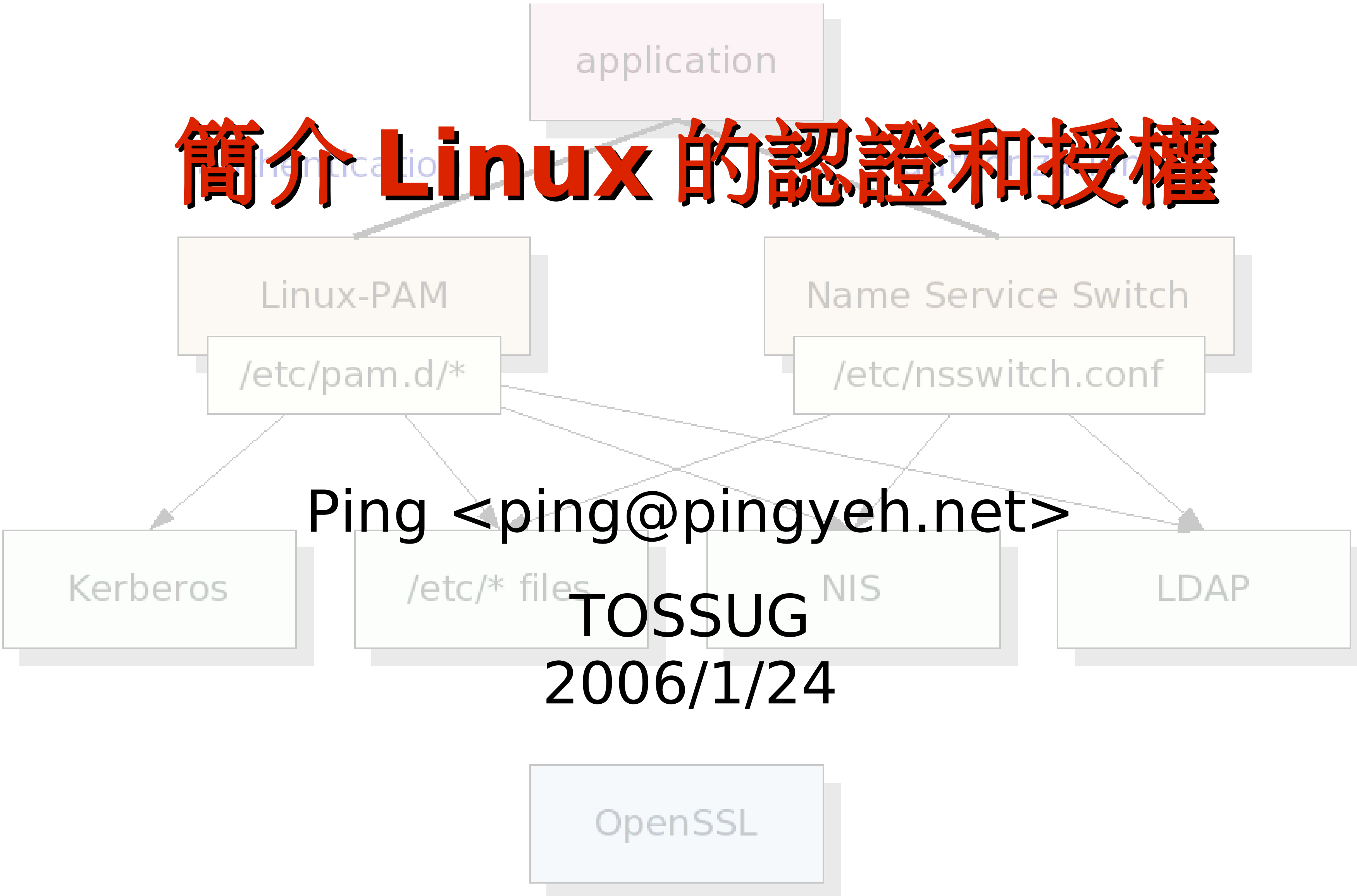
簡介 Linux 的認證和授權

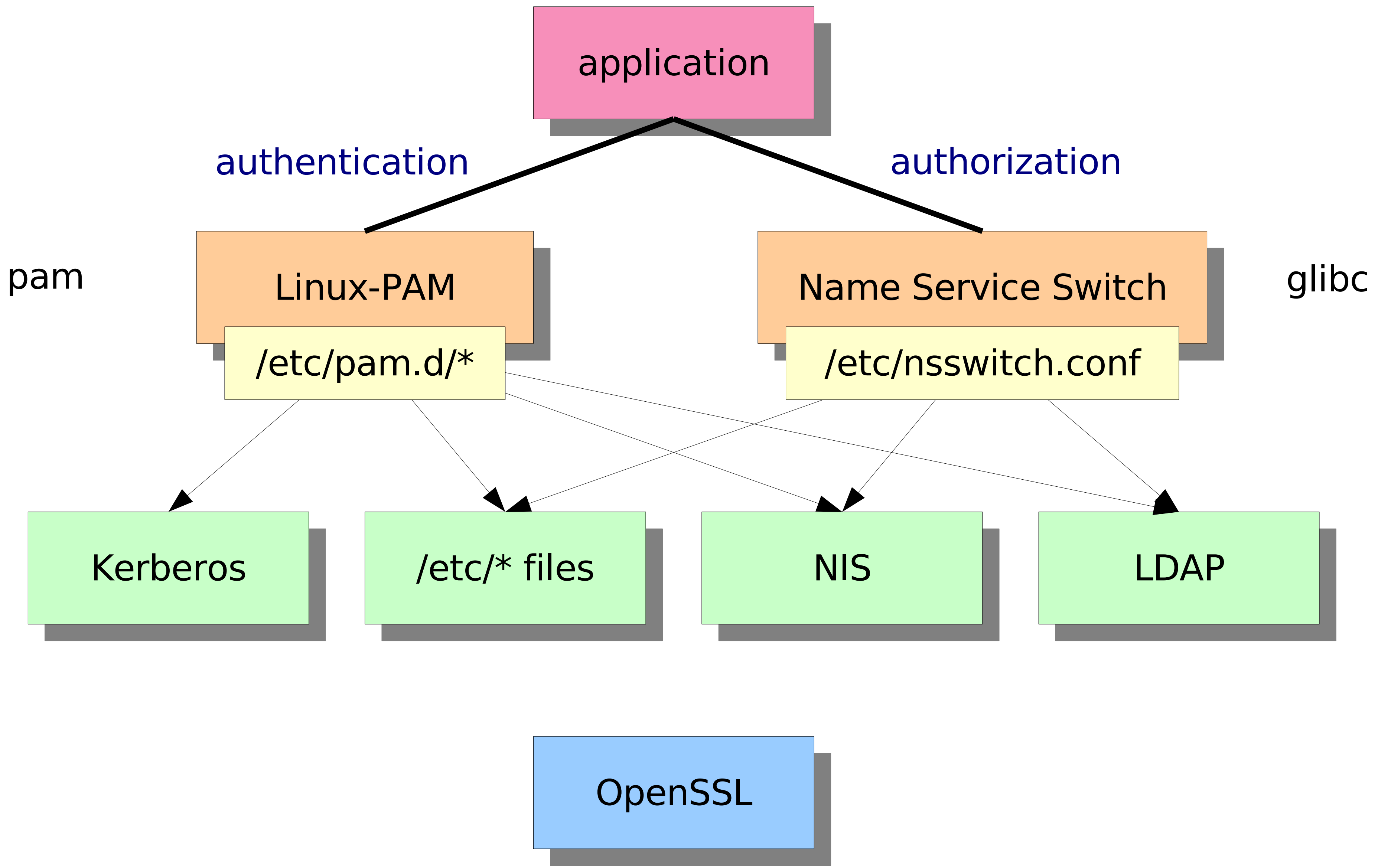


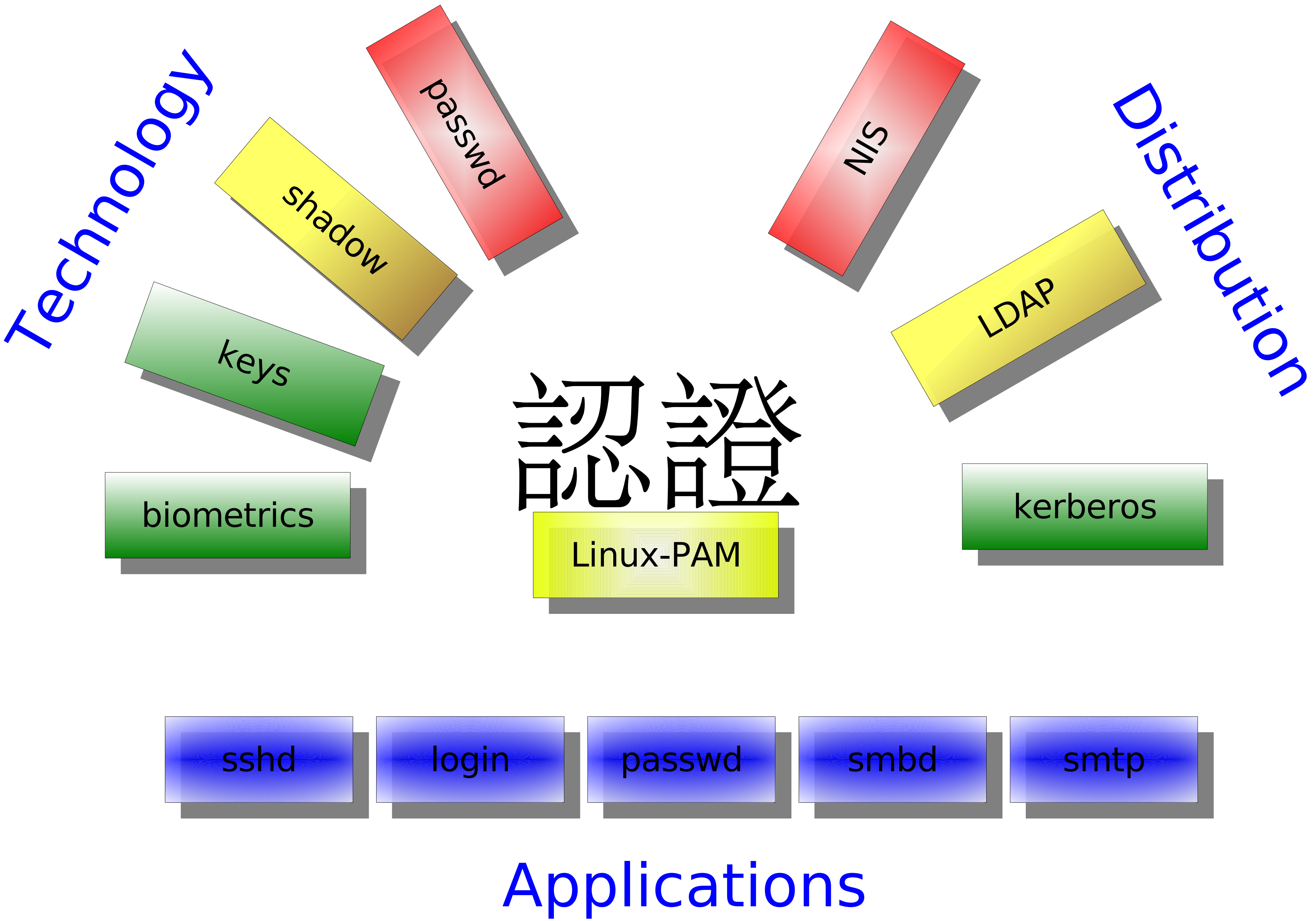
Ping <ping@pingyeh.net>

TOSSUG
2006/1/24

- 認證（Authentication）：建立使用者在系統中的身分
- 授權（Authorization）：決定誰在系統中可以用什麼資源
- 記帳（Accounting）：記錄在系統中誰何時用了什麼

Linux 認證和授權的高層次架構





Technology
passwd
shadow
keys
biometrics
認證
Linux-PAM
NIS
LDAP
kerberos
Distribution
sshd
login
passwd
smbd
smtp
Applications

認證（ Authentication ）

- 建立使用者在系統中的身分
- 方法很多種：
  - 密碼（ username / password ）：證明你知道這個人的密碼（在網路上要注意密碼小偷）
  - 私鑰（ public key / private key ）：證明你擁有這個人的私鑰檔（私鑰檔要小心保管）
  - 生物特徵（ biometric ）：證明你擁有這個人的指紋 / 視網膜 ... （千萬不要被綁架）
