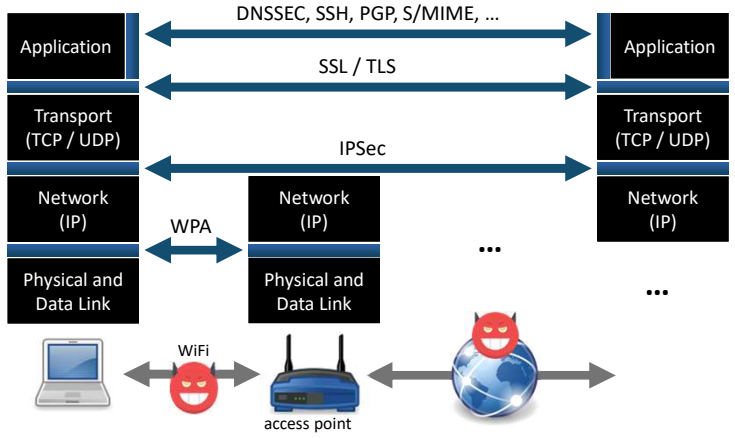
Lecture 16: E-mail Security

Stephen Huang

Content

1. E-Mail Security
2. Pretty Good Privacy (PGP)
3. Secure/Multipurpose Internet Mail Extension (S/MIME)
4. DomainKeys Identified Mail (DKIM)
5. Secure Shell (SSH)

Protocol Stack in Practice

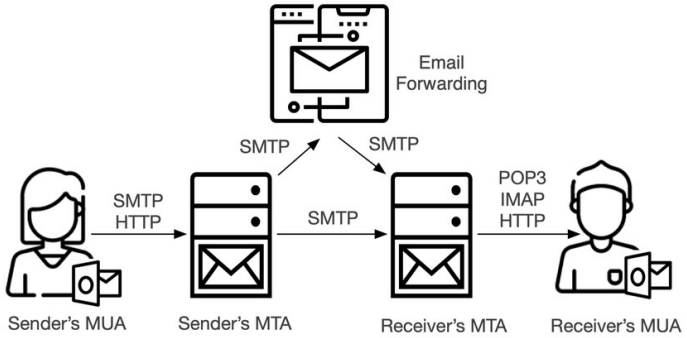


1. E-mail Security

Terminologies

- E-mails are sent to a mail server which is "permanently" available on the network.
- When the recipient's machine connects to the network, it reads the mail from the mail server.
- The e-mail infrastructure consists of a mesh of mail servers, Message Transfer Agents (**MTA**s), and client machines running an e-mail program comprised of a User Agent (**MUA**) and local MTA.
- Simple Mail Transfer Protocol (**SMTP**) is used to forward e-mail messages.
- The recipient retrieves messages from the server using Post Office Protocol (**POP**) and Internet Message Access Protocol (**IMAP**).

The Email Delivery Process



https://www.researchgate.net/figure/The-email-delivery-process_fig1_345806915

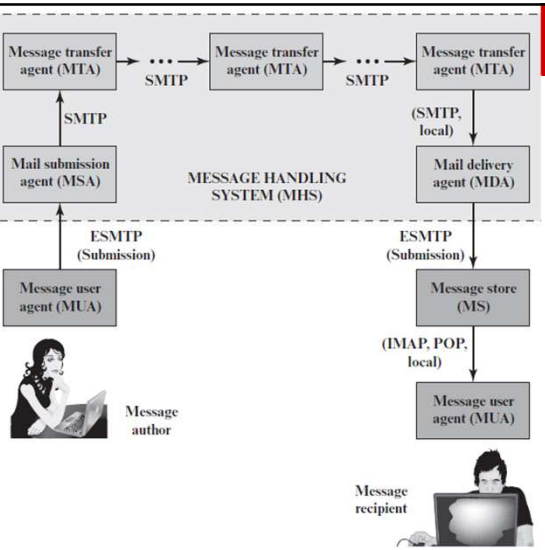


Figure 19.1 Function Modules and Standardized Protocols Used between them in the Internet Mail Architecture

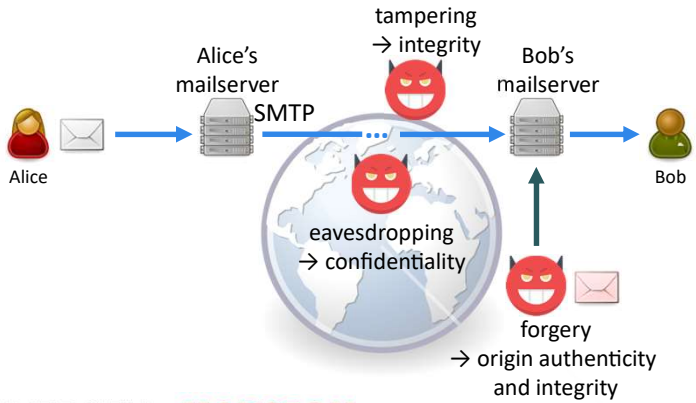
SMTP Transaction Scenario

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: HELO bar.com
S: 250 OK
C: MAIL FROM:<Smith@bar.com>
S: 250 OK
C: RCPT TO:<Jones@foo.com>
S: 250 OK
C: RCPT TO:<Green@foo.com>
S: 550 No such user here
C: RCPT TO:<Brown@foo.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <crlf>.<crlf>
C: Blah blah blah . . .
C: . . . etc. etc. etc.
C: <crlf><crlf>
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```

Figure 19.2 Example SMTP Transaction Scenario

E-Mail Weaknesses

Simple Mail Transfer Protocol (SMTP) standardized in 1982.



