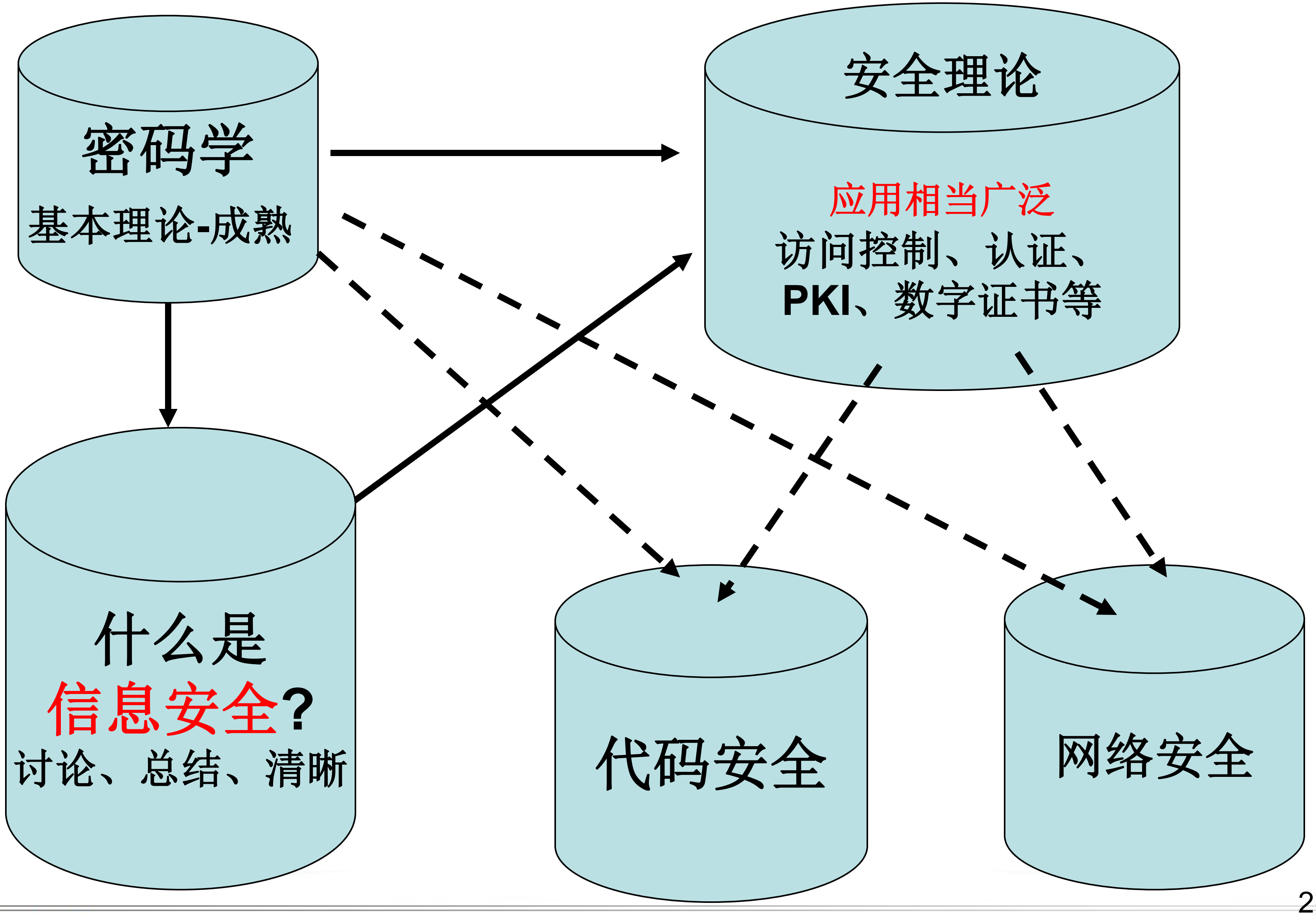
Information Security 09

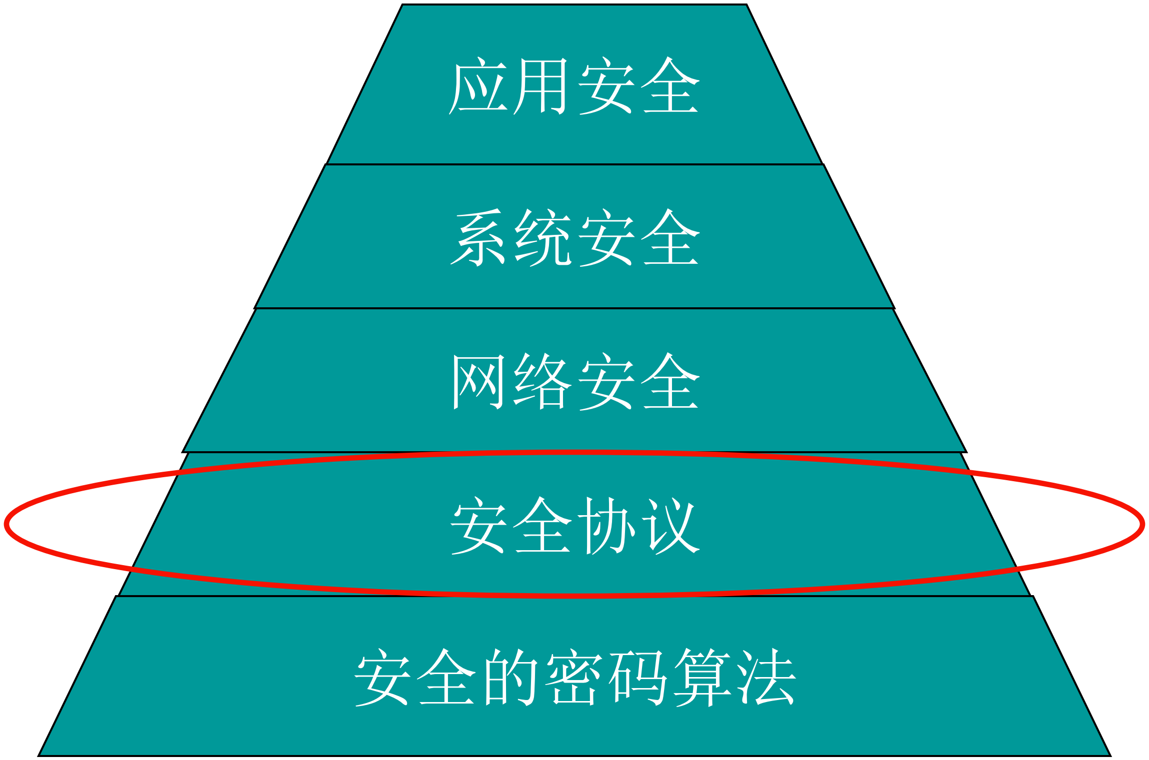
Authentication

Chapter14 and supplements

内容间的联系


密码学
基本理论-成熟
安全理论
应用相当广泛
访问控制、认证、
PKI、数字证书等
什么是
信息安全?
讨论、总结、清晰
代码安全
网络安全

Review: 安全层次



Outline of Talk

- Definitions
- Passwords
  - Unix Passwords
  - One time passwords
- Challenge-response techniques

Definitions

Authentication:

- A *claimant* tries to show a *verifier* that the claimant is as declared
  – identification

- Different from *message authentication*
  – which enables the recipient to verify that messages have not been tampered with in transit (data integrity) and that they originate from the expected sender (authenticity).

Definitions

Authentication

- 消息认证/报文的鉴别
- 身份认证
  - Message authentication has no *timeliness*
  - Entity authentication happens in *real time*
- 双向和单向认证

A good authentication scheme is…

- ***Sound:*** an honest party can successfully authenticate him/herself
- ***Non-transferable***
- ***No impersonation***
- All this is true even when
  - A large number of authentications are observed
  - Eve is able to spoof/eavesdrop
  - Multiple instances are run simultaneously

Basis of Authentication

- Something *known* - passwords, PINs, keys...
- Something *possessed* - cards, handhelds...
- Something *inherent* - biometrics

PINs and keys

- Long key on physical device (card), short PIN to remember
- PIN unlocks long key
- Need possession of both card and PIN
- Provides ***two-level*** security

Outline of Talk

- Definitions
- Passwords
  - Unix Passwords
  - One time passwords
- Challenge-response techniques

Basic password authentication

- Setup
  - User chooses password
  - Hash of password stored in password file
- Authentication
  - User logs into system, supplies password
  - System computes hash, compares to file

Passwords -weak authentication

- Usually fixed
- Stored either in the clear, or "encrypted" with a OWF
- *Rules* reduce the chance of easy passwords
- *Salt* increases search space for a dictionary attack

- There are many examples using password-based authentication
  - how to manage passwords