- Linux 認證的 de facto 標準： PAM

系統的密碼檔

- 沒用 shadow 的 /etc/passwd

```
-rw-r--r--  1 root root 2035  1月  14 23:54 /etc/passwd
pyeh:fWTsTP0Jr1raM:500:500:Ping Yeh,531,p1,p2:/home/pyeh:/bin/bash
```

DES

  - 問題：[illegible]權資訊混雜，為了 uid 和 gid ，/etc/passwd 必須是 world readable ⇨ 任何有帳號的人都拿得到加密過的密碼！

- shadow ：把密碼放在只有 root 可以讀的 /etc/shadow

```
pyeh:x:500:500:Ping Yeh,531,p1,p2:/home/pyeh:/bin/bash
-rw-------  1 root root 1227  1月   4 23:26 /etc/shadow
pyeh:$1$2Rcx4R5R$RXJcsvmEmQVc1/dvLqD89/:13162:0:99999:7:::
```

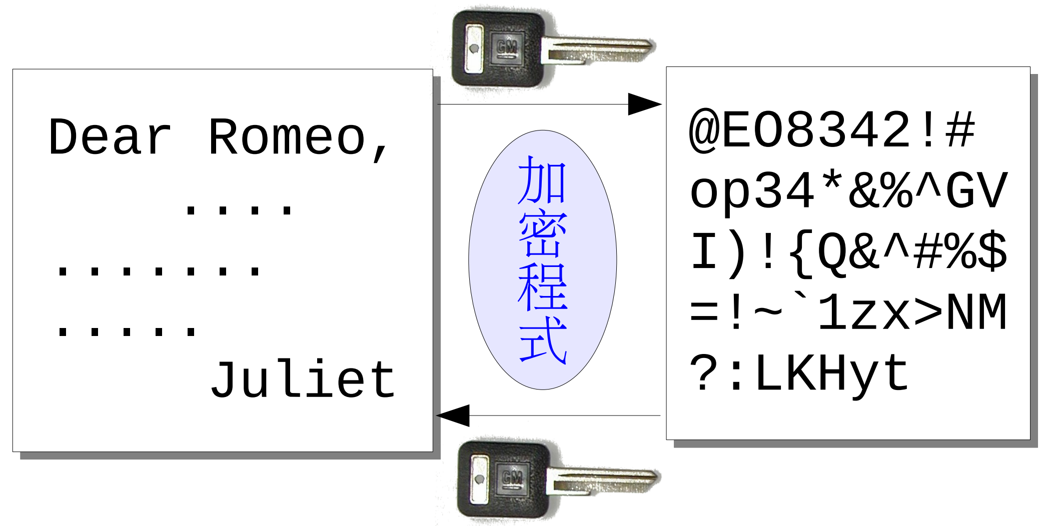
MD5

- 密碼的加密演算法： DES vs. MD5
- 帳號名和密碼可以放在： /etc/* 檔案、 NIS maps 、 LDAP directories 等等地方

