Security Basics

P. J. Denning

For CS471

- All computers have security requirements and mechanisms to implement security policies.
  - memory partitioning
  - sensitive instructions and supervisor state
  - file access controls
  - login
  - cryptographic protocols
  - public key certificates
  - inference controls (in some database systems)
  - data flow controls (in some military secure systems)

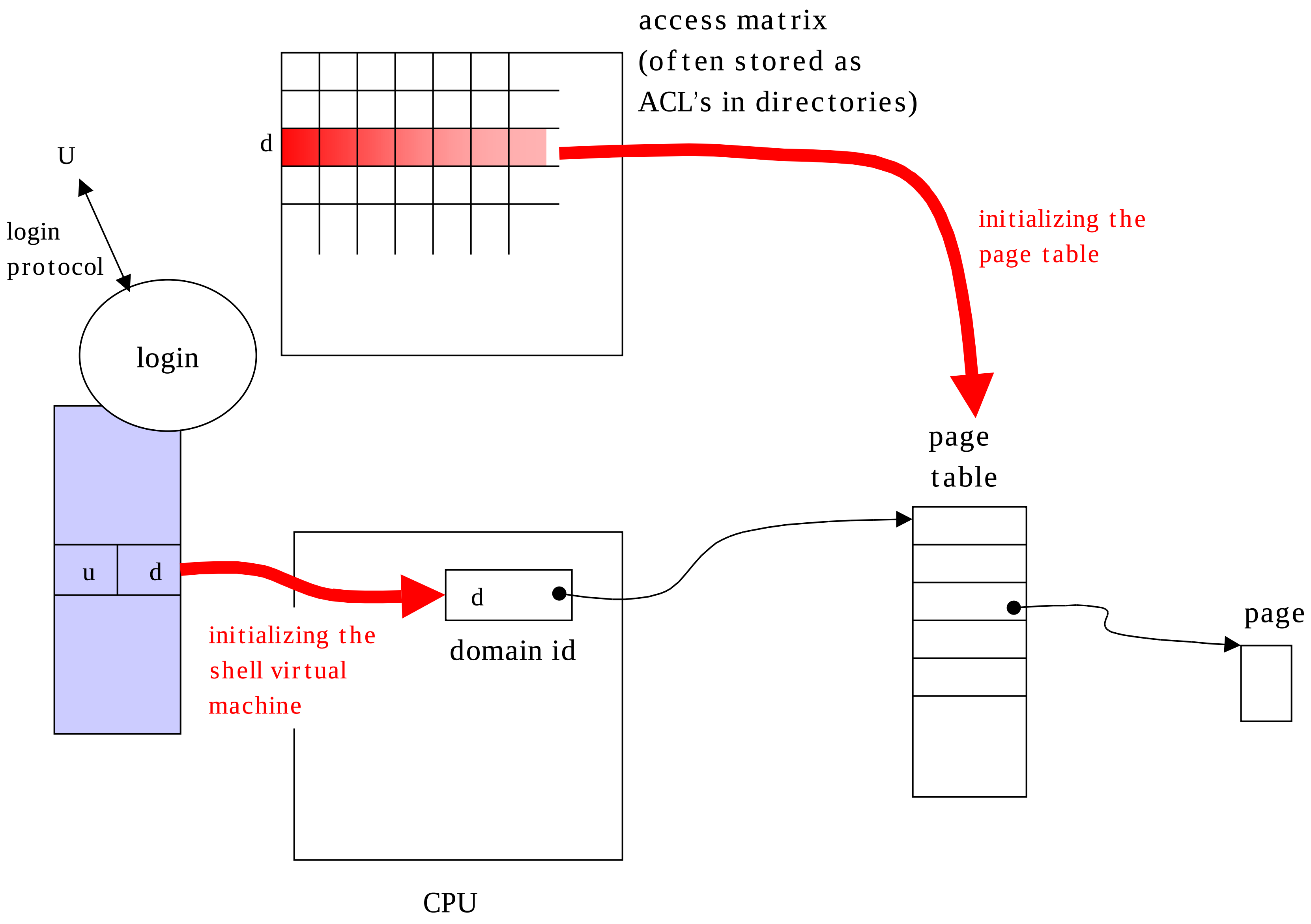
- Policy versus mechanism separation
  - change rules without changing mechanism
  - example: file access controls
- Three levels of mechanism
  - the kernel of a machine’s OS and hardware
  - a network of mutually trusting computers
  - the Internet: who or what can you trust?

Kernel -- Level 1

- Objective: control access to all regions of main and secondary memory.
  - Processes can only access memory objects for which they have been given explicit access.
  - Default is no access (“least privilege”)
- Means:
  - Base-bound descriptors in registers or tables
  - Page tables of virtual memory (logical partitioning)
  - restrict sensitive instructions to supervisor state
  - protected entry via traps
  - file access controls; copied into mapping tables
  - every object access verified (“reference monitor”)

- Previous discussion, "Access Control Basics," used access matrix to specify policy.
  - rows: protection domains (inhabited by processes)
  - columns: objects (including domains)
- Every access checked by kernel; if it conflicts with the policy, a protection violation interrupt occurs.
- Implement access matrix by columns
  - ACL (access control list) associated with object, specifies domains that have access and what kind
- Implement access matrix by rows
  - Object list (capability list) associated with domain, specifies objects accessible
- In practice, use both: ACL info in directories is copied into paging (or segment) tables; kernel and hardware use tables to map addresses and restrict access.

