Security in Operating Systems

Peter J. Denning

July 2019

What is security?

- Security is an assessment of trust that a computer system protects data and assets and is safe to use
- How is this assessment grounded?
  - Record of system safety and reliability
  - Record of successfully resisting malware attacks
  - Trustworthy authentication
  - Encryption
  - Data protection safeguards in place
  - User declarations regarding privacy and sharing honored

- Security has been a major concern in the design of OS since the 1960s. Examples discussed in previous modules of this book pervade the kernel:
  - memory partitioning
  - sensitive instructions and supervisor state
  - file access controls
  - Login protocols
  - Protected service processes
  - Protected entry points
  - Capability addressing
  - Sandboxes and confined domains

- There are more topics in security that have not been covered in previous modules, but will be discussed in this module:
  - authentication protocols
  - digital signatures
  - cryptographic communication protocols between servers
  - public key certificates
- Other security topics not covered here:
  - malware and hacker defenses
  - security practices in the user community
  - inference controls (in some database systems)
  - data flow controls (in some military secure systems)

- Most security methods are integrated with the OS
  - Some like memory protection, process isolation, and access controls permeate the design of the kernel
  - Others such as network access and web access are implemented as protected service processes

Organization of this discussion

- Based on principle: separate policy from mechanism
  - change rules without changing mechanism
  - example: file access controls
- Three levels of mechanism
  - the kernel of a machine's OS and hardware
  - a private network of mutually trusting computers
  - the Internet: who or what can you trust?

Kernel -- Level 1

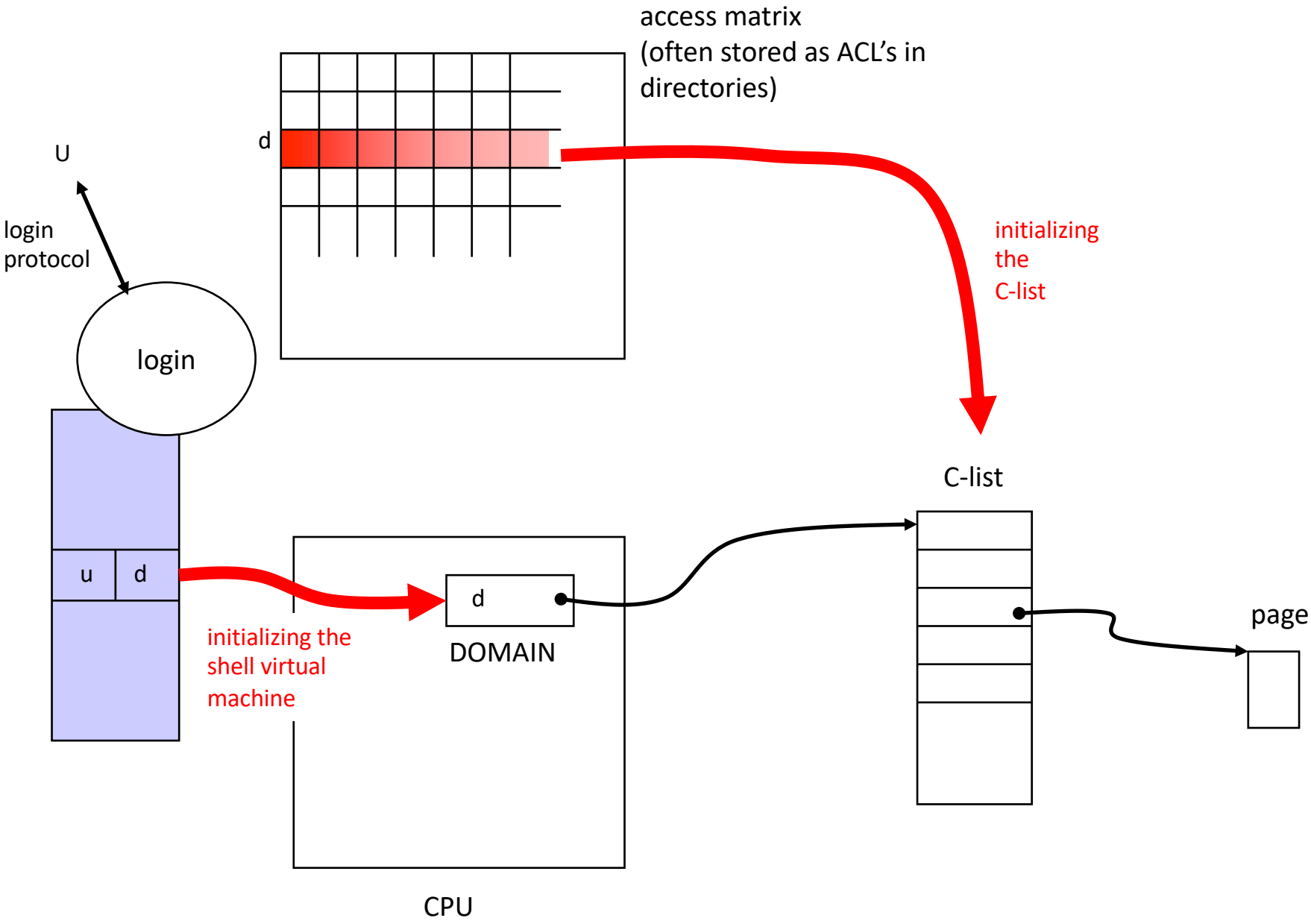
- Objectives:
  - Control access to all regions of main and secondary memory
  - Isolate processes (virtual machines) from each other
  - Assure correct operation of all kernel functions
- Principles:
  - Processes can only access memory objects or regions for which they have been given explicit access.
  - Default is no access ("least privilege")
  - Kernel functions invoked only by protected entry
  - Protected service processes invoked via messages
  - Untrusted software can be confined

Kernel -- Level 1

- Means to these objectives:
  - Base-bound descriptors in registers or tables
  - Page tables of virtual memory (logical partitioning)
  - restrict sensitive instructions to supervisor state
  - protected entry via interrupts
  - protected IPC message system
  - file access controls; copied into mapping tables
  - every object access verified (“reference monitor”)

