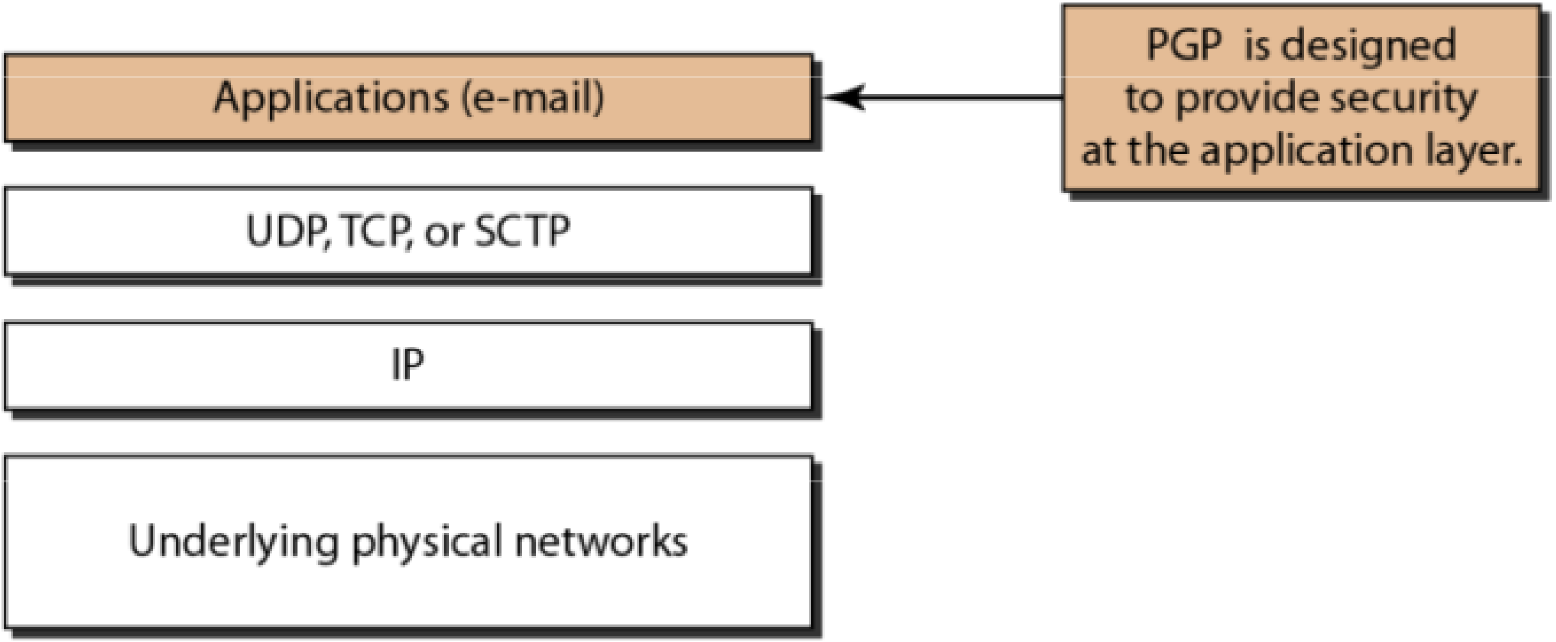
Module 5

Network Security and Applications

Pretty Good Privacy (PGP)



Cryptographic Keys

- PGP uses four types of keys:
  - One-time session symmetric key
  - Public keys
  - Private Keys
  - Passphrase based symmetric keys

Cryptographic Keys

- **Session Key Generation**
  - Each session key is associated with a a single message and is used only for the purpose of encryption or decryption
- **Key Identifiers**
  - The session key is encrypted with the recipient’s public key
  - Only the recipient will be able to recover the session key and hence the message will also be recovered by him