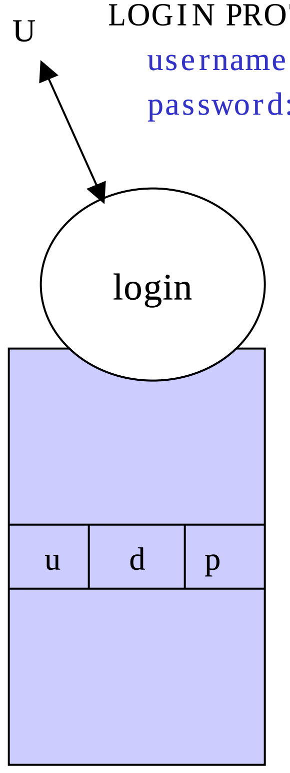
- Mapping tables pointed to by domain id register in CPU stateword.
- When user logs in, the user’s base domain is assigned to the shell started for that user.
- User can access (switch to) other domains if the access matrix permits it (“protected entry”).


access matrix
(often stored as
ACL's in directories)
d
U
login
protocol
login
initializing the
page table
page
table
u
d
d
domain id
initializing the
shell virtual
machine
page
CPU

Local Network -- Level 2

- Files are shared across a set of workstations and servers, connected by a fast local network.
- Location transparency:
  - files visible from any workstation
  - user can log in to any workstation
- User authentication at login is the critical security issue. Attackers seek to break in to unauthorized accounts.
  - password cracking
  - password sniffing

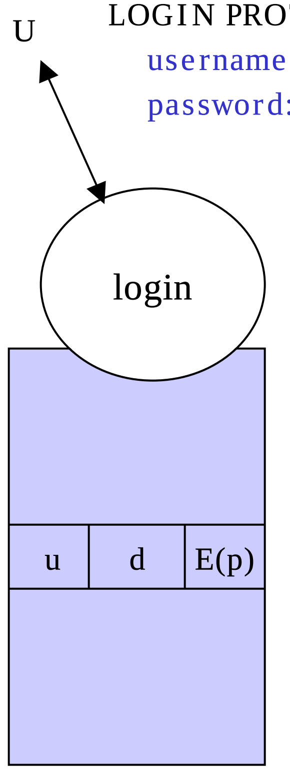
LOGIN PROTOCOL:
username: u
password: p'



password file
(hidden file,
in the clear)

Login allowed if u is listed
and typed password p' = p

User u is shown as owner of
initial shell, which operates in
domain d.



LOGIN PROTOCOL:
username: u
password: p'

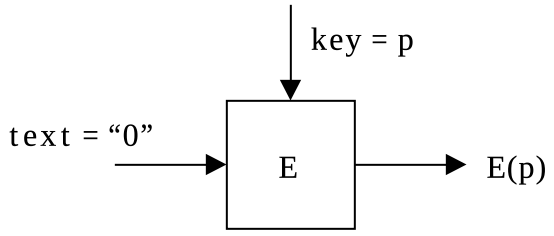
password file
(hidden file,
in the clear)

NOT SAFE!

Breaks if password file revealed.

Solution: use one-way cipher E and store E(p). Login allowed if E(p') matches the password field for user u.

False sense of security: Unix /etc/passwd public!



DES box

Login still not safe

- Attacker does not have to search an astronomical “password space” to find password matching E(p).
  - E.g., 6-letter passwords: 26^6 passwords possible, a space of size approx 2^{30} -- appears intractable.
- BUT: users tend to select words in English =>
  - High probability of cracking password upon guessing all words from English dictionary (250,000 entries)
  - Include common permutations such as reversal of name, two copies of name, etc.

- Many studies have shown that a dictionary attack is virtually certain to succeed on a password file of as few as 100 users. Takes just a few hours.
- Many ways to thwart:
  - Make password file inaccessible (authentication server)
  - Insert delay between login attempts
  - Disconnect after three unsuccessful login attempts
  - Require caps, numbers, and punctuation in password
  - Friendly password crackers
  - Pass phrases

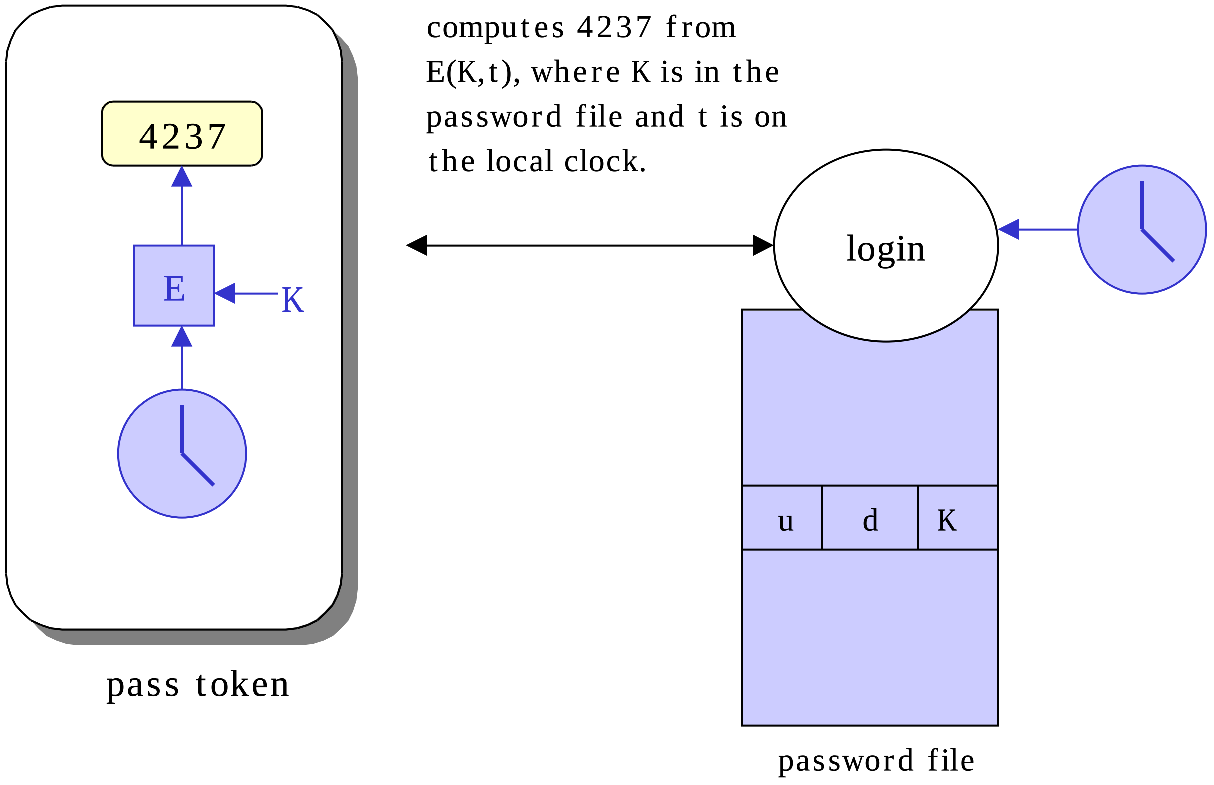
- All this is still not safe!
- Password sniffing: computer listens to ethernet with card set in "promiscuous mode", records all packets on the network; software saves those belonging to login protocol.
- Cracker collects the sniffed passwords later. No guessing required.
- Solution: any computer that runs on the network has a login client that enciphers usernames and passwords before passing them to authentication server for validation. Example: Kerberos.