E-Mail Threats

- Spam: unsolicited messages sent in bulk.
- E-mail scam: advance-fee scam (a.k.a. "Nigerian Prince" scam), job scam, … .

Example: Advance-fee scam e-mail

I am contacting you in respect of a family treasure of Gold deposited in my name

From: **becky** (becky_time5001@rediffmail.com)
You may not know this sender. Mark as safe | Mark as unsafe
Sent: Wed 8/15/07 11:59 AM
To: becky_time5001@rediffmail.com

i am Becky Ofori a Ghanian from Ashanti region Kumasi, Ghana . I am contacting you in respect of a family treasure of Gold deposited in my name by my late father who was a Gold and Diamond merchant.
As a well known business man,and a stong politician ,my father was brutally murdered during the regime of J.J. Rawlings the ex- president of the federal republic of Ghana , as he was accused of mating the general public against the government of the day. Been a poligamous home , and my mother being his last and most loving wife was abandoned after the death of my father by members of his family . As a strong response to my mother carefull and stiff handling of my fathers estate while he was alive. We were kicked aside without benefitting from any of my fathers shared estate. My mum was humiliated and i and my younger brother was left at the mercy of my elder brothers.
Right now we are passing through great difficulties and i only discovered a document which shows that my father while he was alive, deposited a consignment of gold with my name as the next of kim with a security outfit in my country.We have made all inquiry to confirm this fact with the security outfit . Therfore my mum and i have decided to sell this consignment of gold to a potential buyer in overseas to enable us use the proceeds to put our lives on course again by leaving Africa completely to overseas to start life afresh.
I want you to come to Ghana and see for yourself what i am talking about as my beneficiary or help us effect the sale overseas .We are prepared to go into any agreement for percentage compensation for your anticipated help , and we are very much prepared to part with 20% of the sales money for your help and assistance.
On the contrary, if you are a potential buyer ,then a fresh agreement would be reached in respect of this transaction.
I am looking forward to hear from you in this respect as soon as you receive this fax.

E-Mail Threats

- Spam: unsolicited messages sent in bulk.
- E-mail scam: advance-fee scam (a.k.a. "Nigerian Prince" scam), job scam, … .
- Phishing: collecting sensitive information (*e.g.*, passwords, credit card numbers) or delivering malware by impersonating a trusted entity.

Example: Phishing e-mail

- Reported to UH on October 8th, 2019

From: **Jones, John R** <jrjone27@cougarnet.uh.edu>
Date: Tue, Oct 8, 2019 at 12:48 PM
Subject: Financial Aid On Hold - Action Required

This is a reminder that your financial aid is on hold until all of your documents have been received and processed by our office. Please log in to your uh.verifymyfafsa.com/ portal to complete the requested information. Submit all required items by October 12 in order to have your file complete before the first fall disbursement. If you believe you have completed all requirements but are still getting this reminder email, please log into your Financial Aid Account at the link above and click the submit button.

Sincerely,

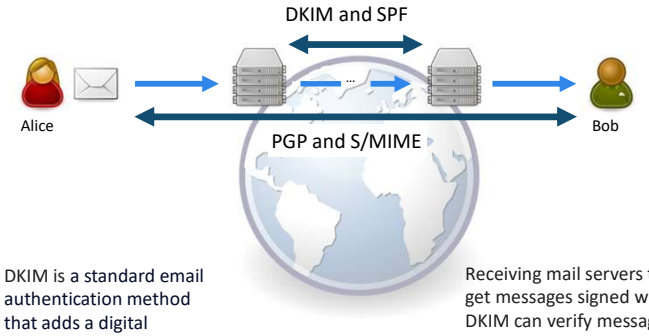
Financial Aid & Scholarship Office
University of Houston, Texas

You have been sent this communication as part of the financial aid process. To opt out of automated communications, click unsubscribe. You may still receive communication that is not auto-generated and sent by the Financial Aid Office. You are still required to complete any outstanding tasks remaining in this site.

E-Mail Threats

- Spam: unsolicited messages sent in bulk.
- E-mail scam: advance-fee scam (a.k.a. "Nigerian Prince" scam), job scam, … .
- Phishing: collecting sensitive information (*e.g.*, passwords, credit card numbers) or delivering malware by impersonating a trusted entity.
- Spear-phishing: phishing directed at specific targets (*e.g.*, users).
  - *examples:*
    - in 2012, attackers penetrated White House internal networks.
    - in 2013, attackers stole 41 million credit card numbers from Target.
    - in 2016, attackers compromised e-mail accounts associated with the Democratic National Committee.