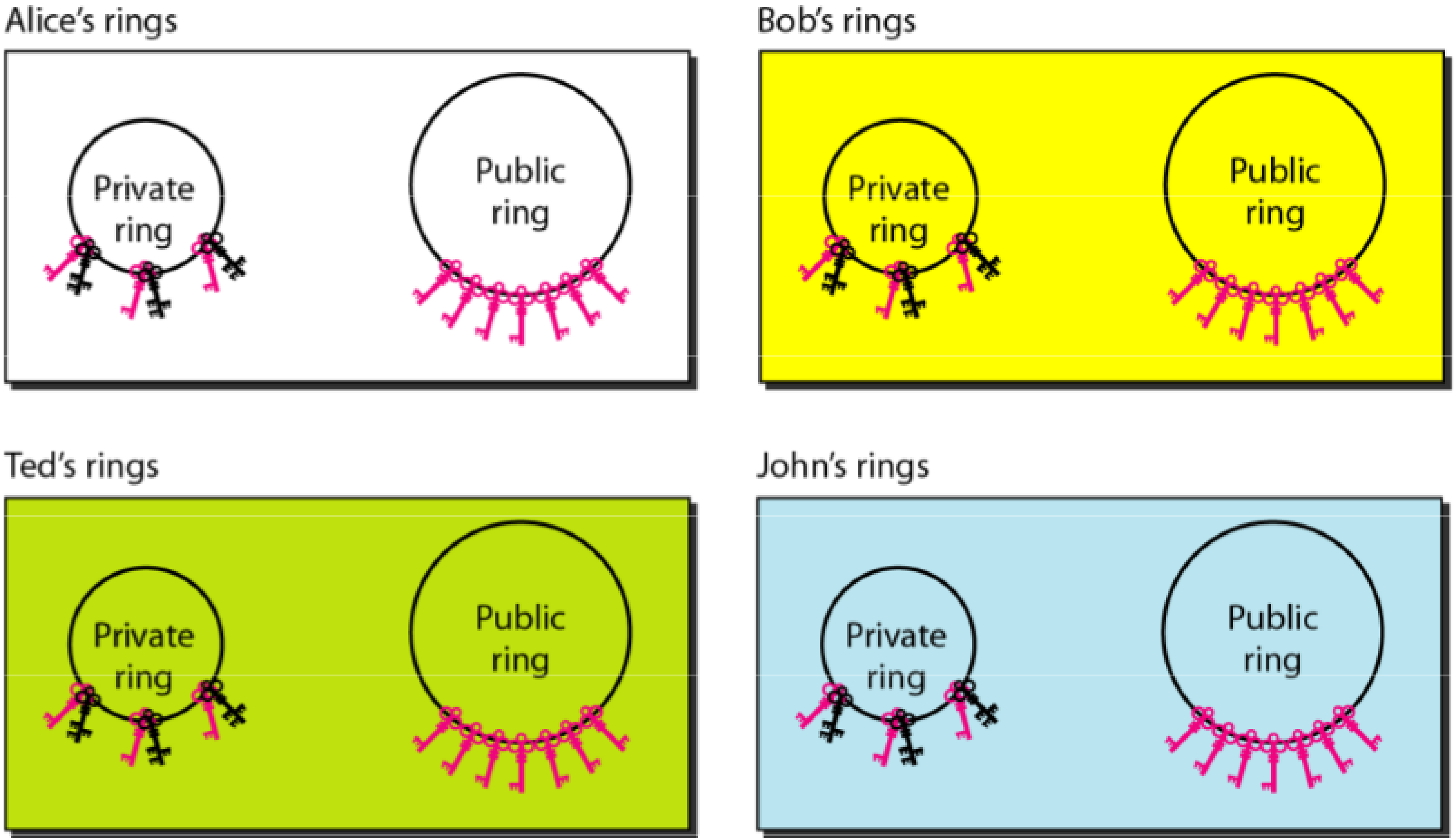
Cryptographic Keys

- A user may have multiple public/private key pairs
- How does the recipient know which of its public key was used to encrypt the session keys?
- Solution- Assign a key ID to each public key which is unique to each user id.
- The key ID associated with each public key consists of its least significant 64 bits
- PGP digital signature also uses this key ID

