Message Authentication and Digital Signature

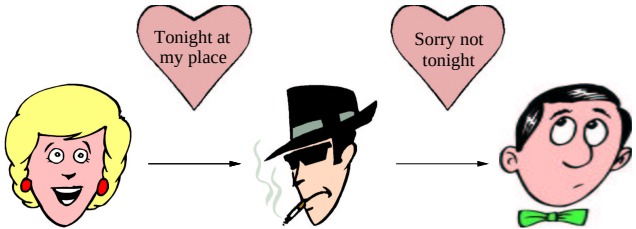
Alessandro Armando

Computer Security Laboratory (CSec)
DIBRIS, Università di Genova

Computer Security

Outline

Message Integrity



Message authentication is concerned with:

- protecting the integrity of a message
- validating identity of originator
- non-repudiation of origin (dispute resolution)

Authentication Functions

Any message authentication or digital signature mechanism relies on an authentication function to generate an *authenticator*, i.e. a value used to authenticate a message.

We will consider the following authentication funtions:

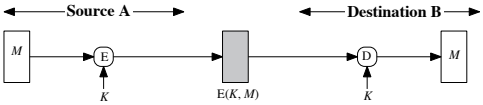
Message encryption. The ciphertext of the entire message serves as its authenticator.

Message Authentication Code. A function of the message and a secret key that produces a fixed-length value that serves as authenticator.

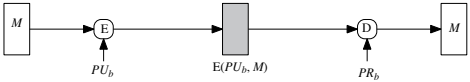
Cryptographic Hash function. A function that maps a message of any length into a fixed-length hash value, which serves as authenticator.

Outline

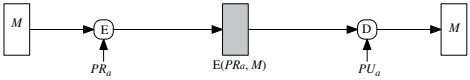
Message Encryption



(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality

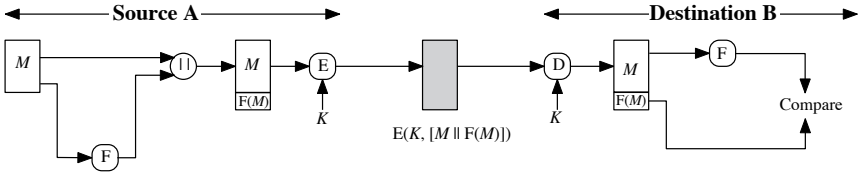


(c) Public-key encryption: authentication and non-repudiation

Error Control with Message Encryption

For the previous approaches to success there must be some automatic means to determine if incoming ciphertext decrypts to intelligible plaintext.

One solution to this problem it to give the plaintext some structure that is easily recognised but that cannot be replicated without recourse to the encryption function, e.g. by appending a checksum to the massage before encryption.



(a) Internal error control

Table 11.1 Confidentiality and Authentication Implications of Message Encryption (see Figure 11.1)

A → B: E(K, M)
- •Provides confidentiality
  - —Only A and B share K
- •Provides a degree of authentication
  - —Could come only from A
  - —Has not been altered in transit
  - —Requires some formatting/redundancy
- •Does not provide signature
  - —Receiver could forge message
  - —Sender could deny message

(a) Symmetric encryption

A → B: E(PU_b, M)
- •Provides confidentiality
  - —Only B has PR_b to decrypt
- •Provides no authentication
  - —Any party could use PU_b to encrypt message and claim to be A

(b) Public-key (asymmetric) encryption: confidentiality

A → B: E(PR_a, M)
- •Provides authentication and signature
  - —Only A has PR_a to encrypt
  - —Has not been altered in transit
  - —Requires some formatting/redundancy
  - —Any party can use PU_a to verify signature

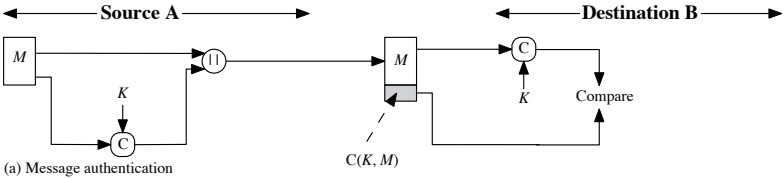
(c) Public-key encryption: authentication and signature