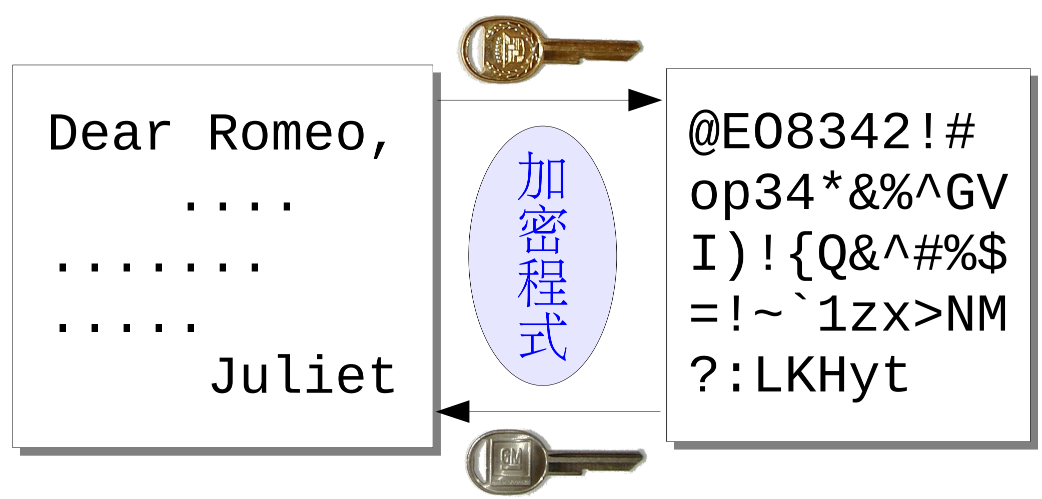
公鑰加密法

文件加密

- 對稱鑰匙加密法
  - 一把鑰匙，用來加密，也用來解密
  - 鑰匙是 Romeo 和 Juliet 共同的小秘密，千萬要保管好！



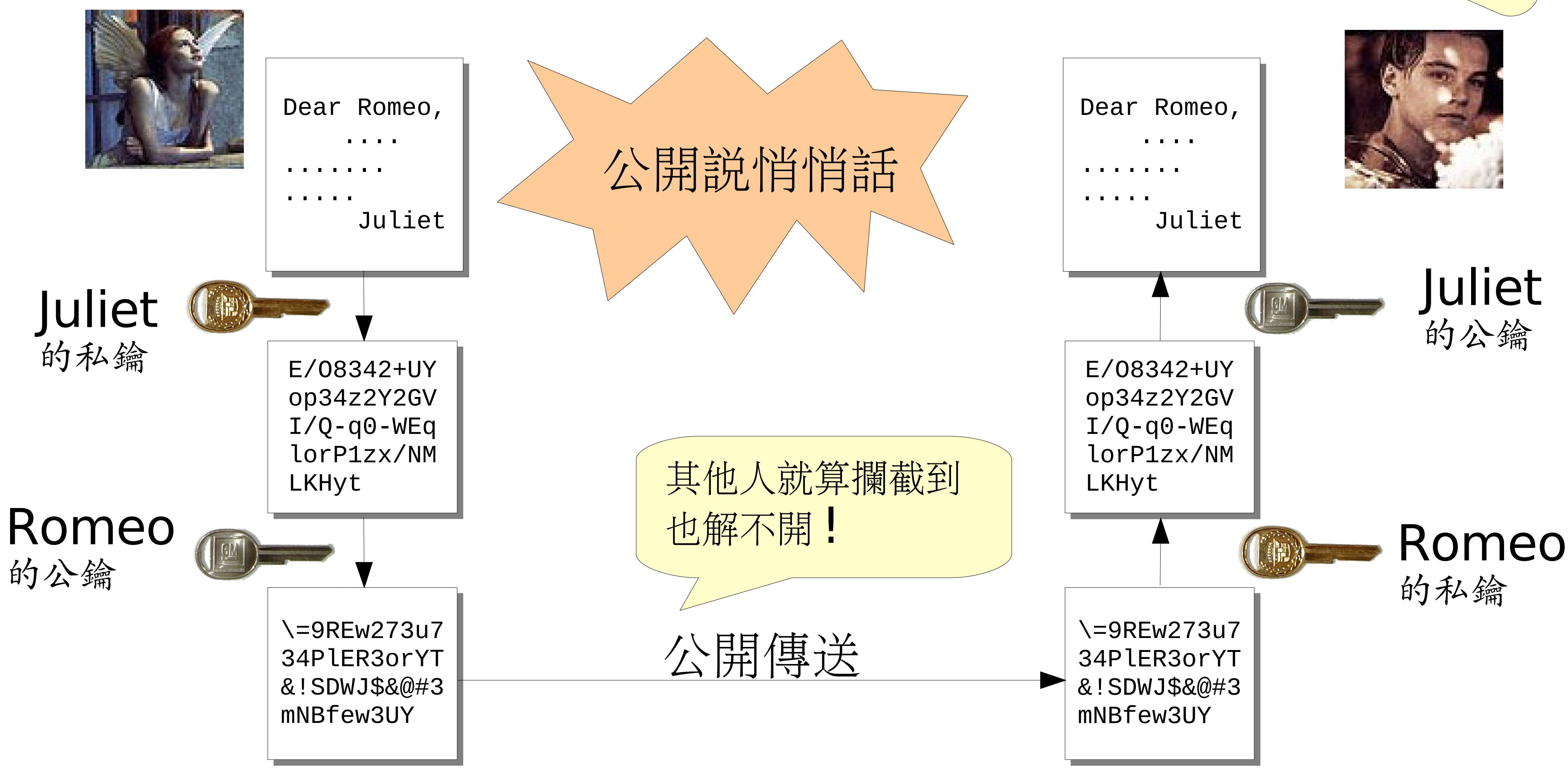
- 不對稱鑰匙加密法
  - 兩把鑰匙，一把加密，另一把解密
  - 鑰匙是自己的秘密， Romeo 不必負責保管 Juliet 的鑰匙



公鑰加密法 (Public Key Cryptography)

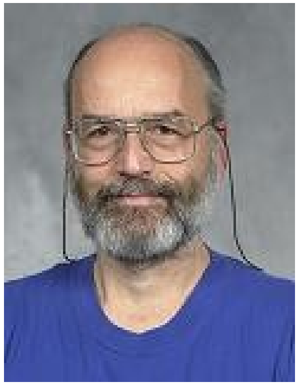
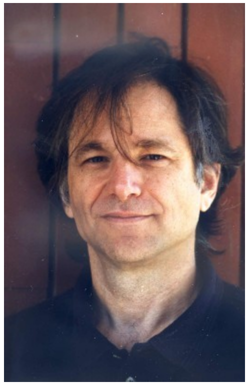
- 使用不對稱加密法，兩把鑰匙，一把自己保管（私 把可以公開給所有朋友（公鑰）

此例只是眾多應用方法之一



最常用的公鑰演算法：RSA

- MIT 的 Ronald Rivest, Adi Shamir 和 Leonard Adleman 在 1977 年發表的公鑰演算法。 1983 年在美國獲得專利 (?!) ，在 2000 年 9 月到期。

- 演算法：
  - 找兩個質數 p 和 q ，求出 *modulus* n = p * q ，令 m=(p-1)(q-1)
  - 找兩個數 e 和 d ： e 和 m 互質， e*d = 1 (mod m)
  - 公鑰 =(n,e) ，私鑰 =(n,d)

