SOFTWARE ARCHITECTURE
10. REMOTE TERMINAL AND ELECTRIC MAIL

Tatsuya Hagino

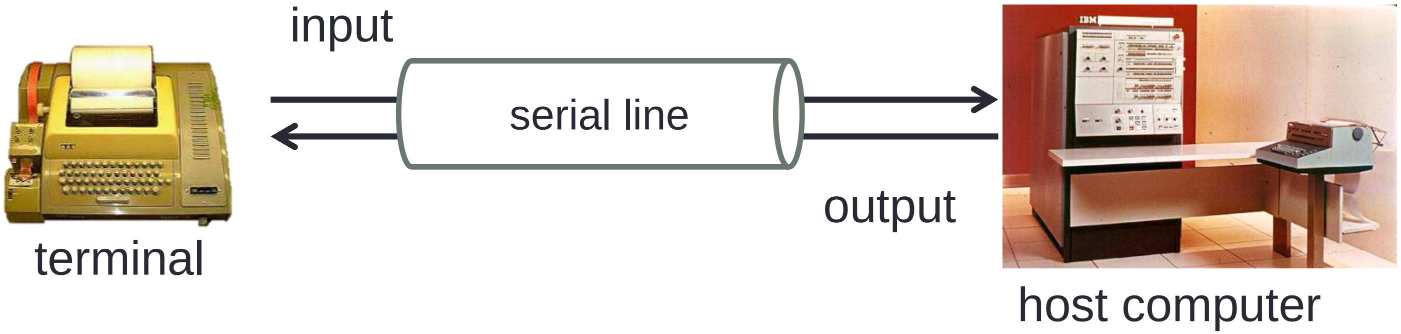
hagino@sfc.keio.ac.jp

lecture URL

https://vu5.sfc.keio.ac.jp/slide/

Terminal to Use Host Computer

- Before personal computer age
  - Connect a terminal to a host computer with a serial line or a telephone line.
  - A terminal is a kind of type writer but key inputs go to the serial line and characters received from the serial line are printed.
  - Character terminal, graphical terminal, etc.

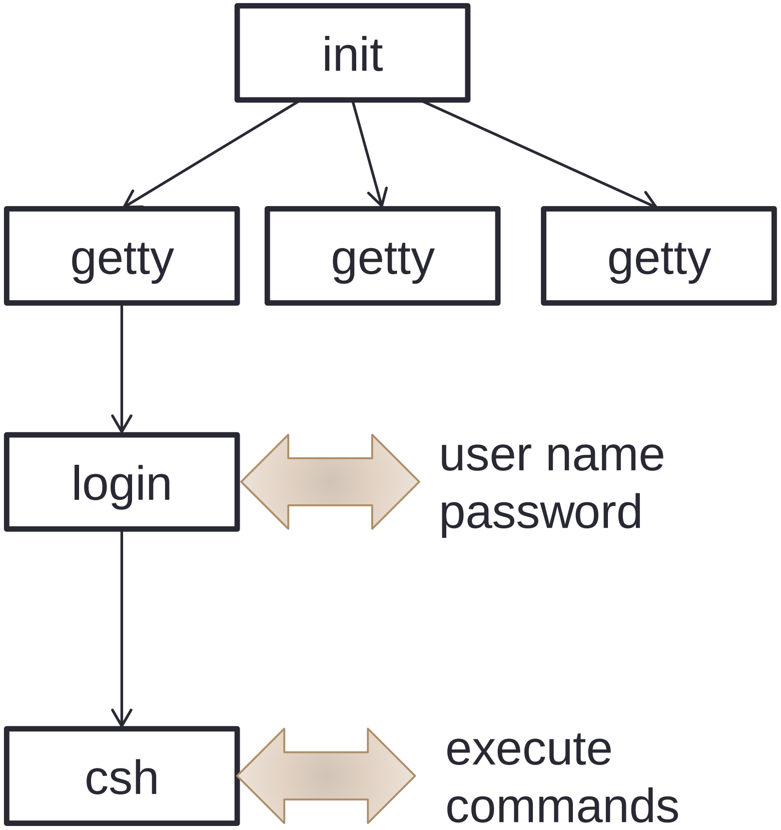


- Personal computer age
  - PC terminal emulator
  - Connected with a serial line or a telephone line through a modem.

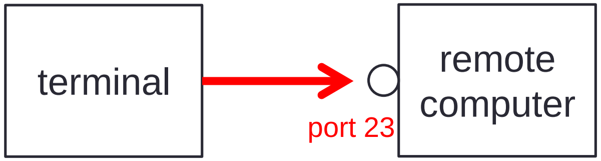
- Internet age
  - Virtual terminal emulator
  - Use Internet to connect to a remote computer.

Terminal Connection in UNIX Host

- First, `getty' program waits for each line for a new connection from a terminal.
  - getty = get tty
- Hand to a `login' program does user authentication by asking a user name and a password.
- Invoke a shell
  - The shell will handle all the interaction from then on.



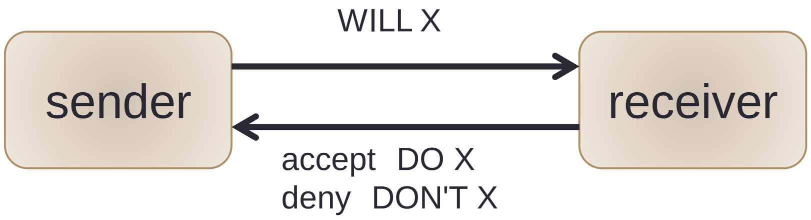
Telnet Protocol



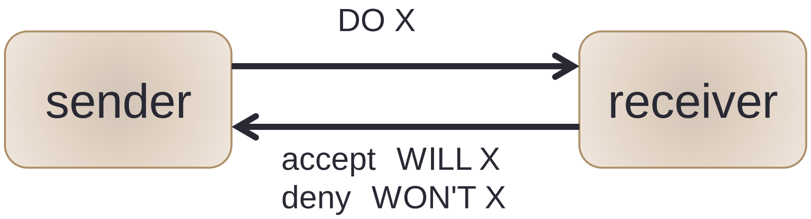
- Implement a virtual terminal for Internet
  - Use internet instead of a serial line.
  - TCP connection (with session and reliability)
    - port 23
- Simple protocol
  - All the characters input from a terminal are sent to the remote computer.
  - All the output form the remote computer are output to the terminal.
  - Replace the serial line, but no authentication of user is handled.
- Options are negotiated between the terminal and the remote computer.
  - DO or DON'T
  - WILL or WON'T