Obviating Password Attacks

- Key assumption about passwords: they are reusable.
- Therefore, dictionary attacks and sniffing can yield a useful “harvest” of passwords that are still valid.
- Is there a way to eliminate reusability? -- That is, to implement one-time passwords?

username: u
password: 4237

Login validates if it also computes 4237 from E(K,t), where K is in the password file and t is on the local clock.



- Number in the token window is the clock time enciphered under a key stored within the token and assigned by the system administrator. It changes once per minute.
- When login asks for password, user enters the number then showing in the token window.
- System checks to see if the number is what it expects from its knowledge of the time and the user's key.

- Deal with clock drift by storing a drift factor a (initially 1) in the password table for user U.
- Validate if E(K,a*t) matches what the user said is on the pass token.
- If no match, check for match with t-1 (or t+1); if that matches, reset a = (t-1)/t (or (t+1)/t).
- Can add keypad to pass token; user types PIN to the token to activate it. Increases cost of token.

- Main problem is cost
  - tokens typically around $10
  - heavy administrative overhead in assigning tokens, recalling tokens, and replacing lost or broken tokens
- Theft is not usually a problem; risk no worse than credit card theft.

Biometrics

- How useful are biometrics as authenticators?
- Biometrics = some body characteristic unique to individual, e.g.,
  - fingerprint
  - voiceprint
  - retinal print
  - signature
- Benefit: hard to fool. (But not impossible.)
- Problem: can’t thwart identity theft by changing passwords or pass tokens.

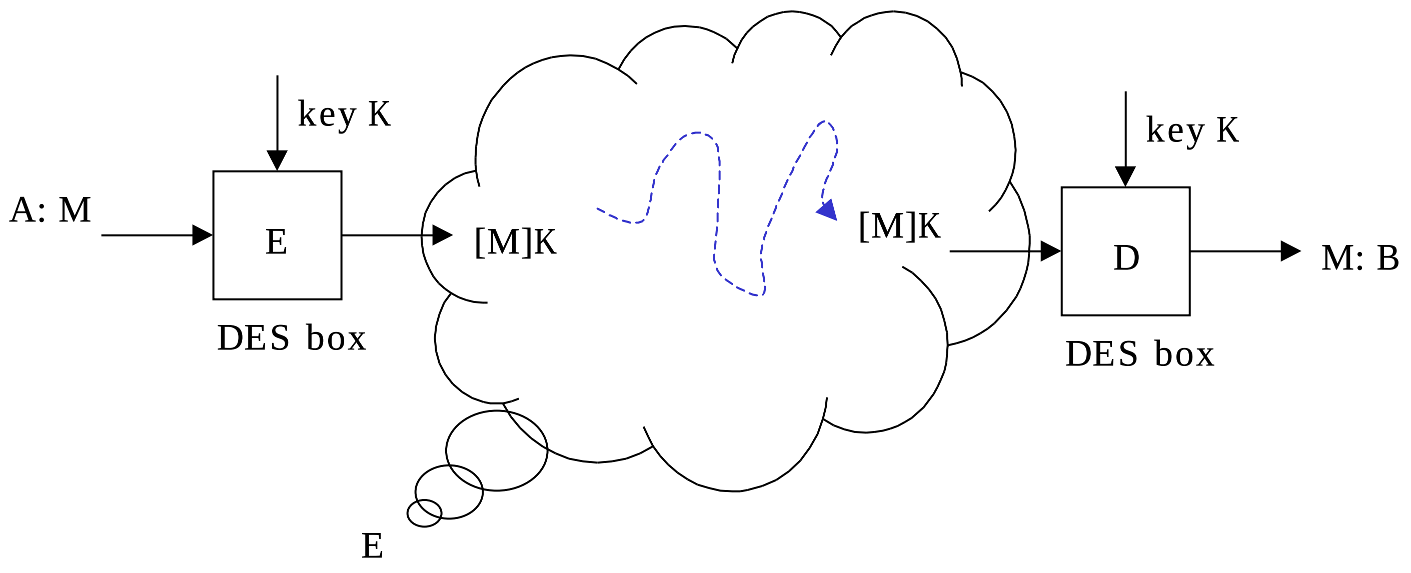
Internet -- Level 3

- Internet raises many new security problems.
- Anonymity: very difficult to learn the physical location of a server or workstation; easy for users to give fake names and return addresses.
- Untraceable connections.
- Scale: any one of 100 million users may access an object.
- Openness: unlike a local network, which is restricted to authorized users, the Internet is open to all. Authority checking must be performed at object.

- Strategy: Protect objects by enforcing access controls locally (at the object) and validating the agent making the access.
- Do this independent of the network routing used to connect the agent to the object.
- Cryptographic end-to-end protocols are the methods of choice. Protocols based on:
  - Single key encryption: fast but limited to secrecy.
  - Public key encryption: much slower but handles signatures and authentication.

Single-Key Cryptography

- The two parties to a conversation are A (Alice) and B (Bob). They are “partners” in the protocols.
- To hold secret conversation, first share a secret key K.
- A enciphers a message M with a function that depends on the key: [M]K = E(K,M).
- B deciphers the message with a function that depends on the key: M = D(K,[M]K)
- Eavesdropper who intercepts [M]K can’t decipher without a brute-force search over all possible keys. Choose key size to make this impossibly long.



E can't find M in a reasonable time from [M]K. Knowledge of the E and D boxes doesn't help.

