ACCESS CONTROL

Prepared By

-Anooja Joy

ACCESS CONTROL BASICS

- **"Access"** is a key concept that implies **flow of information** from a **subject to an object** or from an **object to a subject.**
- **Eg:**
  - When a **user (a subject) updates a data set (an object)**, the information flows from the subject to the object.
  - When **a user reads a record from a data set**, the information flows from the object to the subject.
- **Access control policies** make sure of fundamental requirements of a secure system through **authentication** and **authorization.**
- It is typically provided to **Operating System** and **Databases.**

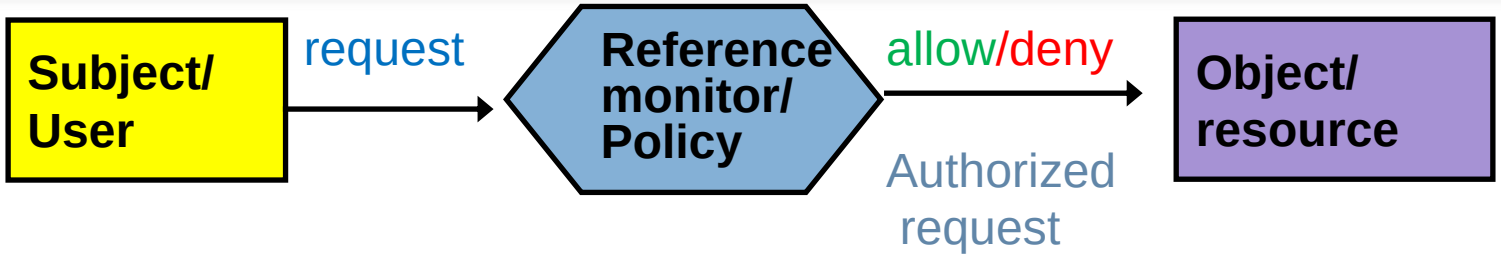
Access Control

- **Access control** is the collection of mechanisms for selective restriction of access to a **place or other resource** ie, determining who has access to specific files, or other system resources. **Its function is to control which (active) subject have access to which (passive) object with some specific access operation.**
- **Examples**
- Passwords, Encryption, Antivirus software, Firewalls, Routers, Protocols to preserve confidentiality and integrity of data.
- The main aim of access conbtrol mechanism is **to control operations executed by subjects** in order **to prevent actions that could damage data and resources.**

Goals of access control

1. Granting access
2. Limiting access
3. Preventing access
4. Revoking access

Abstract Access Control Model



- The very nature of access control suggests that there is an ***active subject*** requiring access to a ***passive object*** to perform some specific access operation.
- In any access control model, the entities that can **perform actions/ inititates requests in the system** are called subjects. Subjects Issues requests to access the object. Subjects can be considered as software entities/ human users/processes, modules, roles and Objects can be files, processes, etc.

Eg: Users, Process, Groups

- The **entities representing resources that hold data** to which access may need to be controlled are called objects.

Eg: memory, files, directories, nodes on a network, I/O devices, interprocess messages etc.

- A reference monitor **grants or denies access.** Reference monitor **makes authorization decisions.** It knows which subjects are allowed to issue which requests.

Eg: Firewall, Operating System, Database, JVM

Abstract Access Control Model

- The **access control mechanism** uses some access control policies to decide **whether to grant or deny a subject access to a requested resource.**
- Access control policies map principals, objects and access rights.
- Access rights refer to the user's ability to access a system resource.
  - read
  - write
  - append
  - execute

Access Modes

- 2 access modes

1. **Observe:** Subject may only look at the content of the object.

   Example: Read, Search

2. **Alter:** Subject may change the content of the object

   Example: Write, Append, Delete

- Any access right can be performed within each access mode.
- The owner of the resource sets the access rights to the resource.

Basic Operations in Access Control

1. **Grant permissions:** Inserting values in the matrix's entries
2. **Revoke permissions:** Remove values from the matrix's entries
3. **Check permissions:** Verifying whether the entry related to a subject *s* and an object *o* contains a given access mode

Access Control Policies

- **Access Control Policy:** specifies how accesses are controlled and access decisions determined.
- It means that subjects (i.e. users, agents) detain privileges (i.e. rcwx) on objects (i.e. data, programs, devices) according to these **AC security policies**.
- Types
  - Discretionary AC.
  - Mandatory AC
  - Role-based AC.
- Mechanism (structure): implements or enforces a policy.
  - Access matrix.
  - AC list (ACL).
  - Capability list.

Discretionary access control (DAC)

- This policy **restricts access to system objects** based on the **identity of the users** and/or the **groups** to which they belong.
- DAC mechanism controls are defined by user identification with supplied credentials during authentication, such as username and password
- The object **Owners/ Users** can set the policies and decide who get to do what with "their" objects but still there is a security policy. For example, User A may provide read-only access on one of her files to User B, read and write access on the same file to User C and full control to any user belonging to Group 1.
- DAC is typically the **default access control mechanism** and **the least restrictive**.
- Each resource object on a DAC based system has an Access Control List (ACL) associated with it. An ACL contains a list of users and groups to which the user has permitted access together with the level of access for each user or group.
- Though Discretionary Access Control provides a flexible environment, drawback to is the fact that it gives the end-user complete control to set security level settings for other users and the permissions can be unnecessarily given to other programs which are inherited by end users and could potentially lead to malware being executed without the end-user being aware of it.

Mandatory access control (MAC)

- **Permissions** are **assigned according to the** security policy. e.g. (Privacy) Hospital records can only be accessed by medical staff. Doctor cannot decide to give non-staff access.
- People are granted access based on an **information clearance**. A **central authority** regulates access rights based on **different security levels.**
- Used within organizations with a strong need for central controls and a central security policy. It's common in government and military environments.
- Mandatory Access Control begins with security labels assigned to **all resource objects** on the system. These security labels contain two pieces of information - **a classification** (TOP SECRET>SECRETET> CONFIDENTIAl >UNCLASSIFIED) and **a category** (which is essentially an indication of the **management level, department** or project to which the object is available).
- When a user attempts to access a resource under Mandatory Access Control the system checks the user's classification and categories and compares them to the properties of the object's security label. If the user's credentials match the MAC security label properties of the object access is allowed.

Role-based access control (RBAC)

- Role Based Access Control (RBAC), also known as Non discretionary Access Controlor Rule based access control , is based on a user's job function or role within the organization to which the computer system belongs.
- Roles can have overlapping responsibilities and privileges. Roles can be updated without updating the privileges of every user on individual basis.
- This widely used method is based on a complex combination of role assignments, authorizations, and permissions.
- Users are then assigned to that particular role. For example, an accountant in a company will be assigned to the Accountant role, gaining access to all the resources permitted for all accountants on the system. Similarly, a software engineer might be assigned to the developer role.
- Permissions are associated with:

  i.roles (= set of actions and responsibilities, associated with particular working activities), and

  ii.users/subjects are coupled to appropriate roles.
- With RBAC one can introduce enterprise-specific security policies.

Attribute-based access control (ABAC)

- Attribute-based access control (ABAC), also known as policy-based access control, defines an access control policy by which access rights are granted to users through the use of policies which combine attributes together. The policies can use any type of attributes (user attributes, resource attributes, object, environment attributes etc.).
- In this dynamic method, access is based on a set of attributes and environmental conditions, such as time of day and location, assigned to both users and resources.
- Attributes are sets of labels or properties that can be used to describe all the entities that must be considered for authorization purposes. Each attribute consists of a key-value pair such as “Role=Manager”.

Access Control Mechanism

- Access control is usually taken care of in following steps, which are authentication, and authorization
  - Ø **Authentication:** Are you who you say you are? Who are you?
    - v Determine whether access is allowed or not
    - v Determining the identity of a user ie, May also involve verification of IP address, machine, time, etc…
    - Enforcement Mechanism: (password/crypto/smartcard/biometricetc.)
  - Ø **Authorization:** What are you allowed to do? Are you allowed to do that?
    - v Once you have access, what can you do?
    - v Enforces limits on actions
    - Enforcement Mechanism: (Acess control)
- Verification of access rights -->***Access Control***
- Granting of access rights --> ***Authorization***