Option Negotiation

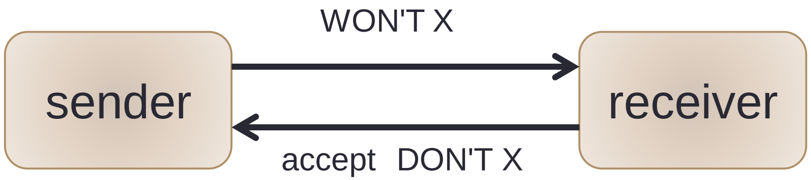
- Enable X at the sender side
  - sender: WILL X
  - receiver: DO X or DON'T X



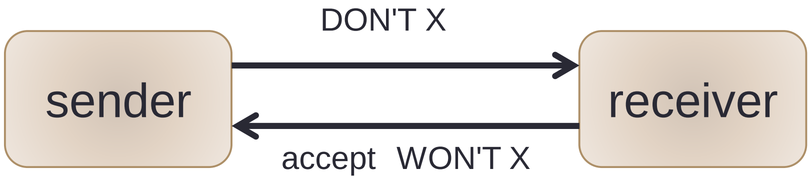
- Enable X at the receiver side
  - sender: DO X
  - receiver: WILL X or WON'T X



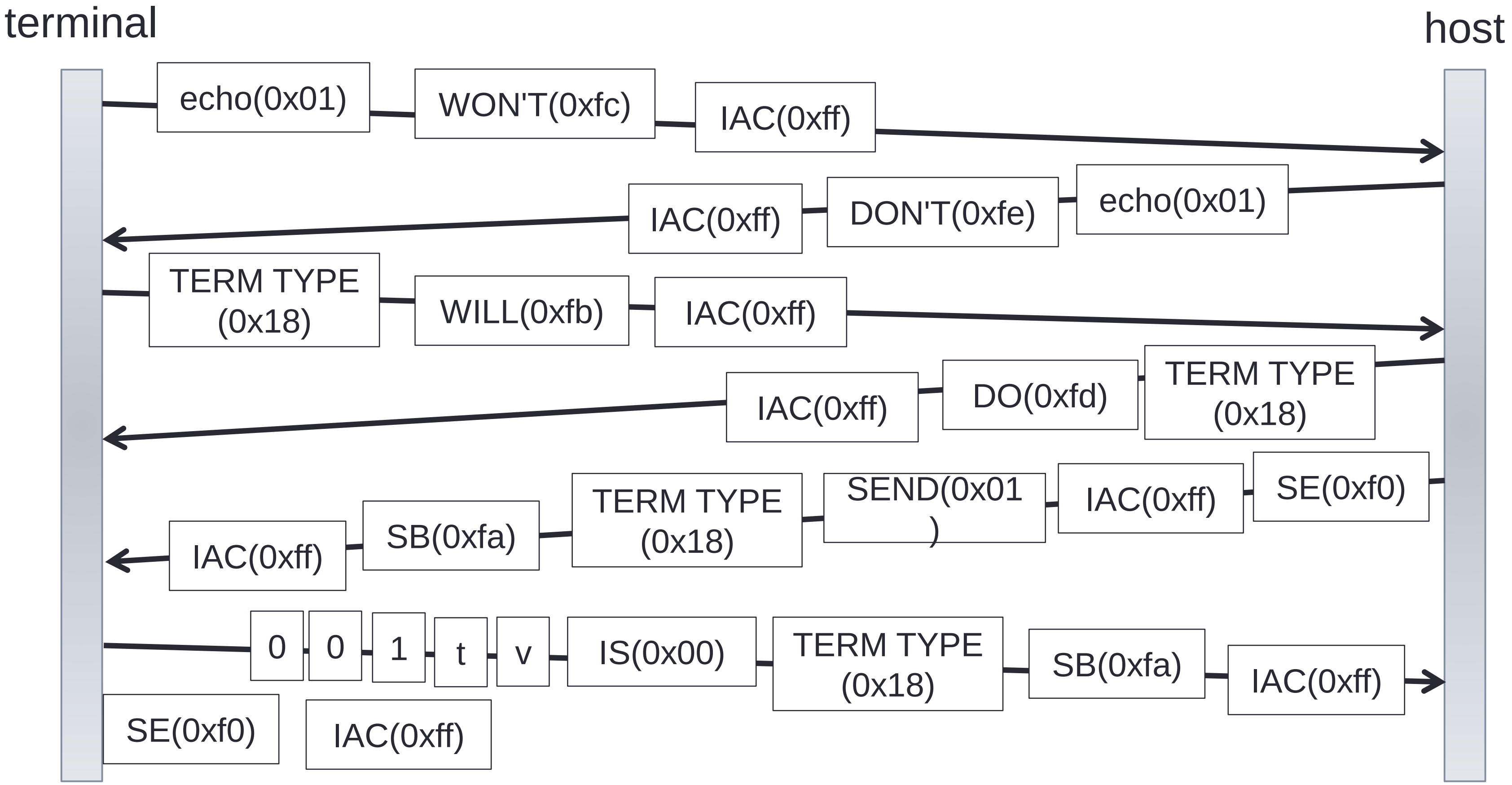
- Disable X at the sender side
  - sender: WON'T X
  - receiver: DON'T X