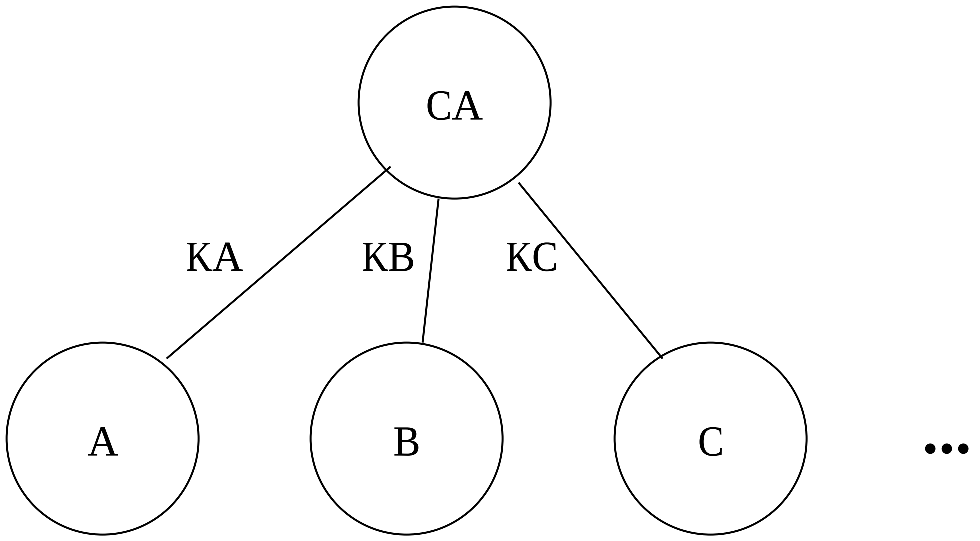
Problem: how do A and B agree on a key K?

- This form of cryptography is thousands of years old.
- Depends on a secure channel for sharing keys.
- Modern instance is the Data Encryption Standard (DES) created around 1975.
- DES provides encryption of 64-bit blocks with a 56-bit key.
- DES streams the bits through a series of 16 rounds of shift registers and “S boxes,” transforming them many times.
- DES chips have data rates of 100 Mbps or higher.
- DES is a good random number generator.
- DES is a good hash function.

- Early concern was breaking DES with massively parallel computer to guess keys. One such was built for $250,000 in 1998; it took about 24 hours per key.
- Triple DES puts three DES chips in series with a feedback loop to get the effect of 112-bit key. Breaking that size key is well beyond a massively parallel computer.
- Internet concern: constructing a secure key exchange channel.

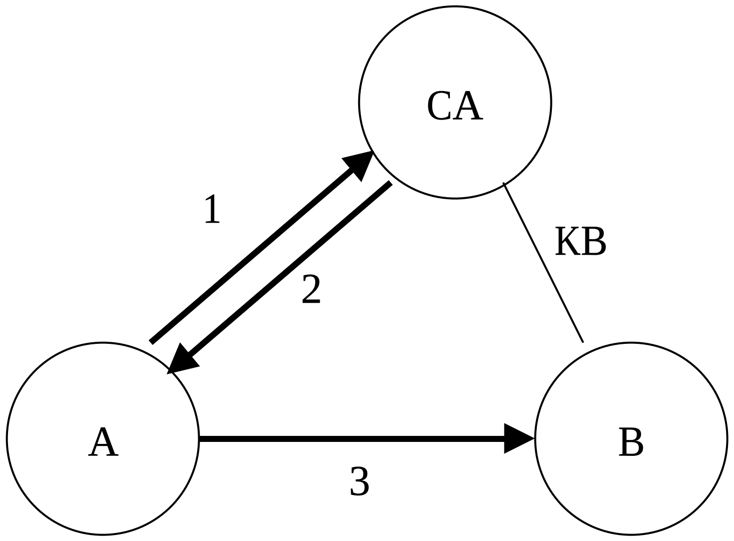
Establishing Secure Channel

- Define Certification Authority (CA).
- CA has a private key KA associated with user A, and known only to A and CA.
- A gets key KA upon identifying self to CA.
- As registered user, A can request CA to create a key K for a session with B, and convince B that only A and B have the key K.



CA must earn high trust

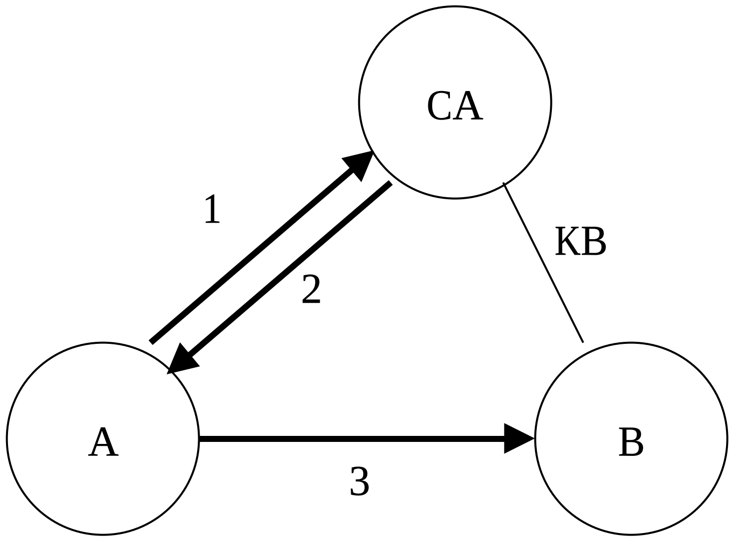
Compromising CA's database compromises entire network



1: ["request session key for B"]KA

2: [[K,["from A:",K]KB]KA

3: "request session", ["from A:",K]KB



1: ["request session key for B"]KA
2: [[K,["from A:",K]KB]KA
3: "request session", ["from A:",K]KB

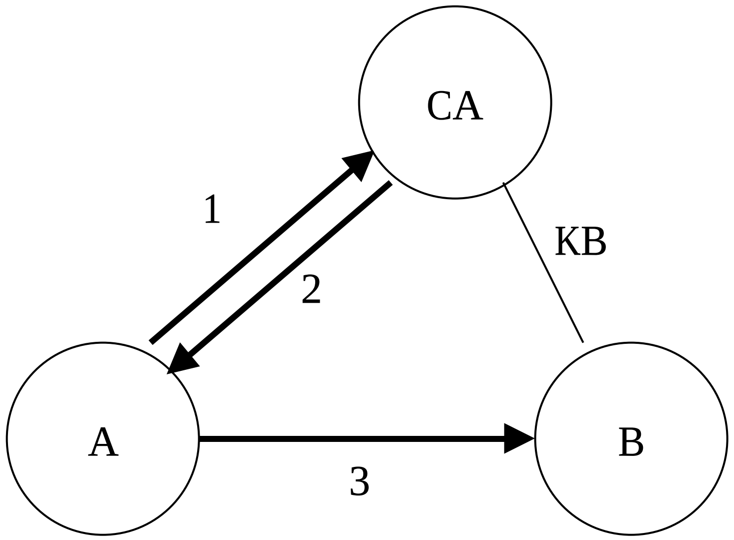
ANALYSIS:

1: A requests CA to generate K; no one else can say this.