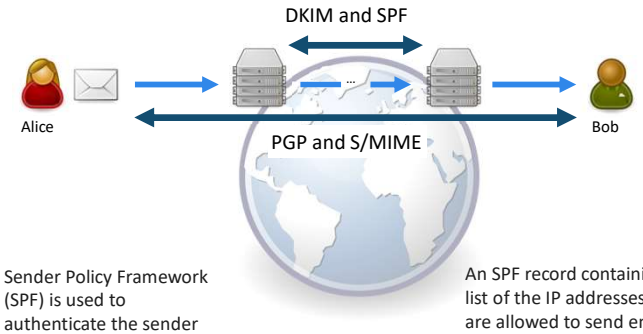
E-Mail Security



DKIM is a standard email authentication method that adds a digital signature to outgoing messages.

Receiving mail servers that get messages signed with DKIM can verify messages actually came from the sender, and not someone impersonating the sender.

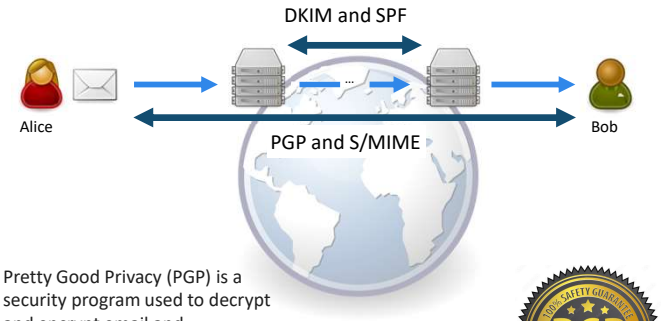
E-Mail Security



Sender Policy Framework (SPF) is used to authenticate the sender of an email.

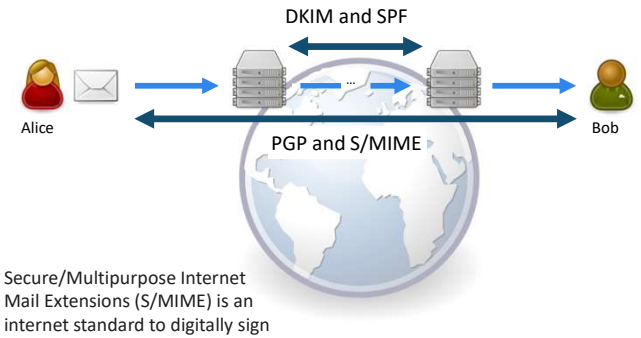
An SPF record containing a list of the IP addresses that are allowed to send email from the domain.

E-Mail Security



Pretty Good Privacy (PGP) is a security program used to decrypt and encrypt email and authenticate email messages through digital signatures and file encryption.

E-Mail Security



Secure/Multipurpose Internet Mail Extensions (S/MIME) is an internet standard to digitally sign and encrypt email messages. It ensures the integrity of email messages remains intact while being received.

2. Pretty Good Privacy (PGP)

- Developed by Phil Zimmermann in 1991
- General-purpose application for secure communication between users
  - confidentiality and integrity protection for files and e-mail,
  - built on widely used asymmetric and symmetric-key cryptographic algorithms,
  - communicating users know each other's public keys → trust.
- IETF standard: OpenPGP
  - first published in 1998, updated multiple times.
- Software
  - PGP went commercial in 1996.
  - GnuPG is a free and open-source implementation of OpenPGP.

PGP Key Management

- Each user may have multiple public-private key pairs → key identifiers are used to specify which key is used

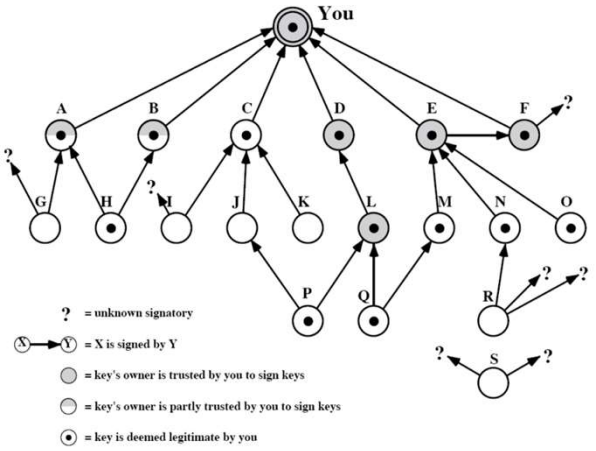
Key storage

- private-key ring: user's own public-private key pairs
  - each entry has user identifier, key identifier, public key, encrypted private key
  - private key is encrypted using a passphrase
- public-key ring: public keys of other users
  - each entry has user identifier, key identifier, public key, trust levels, and signatures from other users
  - public-keys can be verified directly (*e.g.*, delivery on secure channel) or using the "web of trust"

Per User or Per Server?

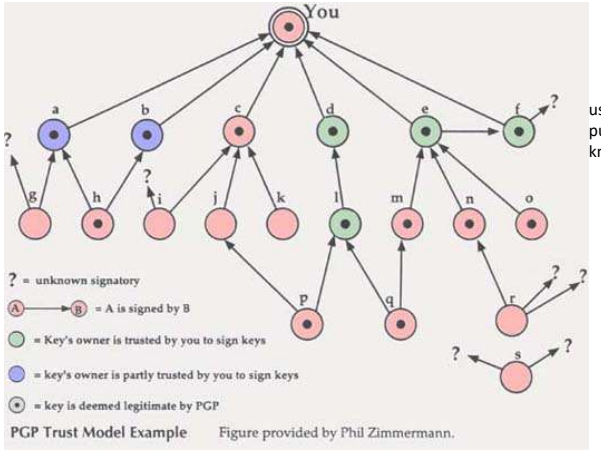
- In PGP, a key pair is typically generated for each individual user rather than for mail servers.
- This means that each user has their own unique key pair, allowing for secure communication.
- However, organizations may also have keys associated with mail servers for tasks such as encrypting server-to-server communication or signing messages sent from the server.