- Disable X at the receiver side
  - sender: DON'T X
  - receiver: WON'T X



Example of Option Negotiation



IAC: Interpret As Command
SB～SE: Sub negotiation

Telnet Control Command

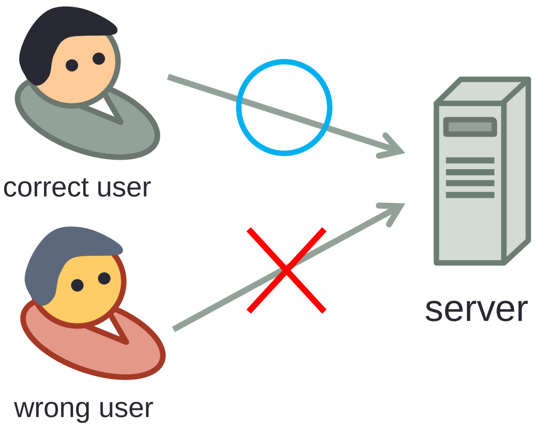
- Are You There
  - IAC, 0xf6
- Erase a character
  - IAC, 0xf7
- Erase a line
  - IAC, 0xf8
- Terminate process (ctrl-C)
  - IAC, 0xf4
- Stop output
  - IAC, 0xf5
- Synchronize (TCP urgent message)
  - IAC, 0xf2

Extension of Telnet

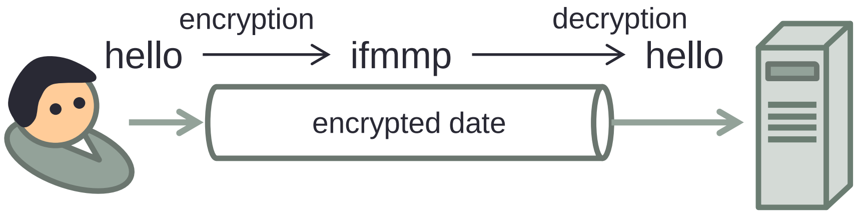
- Problem of Telnet
    - Security problem.
    - User name and password are sent without any encryption.
- Rlogin (remote login)
    - Easy for UNIX systems
    - Introduced in UNIX BSD 4
    - Trust computers listed in ~/.rhosts without password
    - Not much used for security reason
- SSH (Secure Shell)
    - protocol 1, 2
    - authentication: password, challenge-response, RSA, DSA
- Remote desktop
    - Windows version of remote terminal

User Authentication, Encryption and Hash

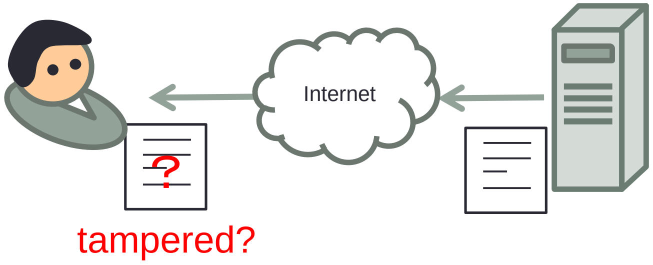
- User authentication
  - Check whether the user is a correct (authorized) user or not.
  - password authentication
  - one time password
  - challenge response authentication
  - RSA (Rivest Shamir Adleman) authentication
  - DSA (Digital Signature Algorithm) authentication



- Encryption
  - Encrypt data so that others cannot understand
  - common key cryptography
    - the key should be secret from others
  - public key cryptography
    - use public key and private key pair



- Hash
  - Make sure data is not tampered.
  - checksum
  - cryptographic hash function (MD5, SHA-1, SHA-2)



User Authentication

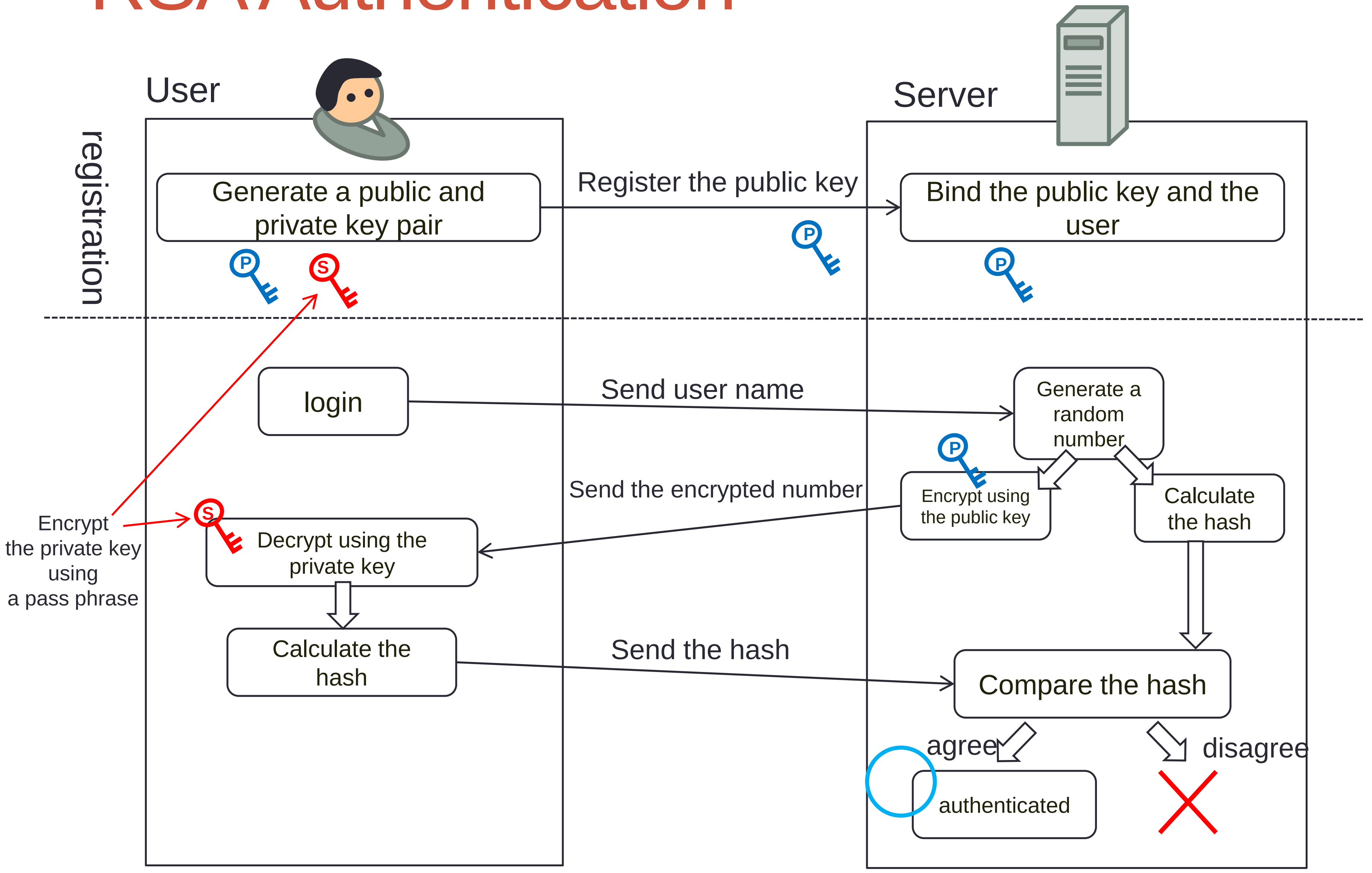
- Password authentication
  - Send the registered password to the server.