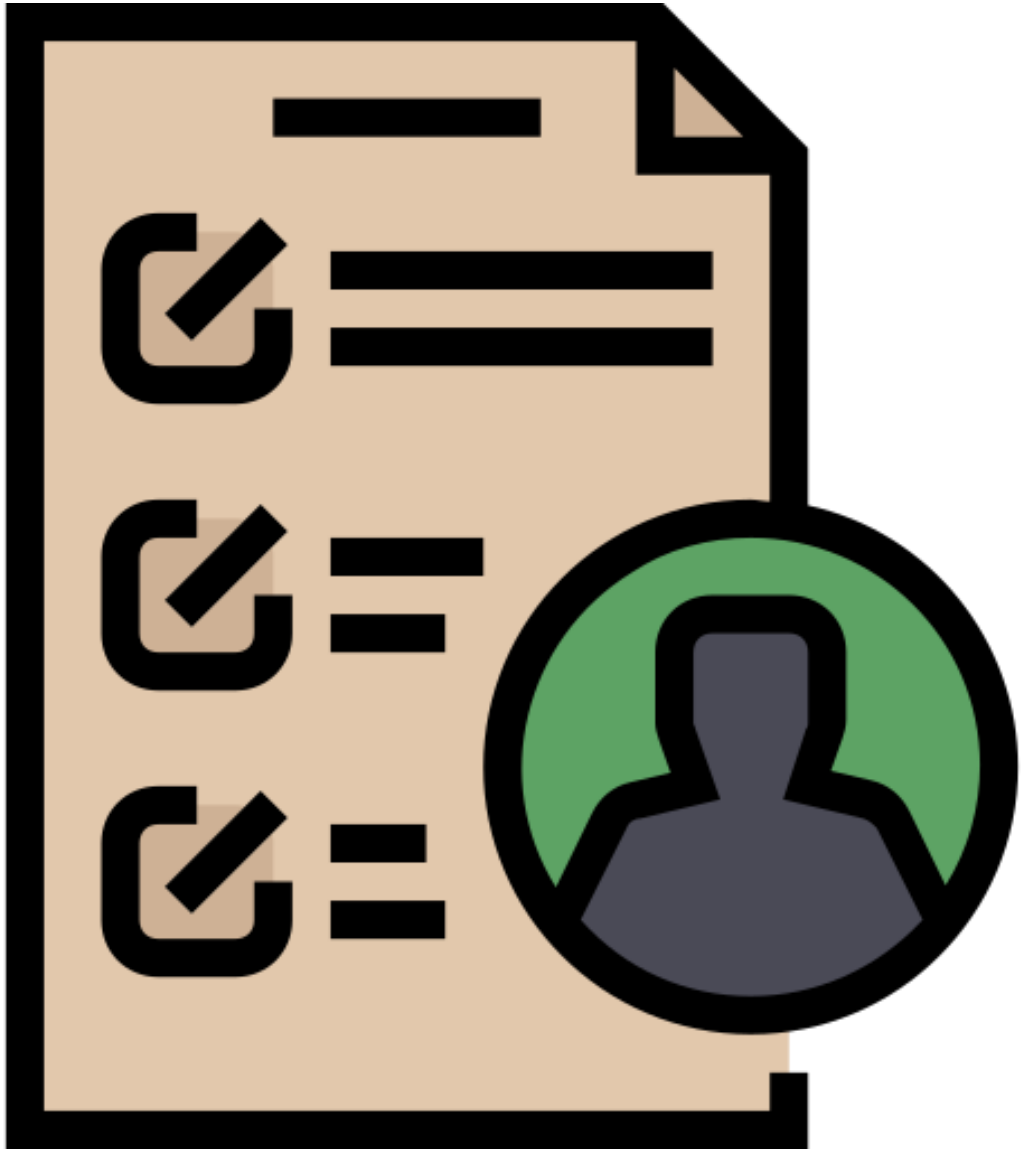
AUTHENTICATION

- Verifies credentials to determine whether users are who they claim to be.
- Users or persons are verified.
- Works through passwords, biometrics, one-time pins, or apps
- Eg: Logins, OTP or a magic link, 2FA/MFA, Single sign-on (SSO)
- It is **visible to the user**
- It is **partially changeable by the user.**
- **Data move** through **ID tokens**

AUTHORIZATION

- Grants or denies permissions to determines what users can and cannot access.
- Users or persons are validated.
- Works through settings maintained by security teams
- Eg: Role-based access controls (RBAC), JSON web token (JWT),SAML(a standard SSO format), OAuth
- It is **not visible to the user**
- It is **not changeable by the user.**
- **Data move** access tokens

AUTHORIZATION



Access Control Structures

- Access control structures are mechanisms for implementing access policies or used for implementation of protection model:

1. Access Control Matrix
2. Capability Tables
3. Access Control Lists
4. Role-Based Access Control
5. Rule-Based Access Control
6. Restricted Interfaces
7. Content-Dependent Access Control

Requirements for access control structures:

- an ability to express control policies
- verifiability of correctness.
- scalability and manageability

Access Control Matrix

- Access control matrix is a basic control structure to implement protection model.
- The *access control matrix* is a matrix with each subject/domain(user/process/procedure) represented by a row, and each object(resources) represented by a column
- Characterizes possible rights of each subject with respect to each object ie, access allowed by subject S to object O is stored in intersection. The entry *M[s, o]* lists the operations that subject *s* may carry out on object *o*.
- *ACM enforces restriction on accounting data.*
- The access control matrix provides a useful way to think about the rights in the system.
- **Examples:** read/write/execute/etc

Access Control Matrix

- Subjects (users) index the rows
- Objects (resources) index the columns

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

Example

Objects

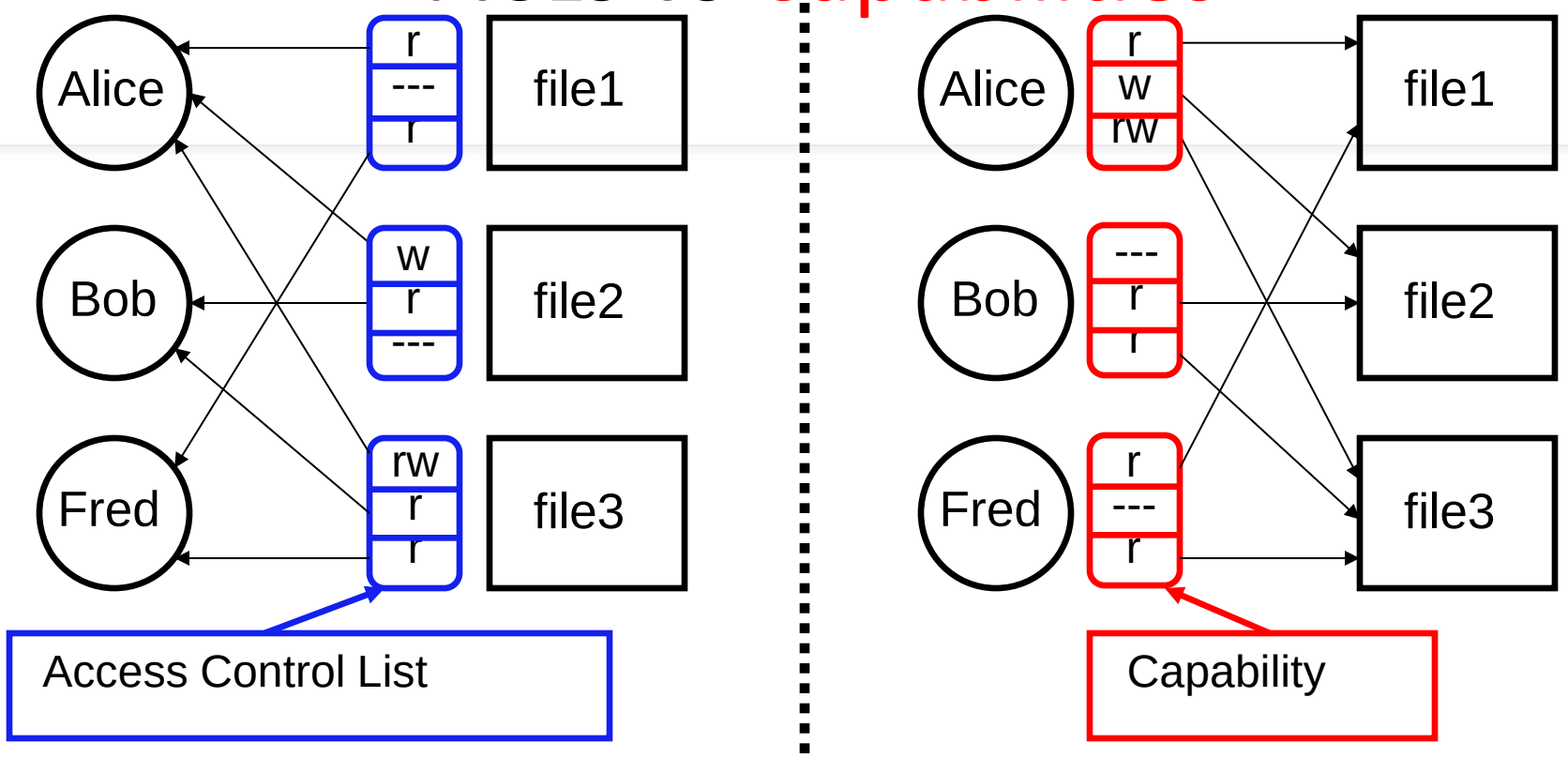
Subjects

	File 1	File 2	File 3	…	File n
User 1	{r,w}	{w}			{r,w}
User 2	{w}	{w}	{r,w}		
User 3				{r}	{w}
…					
User k	{r}	{r}	{r,w}	{r}	{w}

Access Control Matrix

- **Advantages:**
  - clarity of definition
  - easy to verify
- **Disadvantages**:
  - Poor **Scalability.** "Small' change can result in "many" changes to the access control matrix
  - **Poor handling of changes.** One central matrix modified every time subjects/objects are created/deleted, or rights are modified
  - Number of subjects/objects is very large. Could be 100's of users, 10,000's of resources, Then matrix has 1,000,000's of entries. How to manage such a large matrix?