RSA 的加解密法

- 加密：
  - 把訊息 m 用私鑰 (n,e) 加密： $c = m^e \bmod n$
  - 用公鑰 (n,d) 解密訊息： $m = c^d \bmod n$

  - 加密和解密的運算是相同的，只是參數不同
  - 可以用數學證明這樣的運算一定可以解出原始的訊息

- 例子： n=8051, e=235, d=67, message m=99

  加密：c = 99^{235} % n = 3683 ，解密： 3683^{67} % n = 99

RSA 演算法的證明

- 問題： $c = m^e \ (mod\ n), \quad m' = c^d \ (mod\ n) \ \Rightarrow \ m' = m \ (mod\ n)$
- 證明：
  - 中國餘數定理：若 p 除 x 和 q 除 x 都餘 y ，則 (pq) 除 x 亦餘 y
  - Fermat/Euler 定理：若 p 是質數且不整除 x ，則 p 除 x^{p-1} 餘 1

$$x = y \ (mod\ p) \ \wedge \ x = y \ (mod\ q) \ \Rightarrow \ x = y \ (mod\ pq)$$
$$p \in prime \ \wedge \ x \neq 0 \ (mod\ p) \ \Rightarrow \ x^{p-1} = 1 \ (mod\ p)$$

  - 令 $ed = k(p-1)(q-1)+1$

$$m' = c^d \ (mod\ n) = m^{ed} \ (mod\ n) = m^{k(p-1)(q-1)+1} \ (mod\ n)$$
$$= m^{k(p-1)(q-1)} \ mod\,(pq) \cdot m\,(mod\ n)$$

  - 由 Fermat/Euler 定理： $m^{k(p-1)(q-1)} \ (mod\ q) = (m^{k(p-1)})^{q-1} \ (mod\ q) = 1 \ (mod\ q)$
  - 對 p 也相同，再用上中國餘數定理，即得

$$m^{k(p-1)(q-1)} \ (mod\ pq) = 1 \ (mod\ pq) = 1 \ (mod\ n)$$
$$\Rightarrow \ m' = 1 \ (mod\ n) \cdot m \ (mod\ n) = m \ (mod\ n)$$

破解 RSA?

- 例：公鑰 =(8051,235) ，破解私鑰
  - 質數分解 8051 ⇨ 8051=97*83=p*q
    ⇨ (p-1)*(q-1)=7872 ⇨ (7872*k+1) 要被 235 整除，求 k
    ⇨ k=2, d=67 ，私鑰 =(8051,67)

- 破解 RSA 主要是質數分解的問題，用大質數增加難度，用 1024 位元甚至 2048 位元的 modulus
  - $2^{1024}=(2^{10})^{102}*2^4 \sim (10^3)^{102}*16 \sim 16*10^{306}$
  - 暴力破解法：把 1 到 sqrt(n) 之間的數一一試除 n ，直到整除為止。 sqrt(n) ~ $4*10^{153}$ ，假設有超級電腦每秒可以試除 10^{15} 次，需要 $4*10^{138}$ 秒 ~ 10^{131} 年，遠大於宇宙年齡 $1.5*10^{10}$ 年。
  - RSA 廣被接受的主因之一：不是不會破，而是沒時間。

PAM

google "pam" 的第三條

google "pam" 的第二條

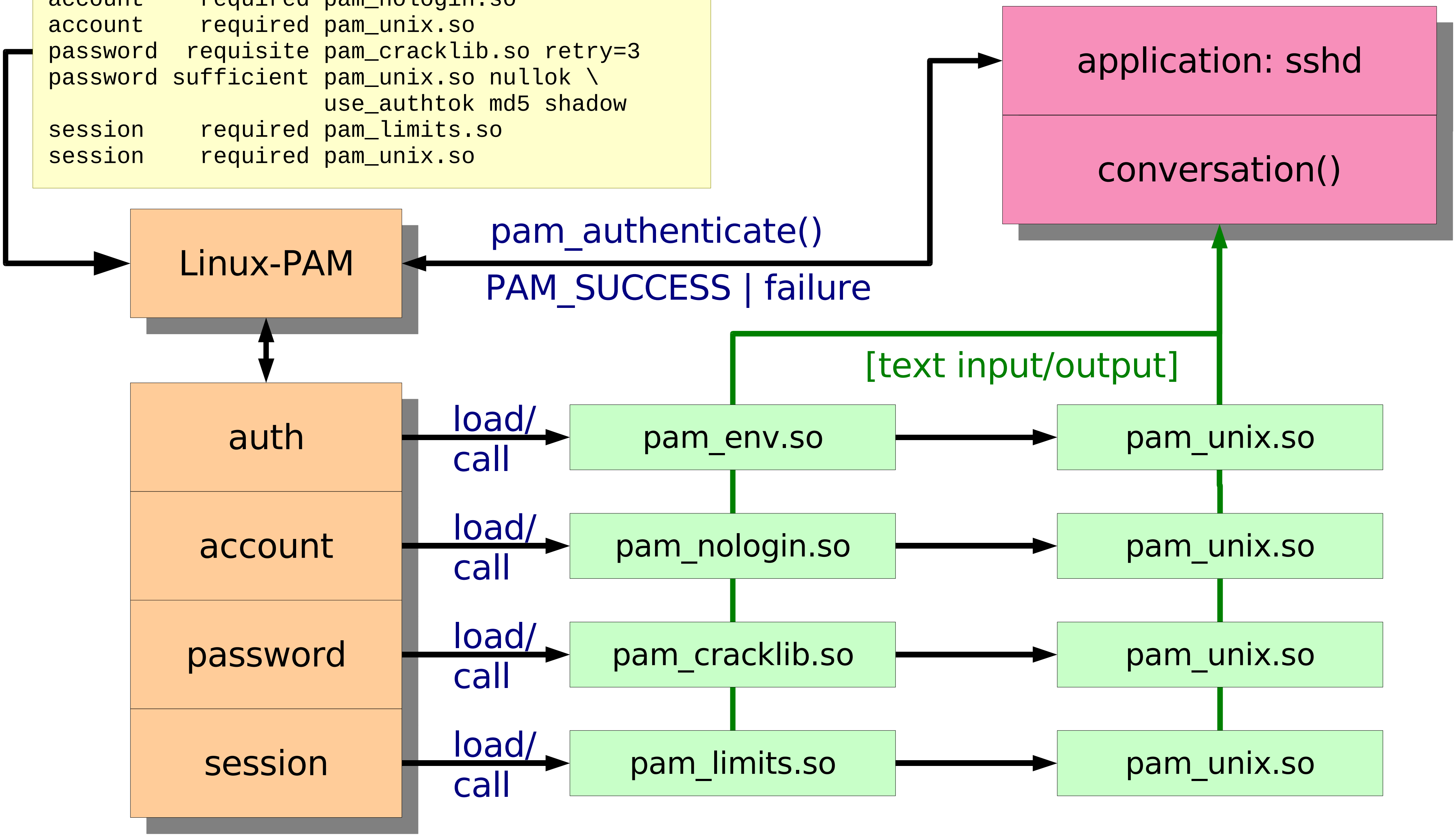
Linux-PAM

- Pluggable Authentication Modules （抽換式認證模組）
- 把『應用程式』和『認證程式庫』分離
  - 應用程式不用知道認證是怎麼做的
  - 系統管理者可以自由調整每個應用程式的認證方式，不用重新啟動應用程式，更不用重開機，只需要修改認證組態檔。
- PAM 的四大功能：
  - authentication management ：認證，並可授與群組等權利
  - account management ：帳號管理，如依系統負載 / 時間 / 登入地點決定是否讓用戶進入
  - session management ：應用開始前和結束後用，如記錄日誌、掛載目錄等
  - passwd management ：改密碼用