A → B: E(PU_b, E(PR_a, M))
- •Provides confidentiality because of PU_b
- •Provides authentication and signature because of PR_a

Outline

Message Authentication Code

- A MAC function accepts a variable-size message M and a secret key K as input and produces a fixed-size output $C(M, K)$.
- It is a many-to-one function as potentially many messages have same MAC but finding these is very difficult
- $C(M, K)$ is called *message authentication code* (MAC) or *cryptographic checksum*.



(a) Message authentication

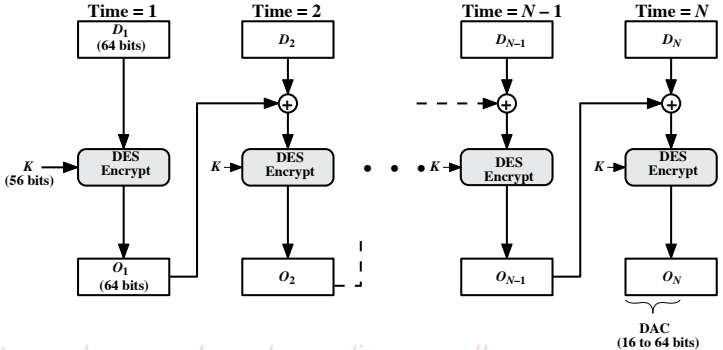
(b) Message authentication and confidentiality; authentication tied to plaintext

(c) Message authentication and confidentiality; authentication tied to ciphertext

Figure 11.4 Basic Uses of Message Authentication Code (MAC)

Data Authentication Algorithm

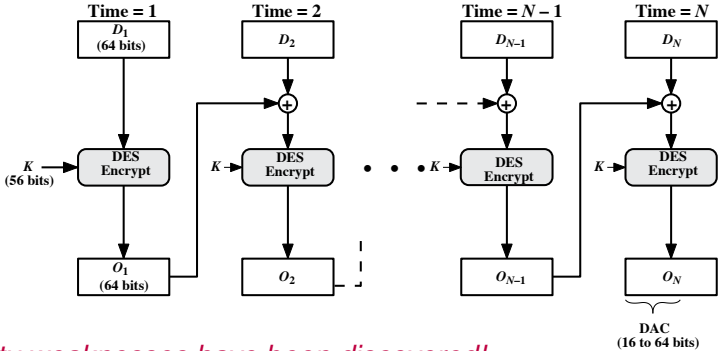
- Based on DES
- One of the most widely used MACs for a number of years
- Input message split in 64 bit chunks $M = D_1 D_2 \cdots D_N$



- But, *security weaknesses have been discovered!*

Data Authentication Algorithm

- Based on DES
- One of the most widely used MACs for a number of years
- Input message split in 64 bit chunks $M = D_1 D_2 \cdots D_N$

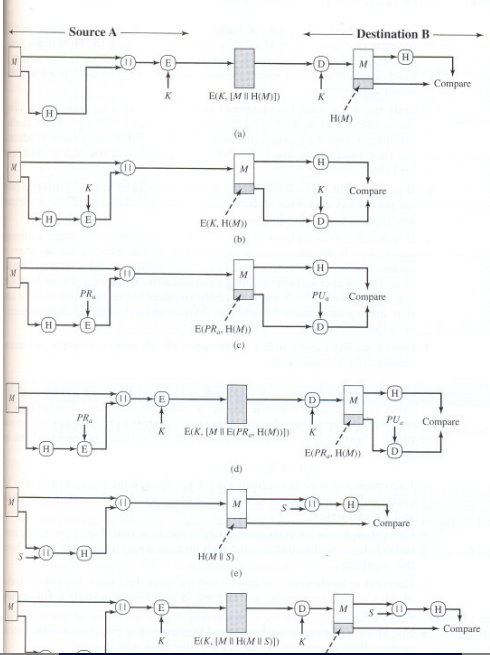


- But, *security weaknesses have been discovered!*

Outline

Cryptographic Hash functions

- As with MAC, a hash function accepts a variable-size message M as input and produces a fixed-size output $H(M)$.
- Unlike MAC, a hash function does not use a key.
- $H(M)$ is called *hash code* (but also *message digest*).