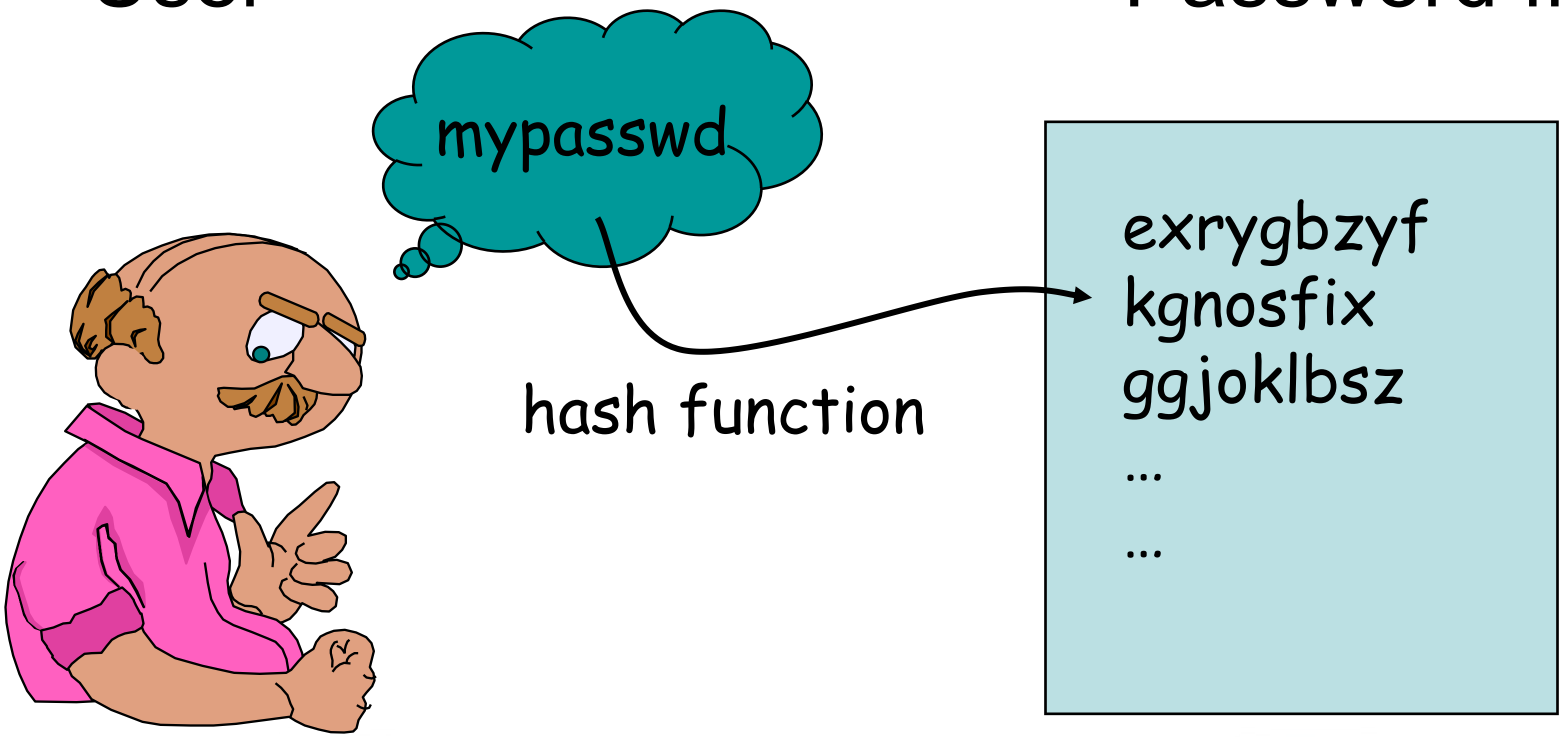
Example: UNIX passwords

/etc/passwd
/etc/shadow

Username: password: UID : GID: USERINFO: HOME: SHELL

User

Password file



Attacks on password schemes

- ***Replay*** of fixed passwords
- Exhaustive ***search***
  - 8 character password has 40-50 bits
- More directed ***dictionary*** *attacks*
  - Crack - widely available tool for doing this
  - Online dictionary attack
    - Guess passwords and try to log in
  - Offline dictionary attack
    - Steal password file, try to find p with hash(p) in file

Dictionary Attack – some numbers

- Typical password dictionary
  - 1,000,000 entries of common passwords
    - people's names, common pet names, and ordinary words.
  - Suppose you generate and analyze 10 guesses per second
    - This may be reasonable for a web site; offline is *much* faster
  - Dictionary attack in at most 100,000 seconds = 28 hours, or 14 hours on average
- If passwords were random
  - Assume six-character password
    - Upper- and lowercase letters, digits, 32 punctuation characters
    - 689,869,781,056 password combinations.
    - Exhaustive search requires 1,093 years on average

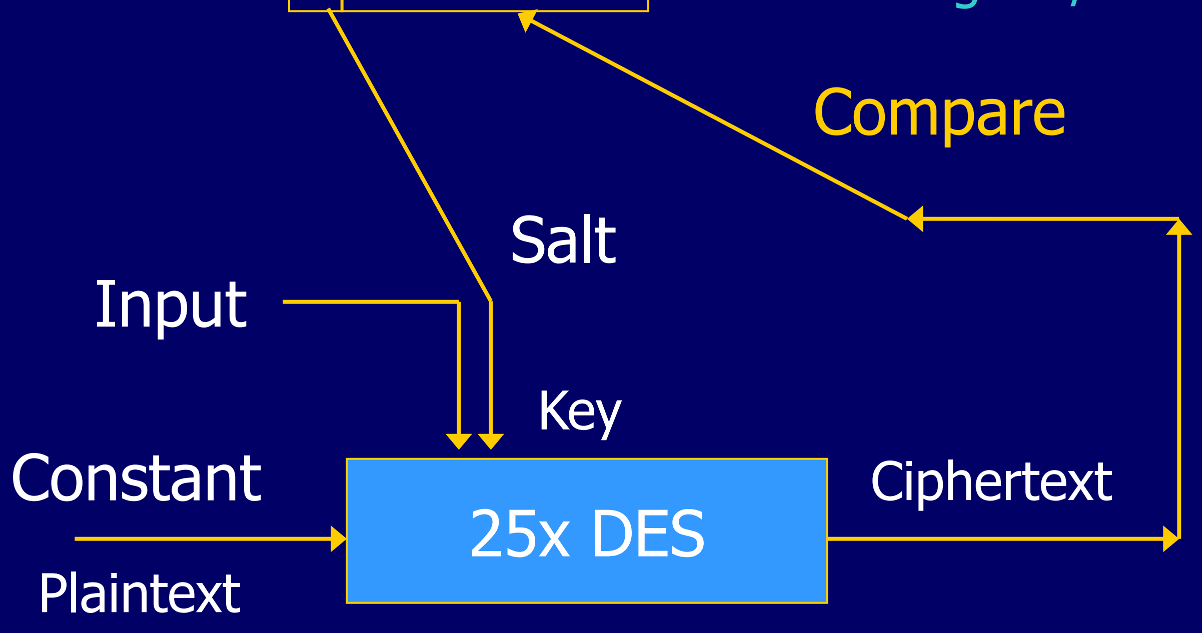
UNIX passwords

- User password serves as key to encrypt known plaintext (64 bit zeroes)
- Encryption - modification of DES, iterated 25 times
- 12 bit salt added - total 64 + 12 = 76 bits
  - Salt taken from system clock, [a-zA-Z0-9./]
  - Alters expansion function of DES
  - char *crypt(const char *key, const char *salt);