- One time password
  - Send a password generated by a password generator and send it to the server.
  - Example: a number calculated from the current time.

- Challenge Response authentication
  - Receive a random number (= challenge) from the server.
  - Calculate a number form the challenge and the user's password.
  - Send the calculated number (= response) to the server.
  - The server does the same calculation to check the response.

- RSA authentication
  - Use RSA (Rivest Shamir Adleman) public key cryptography
  - Generate a public key and private key pair.
  - Register the public key to the server.
  - Decrypt the random number sent from the server using the private key.
  - Calculate the hash and send it to the server.
  - The server checks whether the hash is correct or not.

RSA Authentication

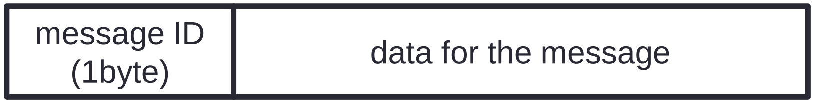

User
Server
registration
Generate a public and private key pair
P
S
Register the public key
P
Bind the public key and the user
P
login
Send user name
Generate a random number
P
Encrypt using the public key
Calculate the hash
Send the encrypted number
Encrypt the private key using a pass phrase
S
Decrypt using the private key
Calculate the hash
Send the hash
Compare the hash
agree
disagree
authenticated

SSH Overview

- Secure communication protocol between terminal and host computer
  - TCP port 22

- Communication data
  - Messages are exchanged between server and client
  - Messages are put into a packet and the packed is encrypted.
  - Multiple channel for multiple communication on a single connection.

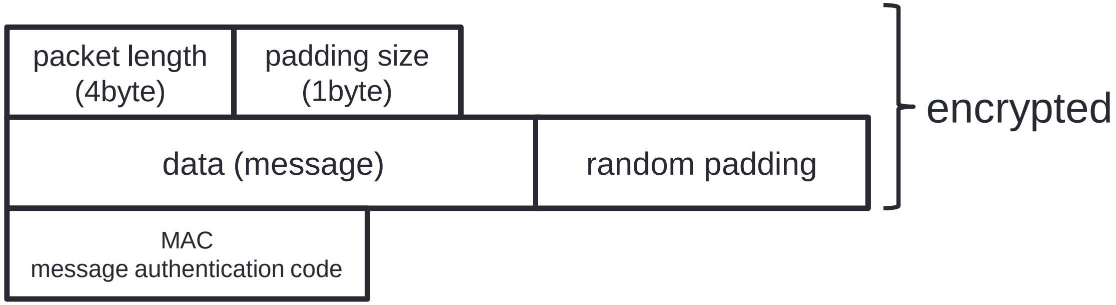
メッセージ



message ID (1byte)

data for the message

パケット



packet length (4byte)

padding size (1byte)

data (message)