Source A
Destination B
M
||
E
K
E(K, [M || H(M)])
D
H
Compare
H(M)
(a)
E(K, H(M))
(b)
PR_a
E(PR_a, H(M))
PU_a
(c)
E(K, [M || E(PR_a, H(M))])
(d)
S
H(M || S)
(e)
E(K, [M || H(M || S)])

Requirements for a cryptographic hash function

The purpose of a hash function is to produce a “fingerprint” of a file, message, or other block of data.

To be useful a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given x.
4. **(One-way property)** For any given value y, it is computationally infeasible to find x such that $H(x) = y$.
5. **(Weak collision resistance** or **pre-image resistance)** For any value x, it is computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$.
6. **(Strong collision resistance** or **2nd pre-image resistance)** It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

Requirements for a cryptographic hash function

The purpose of a hash function is to produce a “fingerprint” of a file, message, or other block of data.

To be useful a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given x.
4. **(One-way property)** For any given value y, it is computationally infeasible to find x such that $H(x) = y$.
5. **(Weak collision resistance** or **pre-image resistance)** For any value x, it is computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$.
6. **(Strong collision resistance** or **2nd pre-image resistance)** It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

Requirements for a cryptographic hash function

The purpose of a hash function is to produce a “fingerprint” of a file, message, or other block of data.

To be useful a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given x.
4. **(One-way property)** For any given value y, it is computationally infeasible to find x such that $H(x) = y$.
5. **(Weak collision resistance** or **pre-image resistance)** For any value x, it is computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$.
6. **(Strong collision resistance** or **2nd pre-image resistance)** It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

Requirements for a cryptographic hash function

The purpose of a hash function is to produce a “fingerprint” of a file, message, or other block of data.

To be useful a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given x.
4. **(One-way property)** For any given value y, it is computationally infeasible to find x such that $H(x) = y$.
5. **(Weak collision resistance** or **pre-image resistance)** For any value x, it is computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$.
6. **(Strong collision resistance** or **2nd pre-image resistance)** It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

Requirements for a cryptographic hash function

The purpose of a hash function is to produce a “fingerprint” of a file, message, or other block of data.

To be useful a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given x.
4. **(One-way property)** For any given value y, it is computationally infeasible to find x such that $H(x) = y$.
5. **(Weak collision resistance** or **pre-image resistance)** For any value x, it is computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$.
6. **(Strong collision resistance** or **2nd pre-image resistance)** It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

Requirements for a cryptographic hash function

The purpose of a hash function is to produce a “fingerprint” of a file, message, or other block of data.

To be useful a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given x.
4. **(One-way property)** For any given value y, it is computationally infeasible to find x such that $H(x) = y$.
5. **(Weak collision resistance** or **pre-image resistance)** For any value x, it is computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$.
6. **(Strong collision resistance** or **2nd pre-image resistance)** It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

Applications

- One-way property is important in message authentication techniques involving the use of a secret value, e.g. in

$$A \rightarrow B : M \| H(M \| S) \qquad \text{where } S \text{ is a secrect between } A \text{ and } B$$

- Weak collision resistance prevents forgery when an encrypted hash code is used, e.g. in

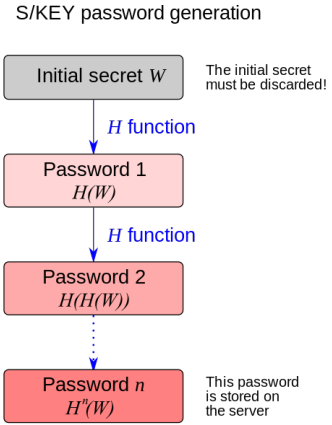
$$A \rightarrow B : M \| E(K, H(M)) \qquad A \rightarrow B : M \| E(PR_a, H(M))$$

- Weak collision resistance also useful to protect password files
  - For password p, store $H(p)$ in password file.
  - Often combined with *salt* s, i.e. store pair $(s, H(s \| p))$, to protect against *dictionary attacks*.