Salt(使用加密技术生成的随机数)

◆Unix password line

walt:fURfuu4.4hY0U:129:129:Belgers:/home/walt:/bin/csh



When password is set, salt is chosen randomly

Advantages of salt

- Without salt
  - Same hash functions on all machines
    - Compute hash of all common strings once
    - Compare hash file with all known password files
- With salt
  - One password hashed 2^{12} different ways
    - Precompute hash file?
      - Need much larger file to cover all common strings
    - Dictionary attack on known password file
      - For each salt found in file, try all common strings
- Now, SHA1 is recommended

Summary: Passwords

- Easy to implement
- Easy to use

• But, The Weakest form of Authentication

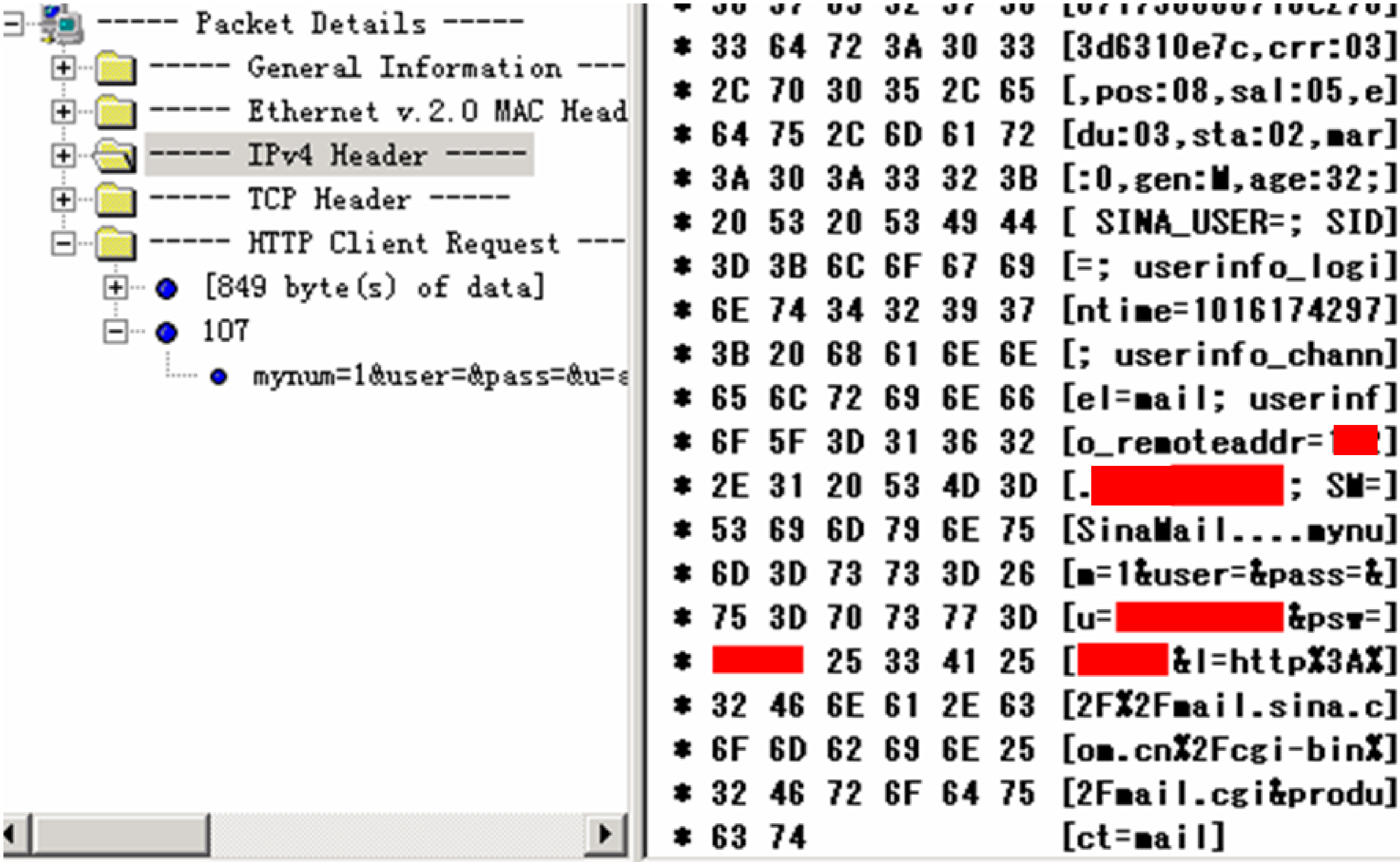
- ???
- 窃取A的password，将在很长一段时间拥有A的权限，直到A发现
- 特别的，网络环境下远程认证
  - 远程登录Unix主机，password传递形式？