ACLs vs Capabilities



- Note that arrows point in opposite directions...
- With ACLs, still need to associate users to files

2. Access control lists (ACLs)

- ACL is an **object-centered description of access rights**: Can be viewed as storing the columns of the access control matrix with the appropriate object, leaving out empty entries
- Ideally, one list per object showing all subjects with access and their rights
- Missing subjects given “default” access. Easy to make an object public

EXAMPLE:

- bill.doc: {Bob: read,write}
- exit.exe: {Alice: execute}, {Bob: execute}
- fun.com: {Alice: execute, read}, {Bob: execute, read,write}

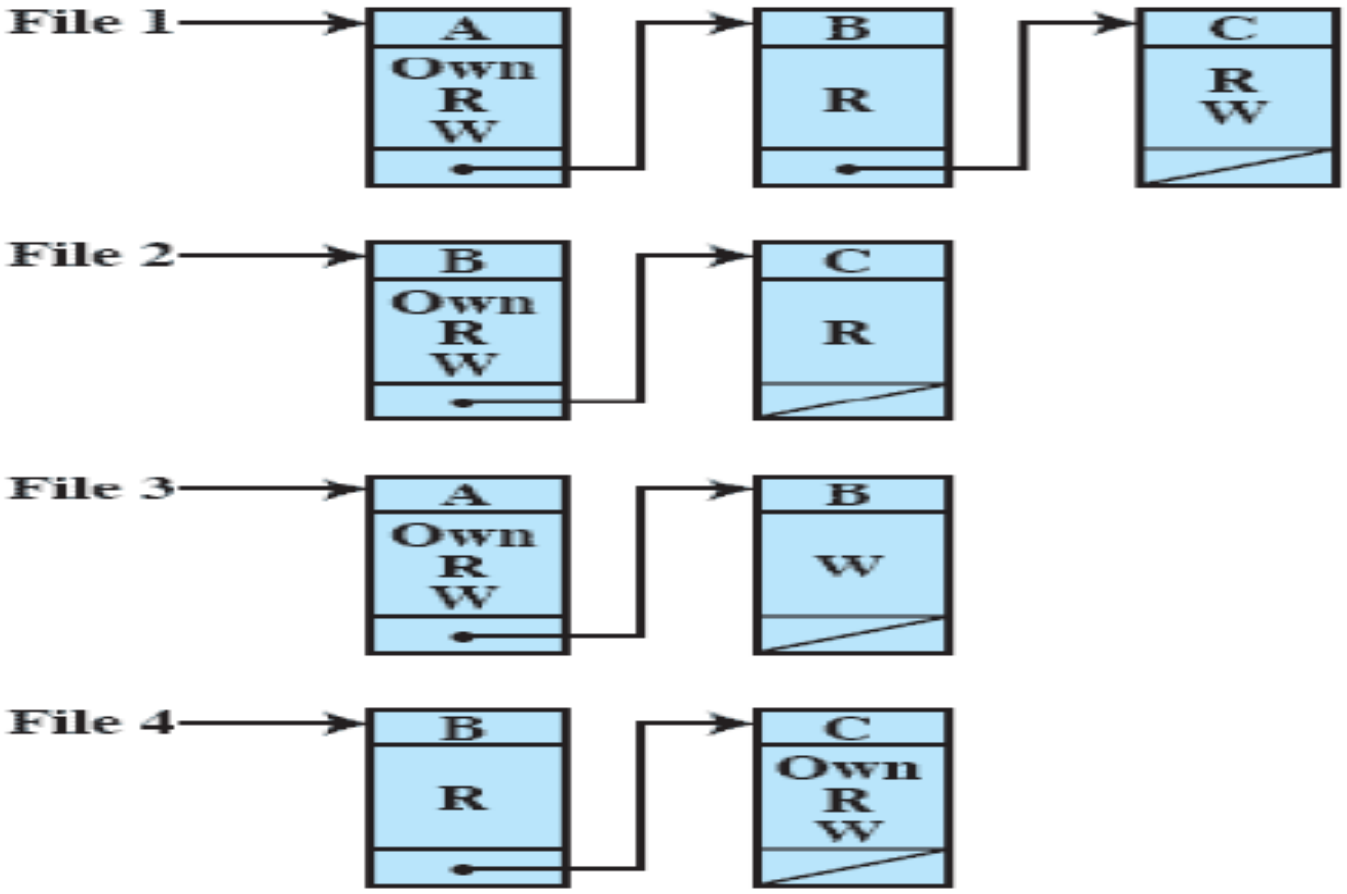
ACL

For example, the ACL corresponding to insurance data

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	r

(Bob, —), (Alice, rw), (Sam, rw), (accounting program, rw).

ACL



(b) Access control lists for files of part (a)

ACL

- **Advantages:**
  - easy access to object access rights
- **Disadvantages:**
  - poor overview of access rights per subject
  - difficulty of revocation
  - difficulty of sharing
  - Need a mechanism for handling conflicts

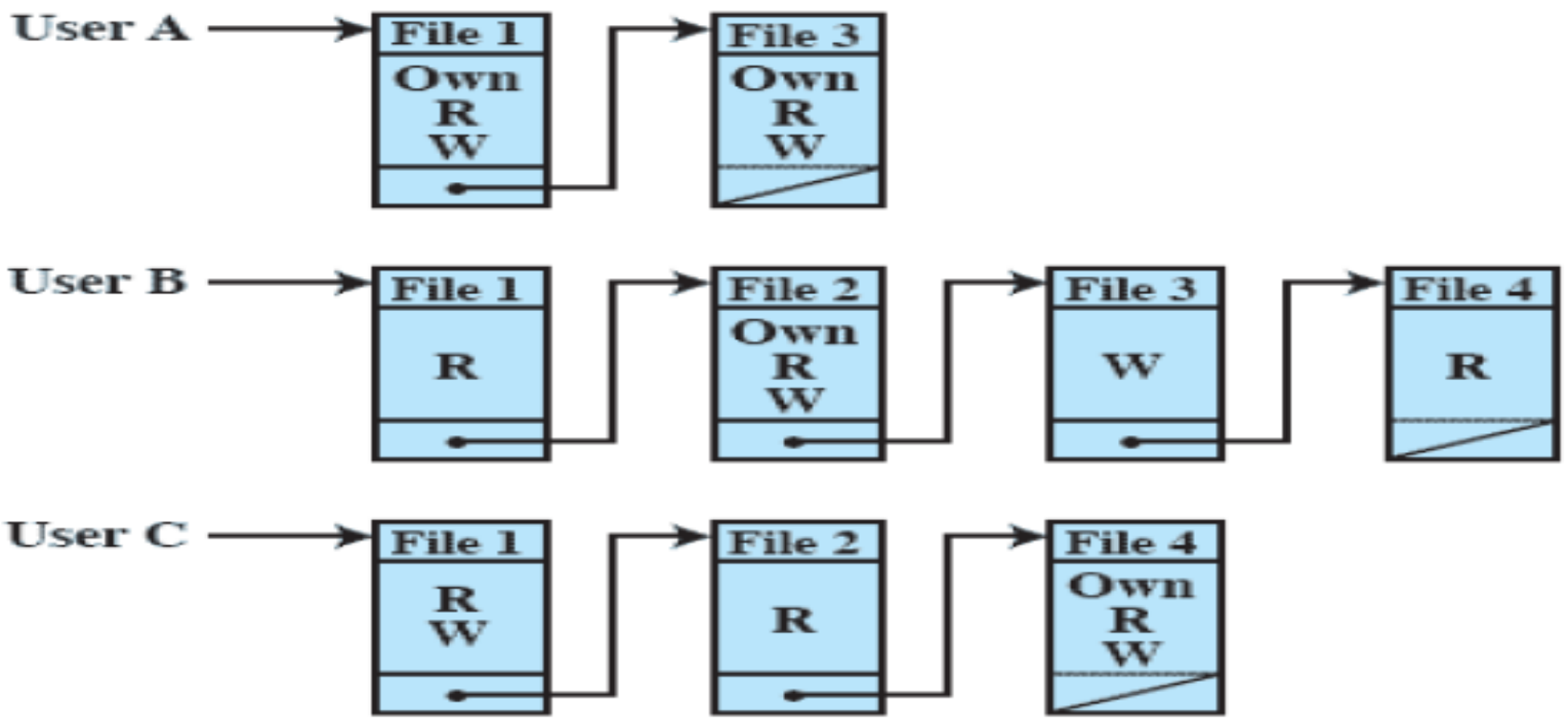
3. Capabilities

- Can be viewed as storing the rows of the access control matrix with the corresponding subject, leaving out empty entries.
- Whenever a subject tries to perform an operation, consult its row of the access control matrix to see if the operation is allowed.
- A capability is an unforgeable token giving user access to an object and describing the level of allowable access
- Capabilities can specify new types of rights

EXAMPLES:

- Alice: {edit.exe: execute}, {fun.com: execute, read}
- Bob: {bill.doc: read,write}, {edit.exe: execute}, {fun.com: execute, read,write}

Capabilities



(c) Capability lists for files of part (a)

Capabilities (or C-Lists)

- Store access control matrix by **row**
- Example: Capability for **Alice** is in **red**

Subjects \ Objects	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

Capabilities: two approaches

Advantages:

- easy ownership transfer
- easy inheritance of access rights

Disadvantages:

- poor overview of access rights per object
- difficulty of revocation
- need for extra integrity protection