    (Examples of "useful" dictionaries can be found here `https://wiki.skullsecurity.org/Passwords`)
- Strong collision resistance useful against the *birthday attack*.

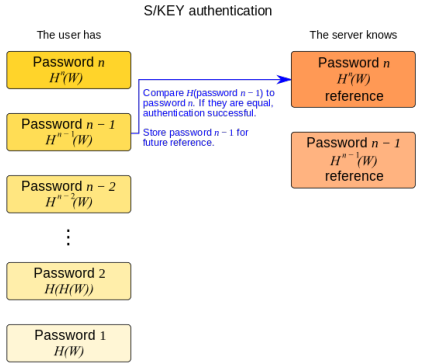
S/Key

- One-time password system developed from dumb terminals and untrusted public computers that does not require to type a long-term password.
- Because each password is only used once, they are useless to password sniffers.
- Passwords are printed or computed by portable device (secure element)



S/Key

- The user provides pwd_i for $i = n - 1, ..., 1$.
- The server computes $H(pwd_i)$ and compares the result to pwd_{i+1}, which is stored as reference on the server.
- Notice, the first password (pwd_n) is discarded by the user.



Birthday paradox

- How many people must be in a room such that the probability p that one has your birthday is $p > 0.5$?
- How many people must be in a room such that the probability p that any two share the same birthday is $p > 0.5$?
- **Generalization:** Let H have 2^m possible outputs. H must be applied to $2^{m/2}$ inputs so that the probability of a collision is greater than 0.5.

Birthday paradox

- How many people must be in a room such that the probability p that one has your birthday is $p > 0.5$?
  The probability that one of n people has your birthday is $n/365$, so $p > 0.5$ for $n \geq 183$.

- How many people must be in a room such that the probability p that any two share the same birthday is $p > 0.5$?

- **Generalization:** Let H have 2^m possible outputs. H must be applied to $2^{m/2}$ inputs so that the probability of a collision is greater than 0.5.

Birthday paradox

- How many people must be in a room such that the probability p that one has your birthday is $p > 0.5$?
  The probability that one of n people has your birthday is $n/365$, so $p > 0.5$ for $n \geq 183$.

- How many people must be in a room such that the probability p that any two share the same birthday is $p > 0.5$? 23 persons!

- **Generalization:** Let H have 2^m possible outputs. H must be applied to $2^{m/2}$ inputs so that the probability of a collision is greater than 0.5.

Birthday paradox

- How many people must be in a room such that the probability p that one has your birthday is $p > 0.5$?
  The probability that one of n people has your birthday is $n/365$, so $p > 0.5$ for $n \geq 183$.

- How many people must be in a room such that the probability p that any two share the same birthday is $p > 0.5$? 23 persons!

- **Generalization:** Let H have 2^m possible outputs. H must be applied to $2^{m/2}$ inputs so that the probability of a collision is greater than 0.5.