PGP Web of Trust



users whose public keys are known directly

Web of Trust



users whose public keys are known directly

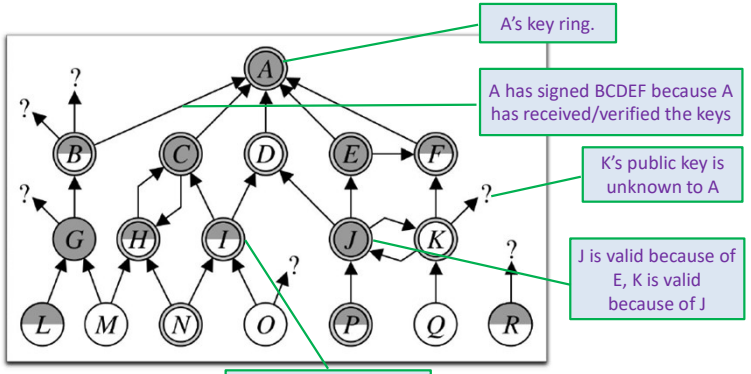
Valid

- To rate a public key valid, PGP either requires
  - the key belongs to the owner of the key ring,
  - a signature from at least one completely trusted introducer with a "valid" public key,
  - signatures from at least two marginally trusted introducers with "valid" public keys.
- Otherwise, the key is rated as "invalid".

Trust

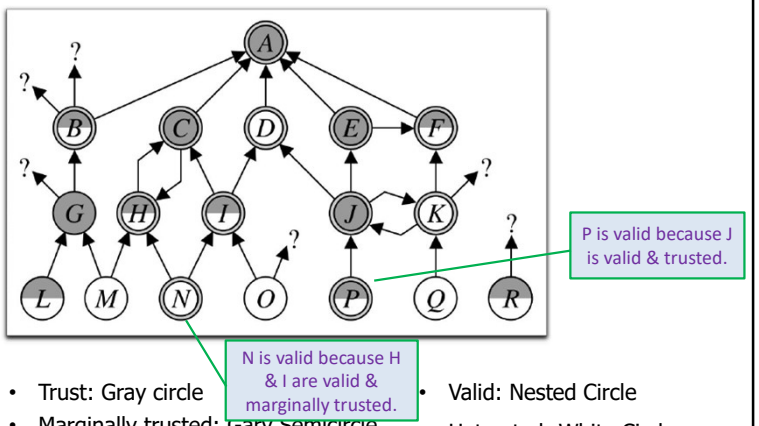
- The PGP trust model is unsatisfactory in many ways.
- Although trust is a gradual quantity that reflects someone's confidence in someone else's reliability, PGP provides only three levels of trust.
  - Completely trusted
  - Marginally trusted
  - Not trusted

Another Example



- Trust: Gray circle
- Marginally trusted: Gary Semicircle
- Valid: Nested Circle
- Untrusted: White Circle

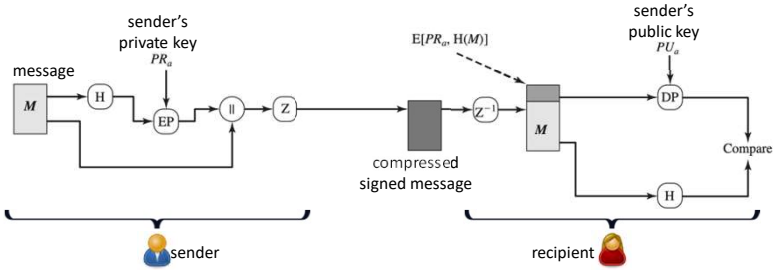
Another Example



- Trust: Gray circle
- Marginally trusted: Gary Semicircle
- Valid: Nested Circle
- Untrusted: White Circle

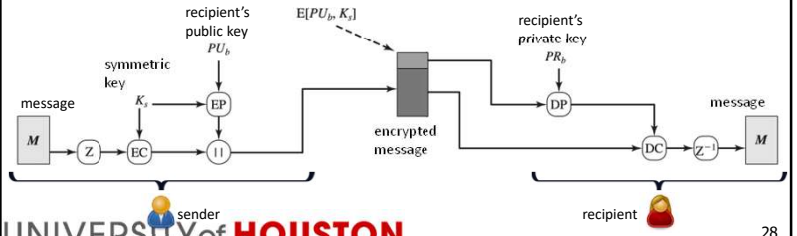
PGP Authentication

- Digital signature using the sender's private key
  - hash using MD5, SHA-1 or SHA-2, and then sign using RSA or DSA
- Message may be compressed after signature
  - ZIP, Bzip2, …