- Access matrix: model of a policy. (see ArtOS2 Ch 6.4, “Access Controls”)
  - rows: protection domains (inhabited by processes)
  - columns: objects (including domains)
- Reference monitor: every requested access checked by kernel; if it conflicts with the policy, monitor throws protection violation interrupt.
- Access Control List (ACL) -- Implements access matrix by columns
  - ACL associated with object, specifies allowed accesses for each domain
- Capability List (C-list) -- Implements access matrix by rows
  - C-list associated with domain, specifies objects accessible in that domain
- In practice, use both: ACL info in directories is copied into kernel mapping tables (C-lists), which map addresses and restrict access

- CPU contains a DOMAIN register pointing to the C-list, used to map addresses to objects. DOMAIN register part of CPU stateword.
  - Generalizes the page-table pointer register
- When user logs in, the user's shell process DOMAIN register is initialized with the user's default base domain.
  - Shell's domain inherited by all children processes
- Process can switch to another domain if the access matrix permits it.
  - Protected entry is the means to safely switch domains
  - C-list contains an "enter capability" for the other domain

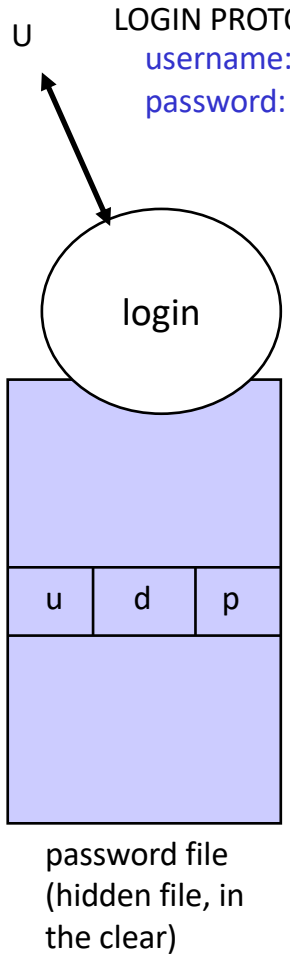


Local Network -- Level 2

- A local (private) network consists of a collection of mutually trusting computers (workstations, servers, desktops, smartphones, and other devices)
- User joins the network with a login protocol as discussed next
- Computers interact with cryptographic protocols that allow them to be continuously certain they are talking with computers they trust
  - VPN (virtual private network) protocols
  - Kerberos (an MIT system) protocols
  - Secure web server protocols (e.g., https)
  - Cryptographic protocols discussed later
- Computers in a local private network can be dispersed over a wide geographic area and be mobile

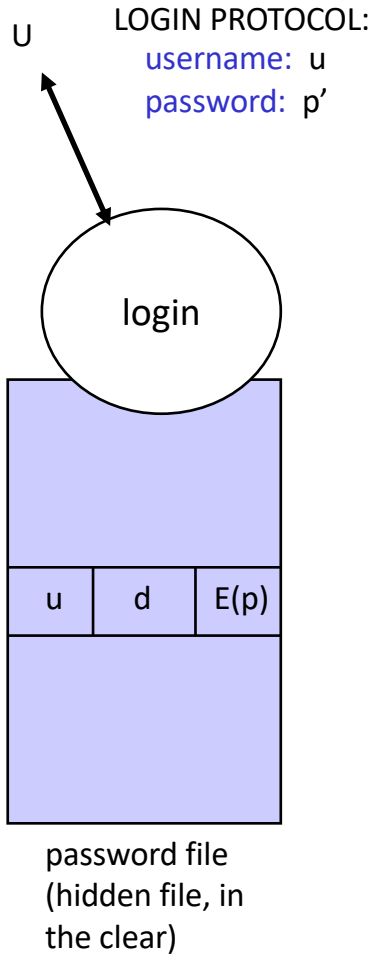
Inside a Local Network

- File sharing is one of main benefits of local private network. Everyone sees the same directories and files with a common name space, typically URLs.
- Communications among the network's computers are secured by cryptographic protocols. Intruders unlikely.
- User authentication at login is the critical security issue. Attackers search for accounts with weak authentication.
  - password cracking
  - password sniffing
  - social engineering (tricking user into releasing private info)
  - malware (Trojan horses)



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

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

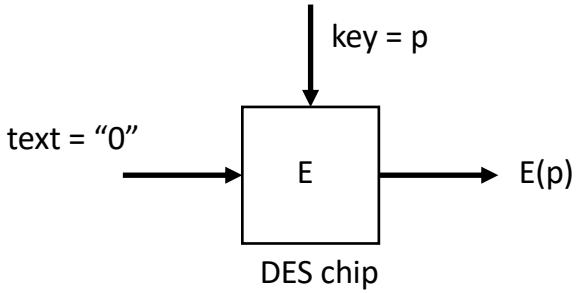


NOT SAFE! THIS IS SAFER:

Breaks if password file revealed.

Solution: use one-way cipher E and store E(p).
Login allowed if E(p') = E(p).

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



One-way password still not safe

- Attacker does not have to search an astronomical "password space" to find password matching E(p).
  - The potential space for 12-letter passwords has 26^{12} = approx 2^{54} passwords. Intractable if using brute-force search.
- BUT: users tend to select words in English =>
  - High probability of cracking password upon guessing all words from English dictionary (250,000 entries) – fast search
  - 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 one-way password file of as few as 100 users. Takes at most 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
  - Check proposed passwords with friendly password cracker
  - Pass phrases

- All this is still not safe!
- Password sniffing: eavesdropper listens to ethernet or other packet-carrying channel, locating packet sequences from login protocol, and saving them.
- Cracker collects the sniffed passwords later and replays them on the network. No guessing required.
- Solution: any computer that runs on the network has a login client that enciphers usernames and passwords before passing them on the network, for example to a Kerberos authentication server.

What are safe ways to log in?