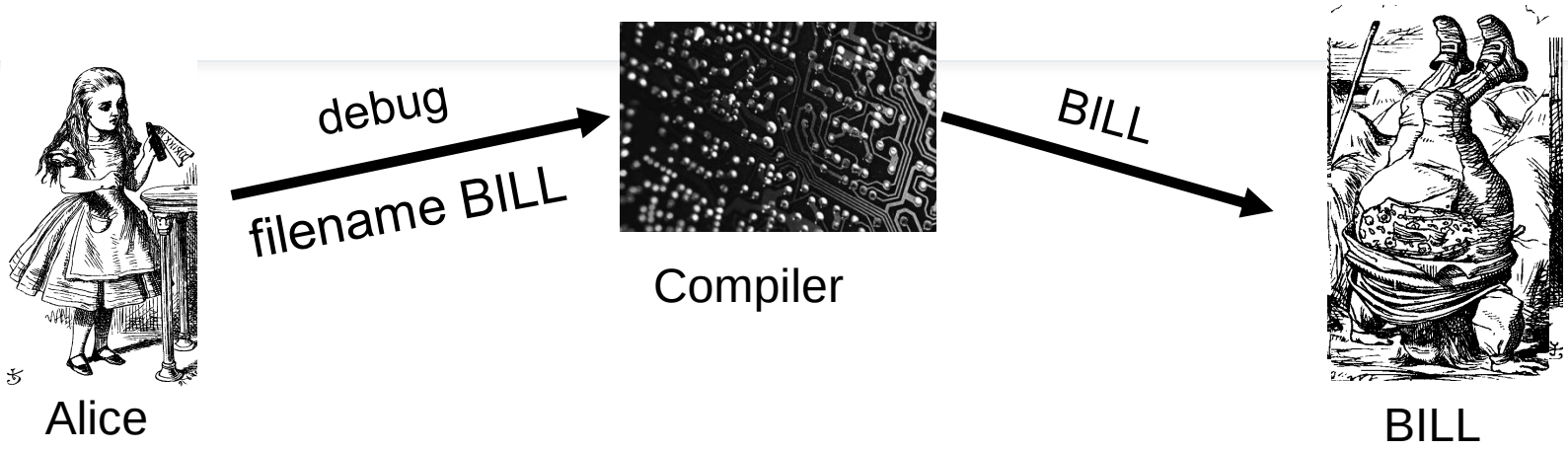
- Ticket is held by the OS, which returns to the subject a pointer to the ticket
- Ticket is held by the user, but protected from forgery by cryptographic mechanisms
  - How?
  - Ticket can then be verified by the OS, or by the object itself

Confused Deputy

- Consider a system with:
  - **2 System resources:** Compiler and BILL file (billing info) that contains critical billing information
  - **1 user:** Alice.
- Compiler can write to any file, while Alice can invoke the compiler and she can provide a filename where debugging information will be written.
- However, Alice is not allowed to write to the file BILL, since she might corrupt the billing information.

	Compiler	BILL
Alice	x	—
Compiler	rx	rw

Confused Deputy



- Suppose that Alice is invoking compiler, and she provides BILL as the debug filename. This command should fail, as Alice doesn't have privilege to access file BILL, does have the privilege to overwrite BILL. The result of Alice's command should be the trashing of the BILL.
- Ccompiler is **deputy** acting on behalf of Alice and Compiler is **confused** since its acting based on it's own privileges or should be acting based on ALice's proivileges.

Confused Deputy

- There has been a separation of **authority** from the **purpose** for which it is used
- With ACLs, more difficult to prevent this
- With Capabilities, easier to prevent problem
  - Must maintain association between authority and intended purpose
- Capabilities — easy to **delegate** authority

ACLs vs Capabilities

- ACLs
  - Preferable when users manage their own files and when protection is data oriented.
  - Protection is data-oriented
  - Easy to change rights to a particular resource
  - more difficult (but not impossible) to avoid the confused deputy
  - ACLs are used in practice far more often than capabilities
- Capabilities
  - Easy to delegate avoid the confused deputy
  - Easy to add/delete users
  - More complex to implement and they have somewhat higher overhead
  - Due to the ability to delegate, easy to avoid the confused deputy when using capabilities

COVERT CHANNEL

Covert Channel

- **Covert channel**: is a communication path not intended as such by system's designers, that can be exploited by a process to transfer information in a manner that violates the system security policy to outside sources.
- It arises in **network communications** and **operating systems** and are **impossible to eliminate**.
- It is hidden from the access control mechanisms of systems since it does not use the legitimate data transfer mechanisms of the computer system (typically, read and write), and therefore cannot be detected or controlled by the security mechanisms.
- There are two kinds of covert channels:
  1. **Storage channels** - A process writes data to a storage location where a process of lower clearance level is able to read it. Communicate by modifying a "storage location", such as a hard drive, system variables and attributes (other than time) to signal information, e.g. – file status
  2. **Timing channels** - Delay between the packets is used to transmit data. Perform operations that affect the "real response time observed" by the receiver. It vary the amount of time required to complete a particular task, e.g. – influencing the number of CPU cycles available to a given process in a given time frame.

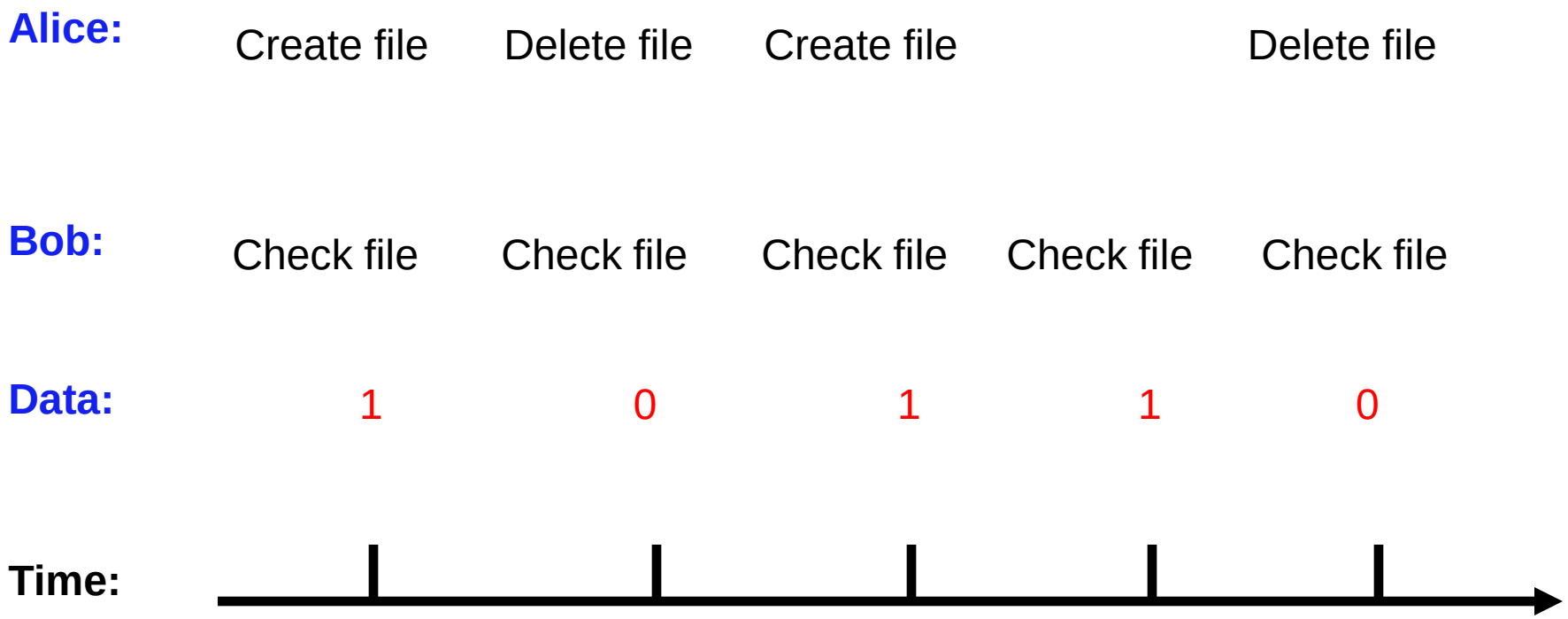
Storage Covert Channel Criterion

1. The sending and receiving entities must have access to the same shared resource or attribute.
2. There must be some means by which the sending entity can force the shared resource or attribute to change.
3. There must be some means by which the receiving entity can detect the change.
4. There must be some mechanism for initiating the communication between the sending and receiving entities and for sequencing the events correctly. This mechanism could be a covert channel with a lower bandwidth.

Storage Covert Channel Example

- This is often accomplished with malware where an individual plants a malicious piece of code on your system and that code then attempts to identify valuable information and then exfiltrate it out of your system using a covert channel.
- Alice has **TOP SECRET** clearance, Bob has **CONFIDENTIAL** clearance
- Suppose the file space is shared by all users
- **Alice** creates file **FileXYzW** to signal **"1"** to **Bob**, and removes file to signal **"0"**
- Once per minute Bob lists the files
  - If file FileXYzW does not exist, Alice sent 0
  - If file FileXYzW exists, Alice sent 1
- Alice can leak **TOP SECRET** info to Bob

Covert Channel Example



Covert Channel

- Consider 100MB **TOP SECRET** file
  - Plaintext stored in **TOP SECRET** location
  - Ciphertext — encrypted with AES using 256-bit key stored in **UNCLASSIFIED** location
- Suppose we reduce covert channel capacity to 1 bit per second
- It would take more than 25 years to leak entire document thru a covert channel
- But it would take less than 5 minutes to leak 256-bit AES key thru covert channel!