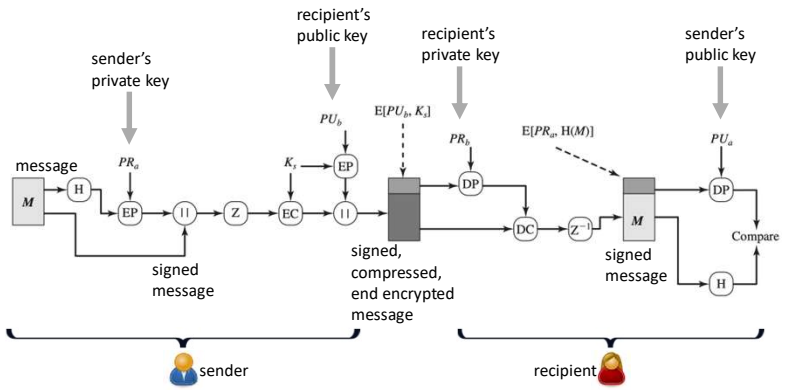


PGP Encryption

- Message may be compressed before encryption
  - ZIP, Bzip2, …
- Generate a new 128-bit random symmetric key for each message
  - encrypt the message with the symmetric key using a block cipher in CFB mode (3DES, Blowfish, AES, …)
  - encrypt the symmetric key with the recipient's public key using RSA or ElGamal



PGP Authentication and Encryption



Definitions

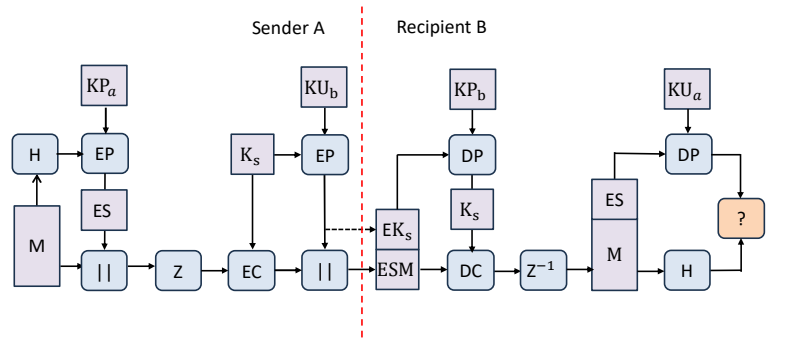
Message & Keys

- M – Message
- ES – Encrypted Signature
- K_s – A random Session Key for Symmetric Encryption
- EK_s – Encrypted Ks
- ESM - Encrypted Signed Message
- KP_b – A private key of user B used in the Public-key encryption
- KP_a – A private key of user A used in Public-key encryption
- PU_a – A public key of user A used in the Public-key encryption
- PU_b – A public key of user B used in the Public-key encryption

Algorithms & Operations

- H – Hash Function
- DP – Public-Key Decryption Algorithm
- EP – Public-Key Encryption Algorithm
- DC – Asymmetric Decryption Algorithm
- EC – Symmetric Encryption Algorithm
- || – Concatenation
- Z – Compression Function
- Z-1 – Decompression Function

Authentication and Confidentiality



3. Secure/Multipurpose Internet Mail Extension

BASIS FOR COMPARISON	PGP	S/MIME
Stands for	Pretty Good Privacy Secure	Multipurpose Internet Mail Extensions
Effectively process	Plain text	Various multimedia files
Depends on	Every user key exchange	Hierarchically validated certifier for key exchange.
Cost	Low	High
Utilization	Personal use	Industrial
Certificates	X.509	X.509V3

Background: Traditional E-Mail Format and Protocol

- E-mail format:

```
Date: March 22, 2022 12:05:30 PM CDT                    Header
From: "Alice" <alice@domain.com>
Subject: Network security
To: bob@otherdomain.com

Hello! I think that security is really interesting.      Body
```

- Transfer: SMTP (Simple Mail Transfer Protocol)
- Limitations of basic SMTP
  - only **7-bit ASCII**, cannot transmit binary objects
    → cannot transmit images, files, etc. or national language / special characters
  - some implementations **remove or add newlines and whitespace characters**
  - messages consist of a **single part** (attachments encoded manually)

Multipurpose Internet Mail Extension

- MIME (Multipurpose Internet Mail Extension)
  - developed in the early 1990s
  - standardized by IETF in 1996
  - fixes the limitations posed by SMTP
- New headers fields
  - Content-Type: type of message content
    - multipart type: body contains multiple parts, each having a header (e.g., images in HTML message, attachments, alternative formats)
    - simple types (e.g., text/plain, image/jpeg, text/html)
  - Content-Transfer-Encoding: how binary data is represented in 7-bit ASCII (e.g., Base64, quoted-printable)

Secure MIME

- Secure / Multipurpose Internet Mail Extension (S/MIME)
  - security enhancement to the MIME e-mail format standard
  - developed in 1995, standardized in 1998
- S/MIME is similar to PGP (Pretty Good Privacy)
  - both S/MIME and PGP enable encrypting and signing messages
  - both have IETF standards
  - both support state-of-the-art algorithms (AES, RSA, SHA-2, …)
  - S/MIME is likely to emerge as the industry standard for commercial and organizational use (*e.g.*, Microsoft Outlook and Gmail support S/MIME)
  - PGP is likely to remain the choice for personal e-mail security