Birthday paradox – proof

The number of ways N we can have k values in the range $1..n$ with no duplicates:

$$N = n \times (n-1) \times \ldots \times (n-k+1) = \frac{n!}{(n-k)!}$$

There are n^k ways to select k items from $1..n$ allowing repetitions. So probability of no duplicates when selecting k items from same range is

$$Q(n,k) = \frac{n!}{(n-k)! \times n^k}$$

Hence, $P(n,k)$ = probability of at least one duplicate in k items is

$$P(n,k) = 1 - Q(n,k) = 1 - \frac{n!}{(n-k)! \times n^k}$$

Can compute $P(365, 23) = .5073$.
In general duplicate encountered in $O(\sqrt{n})$. If $n = 2^m$, then $O(2^{m/2})$

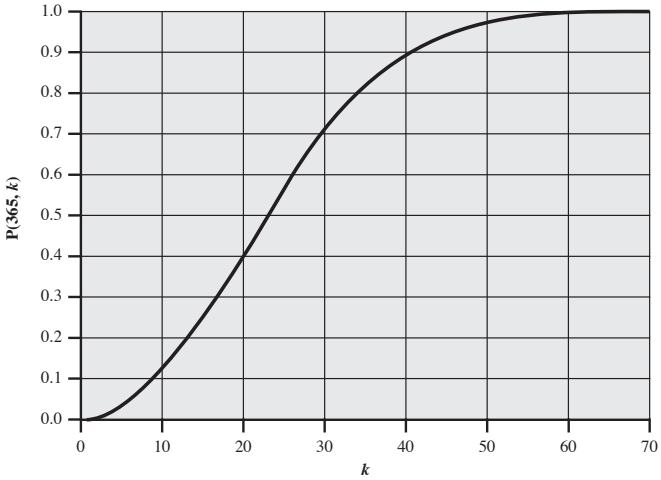


Figure 11.10 The Birthday Paradox

A birthday attack

- One might think a 64 bit hash code is secure. Preimage resistance means, on average 2^{63} messages must be tried.
- Birthday attack for finding collisions
  - Suppose A is willing to sign a contract with B. Then B prepares version x, good for A, and version y, which bankrupts her.
    - B generates "good" messages $x_1, \ldots$.
    - Similarly he dovetails the generation of the "bad" $y_1, \ldots$.
    - He stops when $h(x_i) = h(y_j)$.
  - A signs $h(x_i)$. Later B uses signature for y_j.
- On average, work required is order of 2^{32}.
  So double your hash size if collision avoidance important!

A letter in 2^{37} variations

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}

Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}

Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}

responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}

in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}

to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the

market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the

{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}

of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}

as proof of identity. An order with his signature, which is {appended / attached}

{authorizes / allows} you to charge the cost to this company at the {above / head office}

address. We {fully / --} expect that our {level / volume} of orders will increase in

the {following / next} year and {trust / hope} that the new appointment will {be / prove}

{advantageous}

Constructing a cryptographic hash function

- General scheme proposed by Merkle
- Divide message into fixed size blocks $M = Y_0, \ldots Y_{L-1}$ and apply

$$
\begin{aligned}
CV_0 &= IV \\
CV_i &= f(CV_{i-1}, Y_{i-1}) \\
&\quad \text{for } i = 1, \ldots, L \\
H(M) &= CV_L
\end{aligned}
$$

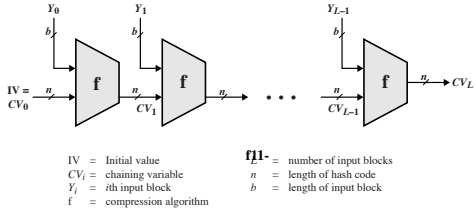


Figure 11.9 General Structure of Secure Hash Code

- **MD5** (Rivest): 128 bit hashes.
  Known weakness.
- **SHA** (US Standard): 160 bit hashes.
  Assumed secure.

Outline

The digital signature problem

- Message authentication protects two parties who exchange messages from any third party.
- However, *it does not protect the two parties against each other!*
- Several forms of dispute between the two are possible.
- For example, suppose A sends an authenticated message to B using a scheme for message authentication shown on slide 11. The following disputes could arise:
  - B may forge a different message and claim that it came from A.
  - A can deny sending the message. Because it is possible for B to forge a message, there is no way to prove that A did in fact send the message.