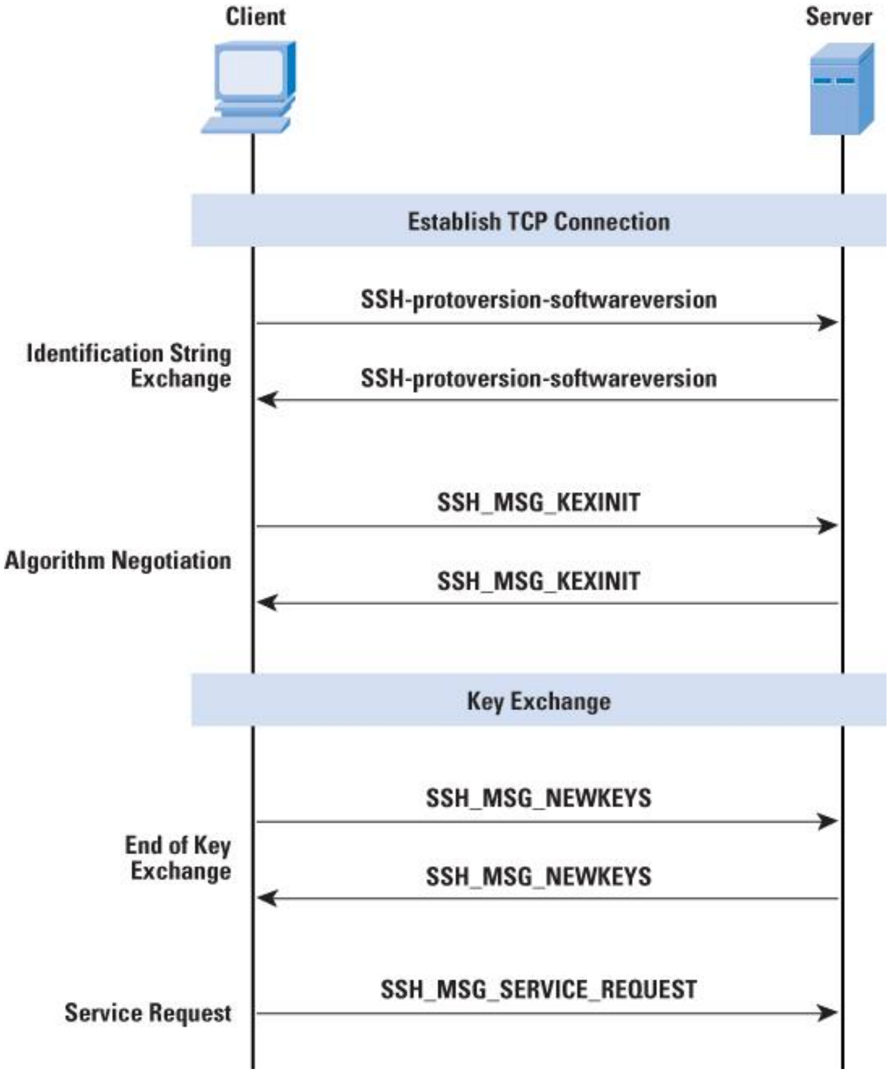
random padding

encrypted

MAC
message authentication code

Message ID	Value
SSH_MSG_DISCONNECT	1
SSH_MSG_IGNORE	2
SSH_MSG_UNIMPLEMENTED	3
SSH_MSG_DEBUG	4
SSH_MSG_SERVICE_REQUEST	5
SSH_MSG_SERVICE_ACCEPT	6
SSH_MSG_KEXINIT	20
SSH_MSG_NEWKEYS	21
SSH_MSG_USERAUTH_REQUEST	50
SSH_MSG_USERAUTH_FAILURE	51
SSH_MSG_USERAUTH_SUCCESS	52
SSH_MSG_USERAUTH_BANNER	53
SSH_MSG_GLOBAL_REQUEST	80
SSH_MSG_REQUEST_SUCCESS	81
SSH_MSG_REQUEST_FAILURE	82
SSH_MSG_CHANNEL_OPEN	90
SSH_MSG_CHANNEL_OPEN_CONFIRMATION	91
SSH_MSG_CHANNEL_OPEN_FAILURE	92
SSH_MSG_CHANNEL_WINDOW_ADJUST	93
SSH_MSG_CHANNEL_DATA	94
SSH_MSG_CHANNEL_EXTENDED_DATA	95
SSH_MSG_CHANNEL_EOF	96
SSH_MSG_CHANNEL_CLOSE	97
SSH_MSG_CHANNEL_REQUEST	98
SSH_MSG_CHANNEL_SUCCESS	99
SSH_MSG_CHANNEL_FAILURE	100

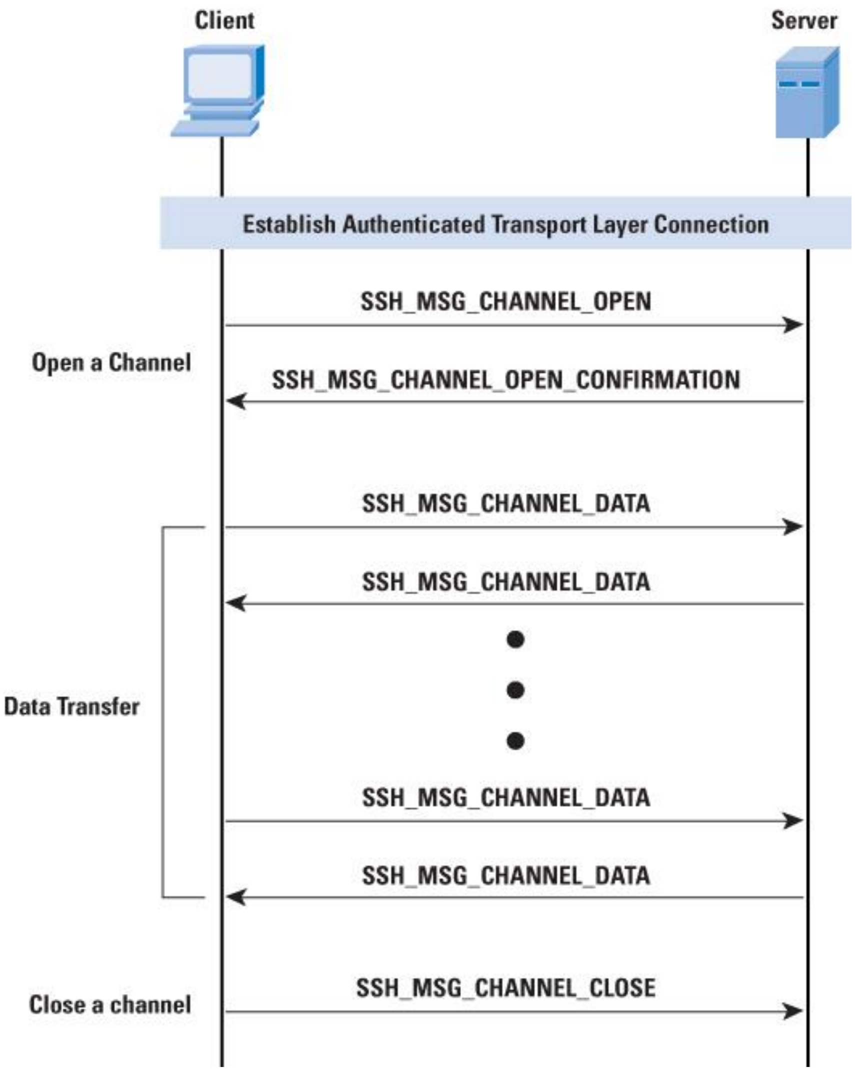
SSH Sequence (1)



- When connected, exchange key to establish a secure communication.

http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-46/124-ssh.html

SSH Sequence (2)