PAM 的運作方式

/etc/pam.d/sshd

```
auth       required pam_env.so
auth     sufficient pam_unix.so likeauth nullok
account    required pam_nologin.so
account    required pam_unix.so
password  requisite pam_cracklib.so retry=3
password sufficient pam_unix.so nullok \
                    use_authtok md5 shadow
session    required pam_limits.so
session    required pam_unix.so
```



PAM 組態檔

- /etc/pam.d/<app_name> = app_name 程式的 PAM 組態
- 四個部分： auth, account, password, session
- 模組堆疊（ module stack ）：每個部分可以堆疊多個模組，依序執行，並指定每個模組的效用：
  - required ：必要條件，模組失敗時繼續執行，最後回傳失敗
  - requisite ：強烈必要條件，模組失敗立刻停止執行，回傳失敗
  - sufficient ：充分條件，模組成功會回傳成功，若之前沒有失敗的必要條件就停止執行
  - optional ：可有可無條件，只有在其他模組都沒意見時有用
- 範例： screen

```
$ cat /etc/pam.d/screen
#%PAM-1.0
auth     required        pam_stack.so service=system-auth
```

常見的 PAM 模組

模組	簡介	認證	帳號	應用前後	密碼
pam_unix.so	unix密碼管理	✔	✔	✔	✔
pam_permit.so	不設防（小心使用！）	✔	✔	✔	✔
pam_deny.so	通通擋下	✔	✔	✔	✔
pam_chroot.so	給登入者一個chroot的環境			✔	
pam_cracklib.so	檢查新密碼的强度				✔
pam_limits.so	設定資源的使用限度			✔	
pam_nologin.so	用/etc/nologin阻擋normal user的登入	✔	✔		
pam_securetty.so	用/etc/securetty指定root可以登入的tty	✔			
pam_tally.so	擋掉常登入失敗的帳號	✔	✔		

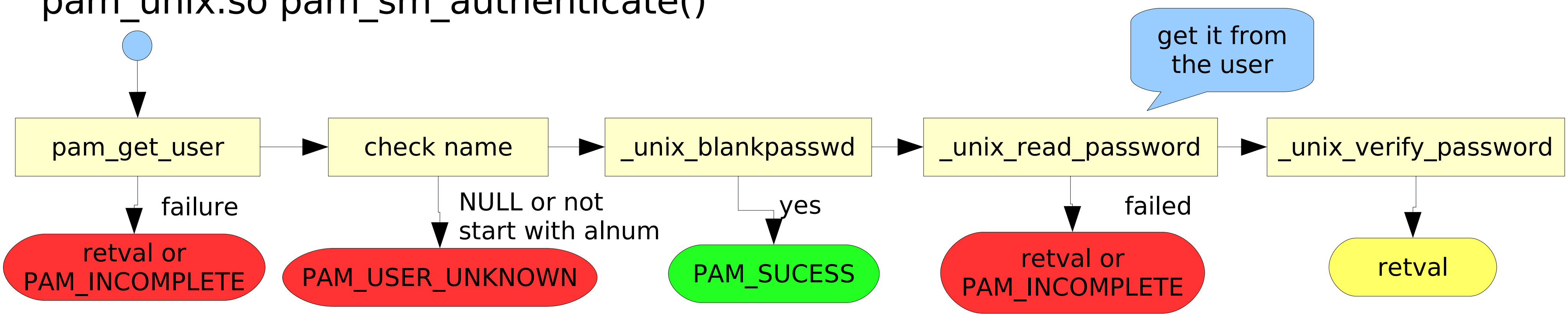
許多模組的設定檔都放在 /etc/security 目錄裡

pam_unix.so

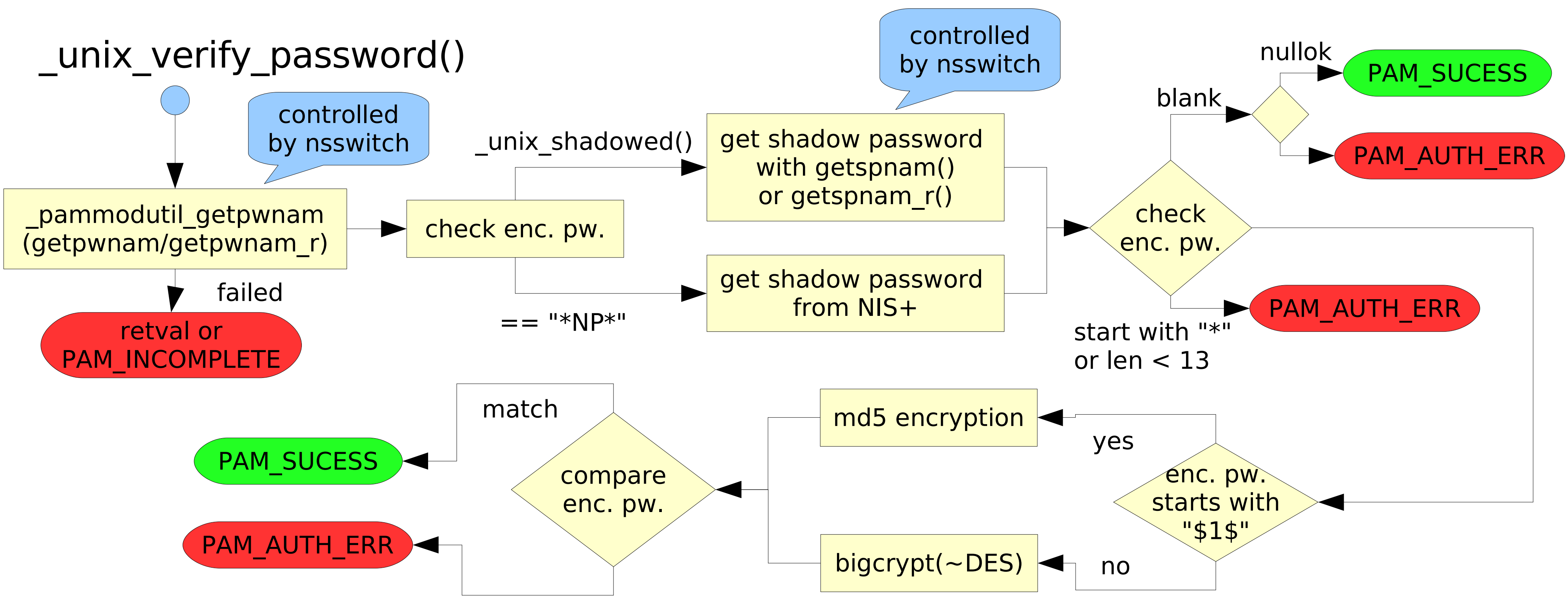
- 認證：用系統呼叫讀取加密的密碼
- 帳號管理：依 shadow 中的密碼生命期，可以提醒用戶換密碼，或在用戶換密碼前不讓用戶進入。
- 應用前後：在系統日誌中記錄用戶名和服務名
- 密碼管理：標準的改密碼程式 (passwd 呼叫 pam_chauthtok() 來完成改密碼的工作)

pam_unix.so 的認證方式

pam_unix.so pam_sm_authenticate()



_unix_verify_password()



例子：主要的 PAM 組態檔 system-auth

- 大部分應用程式的 PAM 組態檔裡都會用 pam_stack.so service=system-auth 語法把 system-auth 堆疊進來

```
$ cat /etc/pam.d/system-auth
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      /lib/security/$ISA/pam_env.so
auth        sufficient    /lib/security/$ISA/pam_unix.so likeauth nullok
auth        required      /lib/security/$ISA/pam_deny.so

account     required      /lib/security/$ISA/pam_unix.so
account     sufficient    /lib/security/$ISA/pam_succeed_if.so uid < 100 quiet
account     required      /lib/security/$ISA/pam_permit.so

password    requisite     /lib/security/$ISA/pam_cracklib.so retry=3
password    sufficient    /lib/security/$ISA/pam_unix.so nullok use_authtok md5
shadow
password    required      /lib/security/$ISA/pam_deny.so

session     required      /lib/security/$ISA/pam_limits.so
session     required      /lib/security/$ISA/pam_unix.so
```

哪些應用程式會用 PAM 認證？

- 用 ldd /path/to/executable | fgrep pam 檢查
- sshd, smbd, pppd, cupsd, ...
- passwd, chfn, chsh, sudo, su, login, xscreensaver, ...

