Key Management

COMP620 Information Privacy and Security

*"Education is the **key** to unlock the golden door of freedom."*

George Washington Carver

Reading

Read chapter 10 of the textbook on Key Management

Notation

- A message encrypted by Alice can be described as:

$$\{\text{Message text}\}\text{Key}_{\text{Alice}}$$

- Sometimes the message will include another encrypted message

$$\{\text{Message1}, \{\text{Message2}\}\text{Key}_{\text{Bob}}\}\text{Key}_{\text{Alice}}$$

Key Exchange

- For symmetric encryption, both sides need to share a key before they can use the encryption
  - For many years the government had a Marine guard carry an encryption key to a remote ambassador
- For asymmetric encryption, you need to be assured that the public key you have is authentic
- Keys can be distributed reliably by
  - Key Servers
  - Digital Certificates
  - Human interaction

Trusted Key Server

- One way to securely share keys is to use a trusted key server
- We assume that every user has a separate encryption key known only to them and the key server
- In the literature, Alice and Bob often rely on their trusted friend Cathy for keys

Nonce

- To avoid replay attacks, several security protocols use a random number to identify a message
- A randomly generated number that is used only once in a security protocol is called a nonce
- A nonce must be randomly generated in a way that cannot be determined by an attacker
- Repetitive use of a pseudo-random number generator is not secure

Cryptographically Random Numbers

- Pseudo random number generators create a stream of apparently random values according to an algorithm
- If you know one value, you can calculate the next
- True random numbers require measuring a naturally random event
- Sources of randomness include radioactive decay, thermal noise, avalanche noise in Zener diodes, clock drift, the timing of actual movements of a hard disk read/write head and radio noise

Session Keys

- An interchange key is associated with an entity *(i.e. human or particular system)*
- A session key is a short term key for a communication session
- By using session keys for a short period of time there is less exposure if the keys are compromised

Forward Search

- Paul can send George one of two messages, *“by land”* or *“by sea”*
- Benedict can encrypt both messages with Paul’s public key
- When Paul sends a message, Benedict can compare it to his messages to see what was sent

- Session keys can prevent forward searches

Public Key Request

- Alice can send a request to a key server asking for Bob's public key

1. Alice → Server: Bob, $nonce_1$
2. Server → Alice : $\{Key_{Bob}, nonce_1,\}key_{Server\ Private}$

- When Alice receives message 2, she can be assured it came from the server because it was encrypted with the server's private key and contains the correct nonce

Needham-Schroeder Key Exchange

- This algorithm allows Alice and Bob to create secure session keys through third party Cathy

1. Alice → Cathy: Alice, Bob, $nonce_1$
2. Cathy → Alice : {Alice, Bob, $nonce_1$, $key_{session}$, {Alice, $key_{session}$}key_{Bob}}key_{Alice}
3. Alice → Bob: {Alice, $key_{session}$}key_{Bob}
4. Bob → Alice: {$nonce_2$} $key_{session}$
5. Alice → Bob: {$nonce_2$-1} $key_{session}$

If Eve can see which message, she be able to send a message to Bob as Alice?

A. Alice → Cathy: Alice, Bob, $nonce_1$
B. Cathy → Alice : {Alice, Bob, $nonce_1$, $key_{session}$, {Alice, $key_{session}$}key_{Bob}}key_{Alice}
C. Alice → Bob: {Alice, $key_{session}$}key_{Bob}
D. Bob → Alice: {$nonce_2$} $key_{session}$
E. It can't be done

Kerberos

- An encryption and authentication protocol developed at MIT in the 1980s
- Used in Microsoft Windows Active Directory
- Uses symmetric key encryption
  - First versions used DES
  - AES is now used
- Kerberos provides authentication and secure access of application servers (i.e. file servers, print servers, etc.)