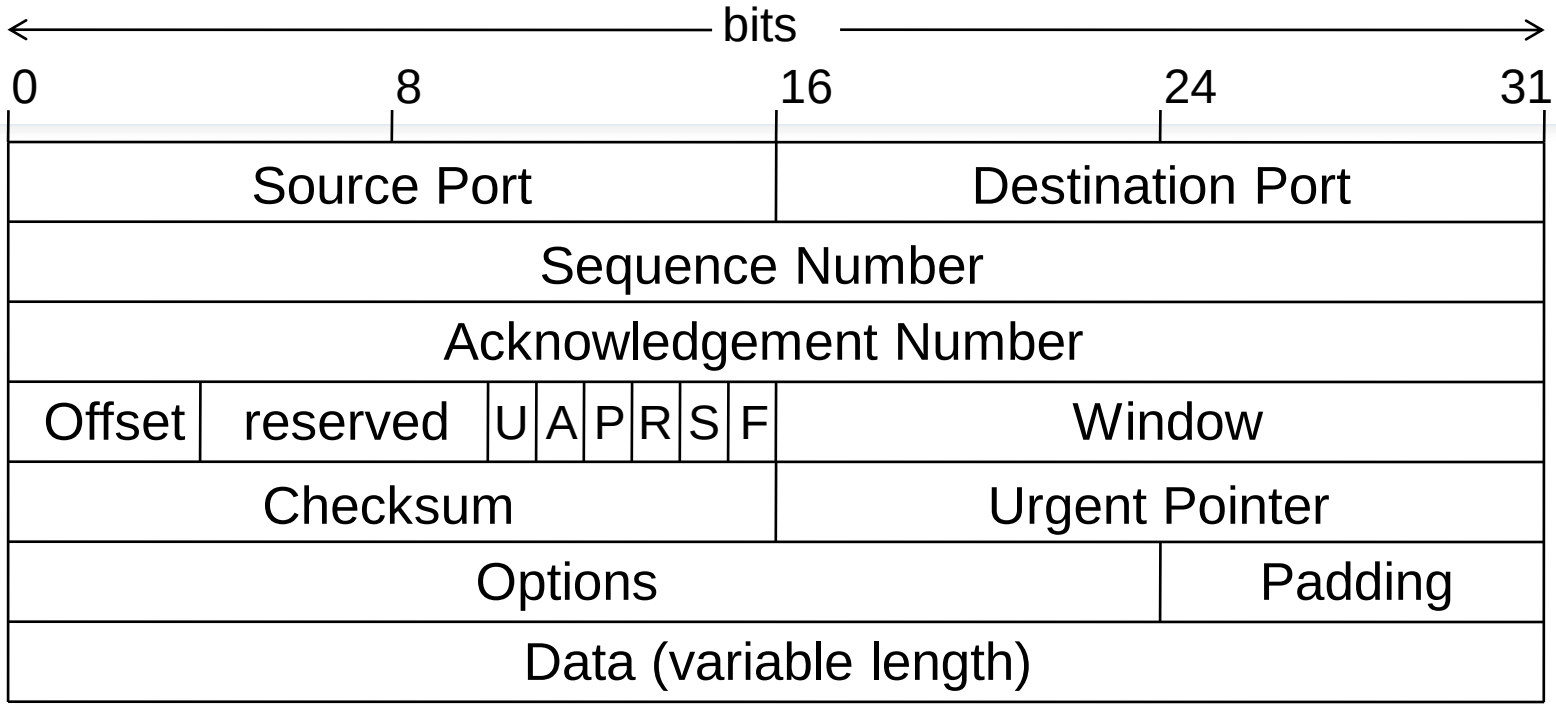
Covert Channel

- **Other possible covert channels?**
  - Print queue
  - ACK messages
  - Network traffic, etc.
- **When does covert channel exist?**
  1. Sender and receiver have a shared resource
  2. Sender able to vary some property of resource that receiver can observe
  3. "Communication" between sender and receiver can be synchronized

TIMING CHANNEL

1. The sending and receiving entities must have access to the same shared resource or attribute.
2. The sending and receiving entities must have access to a time reference such as a real-time clock.
3. The sender must be capable of modulating the receiver's perception of time for detecting a change in the shared resource or attribute.
4. There must be some mechanism for initiating the channel and for sequencing the events transmitted over it. Any time a processor or memory is shared, there is a shared attribute. A change in response time is detected by the receiving process by means of monitoring the clock or its surrogate.

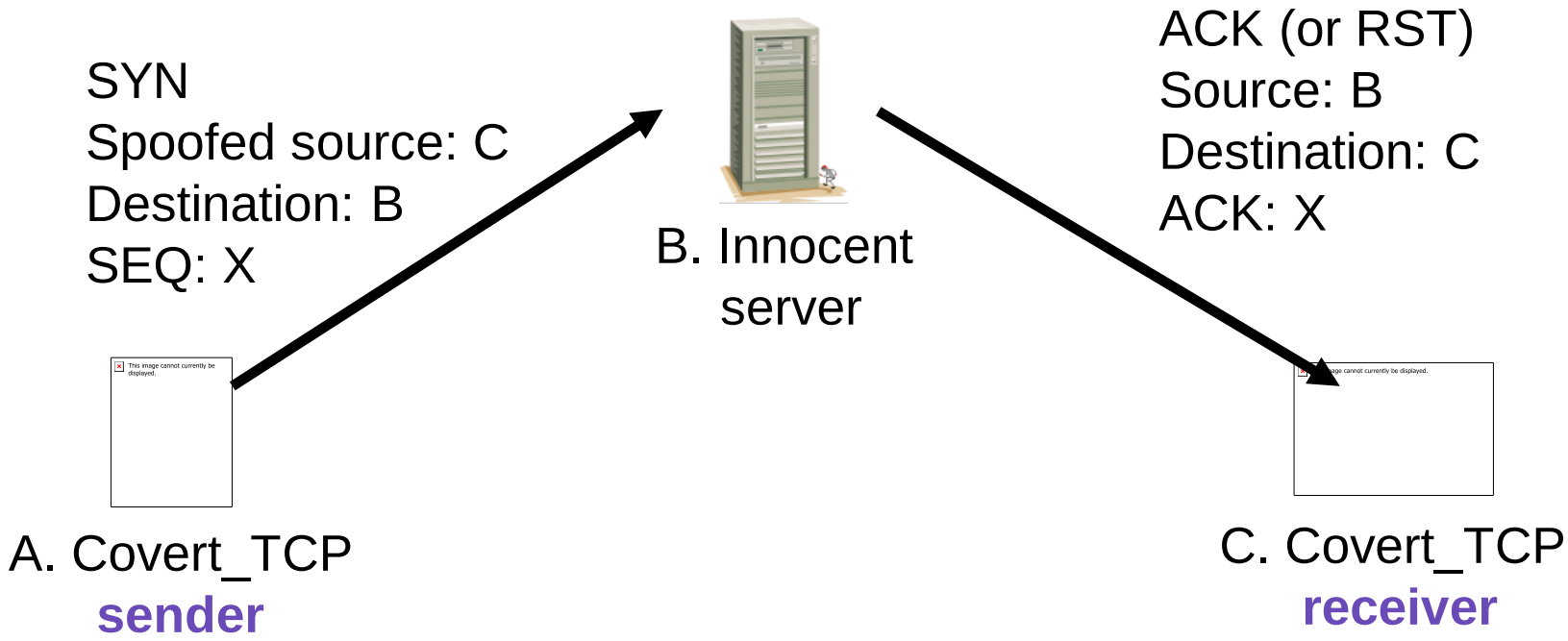
Real-World Covert Channel



- Use “reserved” field in TCP header to transmit ASCII data
- It can use any fileds like
  - Sequence number
  - ACK number
  - Ip packet ID field

Real-World Covert Channel

- Sender hides information in sequence number X and forges source address with address of intended recipient is sent to any server which is innocent.