2: CA sends back K and a special certificate for B.

3: A retains K and forwards certificate to B.

4: B opens certificate, see it's from A and gets key K; only CA could generate certificate.



1: ["request session key for B"]KA
2: [[K,["from A:",K]KB]KA
3: "request session", ["from A:",K]KB

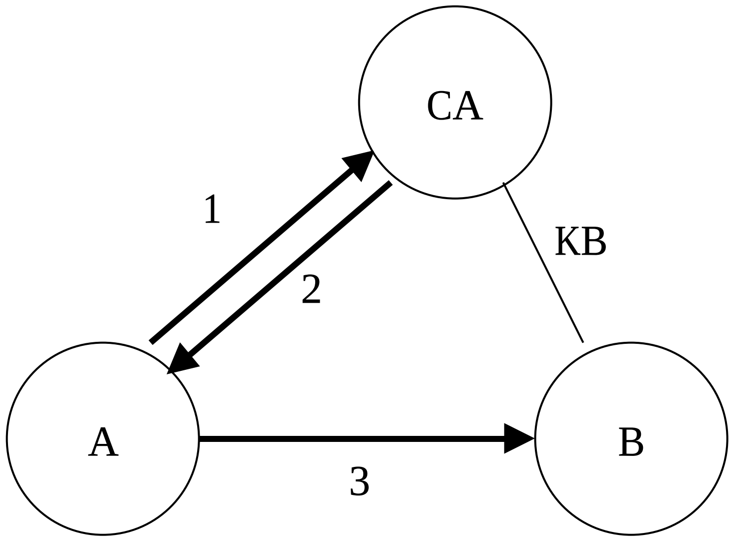
OTHER:

Omitting KA in step 1 enables impostor of A to get K.

Omitting "from A" in step 2 deprives B of assurance A is sending the request.

Omitting certificate from step 3 deprives B of certainty that only A knows key K.

Add timestamps to prevent replay.



1: ["request session key for B"]KA
2: [[K,["from A:",K]KB]KA
3: "request session", ["from A:",K]KB

MAJOR VULNERABILITY:

CA compromise compromises entire network.

Public Key Cryptography

- Invented in 1975 by Whitfield Diffie and Martin Hellman.
- Two keys given to each individual A: public (PA) and secret (SA).
- Knowledge of SA gives no advantage to computing PA.
- Encipherment and decipherment are symmetric:

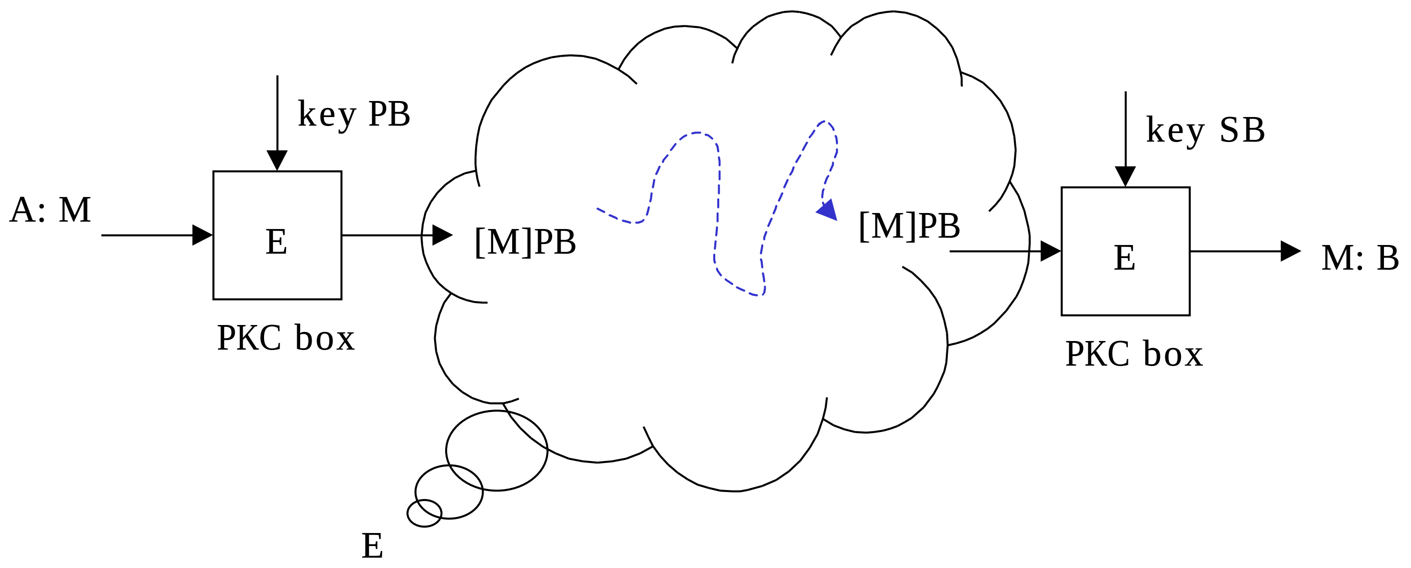
  M = [[M]PA]SA = [[M]SA]PA
- To send a message to A, encipher it with A's public key; A deciphers with secret key. Thus

  M = [[M]PA]SA

  no one else but A can decipher the message.