基于口令的认证+明文传输！！！

- Telnet远程登录
  - 逐个字母发送，明文方式
- POP3邮件登录
- Ftp服务
- ……

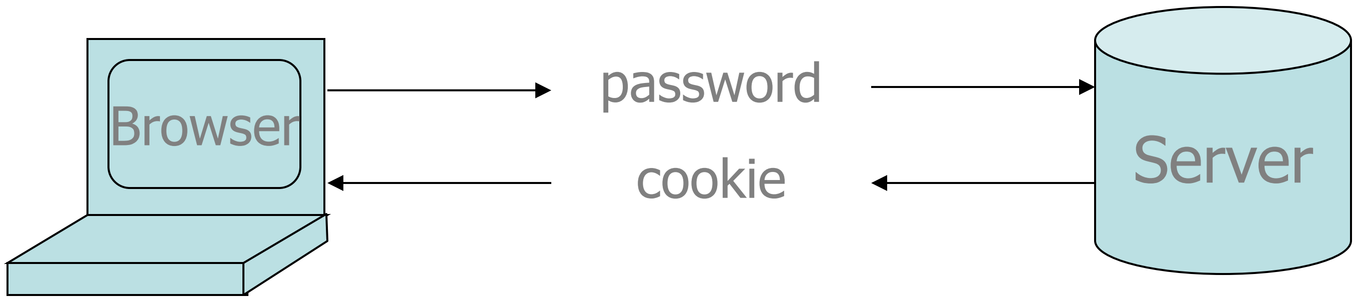
- 嗅探（Sniffer）相当容易

认证例子：sina的邮件登录



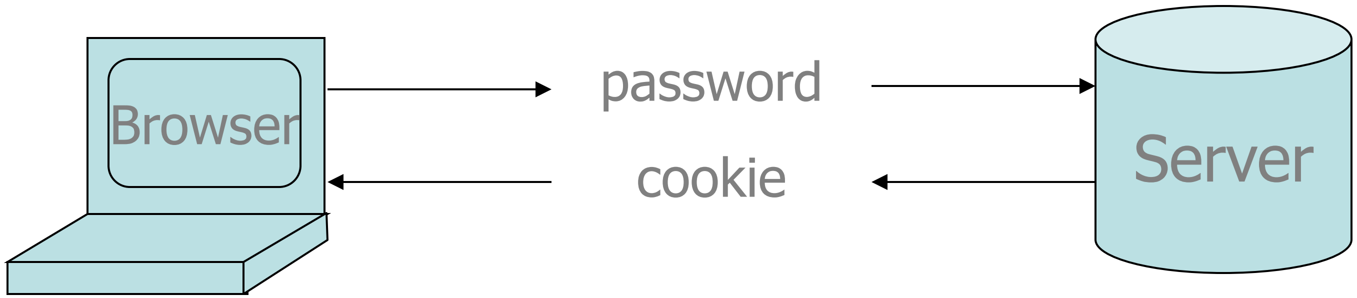
网络环境下的认证

- 基本假设：
  - C/S 模型



- 多server，
  - 同样的口令，还是不同的？
- 单向->双向，
  - Server需要对每个user出示独特的口令吗？

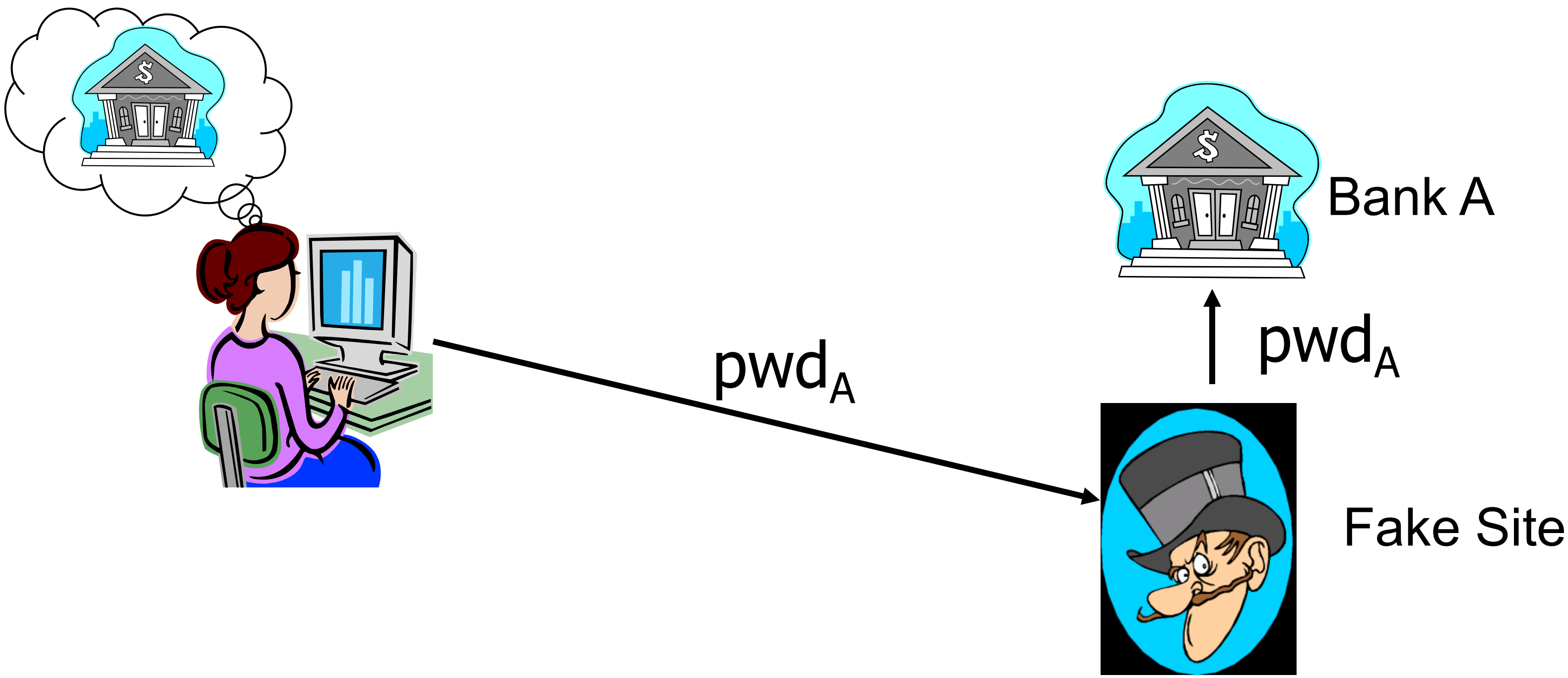
Authentication Problems



- Problems