Key Rings

- The keys need to be stored and organized in a systematic way for efficient and effective use by all parties
- The scheme of organization used by PGP is a pair of data structures used at each node.
- One stores the public/private key pairs owned by that node
- The other one stores the public keys of other users known at this node
- These data structures are known as private key ring and public key ring

PGP-Key Ring


Alice's rings
Private ring
Public ring
Bob's rings
Private ring
Public ring
Ted's rings
Private ring
Public ring
John's rings
Private ring
Public ring

PGP- Key Rings

- Each PGP user has a pair of key rings:
- Public-key ring contains all the public-keys of other PGP users known to this user, indexed by key ID
- Private-key ring contains the public-private key pair(s) for this user, indexed by key ID & encrypted keyed from a hashed passphrase

PGP-Key Ring

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
Ti	PU_i mod 2^{64}	PU_i	$E(H(P_i), PR_i)$	User i
• • •	• • •	• • •	• • •	• • •

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
Ti	PU_i mod 2^{64}	PU_i	trust_flag$_i$	User i	trust_flag$_i$		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = field used to index table

Private Key Ring

- The ring can be viewed as a table in which each row represents one of the public/private pairs owned by the user
- Each row contains:
  - *Timestamp:- The date/time when the key pair was generated*
  - *Key ID:- The least significant 64 bits of public key for this entry*
  - *Public key:- The public key portion of the pair*
  - *Private key:- The private key portion of the pair(this field is encrypted)*
  - *User ID:- The user’s email address*

Private Key Ring

- The private key ring can be indexed either by User ID Key ID
- The private key ring is stored only in the machine of the user who created and owns the key pairs
- The private key itself is not stored in the key ring, instead it is encrypted.
- For encryption, passphrase is used
- When a user wants to access the private key ring to retrieve a private key, he must supply the passphrase.
- PGP will retrieve the encrypted private key, generate the hash code of the passphrase and decrypt the encrypted private key using CAST-128 with the hash code

Private Key Ring-Storing of keys

- Steps:-
    1. User selects passphrase to be used for encrypting private keys
    2. When the system generates a new public/private key key pair using RSA, it asks for the passphrase
    3. Using SHA-1, a 160 bit hash code is generated from the passphrase and it is discarded
    4. The system encrypts the private key using CAST-128 with the 128 bits of the hash code as the key
    5. The hash code is then discarded and encrypted private key is stored in the private key ring

Public Key Ring

- The data structure is used to store the public keys of other users that are known to this user
- The public key ring can be indexed by either User ID or Key ID
- Each row contains:
  - *Timestamp:- The date/time when this entry was registered*
  - *Key ID:- The least significant 64 bits of public key for this entry*
  - *Public key:- The public key for this entry*
  - *User ID:- Identifies the owner of this key. Multiple user IDs may be associated with a single public key*

Public Key Ring

- The ring can be viewed as a table in which each row represents one of the public/private pairs owned by the user
- The private key ring can be indexed either by User ID Key ID
- The private key ring is stored is stored only in the machine of the user who created and owns the key pairs
- The private key itself is not stored in the key ring, instead it is encrypted.

PGP Operational Services

- Authentication
- Confidentiality
- Compression
- E-mail Compatibility

PGP-Authentication (Steps)

- Sender creates a message
- SHA-1 is used to generate 160-bit hash code of message
- Hash code is encrypted with RSA using the sender's private key, and result is attached to message
- Receiver uses RSA with sender's public key to decrypt and recover hash code
- Receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic

PGP-Confidentiality(steps)

- Sender generates message and random 128-bit number to be used as session key for this message only
- Message is encrypted, using CAST-128 / IDEA/3DES with session key
- Session key is encrypted using RSA with recipient's public key, then attached to message
- Receiver uses RSA with its private key to decrypt and recover session key
- Session key is used to decrypt message