High-level requirements on a digital signature mechanism

A digital signature must

- provide the means to verify the author and the date and time of the signature
- authenticate the contents of at the time of the signature
- be verifiable by third parties, to resolve dispute

Requirements on a digital signature mechanism

- A digital signature must be a bit pattern that depends on the message being signed.
- A digital signature must use some information unique to the sender, to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognise and verify the digital signature.
- It must be computationally unfeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudolent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

Outline

Direct Digital Signature

- The destination must know the public key of the source.
- The signature is formed either
  - by encrypting the entire message with the sender's private key as in Figure (c) on slide 6
    or
  - by encrypting the hash code of the message with the sender's private key as in Figure (c) on slide 16.
- Security of the schema depends on security of the sender's private key.
- Every signed message should contain a timestamp (date and time) and compromised keys should be promptly reported to central authority.
- But, an opponent can steal a private key at time T and then forge messages stamped with a time before T ...

Outline

Arbitrated Digital Signature

The problem associated with direct digital signatures can be addressed by using an arbiter (trusted third party).

Table 13.1 Arbitrated Digital Signature Techniques

(1) X → A: $M \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(M)])$
(2) A → Y: $\mathrm{E}(K_{ay}, [ID_X \| M \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(M)]) \| T])$

(a) Conventional Encryption, Arbiter Sees Message

(1) X → A: $ID_X \| \mathrm{E}(K_{xy}, M) \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(\mathrm{E}(K_{xy}, M))])$
(2) A → Y: $\mathrm{E}(K_{ay}, [ID_X \| \mathrm{E}(K_{xy}, M)]) \| \mathrm{E}(K_{xa}, [ID_X \| \mathrm{H}(\mathrm{E}(K_{xy}, M)) \| T])$

(b) Conventional Encryption, Arbiter Does Not See Message

(1) X → A: $ID_X \| \mathrm{E}(PR_x, [ID_X \| \mathrm{E}(PU_y, \mathrm{E}(PR_x, M))])$
(2) A → Y: $\mathrm{E}(PR_a, [ID_X \| \mathrm{E}(PU_y, \mathrm{E}(PR_x, M)) \| T])$

(c) Public-Key Encryption, Arbiter Does Not See Message

Notation:
X = sender M = message

Outline

Summary of PGP Services

Table 15.1 Summary of PGP Services

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed, for storage or transmission, using ZIP.
Email compatibility	Radix 64 conversion	To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix 64 conversion.
Segmentation	—	To accommodate maximum message size limitations, PGP performs segmentation and reassembly.

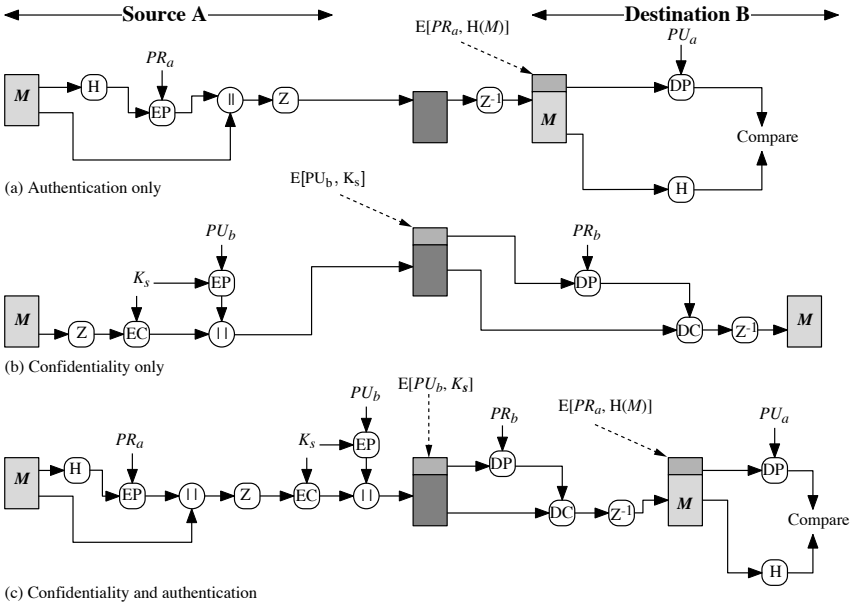
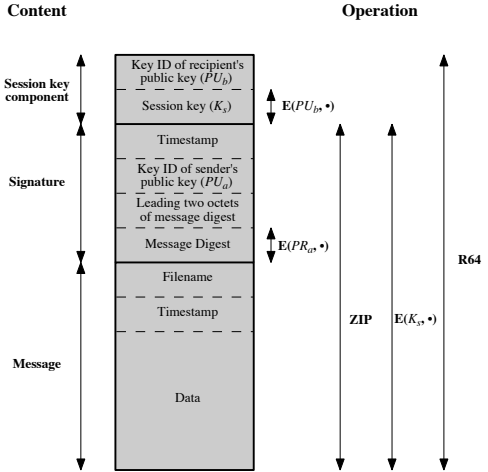