Servers

Kerberos requires two servers for security

- Authentication Server (AS) that validates a user
- Ticket Granting Server (TGS) that provides access to application servers if the user is allowed
- Frequently both server functions are housed in the same machine

Keys

- Every user has a secret symmetric encryption **key** known only to them and the Authentication server. The key is created from their password.
- There is an encryption **key** known only to the AS and TGS
- Each application server has an encryption **key** known only to the server and TGS
- New **keys** are created for connection to servers

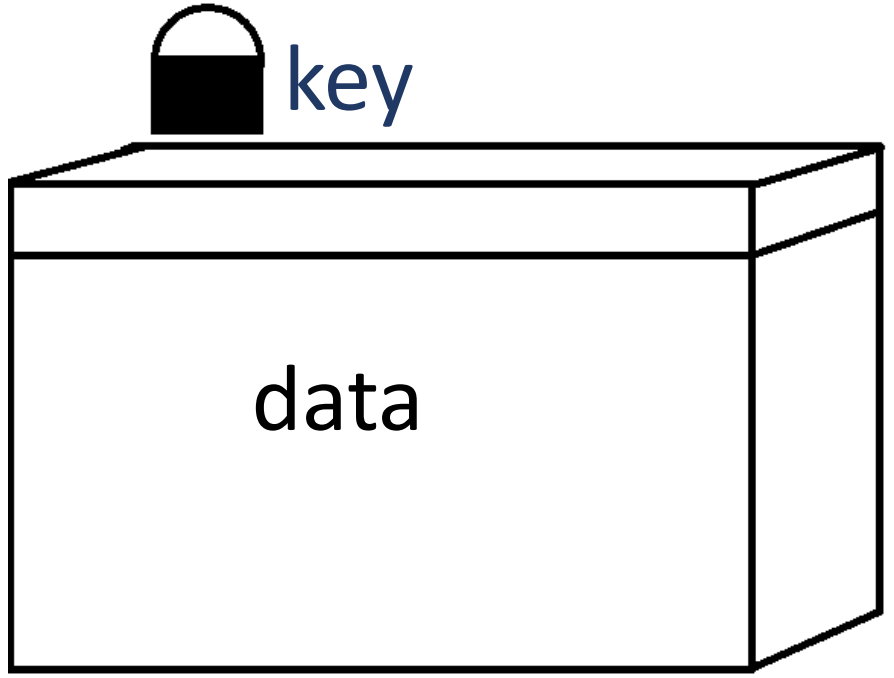
Time Stamps

- All messages contain the current time to prevent replay attacks
- Tickets are only good for a specified length of time

- *Time stamps are not shown in the following diagrams*

Tickets

- A ticket contains information, such as a new encryption key or user ID
- Tickets are encrypted. The client will receive tickets encrypted by keys it does and/or does not know



Overview

- A client is authenticated by the AS
  - This usually happens only once a day
- A client asks a TGS for permission to use an application server
  - A request is required for each server
- There are many application servers in the system for printing, files and other services