RSA System

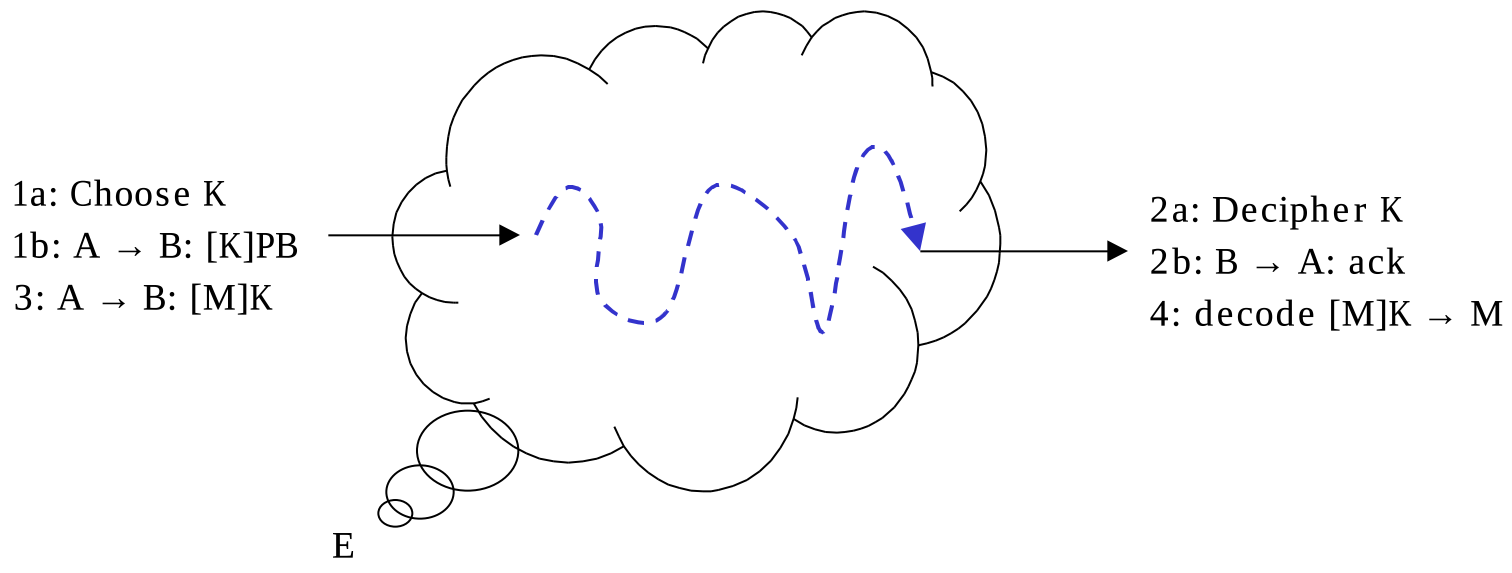
- First solid working version was invented by Rivest, Shamir, and Adleman in 1977 and is today called RSA encryption.
- It is based on choosing two large secret primes p and q (e.g., 200 digits each) and setting n to their product pq.
- The secret key is (d,n) where d is an integer relatively prime to (p-1)(q-1).
- The public key is (e,n) chosen so ed=1 mod (p-1)(q-1).
- Encipherment of M is $C = M^e \bmod n$.
- Decipherment of C is $M = C^d \bmod n$.
- No one has found a way to factor a composite number, like n, into its prime factors in polynomial time. All known factoring algorithms are exponentially hard. Unless you can factor n, you can't find d from the public key.



Eavesdropper E cannot decipher [M]PB since SB is the only way to do that, SB is known only to B, and E cannot compute SB from PB.

DATA RATE of PKC box is much less than DES box -- on the order of 100Kb for PKC and 100Mb for DES.

Solution to Secure Key Channel Problem



A chooses a DES session key K and sends to B enciphered with B's public key. After that they can use DES to encipher all their messages.

Signatures

- PKC allows signatures, a new concept.
- [M]SA can be deciphered by anyone (with PA) but can only be generated by A --

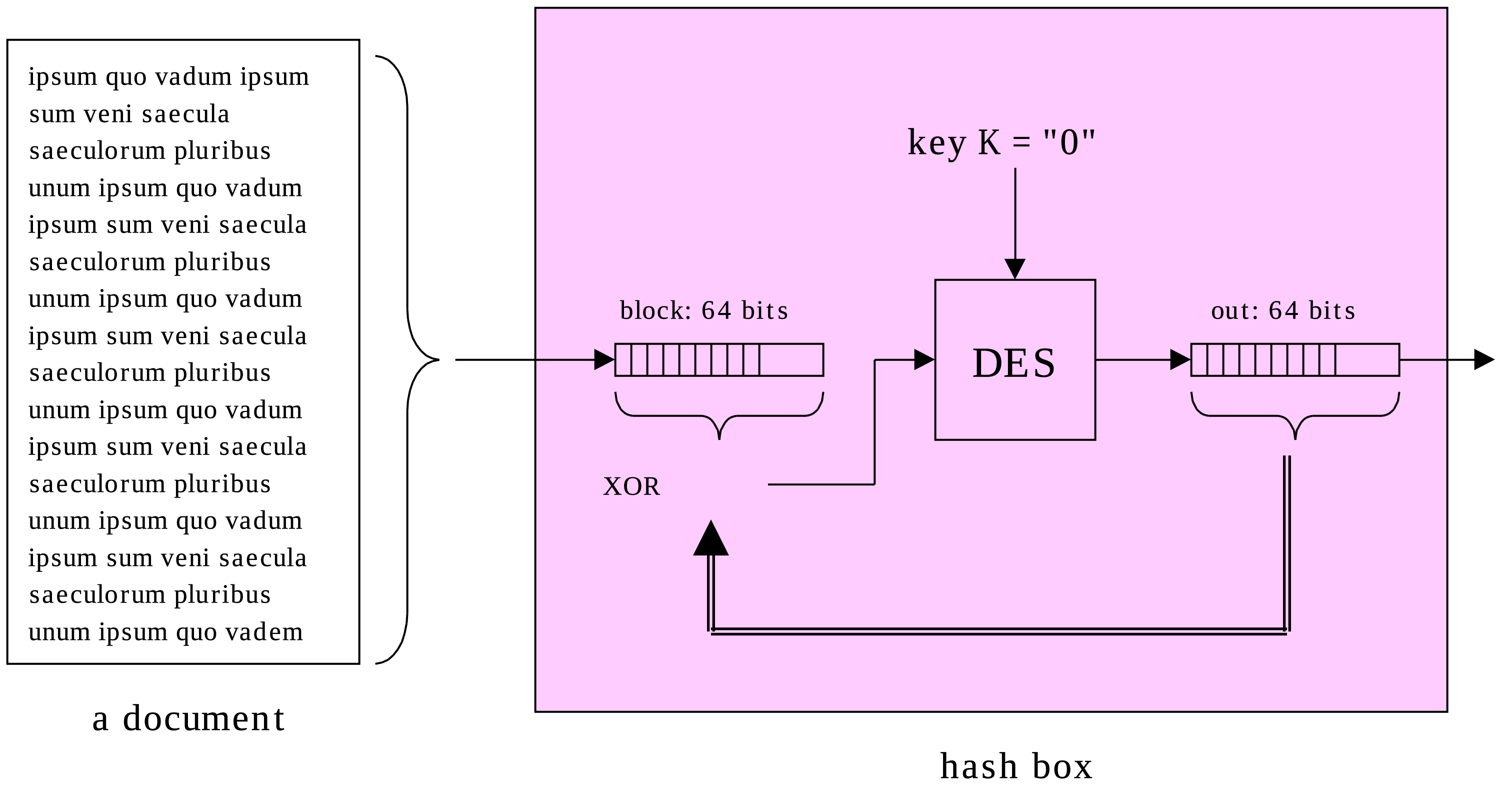
  M = [[M]SA]PA
- Now secrecy and authentication are separate.
  - Secrecy -- encipher with receiver's public key
  - Authentication -- encipher with sender's secret key
- Can do both, as when A sends B --