Figure 15.1 PGP Cryptographic Functions



Notation:
E(PU_b, •) = encryption with user b's public key
E(PR_a, •) = encryption with user a's private key
E(K_s, •) = encryption with session key

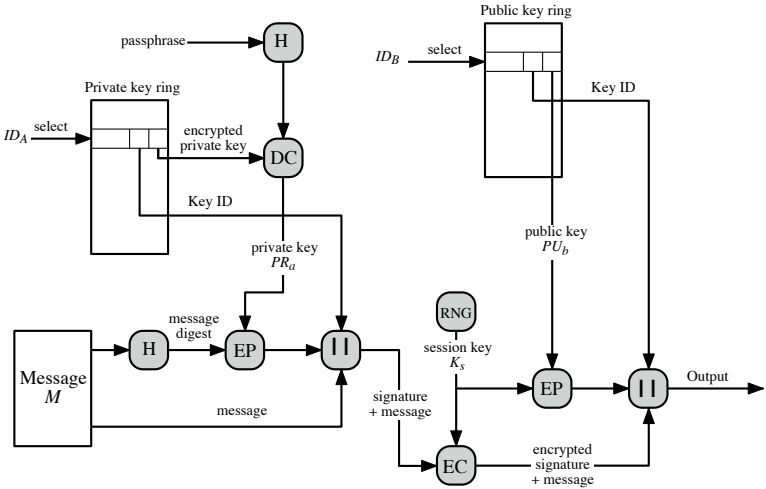


Figure 15.5 PGP Message Generation (from User A to User B; no compression or radix 64 conversion)

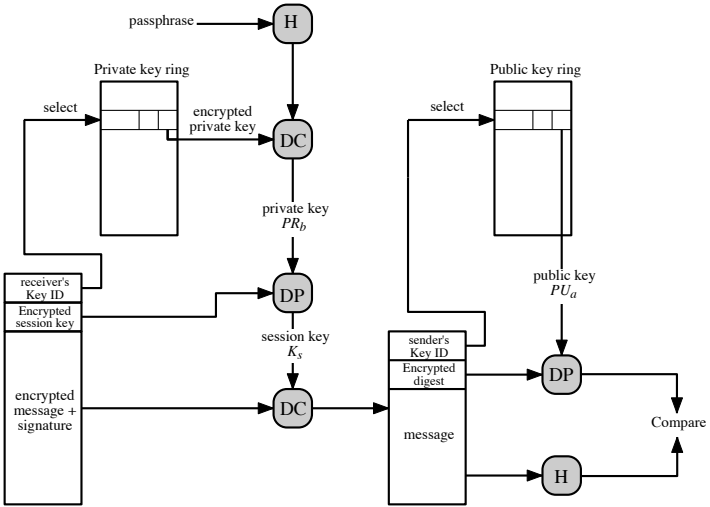


Figure 15.6 PGP Message Reception (from User A to User B; no compression or radix 64 conversion)