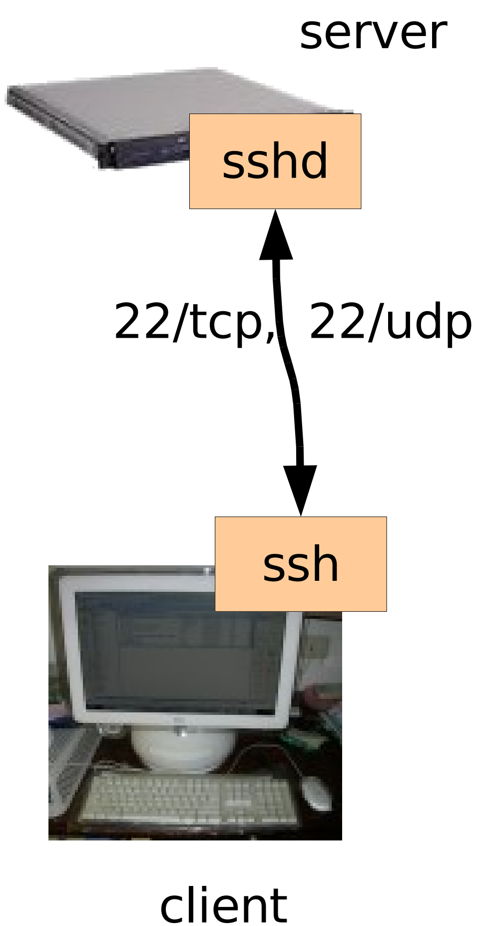
安全取用網路資源

- 資源管理者 / 擁有者：
  - 我怎麼知道他真的是他？
    「在網際網路上，沒人知道你是狗。」
    身份認證 (authentication)
  - 他為什麼可以用我的電腦？
    授權 (authorization)
- 使用者：
  - 我怎麼知道我連上的不是詐騙集團用劫來的 IP 架的站？
    主機認證、服務認證

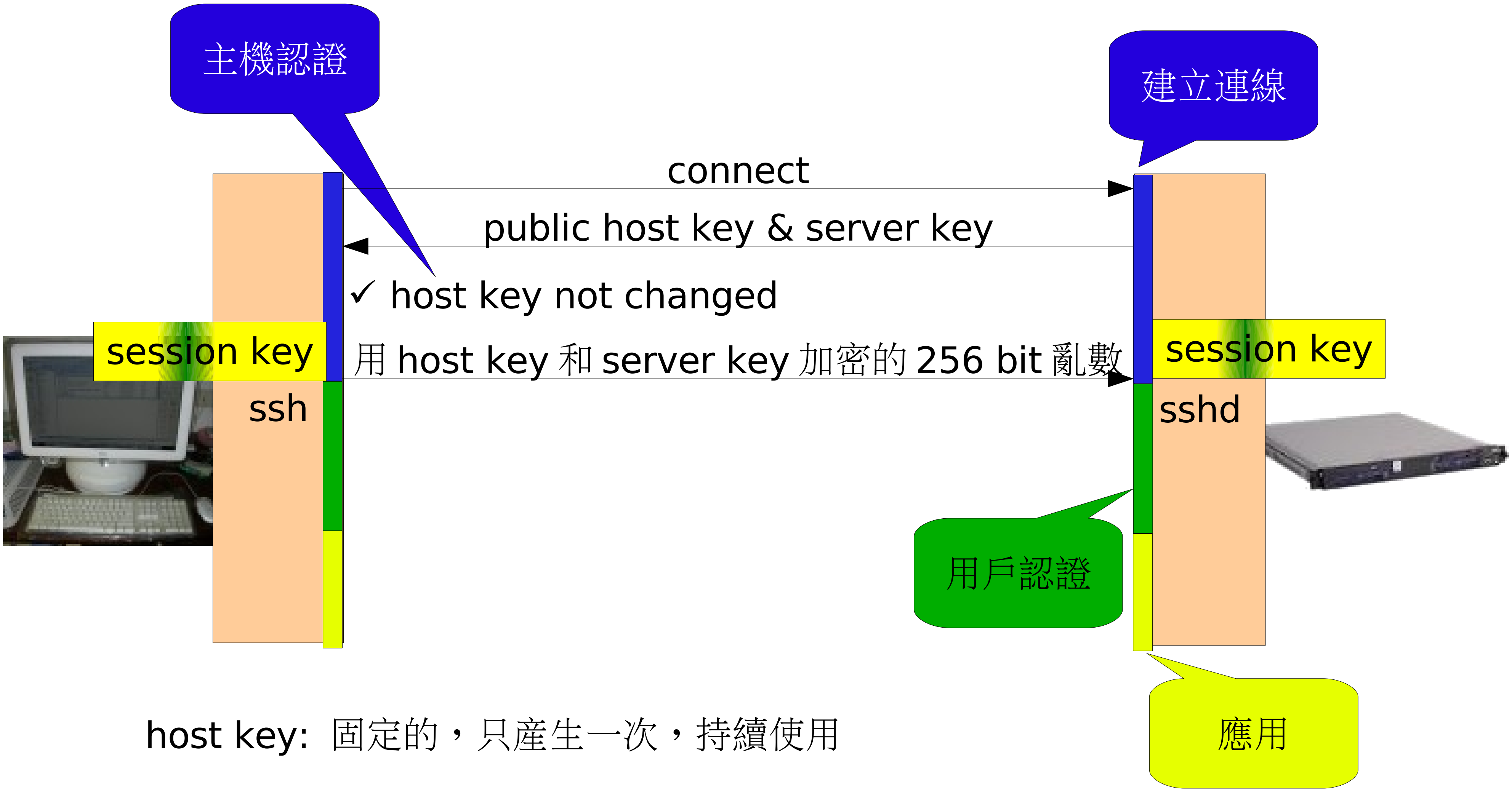
Peter Steiner, page 61 of July 5, 1993 issue of The New Yorker, (Vol.69 (LXIX) no. 20)

Secure Shell: ssh

- 避免讓密碼曝露在網路上，能加密封包，還可以 forward X11 或其他 TCP/IP 封包
- ssh protocol version 1 ：三種認證方式
  - 不需密碼的登入： host* 設定檔
  - 不需密碼的登入： RSA 認證協定
  - 用密碼登入： pam
- ssh protocol version 2 ：和 v1 類似，但增加了加密的演算法，並支援 DSA 認證