  - Network sniffing → Encryption, but key distribution problems
  - Malicious or weak-security website → OWF, hashing
    - Phishing
    - Common password problem
    - Pharming – DNS compromise

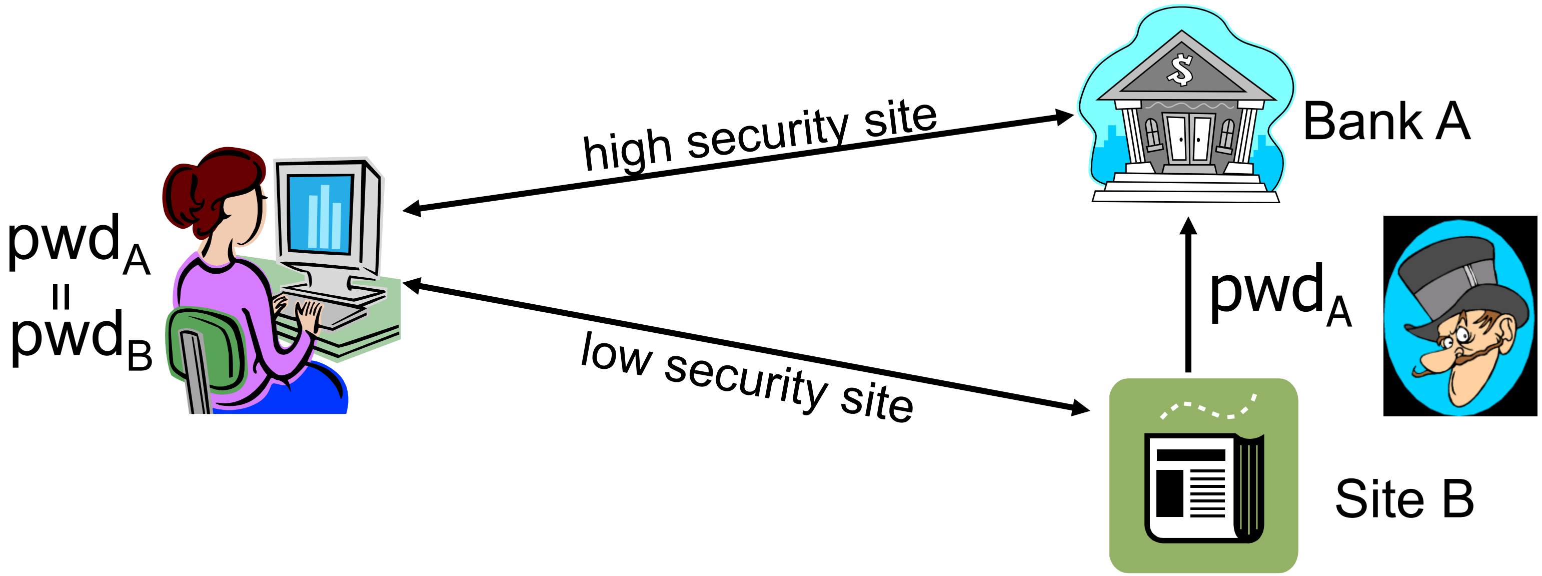
    (Phishing and Common password problem: next few slides)
  - Malware on client machine
    - Spyware
    - Trojan Horse