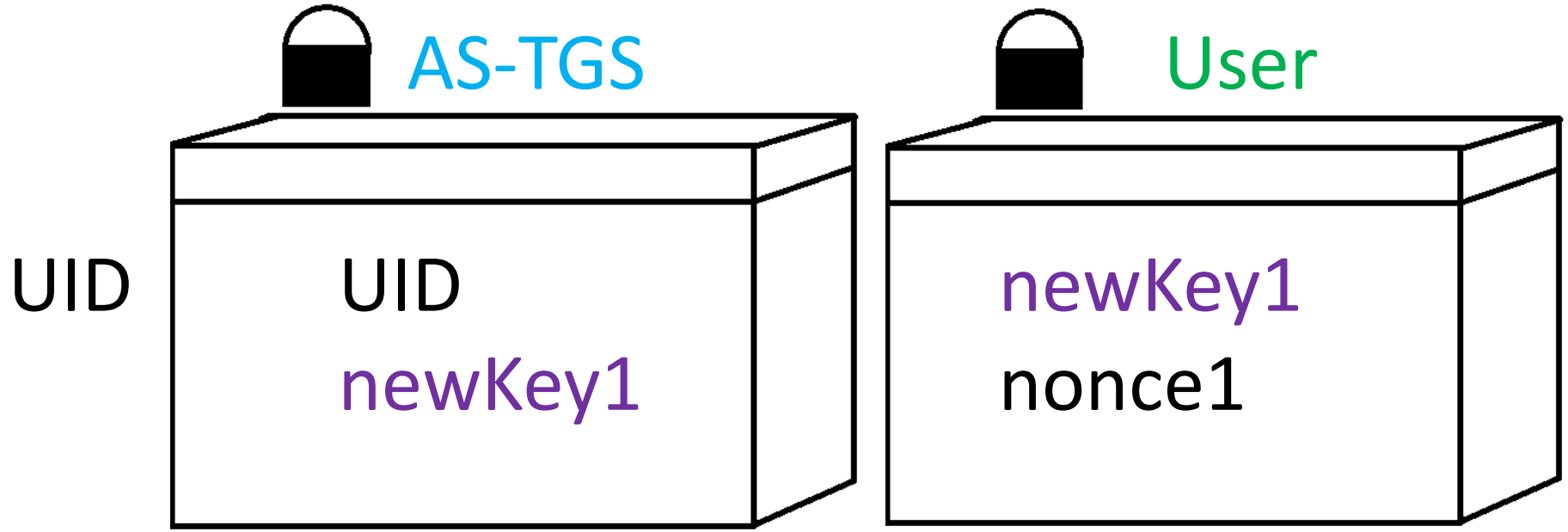
Client Login to AS

- A client first sends its userid (UID) and nonce1 to the authentication server (AS)

- Everything is sent in plaintext

user → AS: UID, nonce1

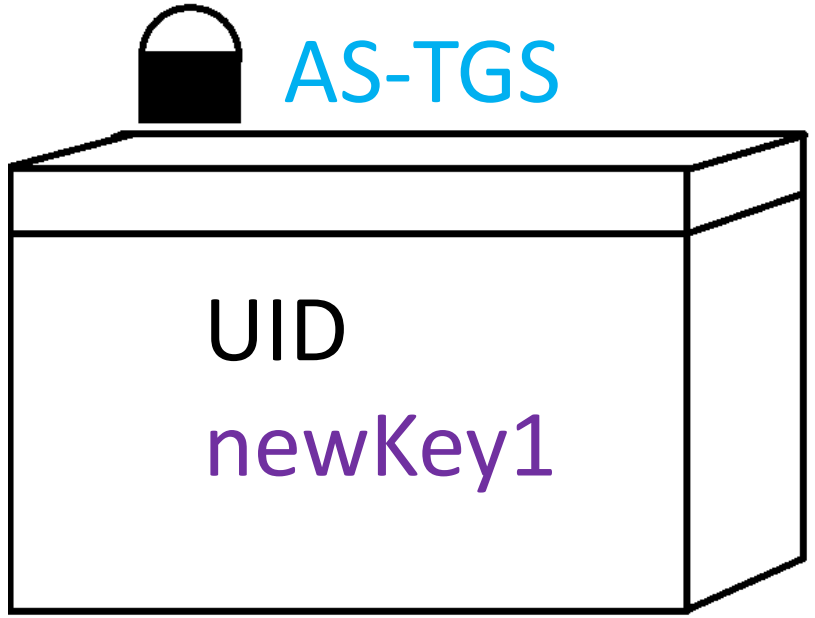
Authentication Server to Client Reply



AS → user: UID, $\{UID, newKey1\}_{Key_{AS\text{-}TGS}}$,
$\{newKey1, nonce1\}_{Key_{user}}$

The user decrypts a portion of the message to get the newKey1.
It checks to make sure the nonce1 is the same one it sent.