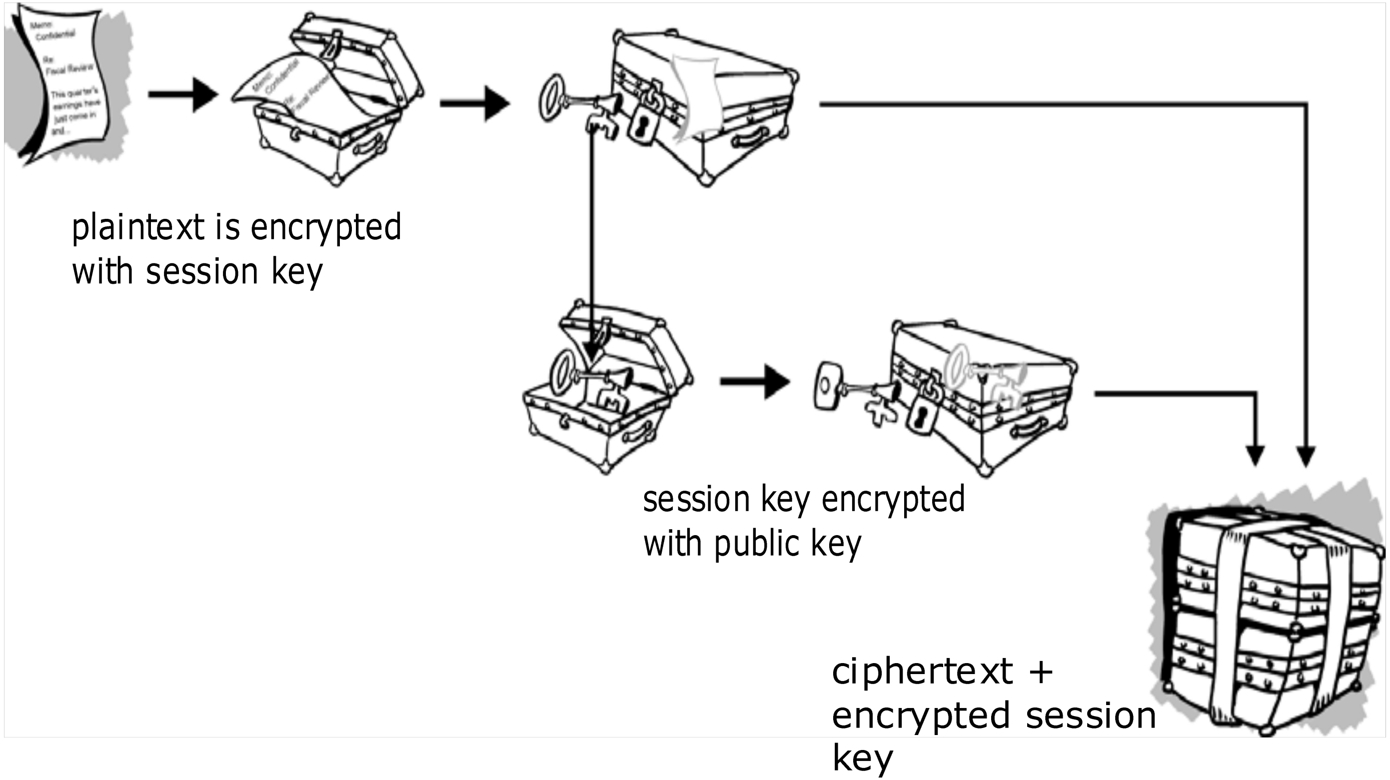
PGP-Confidentiality and Authentication

- Uses both services on same message
  - create signature & attach to message
  - encrypt both message & signature
  - attach RSA encrypted session key