- Two-Factor Authentication: Augment password with a direct “please confirm attempted login” message to the user
- Login tokens
- Biometrics

Two-factor authentication

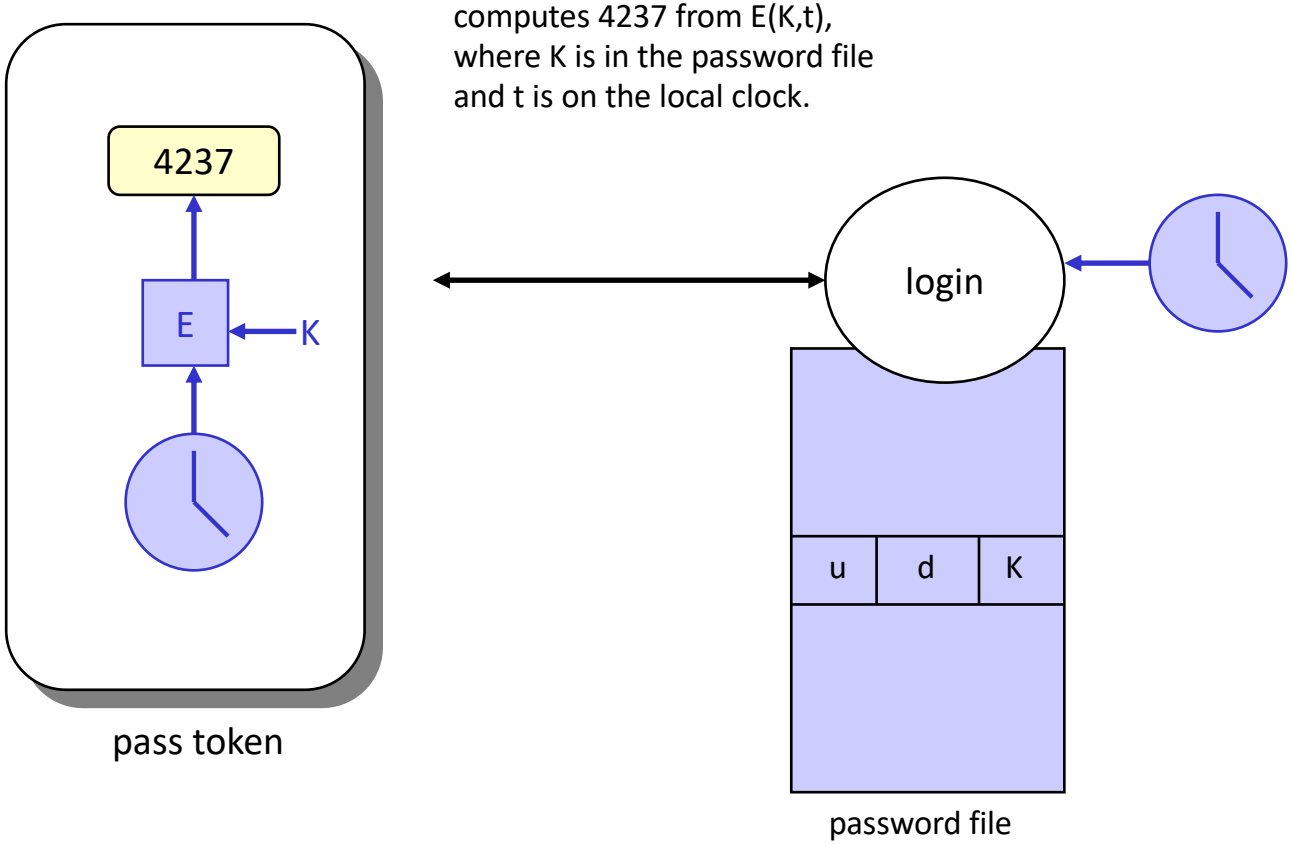
- Password plus “challenge-response”
  - User types name and password
  - Authentication server sends confirmation request (challenge) to user’s previously registered mobile phone
  - Login allowed if password correct and user responds positively to the challenge
- Immune to password attacks

Login tokens

- 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.
- Can also be used as second factor in two-factor authentication: request user to provide number on token's display.

- 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 $5
  - 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

- Another way to log in without passwords
- Biometrics = some measurable characteristic unique to individual
  - fingerprint, voiceprint, retinal print, face print
  - signature
  - keystrokes event recording
  - geolocation
- Hardware to do these things cheap and ubiquitous.
- Benefit: hard to fool. (But not impossible – e.g., with replays or copies of the biometric data.)
- Problem: can't thwart identity theft by changing passwords or pass tokens.