  [[M]SA]PB

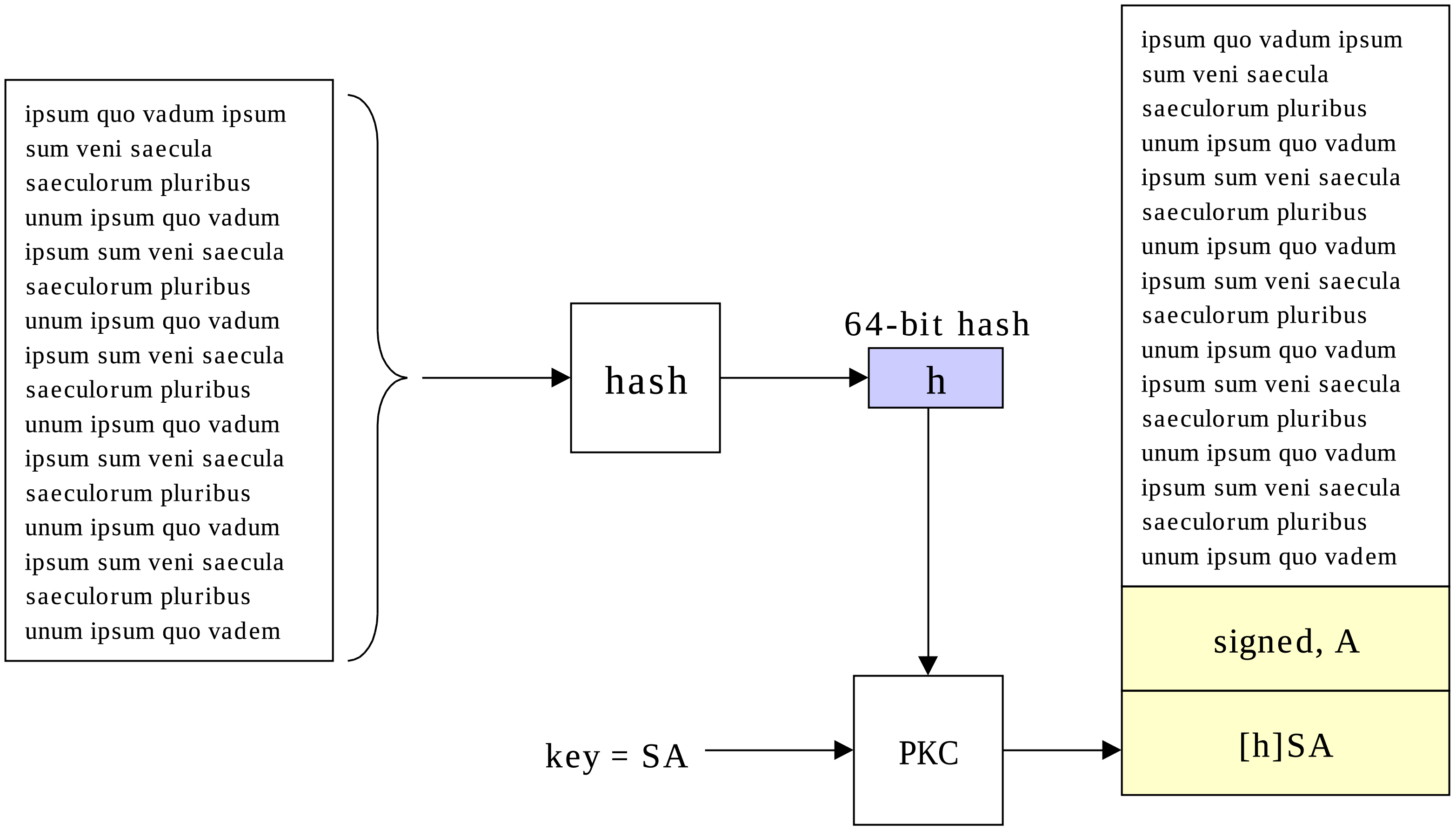
  only B could receive and only A can send.

- The low bandwidth of PKC encoding works against signing large documents.
- Interesting practical case: signing a document (e.g., email) when secrecy is not a concern -- the document is public but we want to be sure that everyone knows A signed it and that A cannot repudiate signature.
- Make a digest of the entire document by passing all bits of the document through a hash function into a single short bitstring -- e.g., 64 bits.
- A signs the digest and includes it with the document.