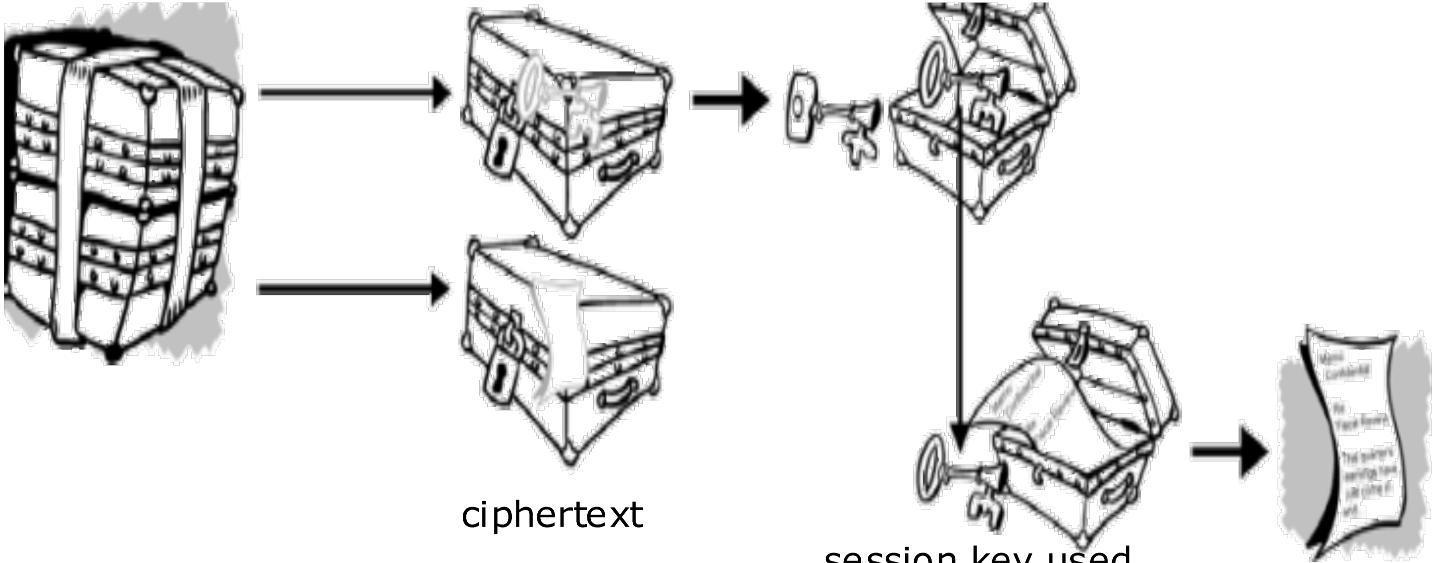
PGP-Confidentiality and Authentication(Steps)

- Sender first signs the message with its own private key
- Next, he encrypts the message with a session key
- That session key is encrypted with the recipient's public key

PGP-Encryption



PGP-Decryption



PGP- Compression

- By default PGP compresses message after signing but before encrypting
- Uses ZIP compression algorithm

PGP- E-mail Compatibility

- Many electronic mail systems can only transmit blocks of ASCII text.
- This can cause a problem when sending encrypted data since ciphertext blocks might not correspond to ASCII characters which can be transmitted.
- PGP overcomes this problem by using **radix-64 conversion.**

PGP- E-mail Compatibility- Radix 64 conversion

- Suppose the text to be encrypted has been converted into binary using ASCII coding and encrypted to give a ciphertext stream of binary.
- Radix-64 conversion maps arbitrary binary into printable characters as follows:
  1. The binary input is split into blocks of 24 bits(3 bytes)
  2. Each 24-bit block is then split into four sets each of 6-bits
  3. Each 6-bit set will then have a value between 0 and 2^6-1 (=63)
  4. This value is encoded into a printable character.