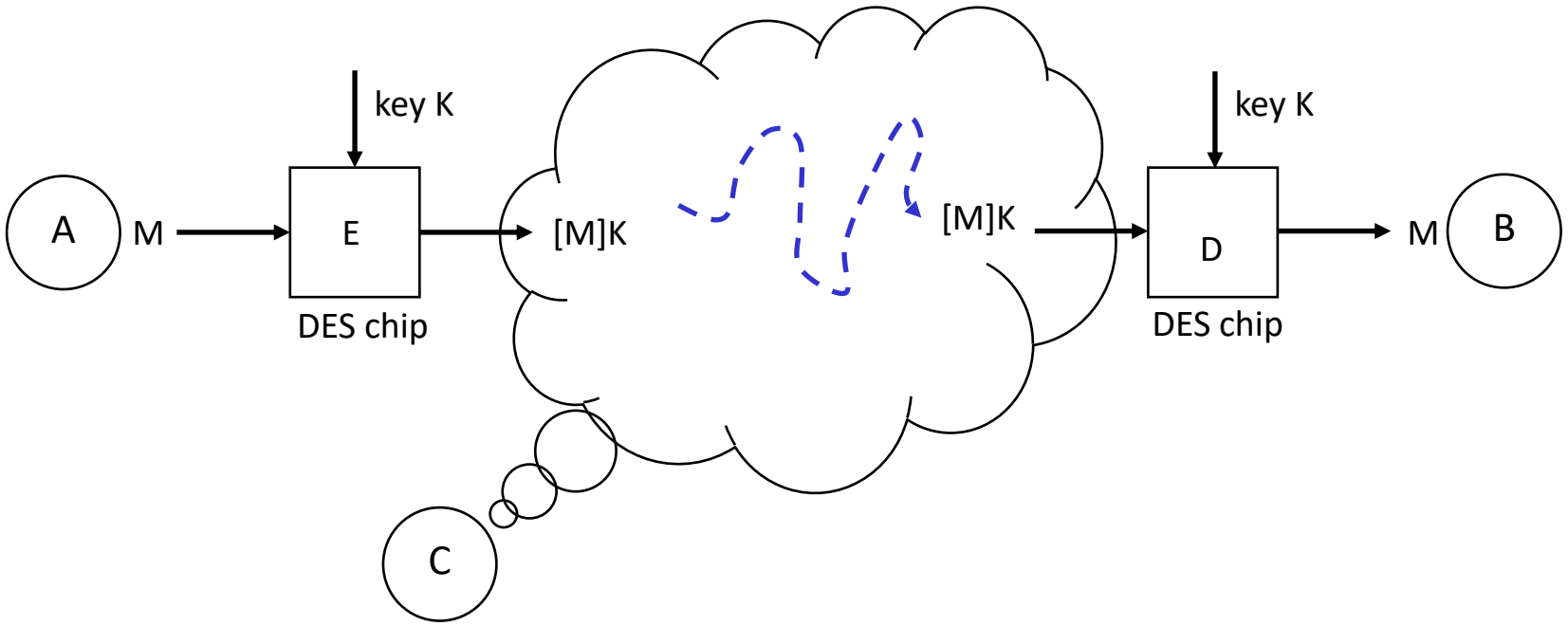
Internet -- Level 3

- Internet raises many new security problems
- Anonymity: Makes it very difficult to learn
  - the physical location of a server or workstation
  - the identity of a sender; easy for users to give fake names and return addresses.
- Untraceable connections and chains of connections
- Obfuscating services
- Scale: any one of 1 billion users may access an object
- Openness: unlike a local private network, which is restricted to authorized users, the Internet is open to all

- 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 one or both of:
  - 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.



C (eavesdropper) can't find M in a reasonable time from [M]K. Knowledge of the E and D chips doesn't help.

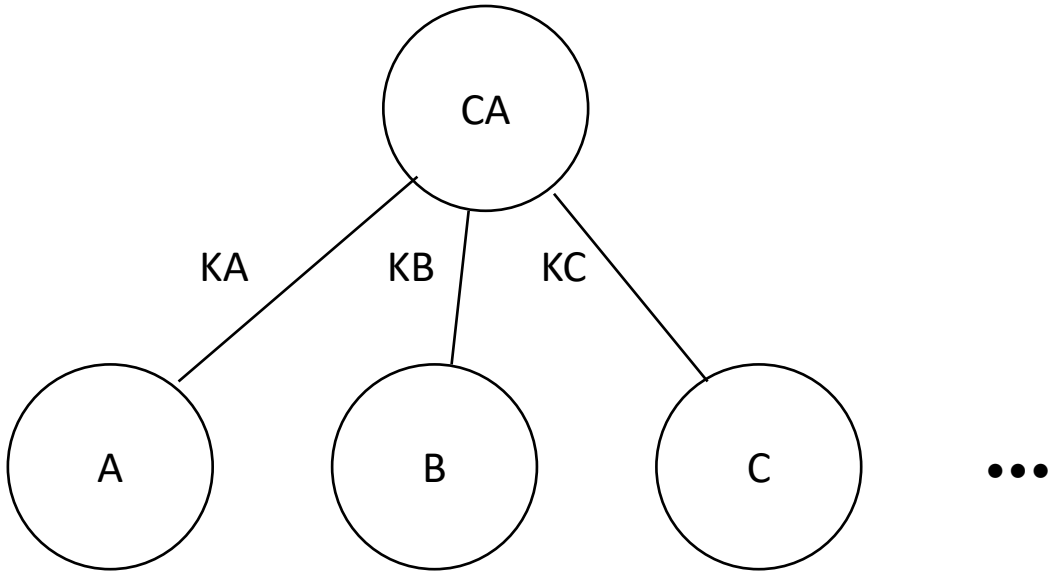
Problem: how do A and B agree on a key K?

- 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 can match data rates of fast local networks.
  - DES is a good random number generator.
  - DES is a good one-way function (as in passwords)
  - DES is a good hash function (signatures, blockchains)

- 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 56-bit 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.

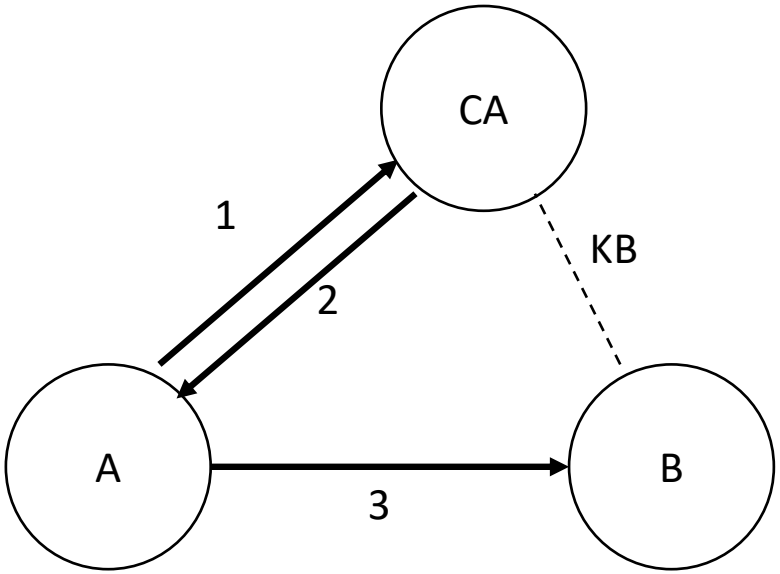
Establishing Secure Channel

- Define Certification Authority (CA).
- CA has a private key KA associated with user A.
- KA known only to A and CA.
- A gets key KA upon registering with 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.