EXAMPLE

Coverrt TCP transfers 1 ASCII character data per packet at a time

Packet one:

18:50:13.551117 nextime.getreal.com 7180> vlast.getreal.com.www: S

537657344:537657344(0) win 512 (ttl 64, id 18432)

Decoding… (ttl 64, id 18432/256) **gives ASCII 72 (H)**

Packet two:

18:50:13.551117 nextime.getreal.com 7180> vlast.getreal.com.www: S

537657344:537657344(0) win 512 (ttl 64, id 17664)

Decoding… (ttl 64, id 17664/256) **gives ASCII 69 (E)**

Packet three:

18:50:13.551117 nextime.getreal.com 7180> vlast.getreal.com.www: S

537657344:537657344(0) win 512 (ttl 64, id 19456)

Decoding… (ttl 64, id 19456/256) **gives ASCII 76 (L)**

EXAMPLE

Packet four:

18:50:13.551117 nextime.getreal.com 7180> vlast.getreal.com.www: S 537657344:537657344(0) win 512 (ttl 64, id 19456)

Decoding… (ttl 64, id 19456/256) **gives ASCII 76 (L)**

Packet five:

18:50:13.551117 nextime.getreal.com 7180> vlast.getreal.com.www: S 537657344:537657344(0) win 512 (ttl 64, id 19456)

Decoding… (ttl 64, id 19456/256) **gives ASCII 79(O)**

Packet six:

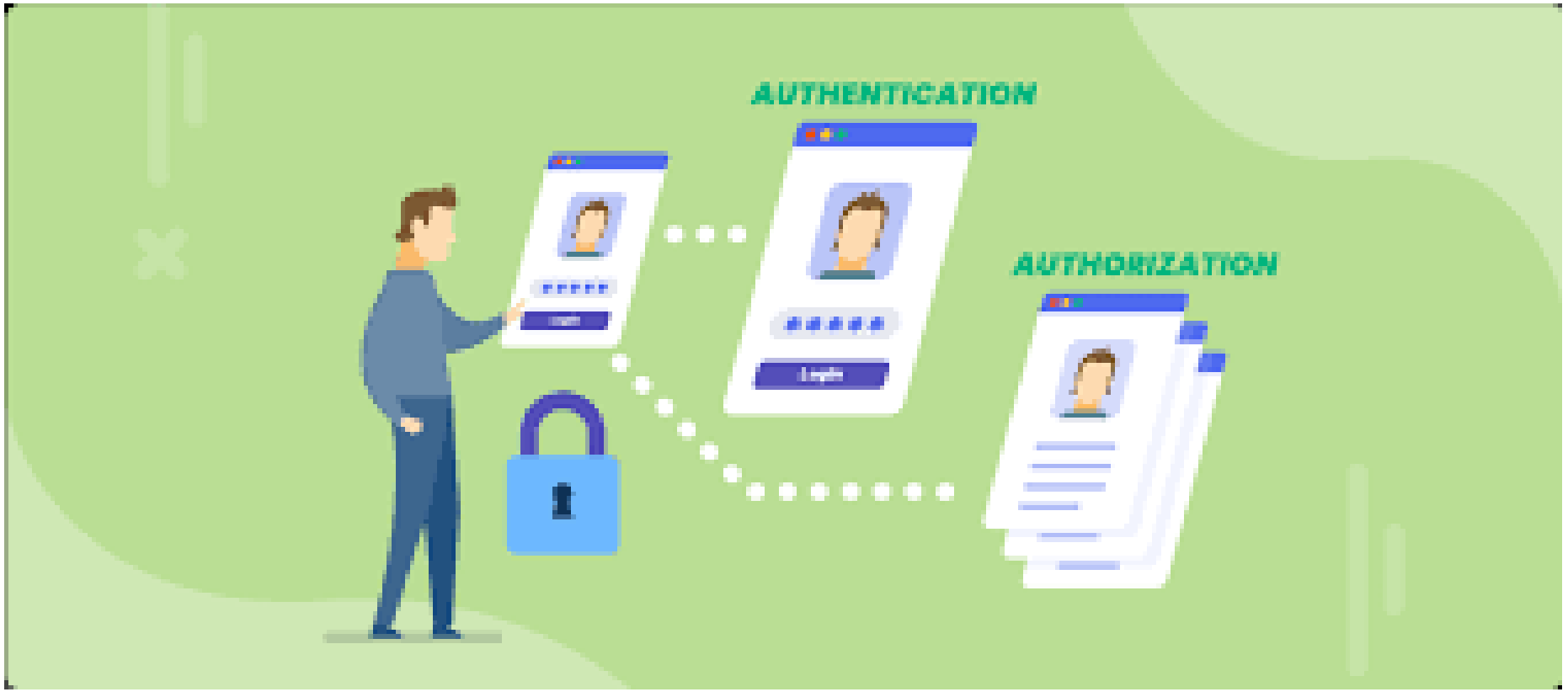
18:50:13.551117 nextime.getreal.com 7180> vlast.getreal.com.www: S 537657344:537657344(0) win 512 (ttl 64, id 2560)

Decoding… (ttl 64, id 2560/256) **gives ASCII 10 (carriage return)**

Covert Channel

- Potential covert channels are everywhere
- Virtually impossible to eliminate covert channels in any **useful** information system
  - DoD guidelines: **reduce covert channel capacity** to no more than 1 bit/second(discuss the example of 100 MB file size in top secret level here)
  - Implication? DoD has given up on *eliminating* covert channels

AUTHENTICATION



AUTHENTICATION PROTOCOLS

AUTHENTICATION

- Alice must prove her identity to Bob
  - Alice and Bob can be humans or computers
- May also require Bob to prove he's Bob (mutual authentication)
- A symmetric key need to be established which can be used as a session key(for confidentiality and integrity)
- May have other requirements, such as
  - Use only public keys
  - Use only symmetric keys
  - Use only a hash function
  - Anonymity, plausible deniability, etc., etc.

Authentication PROTOCOLS

- Authentication over a network is much more complex
  - Attacker can passively observe messages
  - Attacker can replay messages
  - Active attacks may be possible (insert, delete, change messages)
- To use an authentication protocol to allow the devices to get the account information from a central server.
- An authentication protocol is a type of cryptographic protocol specifically designed for transfer of authentication data between two entities(client and server)
- Eg: TACACS+, RADIUS, LDAP,KERBEROS
- We have to design an authentication protocol and do attack-based analysis

Simple Authentication

- Simple and may be OK for standalone system
- But insecure for networked system
  - Subject to a replay attack
  - Bob must know Alice's password

Simple Authentication

Alice

I'm Alice, My password is "frank"

Bob

- More efficient…
- But same problem as previous version

Authentication Attack

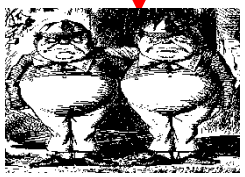
"I'm Alice"

Prove it

Alice

Bob

My password is "frank"



Trudy

Trudy

"I'm Alice"

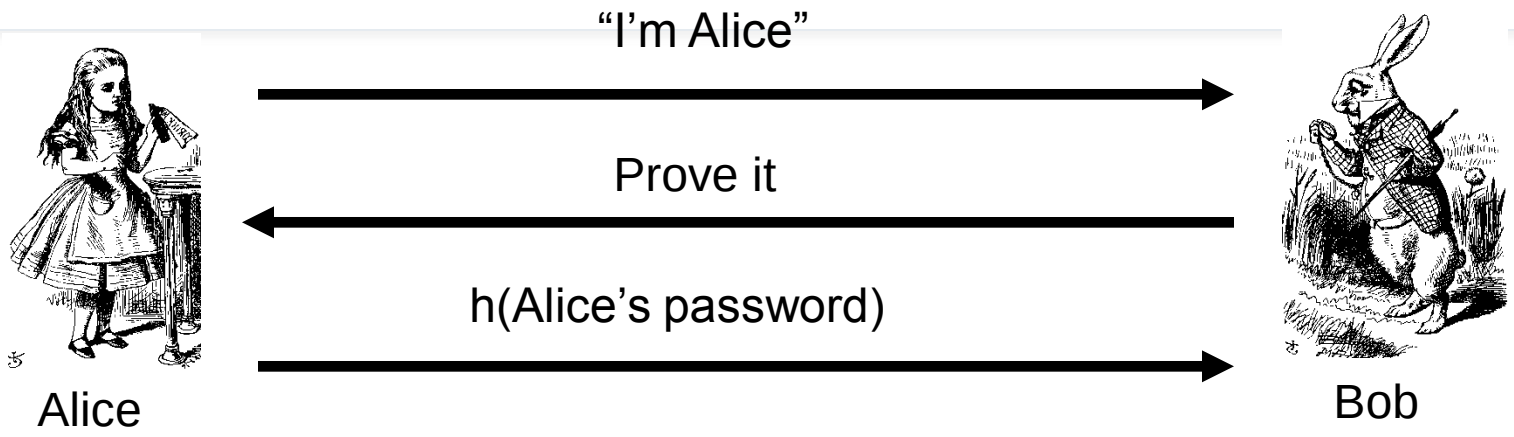
Prove it

My password is "frank"

Bob

REPLAY ATTACK

Better Authentication

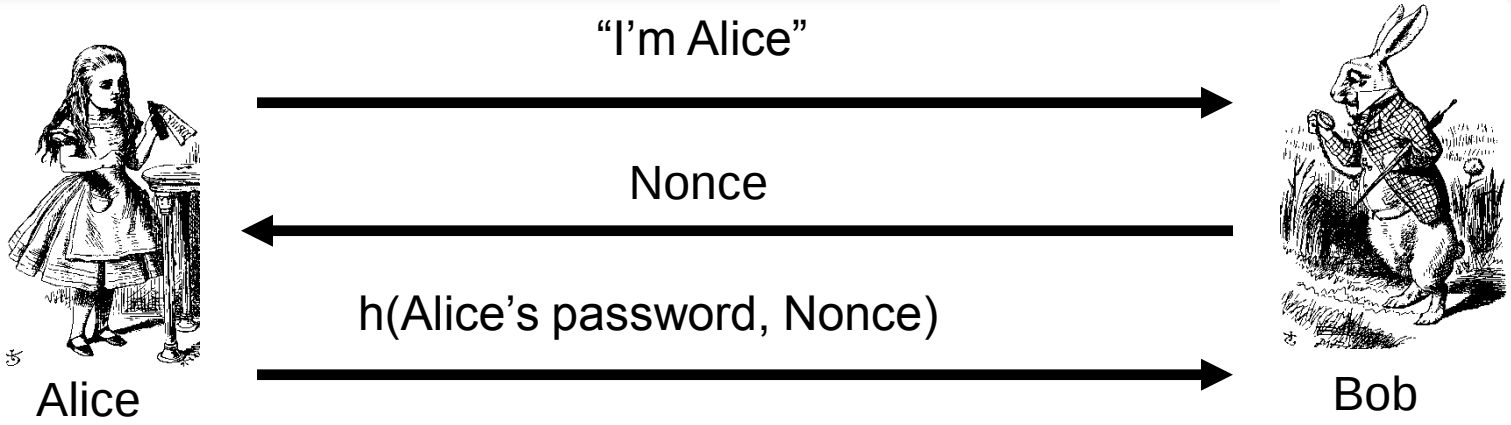


- Better since it hides Alice's password
  - From both Bob and attackers
- But still subject to replay