Client to TGS Request

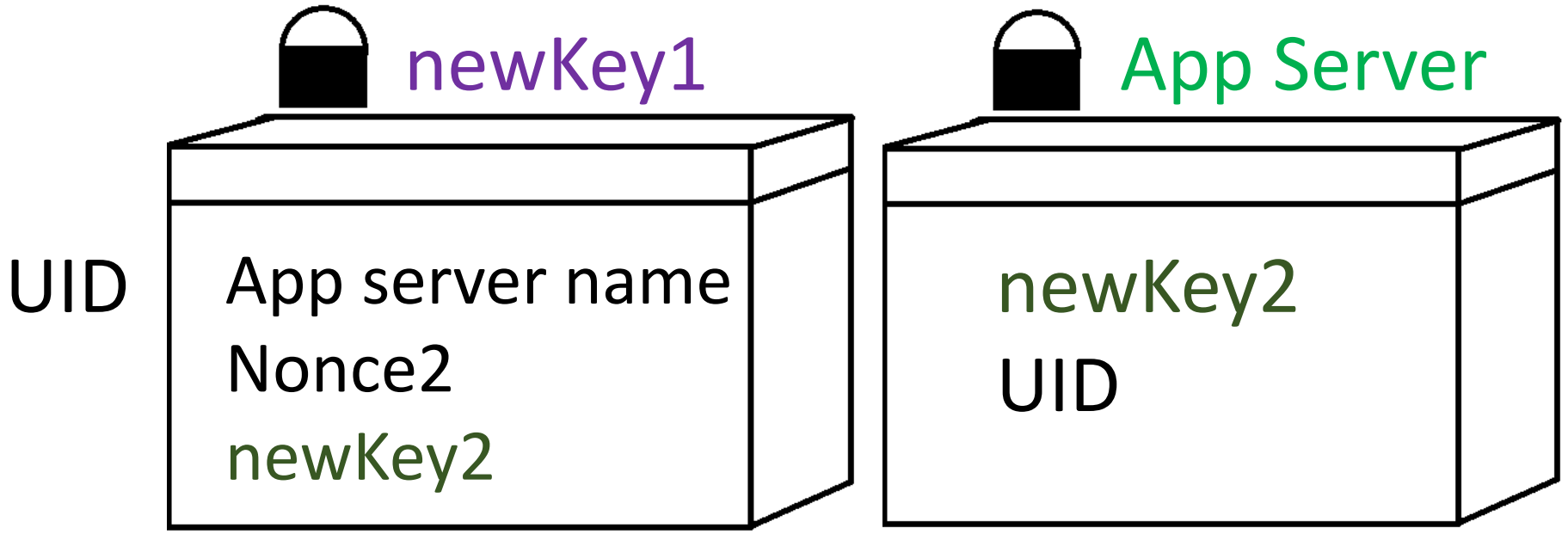


App server name
Nonce2
UID

user → TGS: $\{UID, newKey1\}_{Key_{AS\text{-}TGS}}$, App, Nonce2, UID

TGS trusts data encrypted by AS. The TGS checks if the client is allowed to access the app server.