- Create multiple channels:
  - execution of shell
  - execution of command
  - port forwarding
  - X11 forwarding

http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-46/124-ssh.html

Summary of Remote Terminal

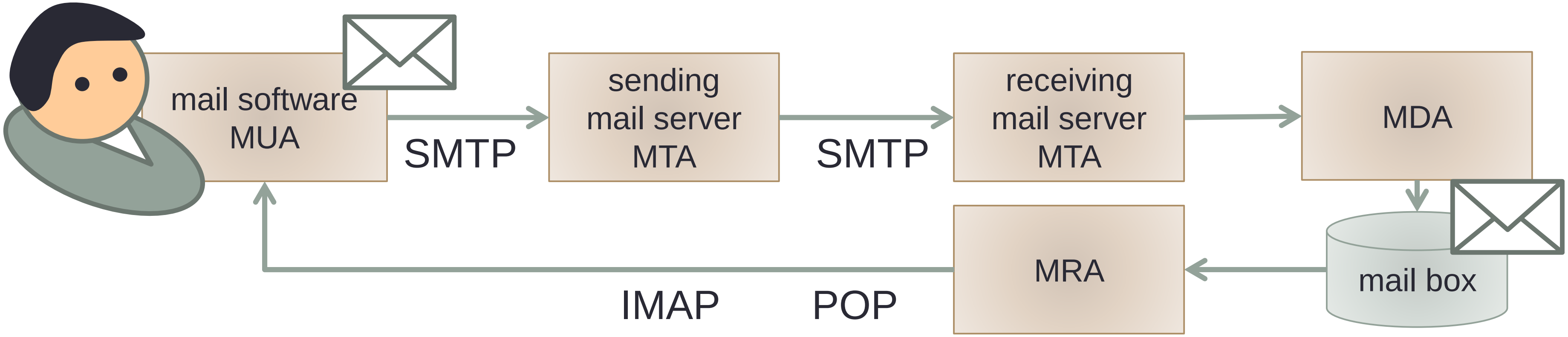
- Telnet
  - One of the oldest TCP protocol
  - Simply implement virtual terminal on internet
  - Security issues

- SSH
  - Secure communication channel
  - Multiple user authentication mechanism supported
  - Multiple channels for one connection

ELECTRIC MAIL

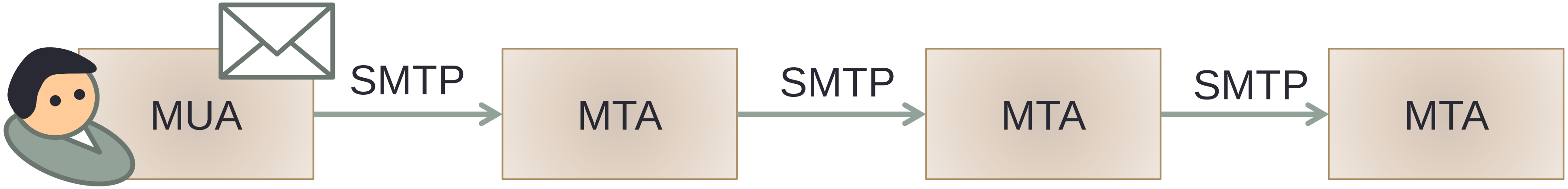
Electric Mail Components

- MUA (Mail User Agent)
  - Mail client software
  - Sending and receiving mails
- MTA (Mail Transfer Agent)
  - Sending mails to destination
- MDA (Mail Deliver Agent)
  - MTA uses this to write mails to mail boxes.
- MRA (Mail Retrieval Agent)
  - Retrieving mails from remote mail box.



SMTP

- Simple Mail Transfer Protocol
  - MUA uses this to send mails to MTA
  - MTA uses this to forward mails to other MTA



- Specification
  - RFC821（1982）is the first spec.
  - Various extensions are added later covered by multiple RFCs.
  - ESMTP (Extended SMTP)

- TCP connection
  - port 25

SMTP Server Mandatory Commands

- HELO
  - Specify the sending host.
- MAIL
  - Specify the sender
- RCPT
  - Specify the receiver
- DATA
  - Mail message
- RSET
  - Reset the server
- NOOP
  - No effect
- QUIT
  - Terminate the connection

Example of Sending Mail

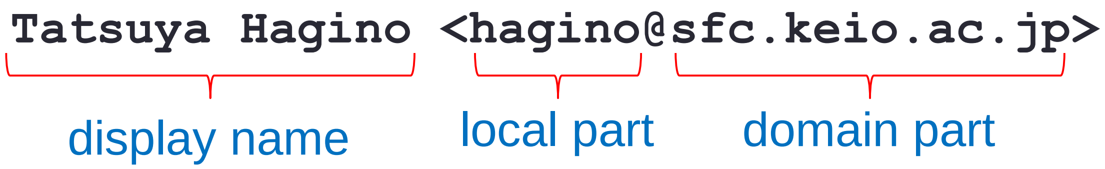
- Connect to the server
  - → 220 smtp.sfc.keio.ac.jp SMPT
- HELO ninna.tom.sfc.keio.ac.jp
  - → 250 ninna.tom.sfc.keio.ac.jp Hello
- MAIL FROM: hagino@sfc.keio.ac.jp
  - → 250 hagino@sfc.keio.ac.jp Sender ok
- RCPT TO: ns@gms.komazawa-u.ac.jp
  - → 250 ns@gsm.komazawa-u.ac.jp Recipient ok
- RCPT TO: timbl@www.org
  - → 220 timbl@www.org No such user
- RCPT TO: timbl@w3.org
  - → 250 timbl@w3.org Recipient ok
- DATA
  - → 354 Enter mail, end with "." on a line by itself

```
From: hagino@sfc.keio.ac.jp
To: ns@gms.komazawa-u.ac.jp
Cc: timbl@w3.org
Subject: Hello

Dear Nobuo and Tim,
(mail body)
.
```

  - → 250 0AA06460 Message accepted for delivery
- QUIT
  - → 221 smtp.sfc.keio.ac.jp closing connection

Mail Address and Mail Server



Tatsuya Hagino <hagino@sfc.keio.ac.jp>

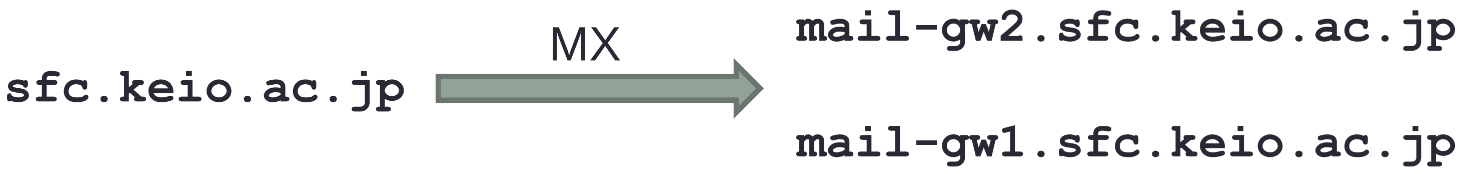
display name

local part

domain part

- A mail address consists of local part and domain part.
  - Use DNS domain name and the part is case insensitive
  - The local part is officially case sensitive, but a lot of systems ignore case.