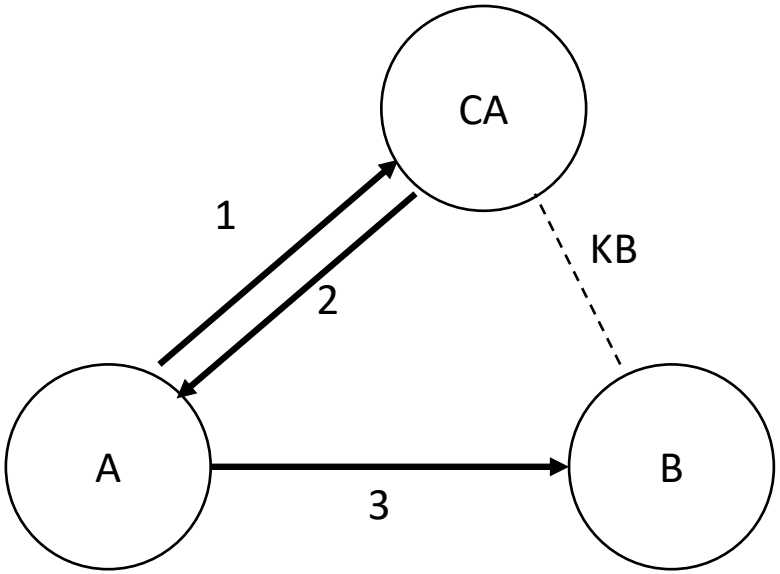
CA must earn high trust

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

Needham-Schroeder Protocol



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

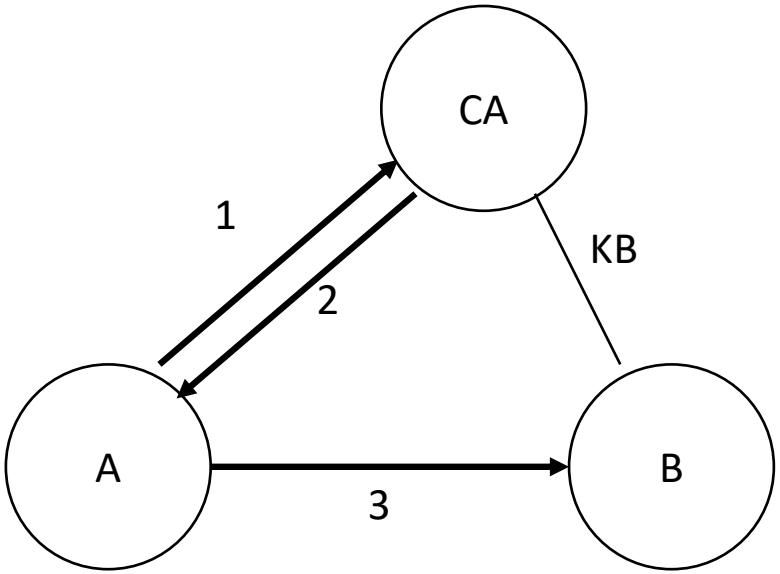
ANALYSIS:

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

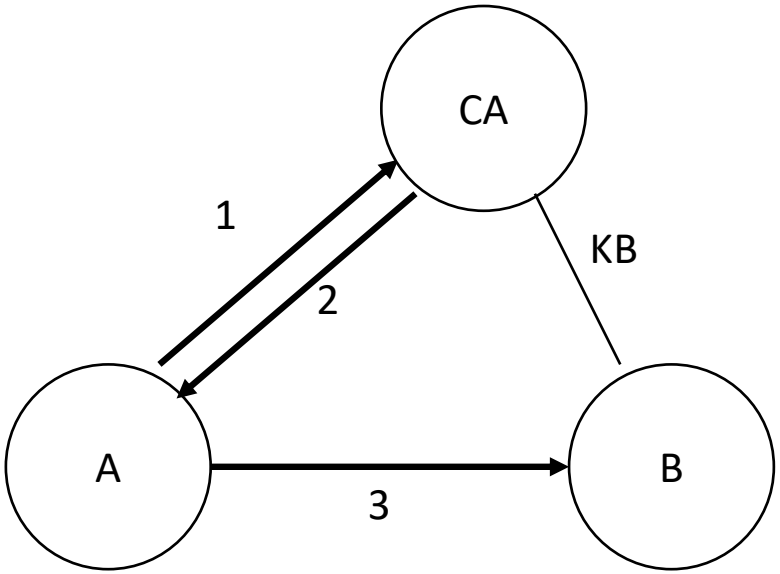
OTHER:

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:

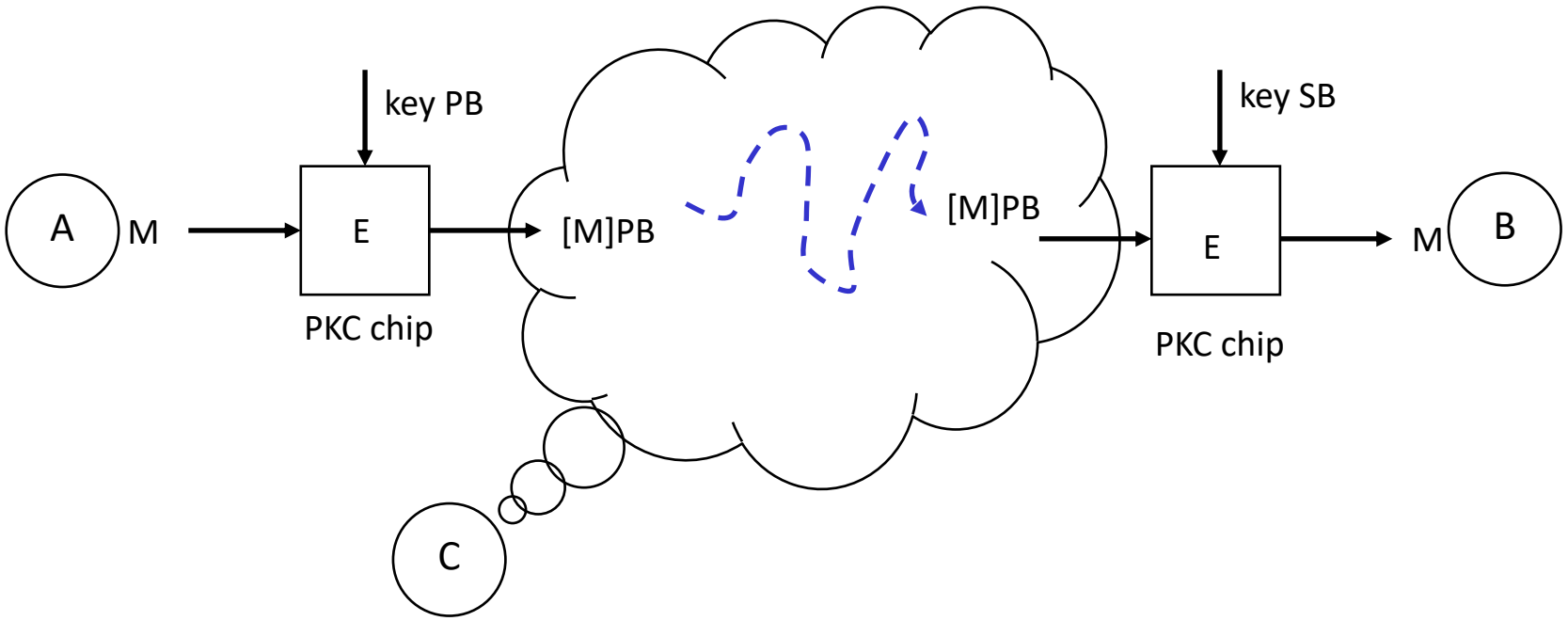
CA compromise compromises entire network.