TGS to Client Reply

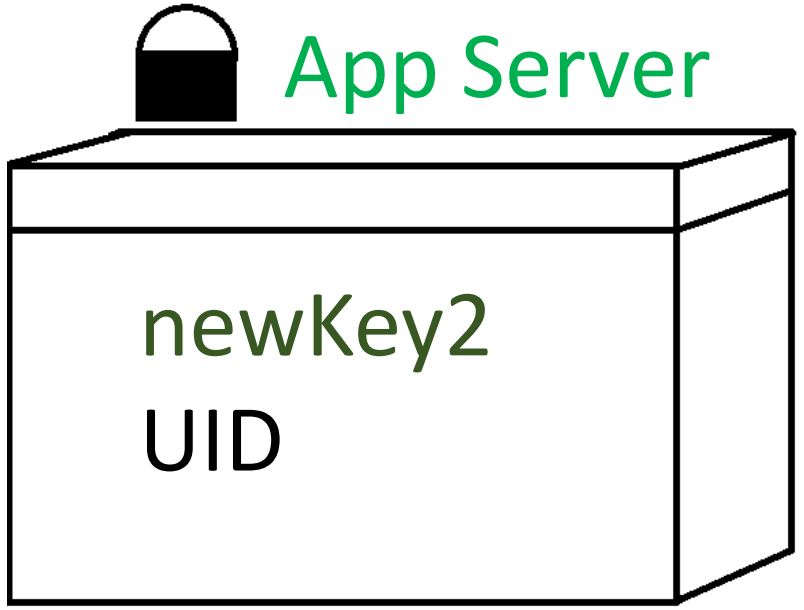


TGS → user: UID, $\{App, Nonce2, newKey2\}_{newKey1}$,
$\{newKey2, UID\}_{Key_{App}}$

The client decrypts a portion of the message to get the newKey2 for the app server. It checks to make sure the nonce2 is correct.

Client Request to App Server

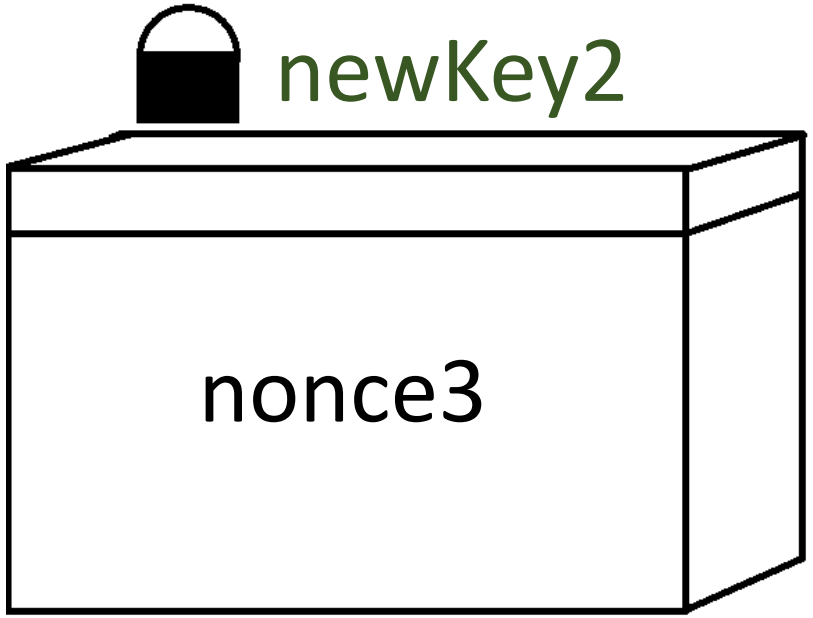
UID Nonce3



user → App: UID, Nonce3, $\{newKey2, UID\}_{Key_{App}}$

The App Server trusts the request because it contains a ticket encrypted using the key known only to itself and the TGS

App Server Reply to Client



App → user: $\{nonce3\}_{newKey2}$

All further communication with the App Server is encrypted using newKey2

How many messages do you have to send to use an application server for the first time?

A. 2
B. 4
C. 6
D. 8
E. 16

Keys for Encryption

- If you use a regular password string entered from the keyboard, it does not make a secure encryption key
- My keyboard has 47 keys or 94 values with shift
- If each character is a byte, then you can only use 94/256 or 36.7% of the possible values
- With 16 bit Unicode characters, you only use 94/64K or 0.14 % of the possible values
- $\log_2(94) = 6.55$ bits

Improved Use of Characters

- The upper bit of all keyboard characters is zero
- When stored as ASCII, the lower 32 values are not used
- The 128 – 32 = 96 values can be concatenated to make a key with fewer predictable bits