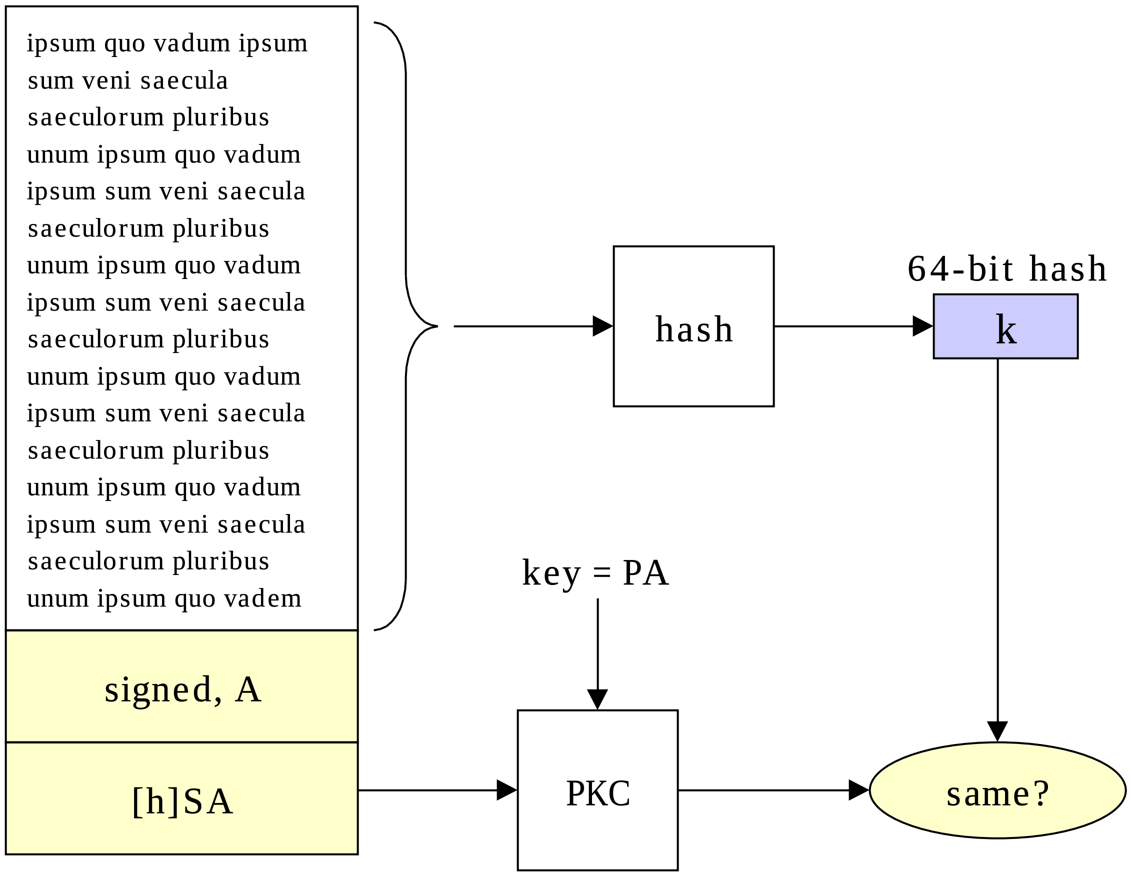
```
out = 0
while blocks remain do {
   block = next64(file)
   out = DES ("0", block     out) }
```



What A Does to Sign Document


ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadem
hash
64-bit hash
h
key = SA
PKC
ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadem
signed, A
[h]SA

What B Does to Verify Document Authenticity


ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadum ipsum sum veni saecula saeculorum pluribus unum ipsum quo vadem
signed, A
[h]SA
hash
64-bit hash
k
key = PA
PKC
same?

The system PGP (Phil Zimmerman's Pretty Good Privacy) validates email this way.

Public Key Certificates

- How are public keys distributed? Mechanism for doing this called Public Key Infrastructure (PKI).
- What about white pages -- a public database contain records of the form (A, PA)?
- White pages not safe.
  - B can register entry (A, PB), thereby fooling others into using PB rather than PA. (Now B can be an eavesdropper.)
  - B can do this even if B needs to identify self at time of registration, if the white pages administrator does not verify that the public key belongs to B alone.

- Certification Authority (CA): issues certificates testifying that PA is A's public key.
- To assure (A,PA) belongs to A, CA must identify A and either
  - CA generates (PA,SA) keys (and discards copy of SA)
  - A provides PA and answers a challenge by CA, e.g., supplies ["challenge string"]SA on demand by CA.
- Still not safe --
  - database could be compromised by attackers who got CA's secret key and could therefore replace certificates.

- Public Key Certificate: A digital object containing
  - declaration “A’s public key is:”
  - A’s public key PA
  - PKC algorithm to be used
  - issue timestamp Ti
  - expiry timestamp Te
  - CA's signature on hash checksum h of the above items

"A's public key is:", PA, RSA, Ti, Te, [h]SCA

NOTE:

Certificate says only: "Someone identifying self as A was issued the public key PA, signed CA."

Certificate does NOT say that the bearer is A. If the bearer matters, the person performing the transaction must check A's identity.

Trust in the certificate depends on the identification procedure used by CA and on the subsequent verification process at time certificate is used.

BEFORE

ipsum quo vadum ipsum
sum veni saecula
saeculorum pluribus
unum ipsum quo vadum
ipsum sum veni saecula
saeculorum pluribus
unum ipsum quo vadum
ipsum sum veni saecula
saeculorum pluribus
unum ipsum quo vadum
ipsum sum veni saecula
saeculorum pluribus
unum ipsum quo vadum
ipsum sum veni saecula
saeculorum pluribus
unum ipsum quo vadem

signed, A

[h]SA

With certificates, A can include copy of its public key certificate with the document, so that any recipient has the public key when it need to verify the document.

NOW

ipsum quo vadum ipsum
sum veni saecula
saeculorum pluribus
unum ipsum quo vadum
ipsum sum veni saecula
saeculorum pluribus
unum ipsum quo vadum
ipsum sum veni saecula
saeculorum pluribus
unum ipsum quo vadum
ipsum sum veni saecula
saeculorum pluribus
unum ipsum quo vadum
ipsum sum veni saecula
saeculorum pluribus
unum ipsum quo vadem

signed, A
”A’s PK is:”,PA,RSA,Ti,Te,[h]SCA

[h]SA

NOTE ABOUT REPUDIATION:

Can A claim that A's secret key was compromised and therefore that a signed document is invalid?

Nothing prevents this -- e.g., A could release SA after signing a document A wishes to repudiate.

Signature puts burden of proof on A to show that signature is not valid. Decided by mediator or court.

Identity theft (someone steals SA) can be a major problem.

PKI

- Public Key Infrastructure: set of protocols, clients, servers, and processes to allow merchants to validate customers.
- PKI trust depends on:
  - trust in the PK certificates (issued by the CA)
  - trust in the per-transaction verification process
- Most PKI does not implement PTV as thoroughly as credit card companies do and can be circumvented. (Recent Microsoft case with VeriSign illustrates.)

finis