S/MIME Functionality

- Functions
  - **Signed data**: the message is digitally signed, and both the signature and the message are encoded (using Base64 representation)
  - **Clear-signed data**: similar to signed data, but only the signature is encoded
  - **Enveloped data**: encrypted message content and encrypted content-encryption key (i.e., session key) for one or more recipients (encoded using Base64)
  - **Signed and enveloped data**: signing and encrypting may be nested
- MIME content types
  - **`application/pkcs7-mime`**: signed or enveloped data
  - **`multipart/signed`**: for clear-signed data, which contains a message and a signature (signature part is `application/pkcs7-signature`)

Clear-Signed Message Example

```
Date: October 19, 2020 2:15:49 PM CDT
From: "Alice" <alice@domain.com>
Subject: Network security
To: bob@otherdomain.com                          ← header
MIME-Version: 1.1
Content-Type: multipart/signed;
protocol="application/pkcs7-signature"; micalg=sha1;
boundary=BOUNDARY
--BOUNDARY
Content-Type: text/plain                         ← message (plaintext in ASCII)
This is a clear-signed message.
--BOUNDARY
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
ghyHhHUujhJ…nj756                                ← signature (binary data encoded in Base64)
--BOUNDARY--
```


S/MIME Public Keys

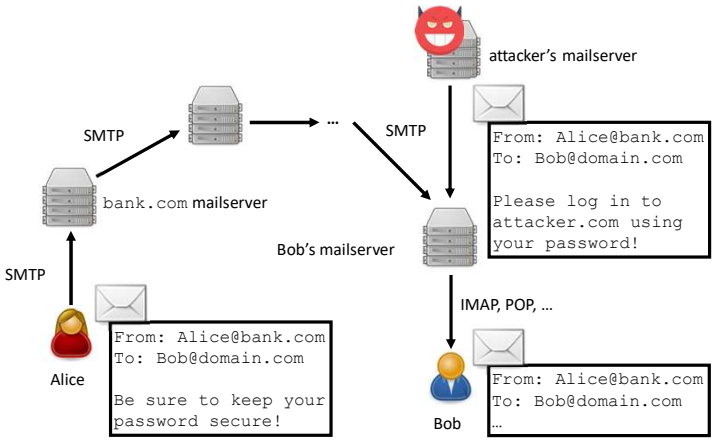
- Public-key certificates
  - based on X.509 digital certificate format
  - similar to PGP, digital certificates are distributed manually
  - however, certificates may be signed by a CA
- Public keys are used for
  - verifying signatures
  - encrypting session keys
    - for enveloped data, the sender generates a random session key
    - The session key is encrypted with each recipient's public key, and the message contents are encrypted (using symmetric-key crypto) with the session key
    - upon receiving the message, a recipient can decrypt the session key and then decrypt the message contents using the session key

in PGP "Web of Trust", users sign other users' keys

4. DKIM

- DKIM is a standard email authentication method that adds a digital signature to outgoing messages.
- Receiving mail servers that get messages signed with DKIM can verify that messages came from the sender, not someone impersonating the sender.

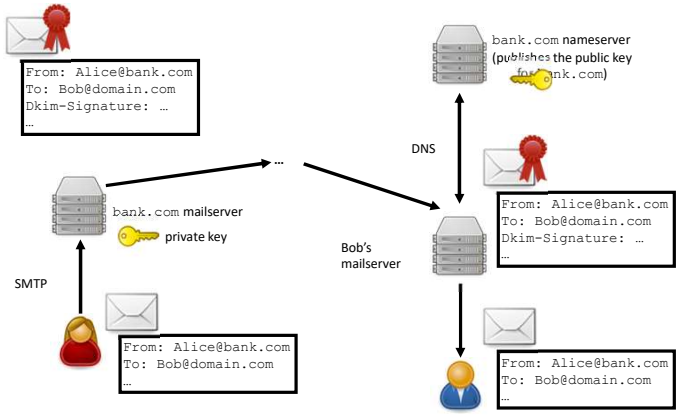
E-Mail Spoofing



E-Mail Spoofing Problem

- *Reminder:* an attacker may use e-mail with a forged sender address for
  - spam: unsolicited advertisement
  - phishing: obtaining sensitive information
- Limitations of PGP and S/MIME
  - depend on the sending and receiving users, who must install or configure software, share public keys, etc.
  - do not sign the message header, only the message contents
- DomainKeys Identified Mail (DKIM)
  - developed by Yahoo and Cisco, standardized by IETF
  - specification for signing e-mail messages
  - implemented on the servers, therefore transparent to the users

DKIM Overview



DKIM Signature and Verification

- Signature: `DKIM-Signature` header field

```
Dkim-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
  d=bank.com; s=gamma;
  h=date:message-id:subject:from:to:content-type;
  bh=5mZvQDy…; b=PcUvPSDy…
```

  - `v`: version
  - `a`: algorithm used for signature (RSA-SHA1, RSA-SHA256)
  - `d`: domain name
  - `s`: selector (if there are multiple public keys)
  - `h`: list of signed header fields
  - `bh`: hash of the body part of the message
  - `b`: signature in Base64
- Verification
  - receiving SMTP server uses DNS to query record *s*.`_domainkey`.*d*
  - nameserver returns the public key corresponding to the signing private key