Key Strengthening

- A key string can be hashed to make a more effective encryption key
- Argon2 is a key derivation function that was selected as the winner of the Password Hashing Competition in July 2015. It was designed at the University of Luxembourg
- It uses repeated hashing of the key
- PBKDF2 is another Password-Based Key Derivation Function

Digital Signatures

Offer similar protections as hand-written signatures in the real world

- Difficult to forge
- Easily verifiable
- Not deniable
- Easy to implement
- Differs from document to document

Digital Signature

- Digitally signed messages can have clearly viewed plaintext in the body of the message, the objective is to verify the sender
- With RSA public key encryption either key can be used to encrypt or decrypt

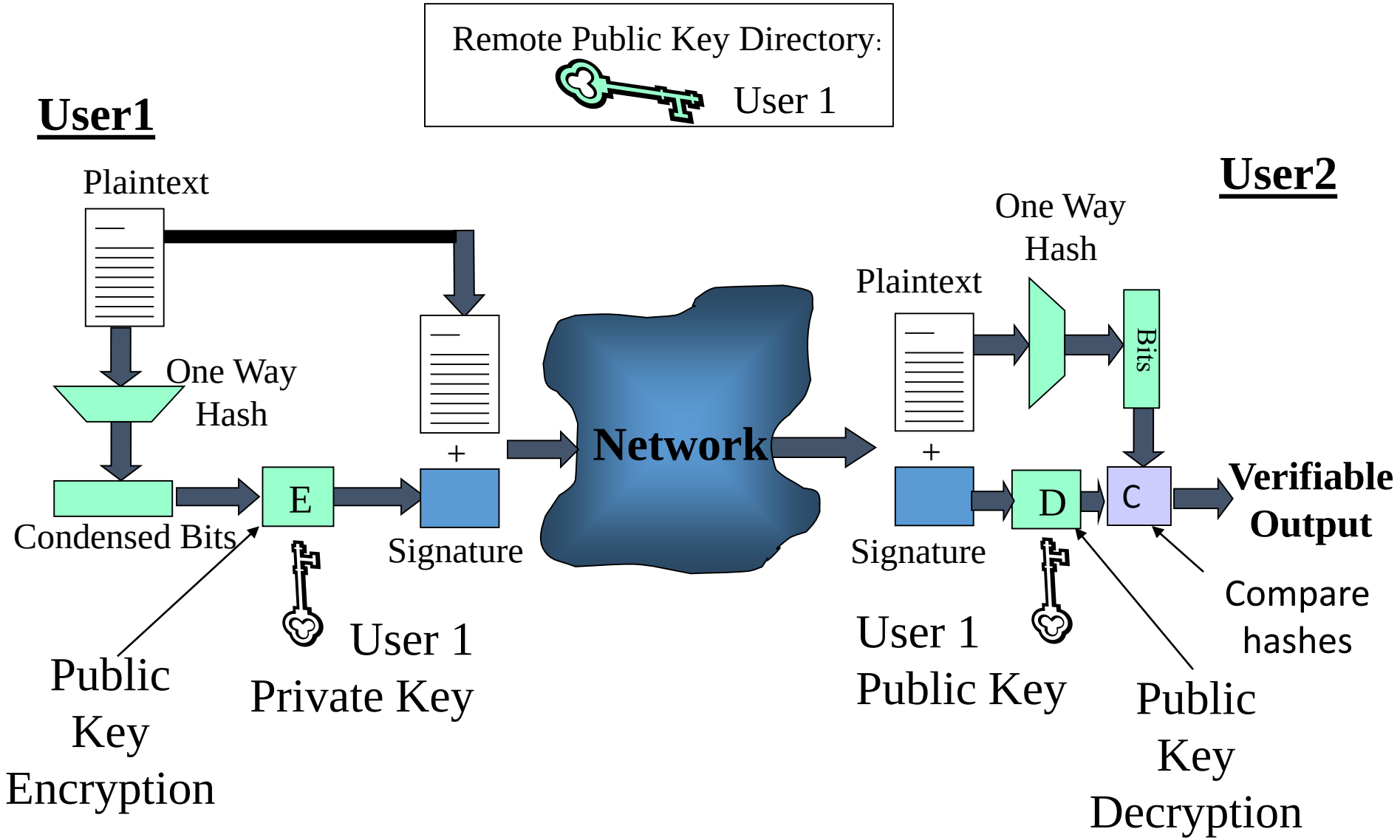
Digital Signatures provide

A. Confidentiality
B. Integrity
C. Availability
D. All of the above
E. None of the above