Challenge-Response

- To prevent replay, challenge-response used
- Suppose Bob wants to authenticate Alice
  - Challenge sent from Bob to Alice
  - Only Alice can provide the correct response which only Alice knows and Bob can verify
  - Challenge chosen is a nonce so that replay is not possible and challenge will be required in order to compute appropriate response so he can identify previous respobses.
- How to accomplish this?
  - Password is something only Alice should know…
  - For freshness, a “number used once” or **nonce**

Challenge-Response

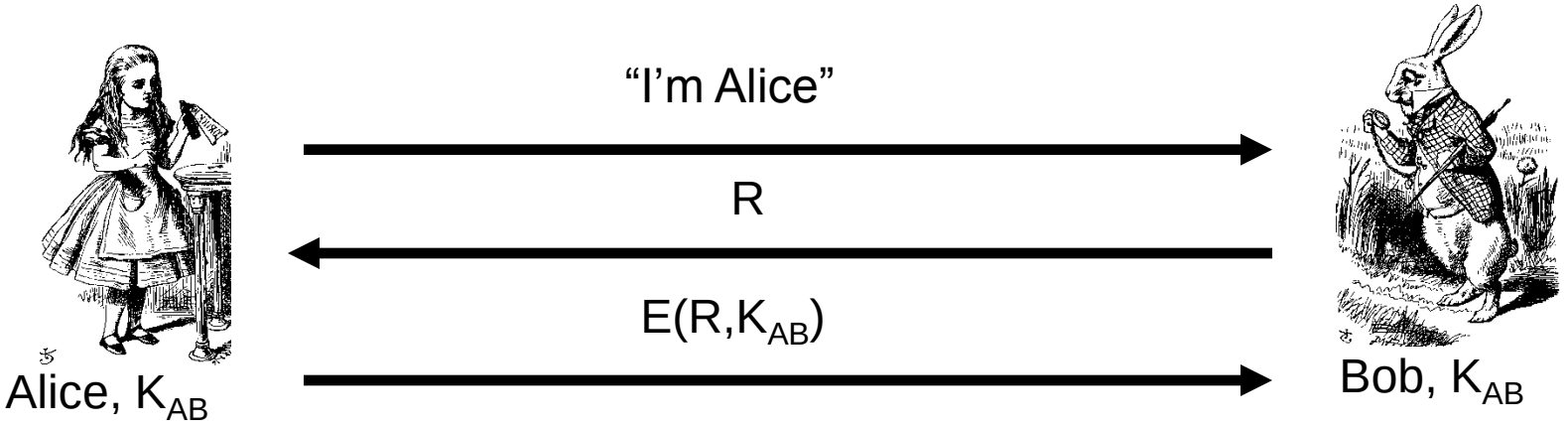


- Nonce is the **challenge**
- The hash is the **response**
- Nonce prevents replay, insures freshness
- Password is something Alice knows
- Note that Bob must know Alice's password

Symmetric Key Authentication & Notation

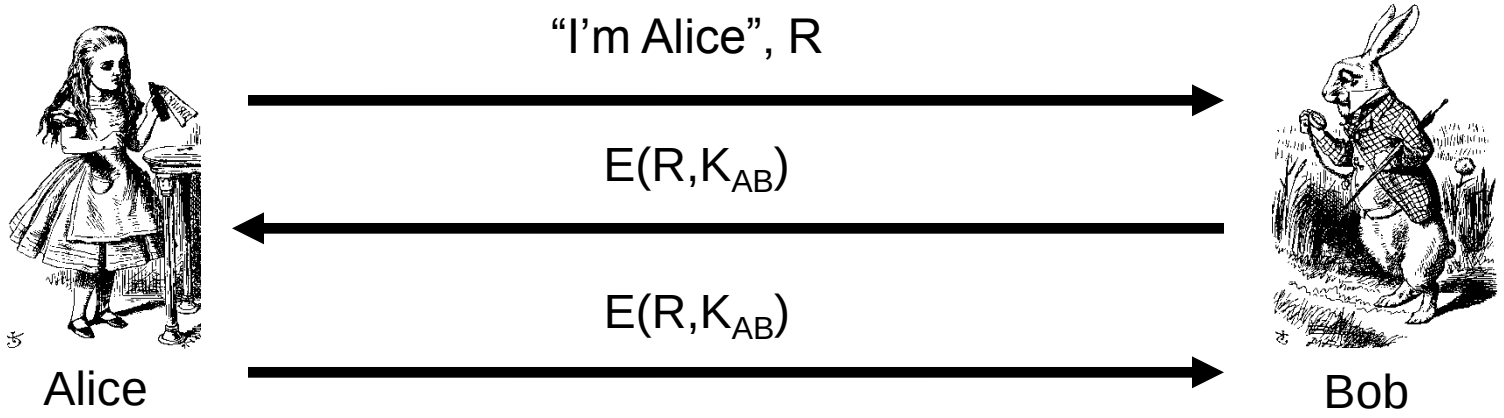
- Encrypt plaintext P with key K ie, **C = E(P,K)**
- Decrypt ciphertext C with key Kie, **P = D(C,K)**
- Here, we are concerned with attacks on **protocols**, not directly on the crypto. We assume that crypto algorithm is secure.
- Alice and Bob share symmetric key K_{AB} . The Key K_{AB} known only to Alice and Bob.
- Authenticate by proving knowledge of shared symmetric key.
- How to accomplish this?
  - Must not reveal key to Trudy.
  - Must not allow replay attack.

Authentication with Symmetric Key



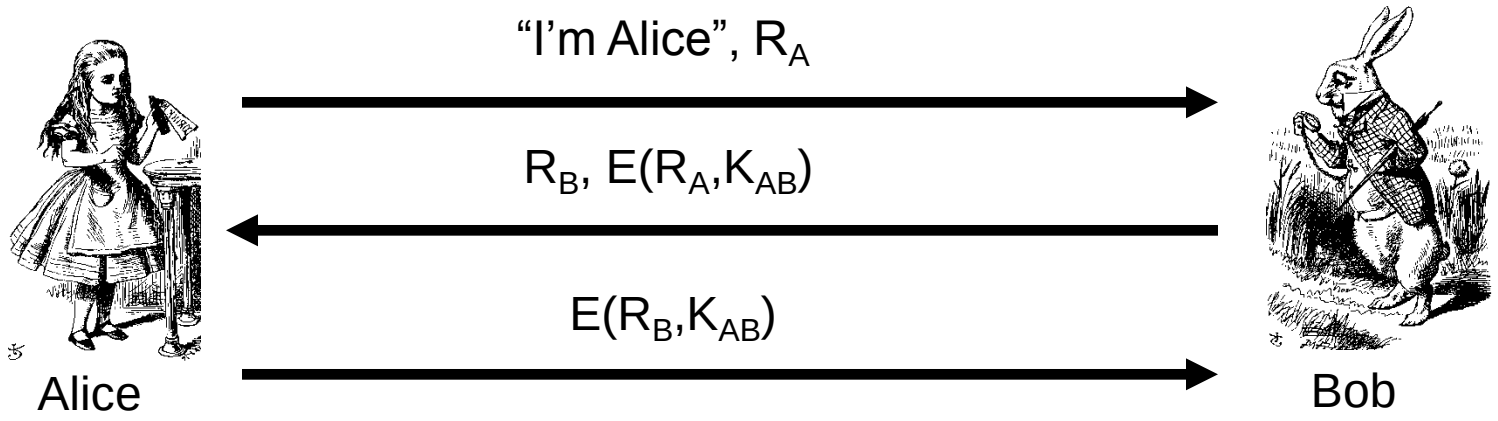
- Encrypt Nonce R with key KAB.
- Alice does not authenticate Bob.Secure method for Bob to authenticate Alice
- Can we achieve mutual authentication?

Mutual Authentication?



- What's wrong with this picture?
- "Alice" could be Trudy (or anybody else)!
- Since we have a secure one-way authentication protocol…
- The obvious thing to do is to use the protocol twice
  - Once for Bob to authenticate Alice
  - Once for Alice to authenticate Bob
- This has to work…

Mutual Authentication?



- Does this provides mutual authentication?