DKIM Conclusion

- Advantages
  - compatible with existing e-mail infrastructure
  - transparent to users
- Result of verification (i.e., valid or not) can be used for e-mail filtering
  - anti-spam
  - anti-phishing
- Supported by many providers (*e.g.*, Yahoo, Gmail, AOL)

E-Mail Authentication Solutions

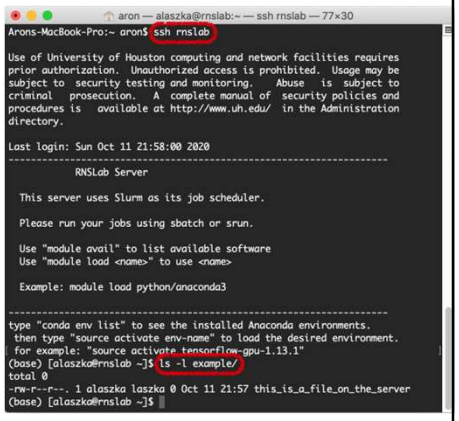
- DomainKeys Identified Mail (DKIM)
- Sender Policy Framework (SPF)
  - another system for preventing e-mail spoofing
  - DNS record lists all authorized sending hosts (i.e., e-mail servers) for a domain
    → receiving server can verify
  - may be combined with DKIM
- Domain-based Message Authentication, Reporting and Conformance (DMARC)
  - built on top of DKIM and SPF, published using DNS record
  - enables domain owners to publish a policy (combination of DKIM and SPF) for verifying the legitimacy of e-mails from the domain

E-Mail Security Conclusion

- PGP and S/MIME
  - typically between users
  - confidentiality, integrity, origin authenticity
- DKIM, SPF, and DMARC
  - typically between servers
  - backwards compatible
  - integrity (DKIM) and origin authenticity
  - supported by many providers

5. Secure Shell (SSH)

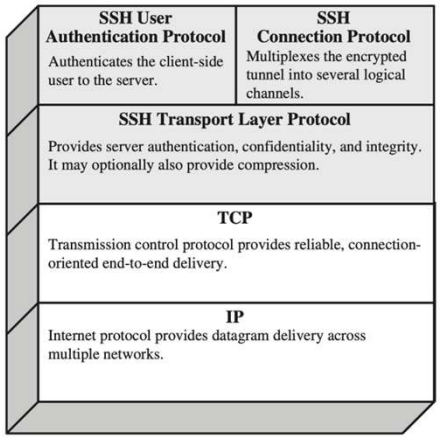
- Motivation: secure remote login
  - earlier remote login protocols (e.g., rlogin, TELNET, rsh) had no security
  - example: enter ssh user@remoteserver.com on the client
    - after authentication, subsequent commands will be executed on the server



Secure Shell (SSH)

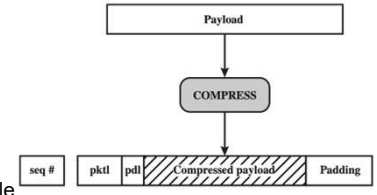
- Motivation: secure remote login
  - earlier remote login protocols (*e.g.*, rlogin, TELNET, rsh) had no security
  - *example:* enter ssh user@remoteserver.com on the client
    - after authentication, subsequent commands will be executed on the server
- SSH-1: developed in 1995 as a freeware
  - later, it evolved into a proprietary software
- SSH-2: standardized by the IETF in 2006
  - improvements in both security and features
- OpenSSH: free and open source implementation of SSH
  - forked from earlier versions of the original SSH program in 1999
  - very popular, available for (and often included in) many operating systems

SSH Protocol Stack



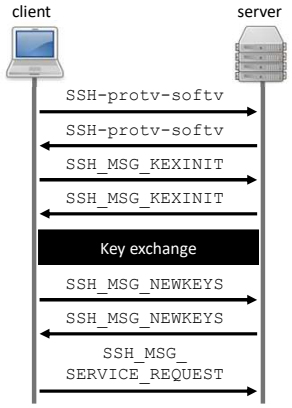
SSH Transport Layer Packet Format

- Sequence number
  - implicit (not sent over)
  - authenticated → prevents replay, etc.
- Padding
  - ensures that length is multiple of the block size (or of 8 bytes if a stream cipher is used)
  - random
    → prevents traffic analysis
- Message authentication code (MAC)
  - computed over entire packet before encryption



SSH Transport Layer: Packet Exchange

- Identification string exchange
  - both the client and the server send:
    `SSH-protocolVersion-softwareVersion`
  - *example:*
    `SSH-2.0-OpenSSH_5.3`
- Algorithm negotiation (`KEXINIT`)
  - both the client and the server send the list of key exchange, encryption, MAC, and compression algorithms that they support
  - chosen one: first on the client's list that is also on the server's list
- End of key exchange (`NEWKEYS`)
  - start using the algorithms and key



SSH Algorithms

- Compression: ZLIB (or none)
- Encryption: 3DES-CBC, AES-CBC, AES-GCM, … (or none)
- Message authentication: HMAC-SHA1, HMAC-MD5, … (or none)
- Key exchange
  - Diffie-Hellman: basic D-H key agreement
  - RSA: client generates a symmetric key and sends it encrypted using the public RSA key of the server
- Server authentication
  - servers signs a hash of all the earlier messages and the new symmetric key
  - servers sends the signature and its public key to the client