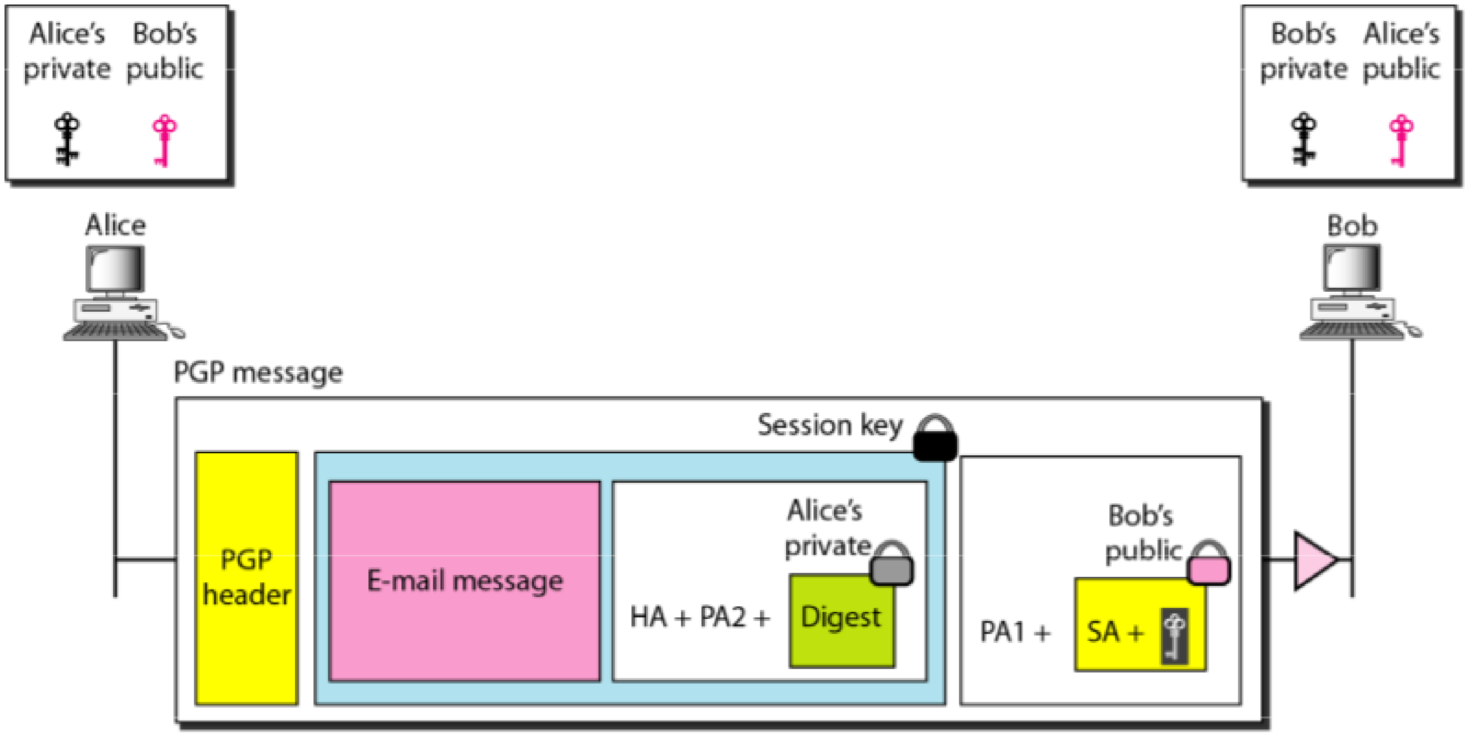
PGP- E-mail Compatilbility-Radix 64 conversion

6 bit value	Character encoding	6 bit value	Character encoding	6 bit value	Character encoding	6 bit value	Character encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

PGP- E-mail Compatilbility-Radix 64 conversion(Example)

- Suppose the email message is: **new**   ASCII: a=97,b=98....
- ASCII format: 110 101 119
- ASCII to Binary: 01101110   01100101 01110111
- After encryption: 10010001 10011010 10001000
- The Radix-64 conversion:
  - The 24-bit block: 10010001 10011010 10001000
  - Four 6-bit blocks: 100100 011001 101010 001000
  - Integer version: 36   25   38   8
  - Printable version: k   Z   m   I

Email authentication and encryption



PA1: Public-key algorithm 1 (for encrypting session key)
PA2: Public-key algorithm (for encrypting the digest)
SA: Symmetric-key algorithm identification (for encrypting message and digest)
HA: Hash algorithm identification (for creating digest)