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.
- SA recovers a message enciphered with PA: M = [[M]PA]SA
- PA recovers a message enciphered with SA: M = [[M]SA]PA
- Only A has SA; no one else can apply SA to a message

RSA System

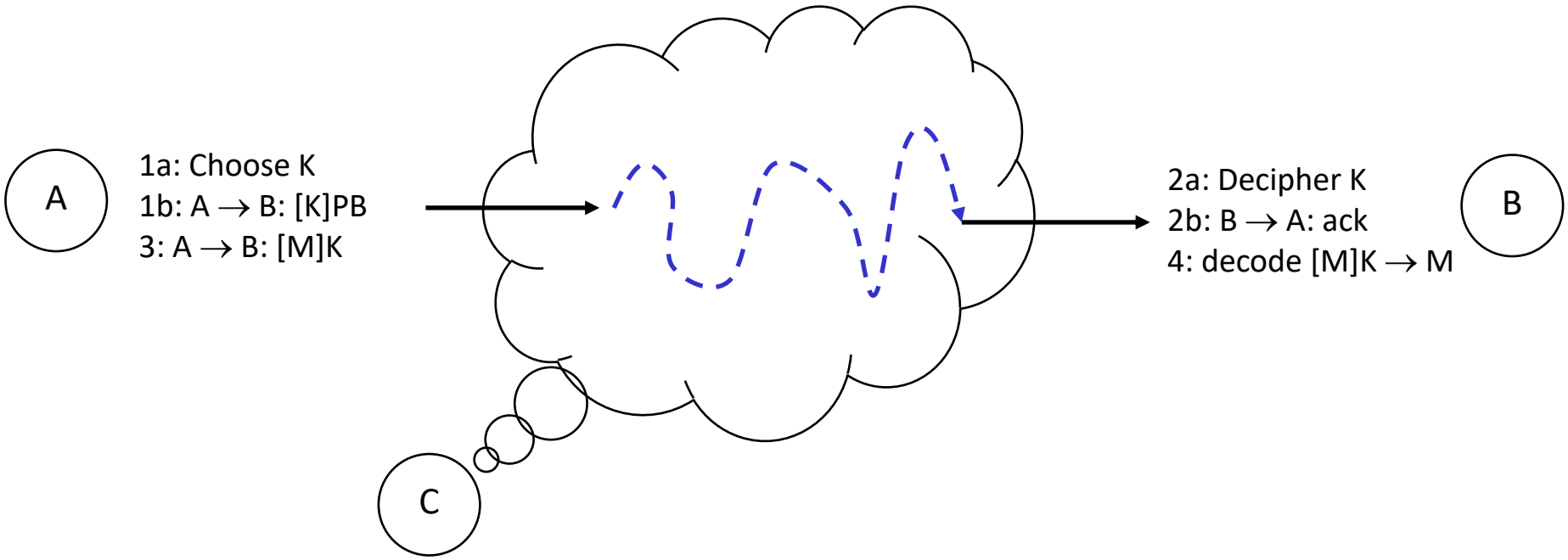
- First solid working version was invented by Rivest, Shamir, and Adleman in 1977 and is today called RSA encryption.
- It is based on choosing two large secret primes p and q (e.g., 200 digits each) and setting n = pq.
- The secret key is (d,n) where d is an integer relatively prime to (p-1)(q-1).
- 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 C cannot decipher [M]PB since SB is the only way to do that, SB is known only to B, and C cannot compute SB from PB.

DATA RATE of PKC chip is much less than DES chip – a factor of 1000 or difference

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. C cannot eavesdrop because K is known only to A and B.

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

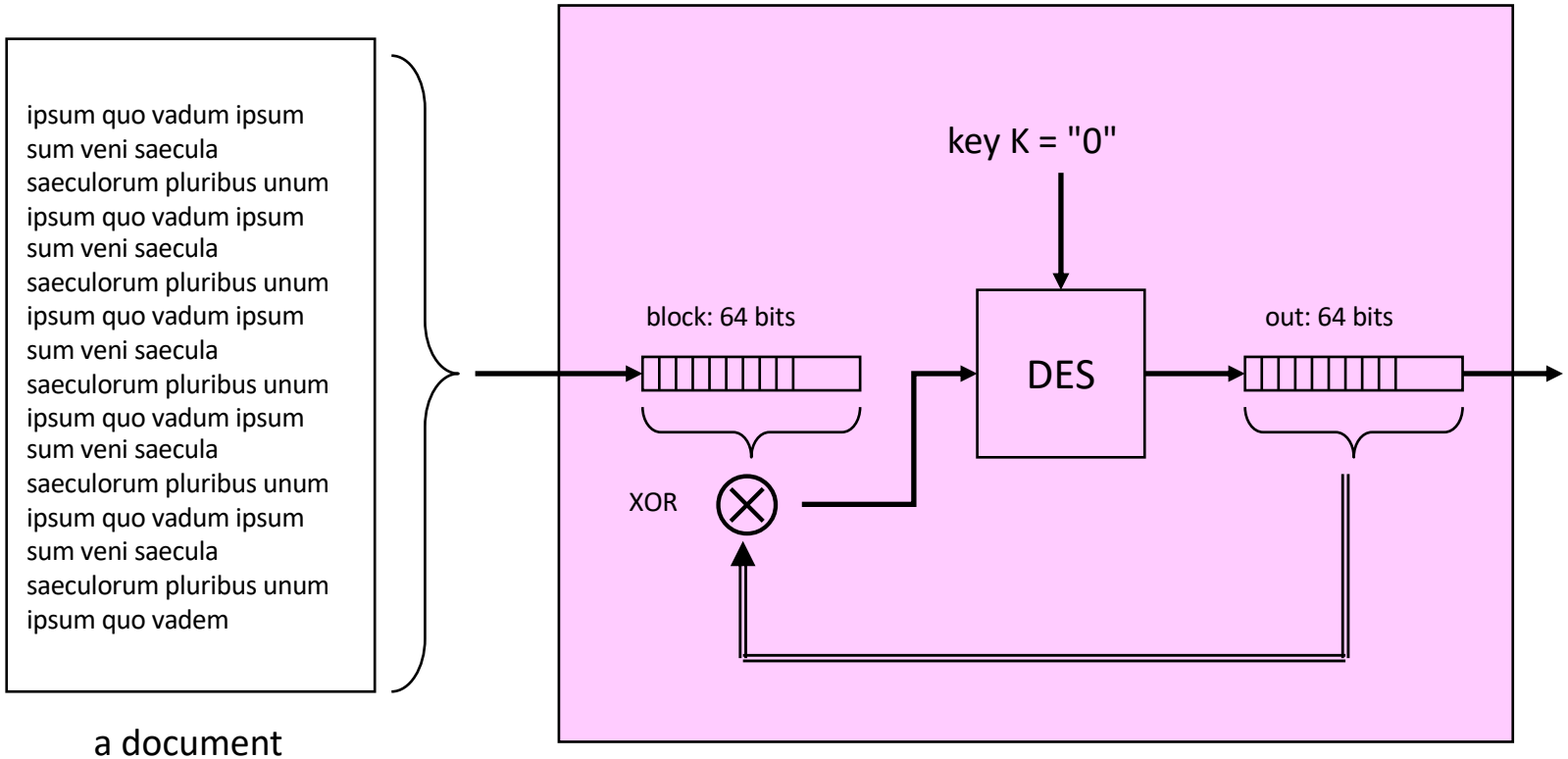
- Now secrecy and authentication are separate.
  - Secrecy -- encipher with receiver's public key
  - Authentication -- encipher with sender's secret key
- Can do both, as when A sends B --