Password Phishing Problem



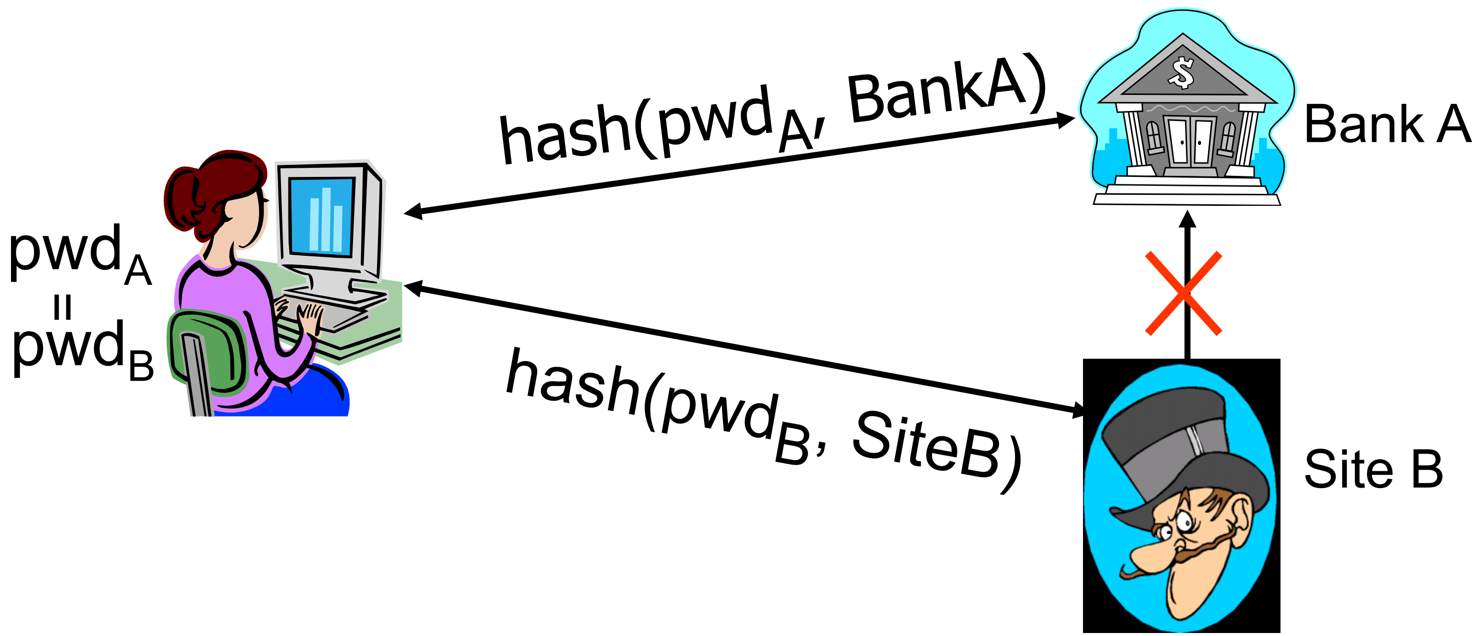
- User cannot reliably identify fake sites
- Captured password can be used at target site

Common Password Problem



- Phishing attack or break-in at site B reveals pwd at A
  - Server-side solutions will not keep pwd safe
  - Solution: Strengthen with client-side support

Defense: Password Hashing



- Generate a unique password per site
  - $HMAC_{fido:123}$(banka.com) ⇒ Q7a+0ekEXb
  - $HMAC_{fido:123}$(siteb.com) ⇒ OzX2+ICiqc
- Hashed password is not usable at any other site
  - Protects against password phishing
  - Protects against common password problem

Outline of Talk

- Definitions
- Passwords
  - Unix Passwords
  - One time passwords
- Challenge-response techniques

One time passwords

- Avoids *replay attacks*
- ***Shared lists*** - pre-distribute list
- ***Sequentially updated*** - create next password while entering current password
- ***Based on one way functions*** - Lamport's scheme...

Lamport's One Time Passwords

- 1981, by Lamport
- Initialization
  - User has a secret *w*
  - Using a OWF *h*, create the password sequence:

$$w, h(w), h(h(w)),\ldots,h^t(w)$$

  - Bob knows only $h^t(w)$
- Authentication：
  - Password for i^{th} identification is:

$$w_i = h^{t-i}(w)$$

S/KEY One-Time Password System

- Based on Lamport's OTP
- Initialization
  - User has a secret: *w*, *seed* (non-secret)
  - Using a OWF *h*, create the password sequence:

$$w,\ h(w,seed),\ h(h(w),\ seed),\ldots,h^t = h(h^{t-1},\ seed)$$

  - Bob server knows: *seed*, Sequence#, h^t
- Authentication：
  - Password for i^{th} identification is:

$$w_i = h^{t-i} = h(w_{i-1},\ seed)$$

使用seed, Sequence#

- 多个server，Password 可重用(使用不同seed即可)
- Server 可发起Challenge:
  - [seed, sequence#]

Attacks on OTPs..

- ***Pre-play attack*** - Eve intercepts an unused password and uses it later
- Make sure you're giving password to the right party
- Bob server must be *authenticated*

Shortcomings of OTPs..