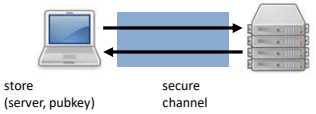
Server Authentication

- Client needs to verify the public-key sent by the server
  (i.e., verify that the public key belongs to the server host)
- Trust models
  - Certificate authority
    - client accepts public keys that are certified by a trusted CA
  - Local database
    - client has a list of known pairs of hosts and public-keys
    - typically, each user has a list stored in its home directory
      - default location:
        `~/.ssh/known_hosts`
      - example content:
        `129.59.256.12 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNo`

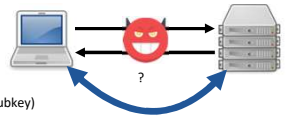
Known Hosts

- First connection: verify and store host and public key

```
$ ssh user@129.7.256.12
The authenticity of host '129.7.256.12 (129.7.256.12)'
can't be established.
ECDSA key fingerprint is SHA256:fc3TjFIKjms5QJnR8+kfTFW7.
Are you sure you want to continue connecting (yes/no)? yes
```

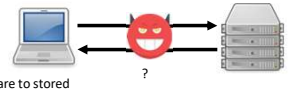


or



verify key manually
(*e.g.*, call sysadmin on the phone)

- Subsequent connections



compare to stored
(server, pubkey)

SSH User Authentication Methods

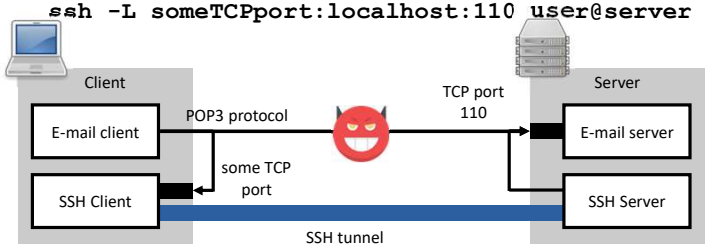
- Passwords
  - The client sends a username and a password
- Public key
  - The client sends a public key and a signature based on the corresponding private key
  - The server checks if the public key is acceptable and verifies the signature
  - Typically, for every user account on the server host, there is a list of acceptable public keys stored at **`~/.ssh/authorized_keys`**
- Host based
  - assumes that the server trusts the client host
    → since the client host has already authenticated the user, the server only needs to verify the identity of the client host
  - The client sends a signature based on the private key of the client host.

SSH Connection Protocol

- Multiplexes a secure connection into a number of logic channels
- Channel types
  - Session
    - remote execution of a command
      (*e.g.*, **`ssh user@server "mv somefile somedirectory/"`**)
    - remote shell (i.e., terminal session)
      (*e.g.*, **`ssh user@server`**
      … user authentication …
      `user@server:~$ mv somefile somedirectory/`)
  - X11: enables X Window System application running on the server to be displayed on the client
  - Direct TCP/IP: local port forwarding
  - Forwarded TCP/IP: remote port forwarding

Local Port Forwarding

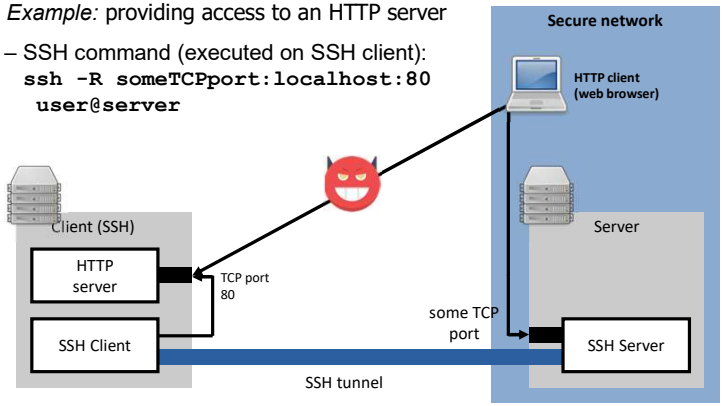
- *Example:* connecting to a POP3 (Post Office Protocol) server securely
  - server listens on TCP port 110
  - SSH command (executed on the client):
    `ssh -L someTCPport:localhost:110 user@server`



- May be used for any TCP-based protocol (*e.g.*, HTTP on port 80)

Remote Port Forwarding

- *Example:* providing access to an HTTP server
  - SSH command (executed on SSH client):
    `ssh -R someTCPport:localhost:80 user@server`



SSH Conclusion

- Applications
  - primary application is remote access to shell accounts
  - Secure copy (SCP): protocol and command-line tool
    - *examples:*
      `scp sourcefile user@server:directory/targetfile`
      `scp user@server:directory/sourcefile targetfile`
  - SSH File Transfer Protocol (SFTP) ≠ FTP over SSH
  - secure channels between TCP ports
- Security
  - SSH-1: multiple design flaws, obsolete
  - CBC-plaintext recovery (SSH-2)
    → use some variation of CTR mode

Next Topic

- E-Mail Security
- Authentication and Access Control