uses the leftmost 64 bits of a public key as a key identifier:

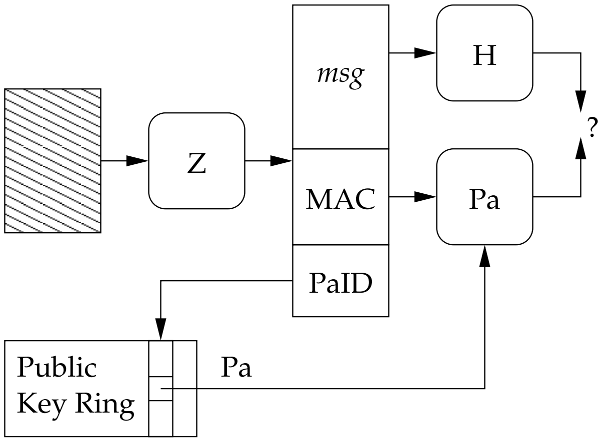
$$ID(P_B) = P_B \mod 2^{64}$$

Notice that such IDs are only *probabilistically unique*.

10 Public-key signature (more accurate picture)



11 Verification



12 Practical weaknesses of PGP (and public-key system in general)

Public-key management is the most important weakness of PGP, and other public-key cryptosystems.

In order for Alice to communicate securely with Bob, Alice must securely obtain and store Bob's public key (and vice-versa).

Of course, keys can be signed, but ultimately the management of keys reduces to the annotation and management of *trust*.

13 Trust management

PGP does not specify how to establish certifying authorities, nor does it establish CAs itself (nor does any company or organization that distributes PGP software).

However, PGP provides means to:

- associate trust rating to keys
- use trust ratings
- transfer trust ratings

14 Public-key ring

timestamp	**key id**	**public key**	**owner trust**	**user id**	**key legitimacy**	**signatures**	**signature trust**
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
T_i	$PU_i \mod 2^{64}$	PU_i	trustflags$_i$	U_i	trustflags$_i$	$\dots$ $\dots$	$\dots$ $\dots$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

15 Trust information

Trust information is assigned to:

- public-key owner, or *ownertrust*: degree to which you trust the owner of this public-key to certify other keys
- (public-key, user-id) pair, or *keylegit*: key legitimacy indicates the degree to which you trust that this key is a valid public-key for this user
- signature, or *sigtrust* (cached copy of *ownertrust*): degree to which you trust this signature

16 Assigning and maintaining trust information

When Alice adds a new public key P_X in her public-key ring, PGP assigns an *ownertrust* value as follows:

- if X is also present in Alice's secret key, then PGP assigns a value of *ultimate trust*
- otherwise PGP asks Alice for her assessment. Alice can specify one of the following values:
  - unknown
  - untrusted
  - marginally trusted
  - completely trusted

Several signatures (certificates) can be attached to a key P_X. When Alice adds a signature $S_Y(P_X)$ for key P_X, PGP checks whether the signer is rated in Alice's public-key ring. If Y is known, Y's *ownertrust* is associated with the signature $S_Y(P_X)$.

The *keylegit* value for a key P_X is *ultimate trust* if at least one of its signatures has an *ultimate sigtrust* value. Otherwise *keylegit* is computed as a weighted sum of the *sigtrust* values of its certificates.

17 Assigning and maintaining trust information

An example of a set of trust rating relations: