Cryptography and Network Security II

Second Course

Lecture 1 : Email security: PGP and S/MIME

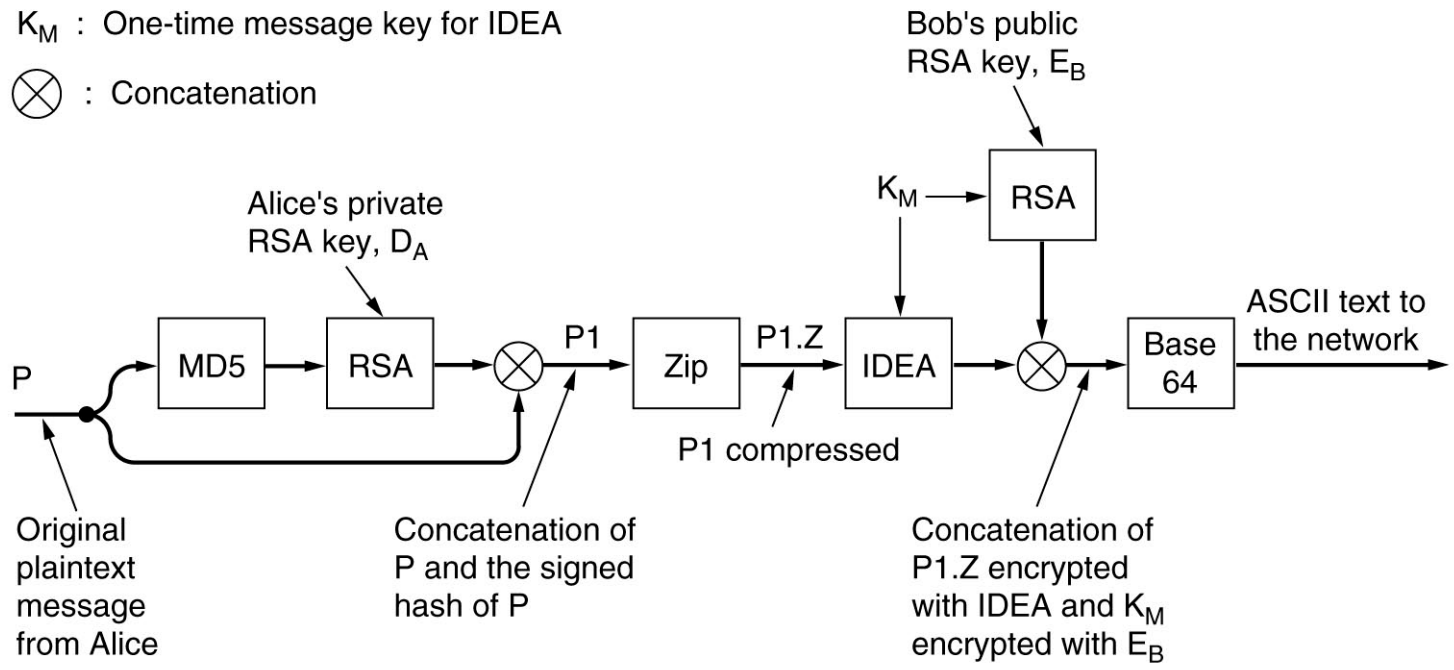
Email

- Very popular network-based application
- Sending an email between two distant sites means that the email has to transit dozens of machines on the way
  - Those machines may read and record the message
  - Privacy is thus non-existent by default
- There are systems for secure e-mails
  - PGP
  - S/MIME
  - PEM
  - …

Pretty Good Privacy – PGP

- Essentially the product of one single person – Phil Zimmermann
- PGP was released in 1991
  - Complete email security package providing privacy, authentication, digital signatures, and compression in an easy-to-use form
  - Complete package, including source code distributed freely on Internet: e.g., www.pgpi.org
  - Available on Unix, Linux, Windows, Mac OS
  - Based on IDEA for encryption (128-bit key), RSA for key management, MD5 for data integrity
- Big controversy related to PGP
  - Zimmermann did nothing to stop people from posting PGP on websites – US government claimed that he violated US laws prohibiting the export of munitions and investigated the case for 5 years before dropping it
    - He never posted PGP on a website
    - Wining a trial meant to convince a jury that posting privacy code on a website is covered by a law prohibiting export of tanks, submarines, nuclear weapons, etc.
    - Publishing the code in a book is not covered by the same law
  - Patent infringement: RSA Security Inc claimed that PGP's use of RSA infringed on its patent (eventually settled); same problems with using IDEA
- Many version of PGP exist – users can download and modify the code. Discuss here original PGP. Other versions: Open PGP, GNU Privacy Guard, etc.

Using PGP to encrypt and sign a message



Using PGP to encrypt and sign a message

- Sending a message
  - Alice first hashes her message with MD5 and then encrypts the hash with her private RSA key
  - Encrypted hash and message are concatenated and encrypted with ZIP (based on Lempel-Ziv algorithm)
  - PGP asks the user for a random input from the keyboard
    - Based on the input and on the typing speed, PGP generates a 128-bit IDEA message key K_M
    - Encrypt P1.Z with IDEA with key K_M
    - Key K_M is then encrypted with Bob's public key
  - The two components are concatenated and converted to base64
    - Some email software only allows sending ASCII text
    - Converting to base64 will give the symbols [A-Z][a-z][0-9][+/]; pad with "=" if necessary
- Receiving a message
  - Bob reverses the base64 encoding and decrypts the IDEA key with his own private key
  - Using this key he decrypts the message to get P1.Z and decompresses it
  - Separate the plaintext from the encrypted hash and decrypt the hash with Alice's public key
  - Compute the hash of the message and check the match with the received hash

Conversion to base 64
- Radix-64 conversion -

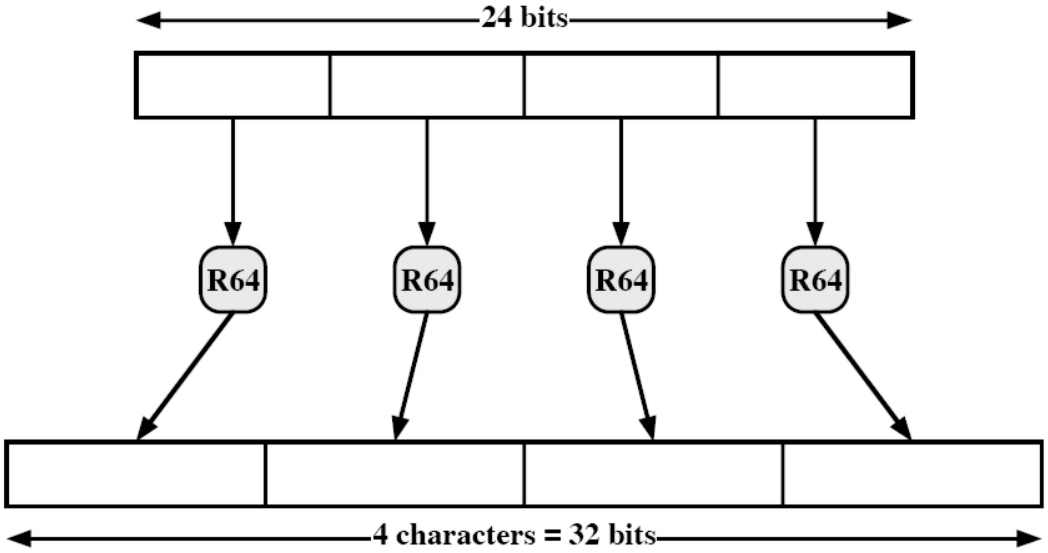


Figure 15.11 Printable Encoding of Binary Data into Radix-64 Format

Conversion to base 64
- Radix-64 conversion -

Table 15.9 Radix-64 Encoding

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	/
						(pad)	=

Encryption or signing with PGP

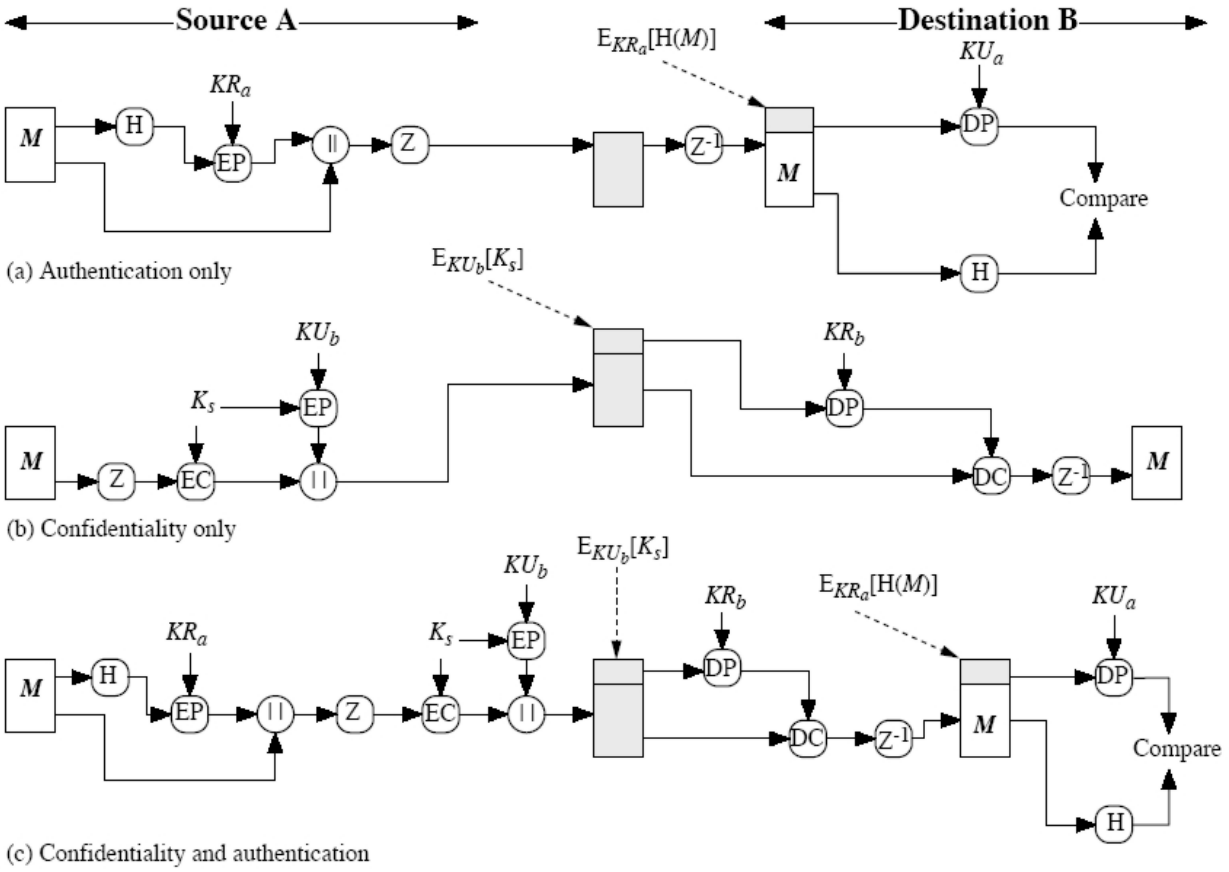
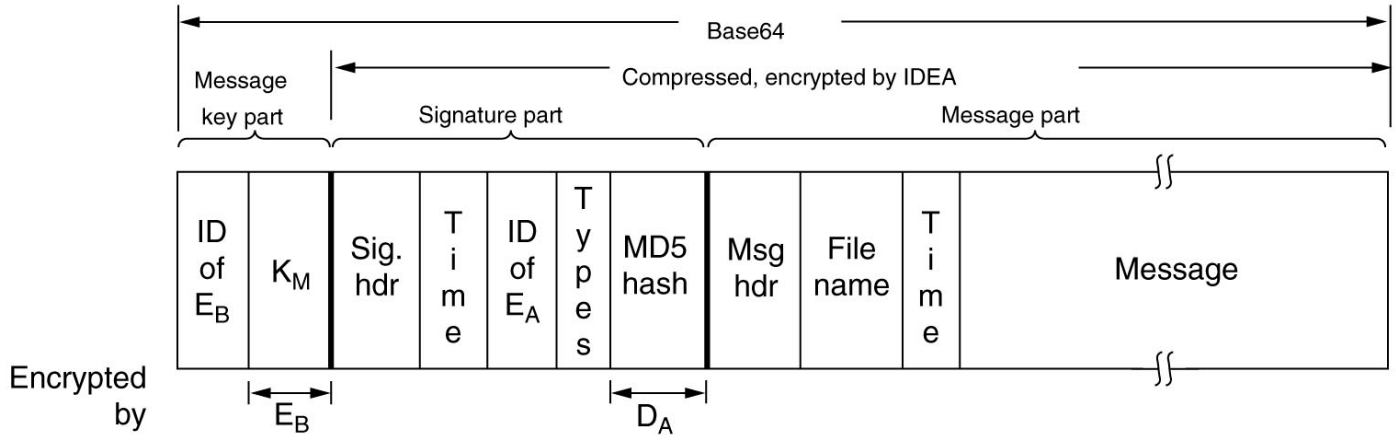


Figure 15.1 PGP Cryptographic Functions

Performance

- RSA is used in two places: to encrypt the 128-bit MD5 hash and the 128-bit IDEA key
- The heavy encryption is done by IDEA, which is much faster than RSA
- PGP supports 4 RSA key lengths and it is up to the user to select the appropriate one according to his needs
  - **Casual** (384 bits) – easily broken today
  - **Commercial** (1024 bits) – breakable by “three-letter” organizations
  - **Military** (2048 bits) – not breakable by anyone on Earth
  - **Alien** (4096 bits) – not breakable by anyone on other planets, either?!
    - Since RSA is only used on 256 bits, everyone should use **alien** keys

Format of a classic PGP message



- Key part contains the key and a key identifier (users can have multiple public keys)
- Signature part contains a header, followed by a timestamp, the ID of the sender's public key that should be used for decrypting the signature hash, some type information to identify the algorithms used (for more flexibility), and the encrypted hash
- Message part contains a header, the default name of the file if the receiver is saving it on the disk, a message creation timestamp, and the message

Key management in PGP
- private key ring -

- Each user maintains two data structures locally: a private key ring and a public key ring
- Private key ring contains one or more personal private-public key pairs
  - One may have more than one pair
  - Each pair has an identifier – the low-order 64 bits of the public key
    - Users are responsible to create public keys with different identifiers
  - The private keys on disk are kept encrypted using a special (arbitrarily long) password
    - PGP asks the user for a password when creating the private-public pair
    - Using SHA-1, a 160-bit hash of the password is created and the password is discarded
    - The private key is encrypted using IDEA and using the hash as a key
    - Whenever the user needs access to his private key, the password is given, and the key is recovered based on the hash
  - The public key can be uploaded on dedicated servers: e.g., keyserver.pgp.com, europe.keys.pgp.com

Key management in PGP
- public key ring -

- Public key ring contains public keys of the user's correspondents and their IDs, plus **an indication of how strongly the user trusts the key**
  - When the user inserts a new public key, PGP is asserting the trust the user has in that key: **key legitimacy field** (computed by PGP)
    - The key could be signed by other users; if those users are already known to PGP (through their public keys), then the trust the user has in those users influences the key legitimacy
    - If the key is not signed or it is signed by unknown users, then it will be considered invalid
    - To validate somebody's key, the user must sign the key himself thus asserting to PGP that he is convinced the key is valid
      - To convince oneself, one could: physically get the key from its owner, get the key by email and verify it (its SHA-1 hash) by telephone, get the key from a mutually trusted individual, get it from a trusted certificate authority, etc.
  - PGP has support for X.509 key certificates
  - To revoke a public key, the user should issue a key revocation certificate, signed by the owner – this is just like a normal signature certificate; the user should send this revocation certificate to everybody he knows as quickly as possible
    - An opponent who has compromised a private key may also revoke it himself – this seems unlikely though

Key management in PGP
- using keys for encryption and authentication -

- Signing a file/message
  - PGP retrieves the sender's private key asking first for the password protecting it
  - Signature is constructed based on the private key
- Encrypting a file/message
  - PGP generates a session key and encrypts the message
  - PGP retrieves the recipient's public key from the public key ring and encrypts the session key
- Decrypting a file/message
  - On the receiver's side, PGP retrieves his private key asking for the password protecting it
  - PGP recovers the session key and decrypt the message
- Authenticating a file/message
  - PGP retrieves the sender's public key from the public-key ring
  - PGP recovers the transmitted message digest
  - PGP computes the message digest for the received message and checks the match

Security of PGP

- No publicly known attack against PGP
- Various indications that even government agencies find it impossible (or have big difficulties) breaking PGP
  - 2003: incident involving the Red Brigades (Italy) indicated that neither Italian police neither FBI could break PGP
  - December 2006: US custom agents could not break PGP used to encrypt some presumably illegal files on a computer
  - November 2009: British citizen jailed for 9 months for refusing to give the encryption keys to some of his PGP-encrypted files
- Commercial exploitation
  - 1996: Zimmermann founded PGP inc through a merge with Viacrypt
  - 1997: bought by Network Associates Inc (now McAfee)
  - 2001: Zimmermann leaves Network Associates
  - 2002: several from Zimmermann's old team found PGP corporation and eventually buys the PGP assets from McAfee
  - 2010: Symantec Corp buys PGP for 300 million dollars

S/MIME

- Secure/Multipurpose Internet Mail Extension
  - Security enhancement to the MIME internet e-mail format standard
  - Originally developed by RSA Labs
- Both PGP and S/MIME are under consideration for IETF standard
  - S/MIME is likely to emerge as the industry standard.
  - PGP for personal e-mail security
- S/MIME defined in RFC 2630, 2632, 2633

Simple Mail Transfer Protocol (SMTP, RFC 822)

- **RFC 822:** defines a format for text messages sent by email: header + body
  - **The header consists of several lines, each starting with a keyword followed by a semicolon: “Date:”, “From:”, “Subject:”, “To:”, “Cc:”, “Message-ID:” and ends with a blank line**
  - **The body: unrestricted text**
- **SMTP Limitations – cannot transmit, or has a problem with:**
  - executable files, or other binary files (jpeg image)
  - “national language” characters (non-ASCII)
  - messages over a certain size
  - ASCII to EBCDIC translation problems
  - lines longer than a certain length (72 to 254 characters)
- Solutions for each of these problems exist but none is standard
- Each SMTP message has a header specifying “from”, “to”, “subject”, “date”
- **MIME (Multipurpose Internet Mail Extension)** is an extension to the RFC 822 framework addressing limitations of the **SMTP**
- MIME defines a standard way to deal with email – RFC 2045, 2046
  - Each message has a header with a number of fields describing the content of the email
  - A number of content formats defined
  - Transfer encodings defined

Header fields in MIME

- **MIME-Version:** Must be “1.0” -> RFC 2045, RFC 2046
- **Content-Type:** describes the data with sufficient detail so that the receiving user can deal with the data; more types being added by developers (application/word)
- **Content-Transfer-Encoding:** How the message has been encoded (radix-64) to make it suitable for mail transport
- **Content-ID:** Unique identifying character string.
- **Content Description:** Text description of the object within the body; needed when content is not readable text (e.g.,mpeg)

S/MIME Functions

- S/MIME offers similar services as PGP: sign and/or encrypt
  - **Enveloped Data:** Encrypted content and encrypted session keys for recipients.
  - **Signed Data:** Message Digest encrypted with private key of “signer”
    - Can only be viewed by a recipient with S/MIME capabilities
  - **Clear-Signed Data:** Signed but not encrypted.
    - Can be seen (but not verified) also by recipients without S/MIME capabilities
  - **Signed and Enveloped Data:** Various orderings for encrypting and signing – sign first, then encrypt, or encrypt first, then sign, signed or clear signed-data, etc.

Cryptographic algorithms in S/MIME

- **Message Digesting:** SHA-1 and MD5
- **Digital Signatures:** DSS (based on ElGamal) and RSA (keys from 512 to 1024 bits)
- **Secret-Key Encryption:** Triple-DES, AES (if available on the user's computer; not supported in Windows XP), RC2/40 (for backward compatibility)
- **Session Key Encryption:** ElGamal, RSA (key from 512 to 1024 bits)
- **MAC:** HMAC with SHA-1

- In all the above, the first listed algorithm is a "MUST", the second is recommended to be available
- S/MIME messages have extra MIME content types to describe the content of the message
- S/MIME secures a MIME entity (not the header though) with signature and/or encryption – if the MIME content type is multipart, then a MIME entity is one or more subparts of the message
- Signing/encrypting a message works in the same way as with PGP

Key management

- S/MIME uses Public-Key Certificates - X.509 version 3 signed by Certification Authority
  - The user is responsible to get the certificates needed to validate a new public key
- S/MIME key management functions to be performed
  - **Key Generation** – generate keys for ElGamal, DSS, and RSA
  - **Registration** - Public keys must be registered with X.509 CA
  - **Certificate Storage** - Local (as in browser application) for different services; may also be offered by the net administrator for a number of local users
- Public key certificates are offered by several companies (CAs): Nortel, Verisign (most popular), GTE, US Postal Service

Public-key certificates at Verisign (now Symantec) (www.verisign.com)

- **Product name: Verisign Digital ID; to change name in 2012 to Norton-secured certificates**
  - Owner's public key
  - Owner's name or alias
  - Expiration date of the digital ID
  - Serial number of the digital ID
  - Name of the certification authority
  - Digital signature of the certification authority
  - Address
  - E-mail address
  - Basic registration information – country, zip, age, gender, etc.
- **Several classes of certificates**
  - **2004**
    - **Class-1 (14.95 $/year in April 2004):** Buyer's email address checked by Verisign by emailing vital info
    - **Class-2:** Postal address is confirmed as well, and data checked against directories
    - **Class-3:** Buyer must appear in person, or send notarized documents.
  - **2012**
    - Digital ID: 19.95$ for 1 year