  [[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 mapping 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.

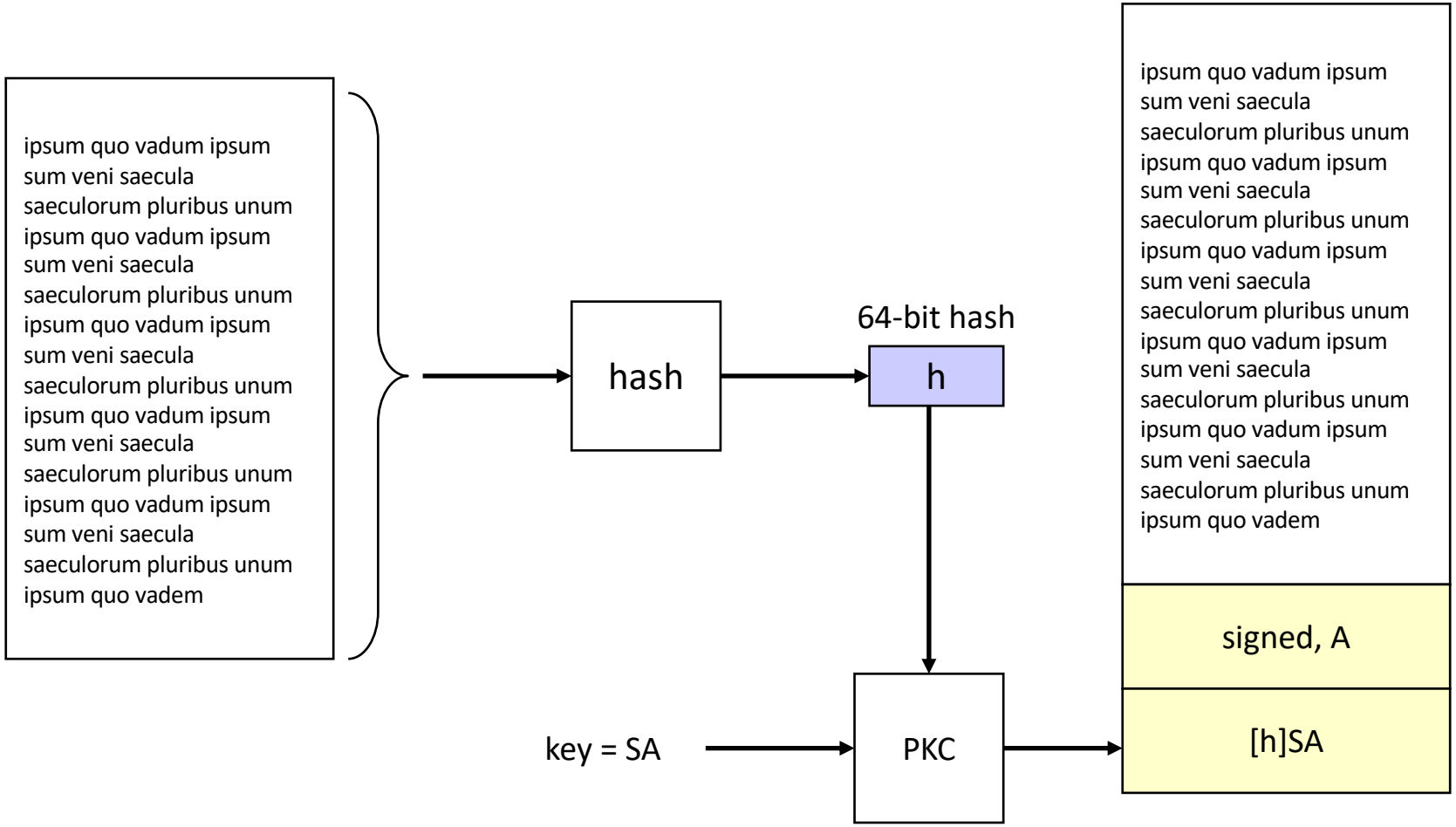
```
out = 0
while blocks remain do {
  block = next64(file)
  out = DES (“0”, block ⊗ out) }
```