建立 ssh 連線



host key: 固定的，只產生一次，持續使用

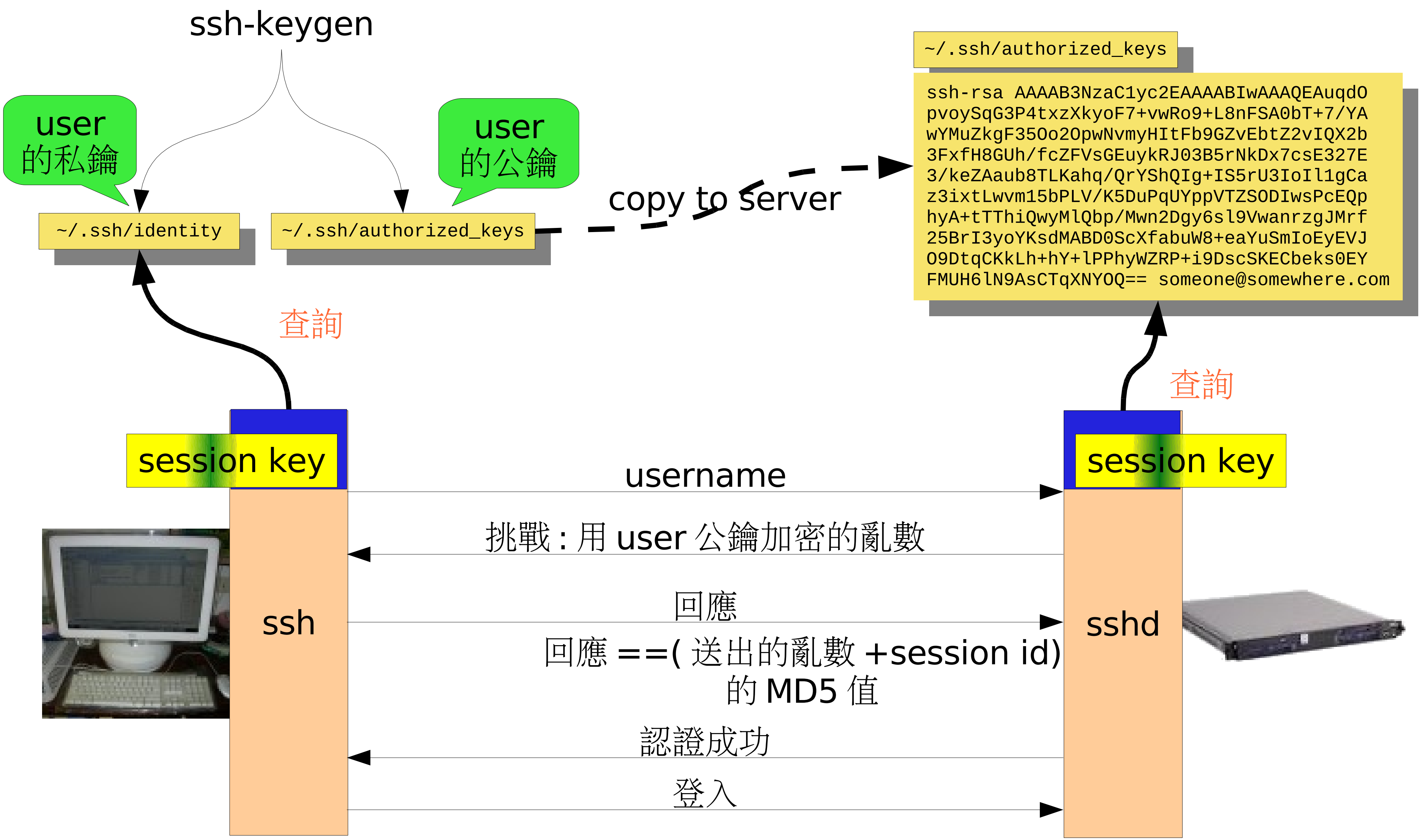
server key: 浮動的，用過約一個小時後就重新生一個

連線建立後，所有通訊都用 session key 加密

用設定檔『認證』

- client 的 host key 沒列在 /etc/ssh/ssh_known_hosts 或 ~/.ssh/known_hosts 中 ⇨ ✗ (假 host)
- client 的 hostname 有列在 /etc/hosts.equiv 或 /etc/ssh/shosts.equiv 中 + 相同 username ⇨ ✓
- client 的 hostname 和 username 有列在 ~/.rhosts 或 ~/.shosts ⇨ ✓
- 不很安全

ssh: 用 RSA 公私鑰認證



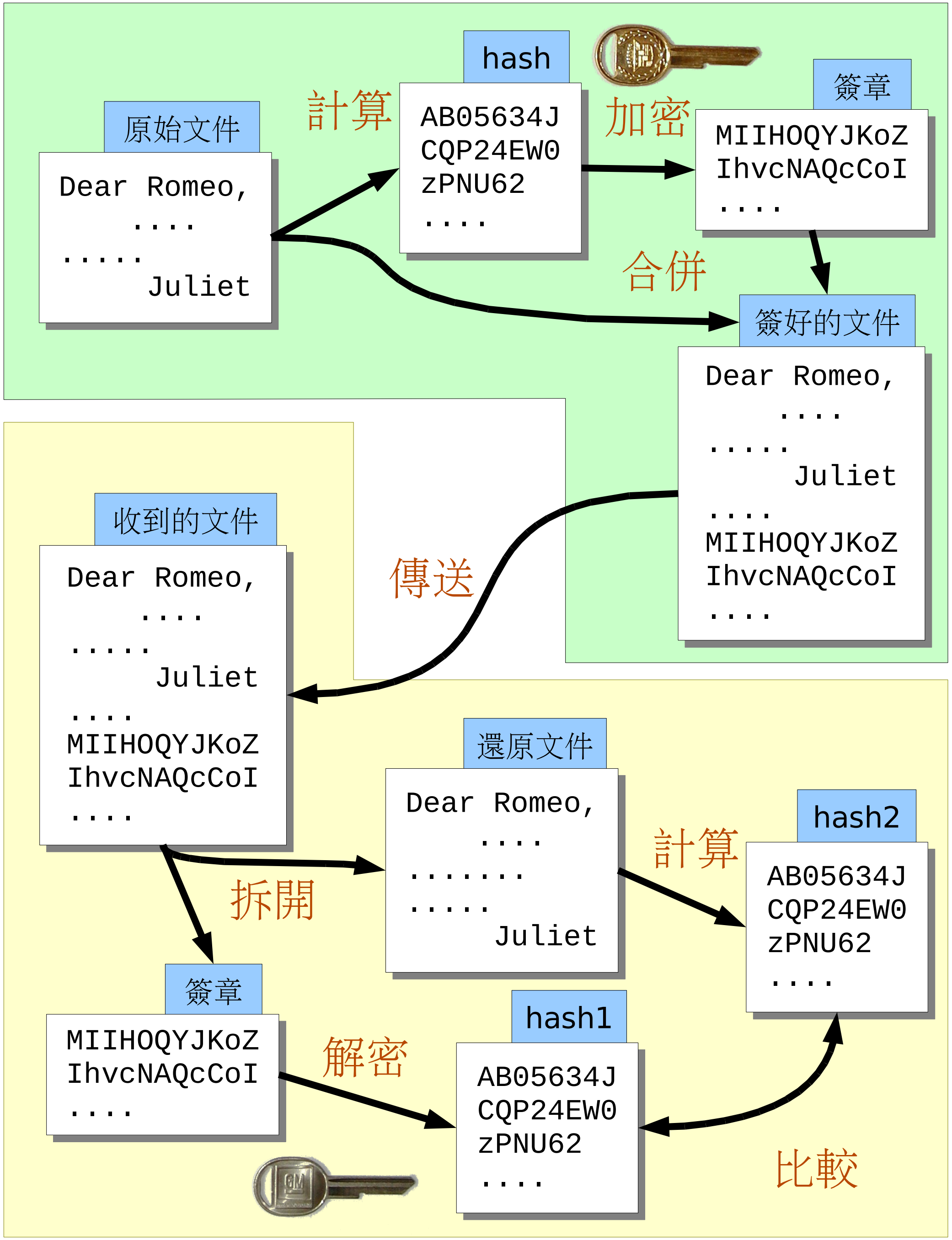
公鑰好不好？

公鑰真不真？

公鑰要不要？

數位簽名 (digital signature)