- 使用500-1000次需要Reinitialization
  - 开销不小
- 不支持双向认证
- 保密性没考虑

Outline of Talk

- Definitions
- Passwords
  - Unix Passwords
  - One time passwords
- Challenge-response techniques
  - Also "one-time"

Challenge-response authentication

- Alice is identified by a *secret* she possesses
- *Bob* needs to know that Alice does indeed possess this secret
- *Alice* provides ***response*** to a time-variant ***challenge***
- Response depends on ***both*** secret and challenge

- To defense sniffer attack, replay attack

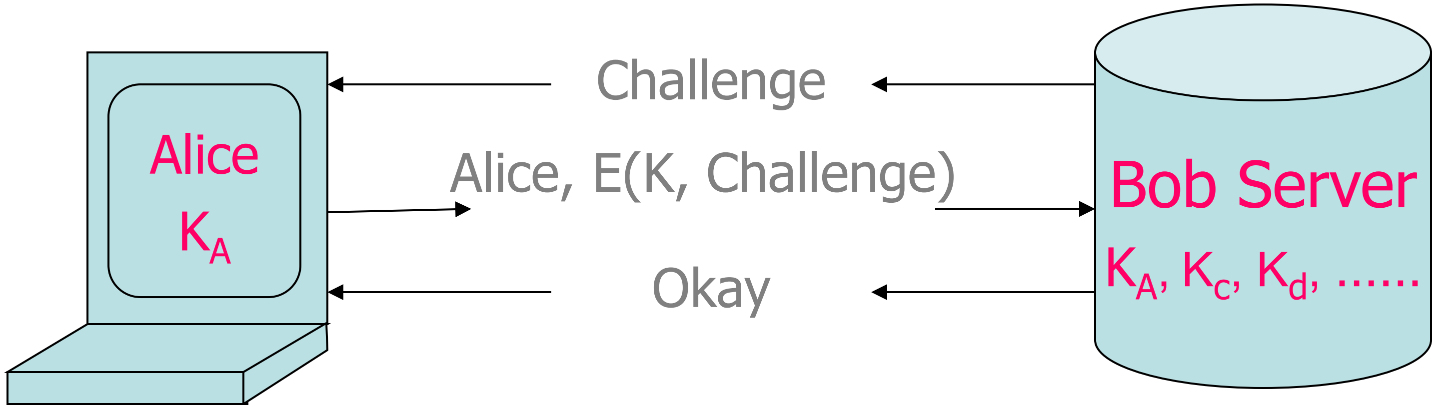
Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

using Symmetric Key Encryption

- Alice and Bob share a key *K*



单向: Using random numbers

- Bob → Alice: r_b
- Alice → Bob: $E_K(r_b, B)$
- Bob checks to see if r_b is the one it sent out
  - Also checks "*B*" - prevents reflection attack
- r_b must be ***non-repeating***

单向: Using timestamps

- Time-Based Implicit Challenge

- Alice → Bob: $E_K(t_A, B)$
- Bob decrypts and verified that timestamp is OK
- Parameter *B* prevents replay of same message in B → A direction

双向: using random numbers

- Bob → Alice: r_b
- Alice → Bob: $E_K(r_a, r_b, B)$
  - Alice Challenge Bob
- Bob → Alice: $E_K(r_a, r_b)$
- Alice checks that r_a, r_b are the ones used earlier

Shortcomings..

- 多Server, 要和不同的Server共享不同的Key
  - Key Distribution ?
  - Key management ?

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Challenge-response based on keyed OWFs

- Instead of encryption, used keyed MAC h_K
- Check: compute MAC from *known quantities*, and check with message
- SKID2 (unilateral), and SKID3(mutual)

Mutual authentication using keyed MAC – SKID3

- Bob $\rightarrow$ Alice: r_b
- Alice $\rightarrow$ Bob: r_a, $h_K(r_a, r_b, B)$
- Bob $\rightarrow$ Alice: $h_K(r_a, r_b, A)$

Unilateral authentication using keyed MAC – SKID2

- Bob → Alice: r_b
- Alice → Bob: r_a, $h_K(r_a, r_b, B)$

- Same as SKID3 without last exchange

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Authentication based on public key decryption

Witness to chosen random *r*

Challenge to Alice – encrypted with her public key

- Bob → Alice: $h(r)$, B, $P_A(r, B)$
- Alice → Bob: r

Alice decrypts challenge to get *r*. Checks with *h(r)*. Sends *r* back for Bob to check.

Mutual Authentication based on PK decryption

- Alice $\rightarrow$ Bob: $P_B(r_A, B)$
- Bob $\rightarrow$ Alice: $P_A(r_A, r_B)$
- Alice $\rightarrow$ Bob: r_B

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Unilateral Authentication using Signatures

Alice $\rightarrow$ Bob: $cert_A, t_A, B, S_A(t_A, B)$

Bob checks:

- Timestamp OK
- Identifier "B" is its own
- Signature is valid (after getting public key of Alice using certificate)

Unilateral Authentication using Signatures

Bob $\rightarrow$ Alice: r_B

Alice $\rightarrow$ Bob: $cert_A, r_A, B, S_A(r_A, r_B, B)$

Bob checks:

- Identifier "B" is its own
- Signature is valid (after getting public key of Alice using certificate)
- Signed r_A prevents chosen-text attacks

Mutual Authentication using Signatures

Bob $\rightarrow$ Alice: r_B

Alice $\rightarrow$ Bob: $cert_A, r_A, B, S_A(r_A,r_B,B)$

Bob $\rightarrow$ Alice: $cert_B, A, S_B(r_A,r_B,A)$