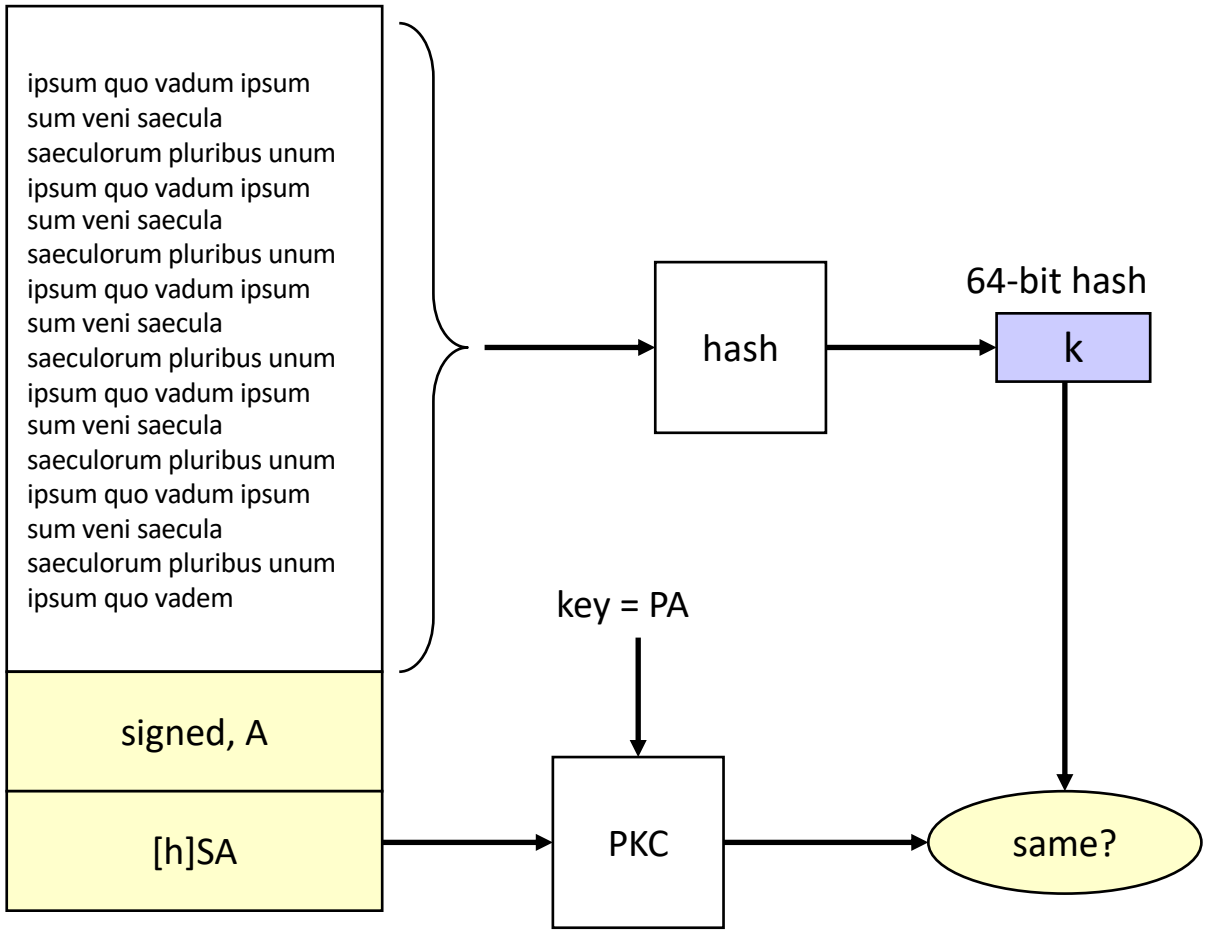
a document

hash box

What A Does to Sign Document



What B Does to Verify Document Authenticity



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

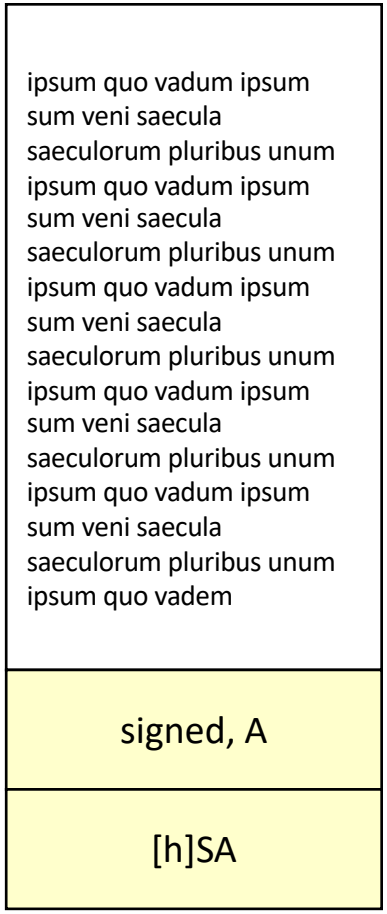
NOTE:

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.

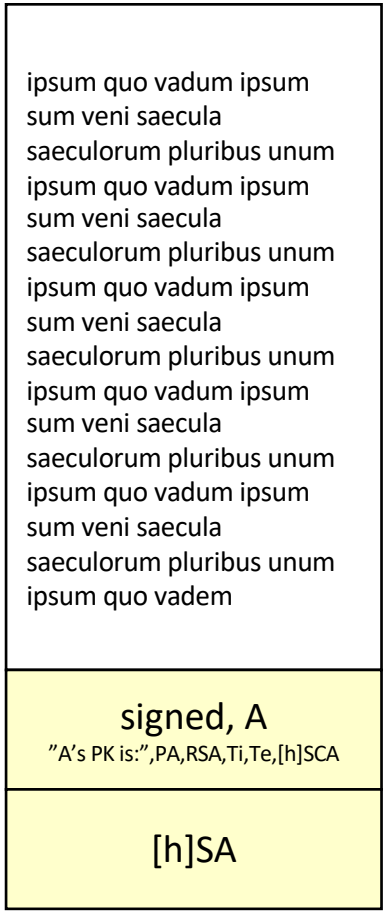
Trust in the certificate depends on the identification procedure used by CA and on the subsequent verification process at time certificate is used.

BEFORE



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



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.)

Summary

- Security is a complex topic, of long concern to OS designers
- Security considerations pervade the design of the kernel and of service processes that communicate with other computers, either privately or via public Internet
- Cryptographic protocols are essential to establish trust in communications between computers