- 簽名：用私鑰加密過的文件 hash
  - 不同文件的簽名就不同
  - 有簽名的文件 = 文件 + 簽名
- 收到簽名文件後，可以驗證文件的完整性 (integrity)
  - 簽名後有人改過的話， hash 會不一樣，除非那人有 Juliet 的私鑰重新簽名
- 簽名機制有效的前題：你拿到的公鑰是真的！



用電子憑證來認證人和主機

- 什麼是認證？
  - 確認某人／某物的真實性
- 什麼是電子憑證？
  - 電子版本的「身份證」或「護照」
  - 個人憑證中包括以下欄位：發證機構、有效期、持證人的名字 ... 等等
  - 由發證機關以數位簽章技術在證件上「蓋章」完成

- 你檢查一張憑證，如果憑證完整、你信任憑證的發證機構、憑證也沒過期，你就可以閉一隻眼相信這個持證者了。

公鑰可信嗎？

- 怎麼知道拿到的公鑰是真的？
  - 我信任的 A 說我拿到的 B 的公鑰是真的，我就相信
- 兩種最常見的 A ：
  - web of trust ： A 是我信任的朋友，或者我信任的 Z 說他信任 A ，或 ... ： Pretty Good Privacy (PGP)
  - A 是個有公信力的機構： Public Key Infrastructure (PKI)
- 想法類似：我拿到的 B 的公鑰上必需有我信任的人的簽名。
- 作法差很多。

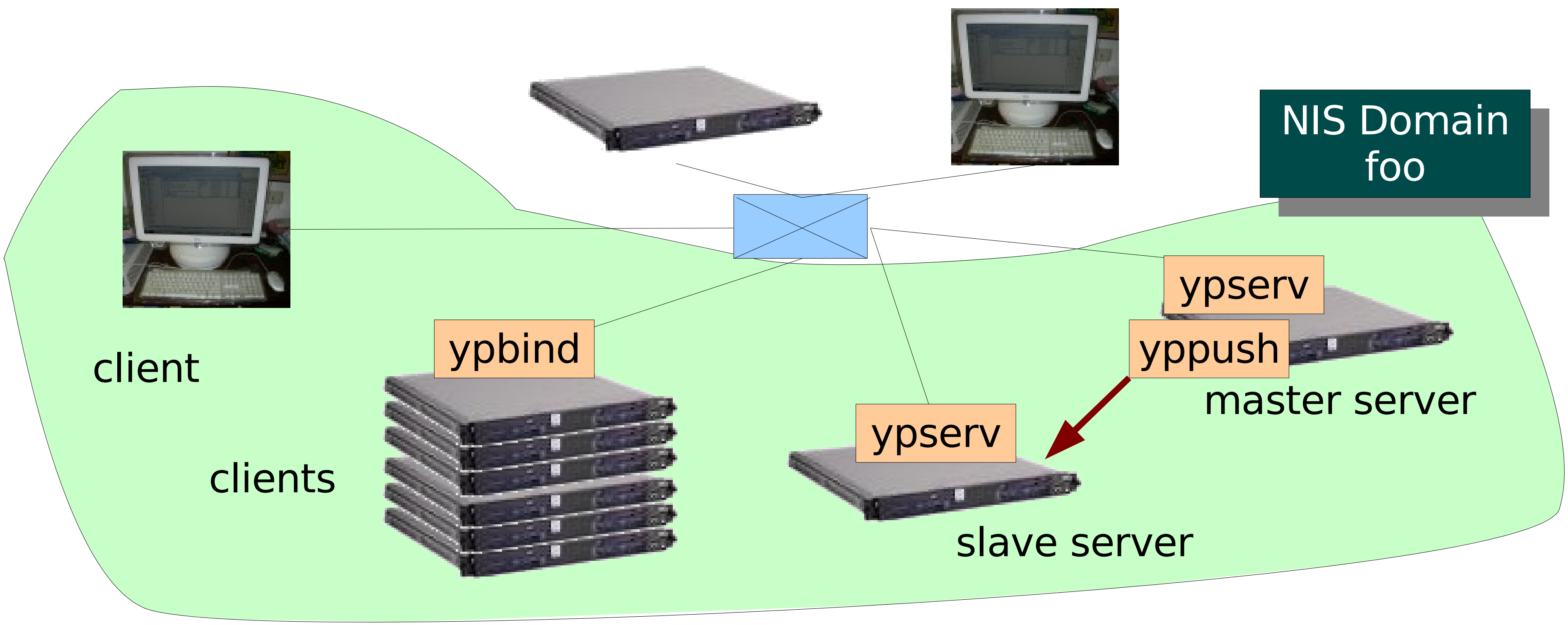
憑證 (certificate)

- PGP 公鑰：由他人簽名過的公鑰
  - 公鑰放在 key server 上（如 pgp.mit.edu ）
  - 在 key signing party 上驗明正身後交換 fingerprint ，回去再簽
  - 一個公鑰可以由許多人簽
  - 見 GnuPG Keysigning Party Howto

- PKI 憑證：由發證機構 (CA) 簽名過的身分證明文件
  - 包含持證人的公鑰和姓名等其他資訊
  - 可以當作公鑰散佈和使用
  - 是 Public Key Infrastructure 的一部分
  - 許多認證機制都使用 PKI ： LDAP, Kerberos

NIS

- Network Information Service (舊名 Yellow Pages)
  - 由 Sun Microsystems 設計的網路資訊服務， client-server 架構
  - 提供 passwd 、 group 、 hosts... 等等資訊給同一個 NIS domain 的 host
  - 在叢集環境中常和 NFS 搭配來統一帳號和密碼。



設定 NIS

- server side ：
  - ypinit
  - "ypmake"
- client side: authconfig ，勾選 NIS 然後填入 NIS domain 名稱和 NIS server 的主機名即可。
  - ypcat
- NIS 內建在 glibc 的 nsswitch 中

LDAP

- Light-weight Directory Access Protocol
- Many server softwares
  - Openldap, Sun iplant, CA etrust, MS Active Directory, Novell NDS, IBM, Oracle, ...
- Many client softwares
  - Authentication: pam, plone, apache, samba
  - User information: NSS
  - Mail relay: sendmail
  - Address book: mutt, mozilla
  - Roaming: netscape
  - And many others

LDAP (cont.)

- 可以認證，也可以做授權用
- 基本上，可以很容易把 /etc 中的設定檔移植進 LDAP (PADL 的 migration scripts)
- LDAP server 的設定和背後的觀念比較複雜，下次再談

Kerberos （地獄看門三頭狗）

- 在開放的網路上認證使用者
  - 之前所講的認證方式，都需要使用者在網路上送出密碼 (ssh?)

時時勤拂拭，莫使惹塵埃

  - 除了改密碼的時候，密碼不會在網路上以任何形式出現

本來無一物，何處惹塵埃！

- kinit ：拿到一個 TGT(ticket granting ticket) ，密碼不離開電腦 (類似對稱鑰匙的做法)
- 之後用 TGT 拿各種服務的 ticket

參考資料

- The Linux-PAM System Administrators' Guide
- PGP FAQ: http://www.faqs.org/faqs/pgp-faq/part1/
- GnuPG Keysigning Party Howto: http://www.cryptnet.net/fdp/crypto/gpg-party.html
- http://en.wikipedia.org/wiki/Web_of_trust
- Crypto FAQ, RSA Laboratories, http://www.rsasecurity.com/rsalabs/node.asp?id=2152
- A proof of RSA encryption/decryption: http://pajhome.org.uk/crypt/rsa/maths.html