Digital Signature Process

- A hash of the data is created. The name of the sender is appended to the hash.
- The hash is encrypted with the private key of the sender
- The hash is appended to the data and transmitted together
- The receiver decrypts the hash with the public key of the sender
- The receiver calculates a hash of the message and compares it to the received hash

Digital Signature



Digital Signature Use

- Digitally signed email verifies the sender
- Signed applets or programs come from a known source and have not been modified
- Digitally signed programs cannot be modified or infected with a virus without detection
- Digitally signed documents cannot be changed without detection

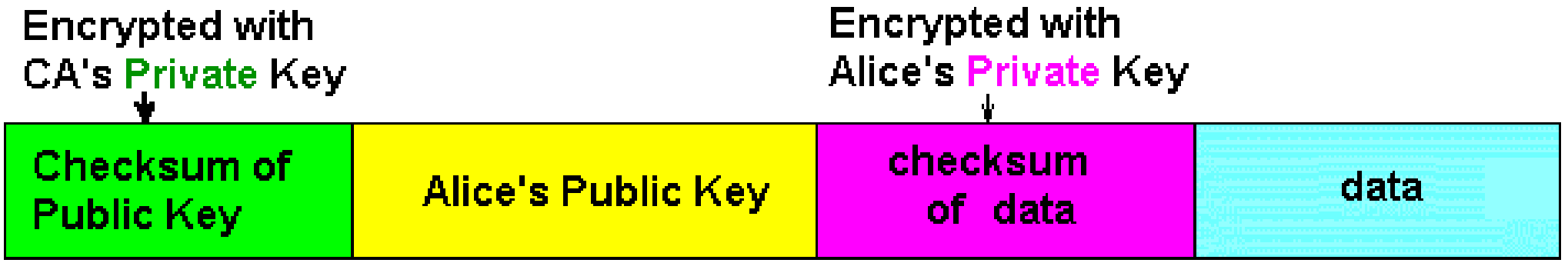
Signed Programs

- Many programs that are available on the web are digitally signed so the users can have some sense of trust to run them
- A program with an invalid signature should be treated with great caution