- Mail server
  - Find the mail server from the domain part.
  - Use MX (Mail Exchange) of DNS to specify the mail server for the domain.



sfc.keio.ac.jp

MX

mail-gw2.sfc.keio.ac.jp

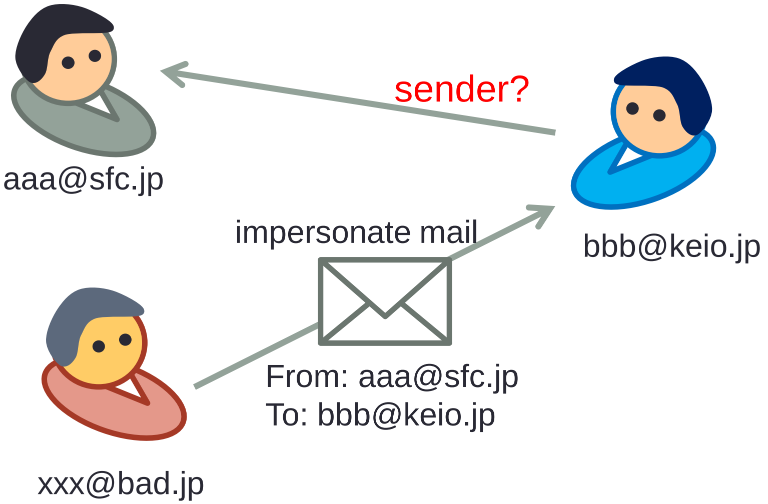
mail-gw1.sfc.keio.ac.jp

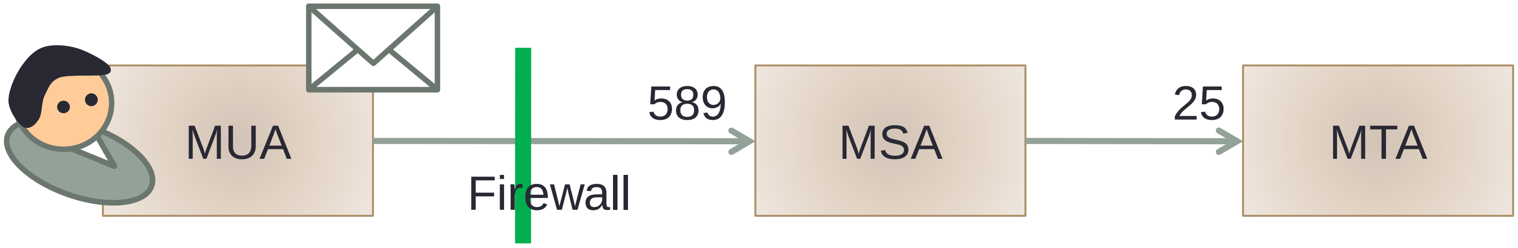
SMTP Security

- Anybody can send mail with any name.
  - Can easily create impersonate mails.
  - Spam

- Limit MUA to connect MTE
  - POP before SMTP
  - SMTP AUTH

- Prohibit using outside MTA
  - Outbound port 25 blocking at firewall
  - MSA (Message Submission Agent)





- Encrypt message body
  - SMTP over SSL

Mail Server Software

- sendmail
  - Developed at U.C. Berkeley in 1980s.
  - Can handle other electric mail protocols like UUCP.
  - Configuration file sendmail.cf consists of rewrite rules.
- qmail
  - Fast, simple and robust
  - Consists of multiple small programs.
  - Easy to setup
  - Use Maildir format for mail boxes.
- postfix
  - Try to keep the compatibility with sendmail
- courier-MTA
  - Open source software
  - Replacement of qmail
- exim

Mail Message Format

- Header
  - RFC822
    - **`To:`** mail receivers
    - **`From:`** mail sender
    - **`Date:`** date
    - **`Subject:`** mail subject

- Body
  - MIME （Multipurpose Internet Mail Extensions）
    - **`MIME-Version: 1.0`**
    - **`Content-Type: type/subtype; parameter`**
    - **`Content-Transfer-Encoding: mechanism`**
    - **`Content-ID: message-id`**
    - **`Content-Description: text`**

Content-Type

- `text/plain;charset=us-ascii`
  - normal text
- `text/enriched`
  - He is a <bold>Japanese</bold>
- `multipart/mixed;boundary=“-3D3C5DF17D08”`
  - consists of multiple parts
- `message/rfc822`
  - electric mail
- `application/octec-stream;name=text.lzh;type=lzh`
- `application/postscript`
- `image/jpeg`
- `audio/basic`
- `video/mpeg`

Use of Japanese in Electric Mail

- Body
  - Default encoding is iso-2022-jp
  - Can specify encoding by MIME (UTF-8, Shift_JIS, ...)