Digital Certificates

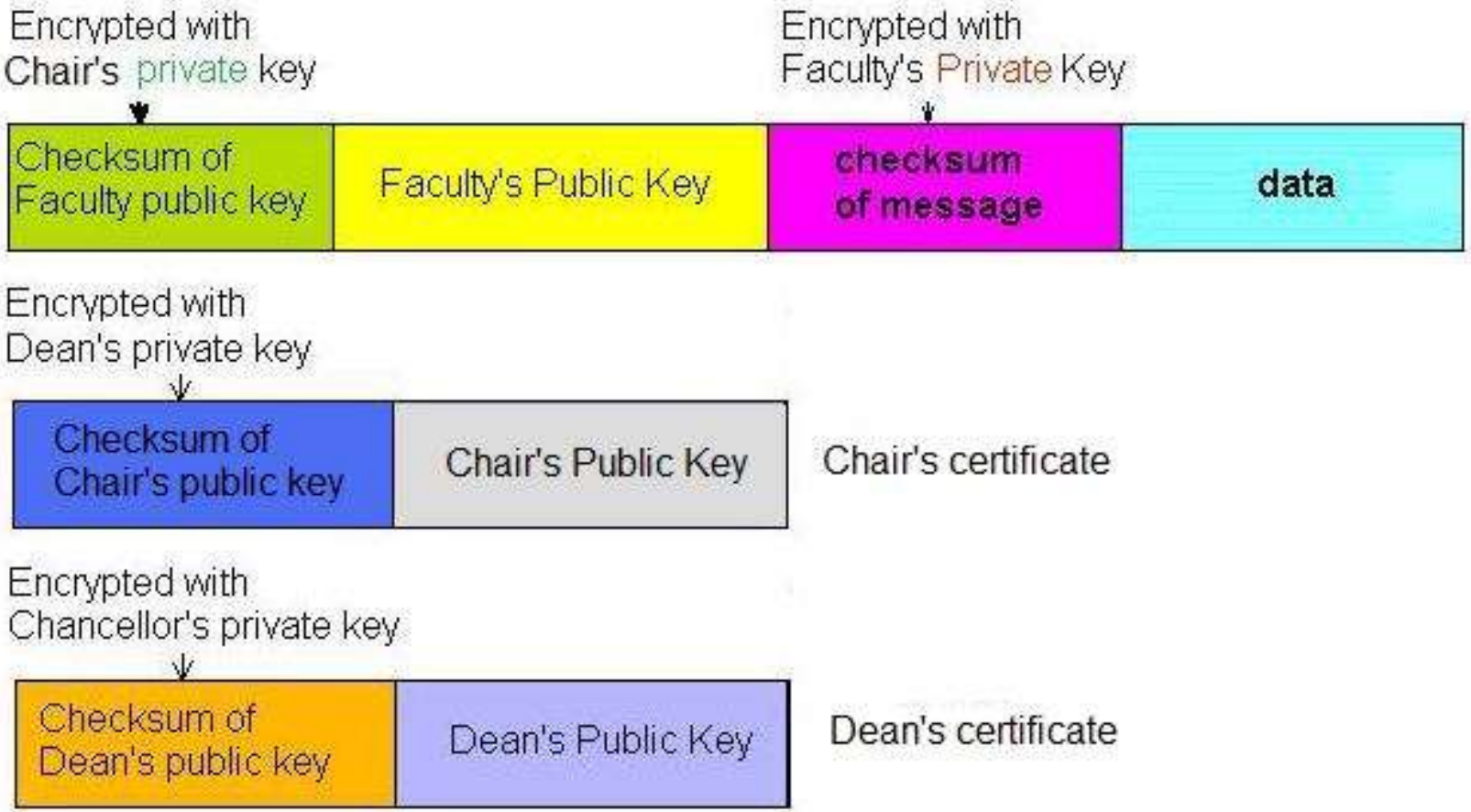
- A digital certificate contains a user's public key along with some information about the user, such as their email address
- The digital certificate is digitally signed by a Certificate Authority
- Certificate Authorities are venders of digital certificates
- Clients must know the public key of the Certificate Authority

Digital Certificates



University Certificates

- A chain of trust can be established from a single point in an organization.
- Only the top public key is needed by everyone



X.509

- X.509 is a standard that defines the format of public key certificates defined by the International Telecommunications Union's Standardization sector (ITU-T)
- Used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS

Buying Digital Certificates

- Several companies sell digital certificates
- They are usually expensive for servers
- Free certificates are available for personal email

Creating Digital Certificates

- Security tools available with Java SDK

Tool Name	Brief Description
keytool	Manage keystores and certificates.
jarsigner	Generate and verify JAR signatures
policytool	GUI tool for managing policy files.

Sign Something

- Acquire a digital certificate
- Send me a digitally signed email

or

- Sign a Jar file and email it to me
- Your name must be associated with the certificate
- Due by midnight on Tuesday, September 4

Programming Assignment

- The programming assignment to encrypt and decrypt a file is due before midnight on Thursday, August 30
- Upload your source code to Blackboard along with a short explanation of the algorithm used and the format of the file
- This is a team project to be done by pairs of students

Reading

Read chapter 10 of the textbook on Key Management