- Header
  - ASCII only
  - For subjects, encode Japanese into ASCII
    - base64
      - `Subject: ?ISO-2022-JP?B?GyRCMGY4fU1NGyhC?=`
    - quoted-printable
      - `Subject: ?ISO-88591?Q?Keld_J=F8rn_Simonsen?=`

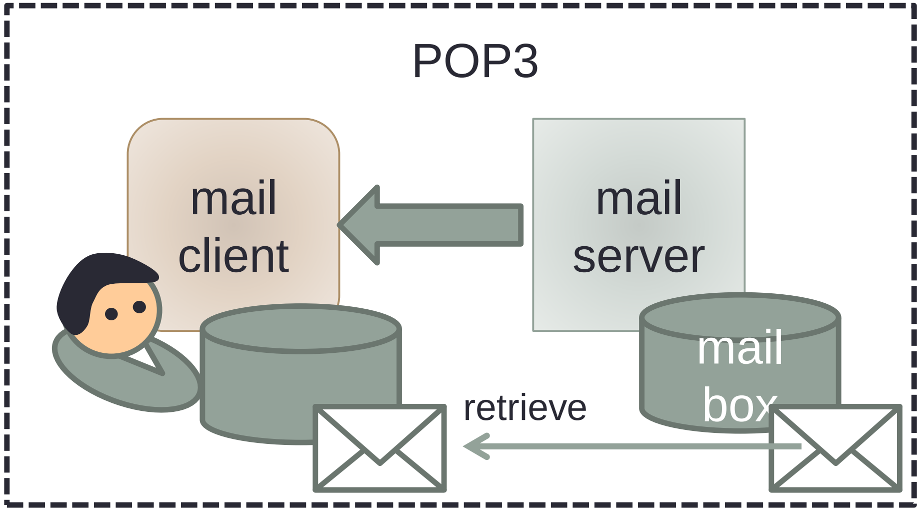
Receiving Electric Mails

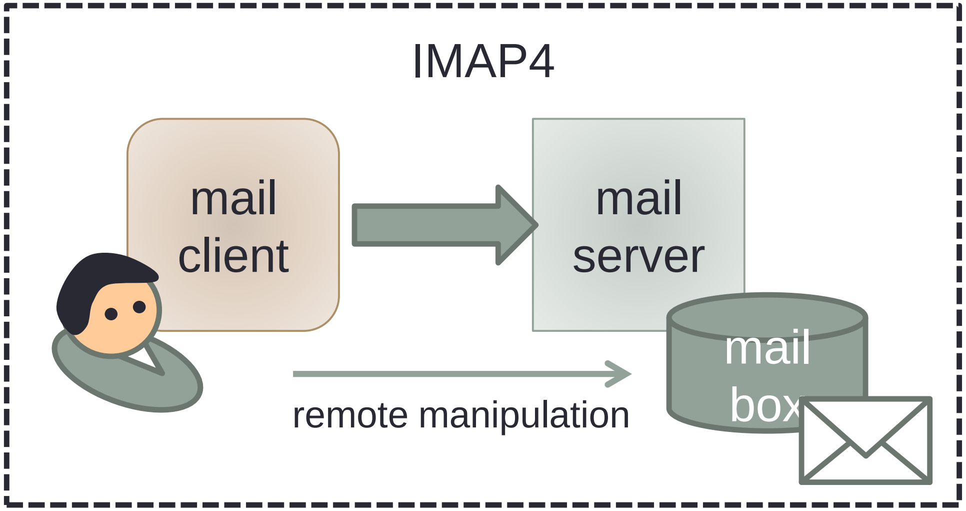
- Directly access mail boxes

- Retrieve mails from remote mail boxes

- POP3
  - Post Office Protocol
  - Retrieve mails to MUA
  - MUA manages mails locally.
  - Need to keep mails in mail boxes for sharing with other MUA

- IMAP4
  - Internet Message Access Protocol
  - Manipulate mails in mail boxes.
  - MUA has mail cache
  - Multiple MUA can chare





Example of POP

```
       +OK Qpopper at mail.sfc.keio.ac.jp starting.
USER hagino
       +OK Password required for hagino.
PASS password
       +OK Welcome hagino!
STAT
       +OK 3 1230
RETR 1
       (the first mail)
DELE 1
       +OK Message 1 marked for deletion
QUIT
       +OK 1 message expunged. Bye!
```

Example of IMAP

```
      +OK IMAP4rev1 server ready
A121 CAPABILITY
      * CAPABILITY IMAP4rev1 AUTH=X509
      A121 OK CAPABILITY completed
A123 LOGIN hagino password
      A123 OK LOGIN completed
A125 LIST ~/Mail/%
      * LIST (¥Marked) "/" ~/Mail/Inbox
      * LIST () "/" ~/Mail/Stuff
      A125 OK LIST completed
A127 DELETE ~/Mail/Stuff
      A127 OK DELETE Completed
A129 SELECT ~/Mail/Inbox
      * 23 EXISTS
      * 12 RECENT
      * OK [UNSEEN 3] Messages 3 is first unseen
      * OK [UIDVALIDITY 5732875] UISs valid
      * FLAGS (¥Answered ¥Flagged ¥Deleted ¥Seen ¥Draft)
      A129 OK [READ-WRITE] SELECT completed
```

```
A131 FETCH 3 BODY[TEXT]
      * FETCH (Body[TEXT]{62}
      (mail body)
      )
      A131 OK FETCH completed
A133 LOGOUT
      * BYE IMAP4rev1 Server logging out
      A133 OK LOGOUT completed
```

Other Topics of Electric Mail

- Mail forwarding
  - Forward mails to other addresses.
  - On UNIX, specify in `~/.forward` (MDA handles)
  - `/etc/aliases` specifies system wide forwading (MTA handles)

- Mailing list
  - Send mails to multiple receivers.
  - Create a mailing list containing receiver's addresses.

- Sorting mails
  - Automatically process received mails and sort them into different folders.
  - Forward mails which match the specified condition.
  - Automatic reply when you are not around. (`vacation` program)
  - Example: procmail

- Encrypted mails
  - Encrypt mail body
  - Header is not encrypted.
  - Example: encrypt mail body with receiver's public key.

- Digital signature
  - Protect from impersonation
  - Make sure mail body is not tampered.
  - Example: encrypt message digest with sender's private key.

Summary of Electric Mail

- One of the most popular TCP protocol
  - Multiple RFC

- Setting up MUA
  - Different protocols for sending and receiving electric mails.
  - SMTP
  - POP/IMAP

- Security issues
  - SPAM