Mutual Authentication Attack

Trudy

1. “I’m Alice”, R_A

2. R_B, $E(R_A,K_{AB})$

5. $E(R_B,K_{AB})$

Bob

Trudy

3. “I’m Alice”, R_B

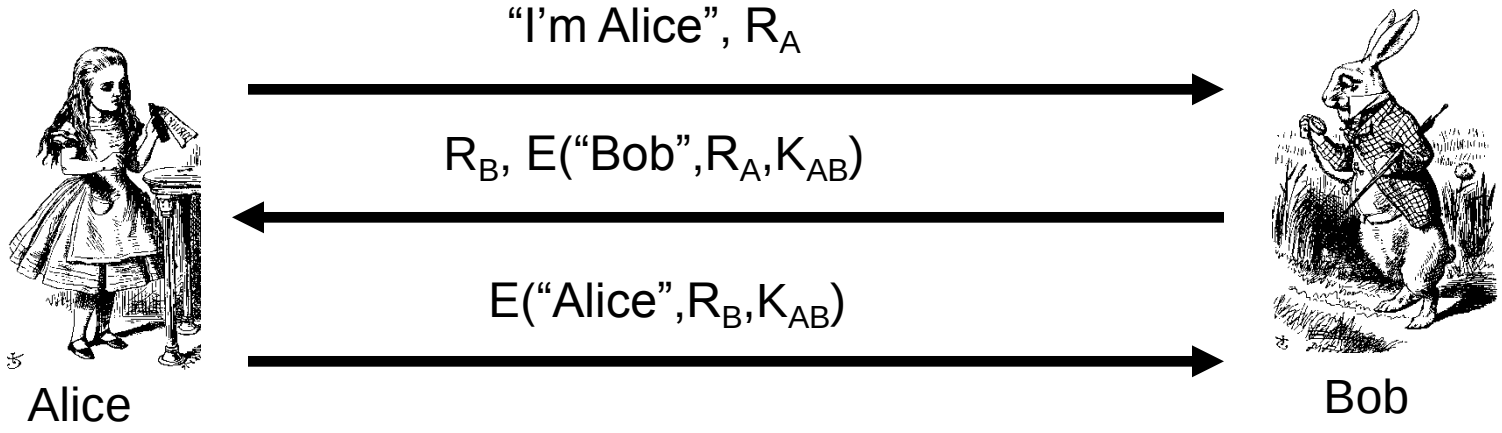
4. R_C, $E(R_B,K_{AB})$

Bob

Mutual Authentication

- Our one-way authentication protocol **not** secure for mutual authentication
- Protocols are subtle!
- The “obvious” thing may not be secure
- Also, if assumptions or environment changes, protocol may not work
  - This is a common source of security failure
  - For example, Internet protocols

Symmetric Key Mutual Authentication



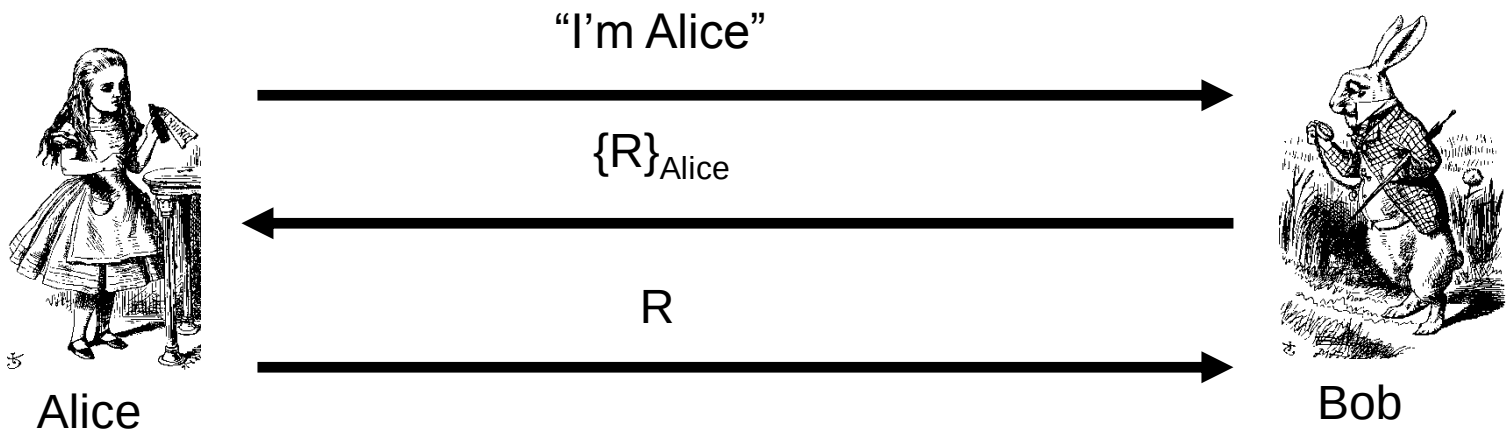
Public Key Notation

- Encrypted the user's identity together with the nonce. It prevent Trudy's MITM attack since she cannot use a response from Bob for the third message—Bob will realize that he encrypted it himself.

Authentication Using Public Keys

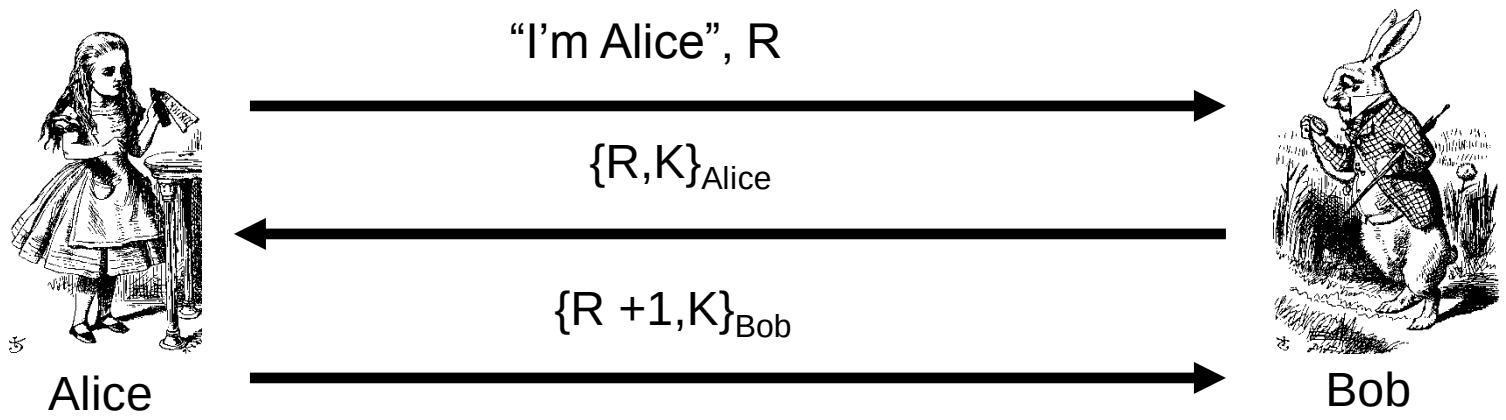
1. Encrypt M with Alice's public key: C= $\{M\}_{Alice}$ Also sign M with Alice's private key: [M]Alice

- Encryption and decryption are inverse operations
  - [{M}Alice]Alice = M
  - {[M]Alice }Alice = M
- Anybody can do public key operations
- Only Alice can use her private key (sign)



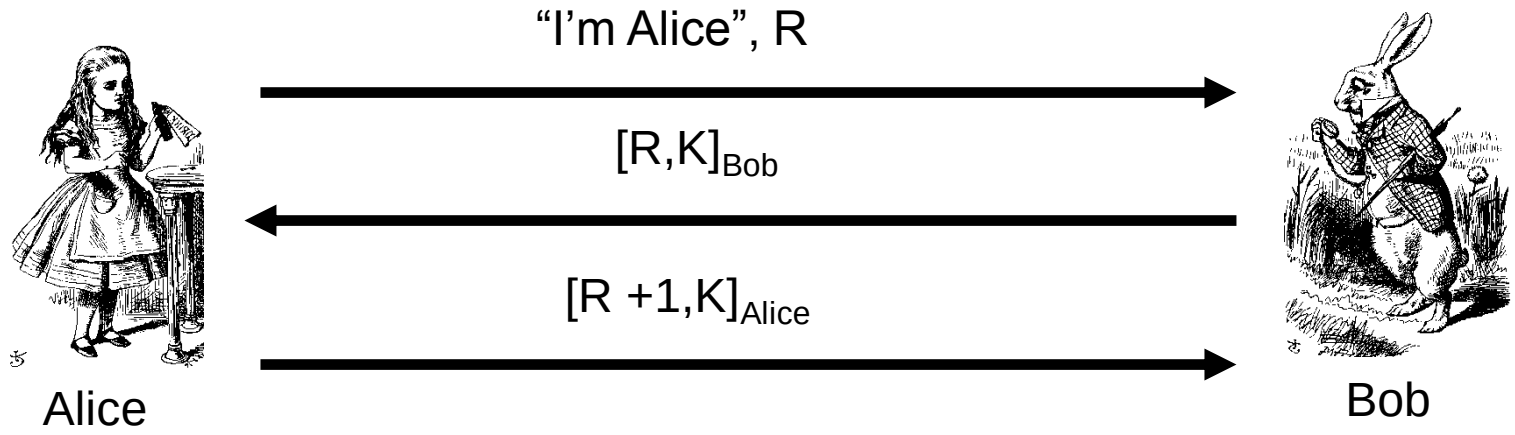
Session Key

- Usually, a session key is required
  - Symmetric key for a particular session
- Can we authenticate and establish a shared symmetric key?
  - Key can be used for confidentiality
  - Key can be used for integrity



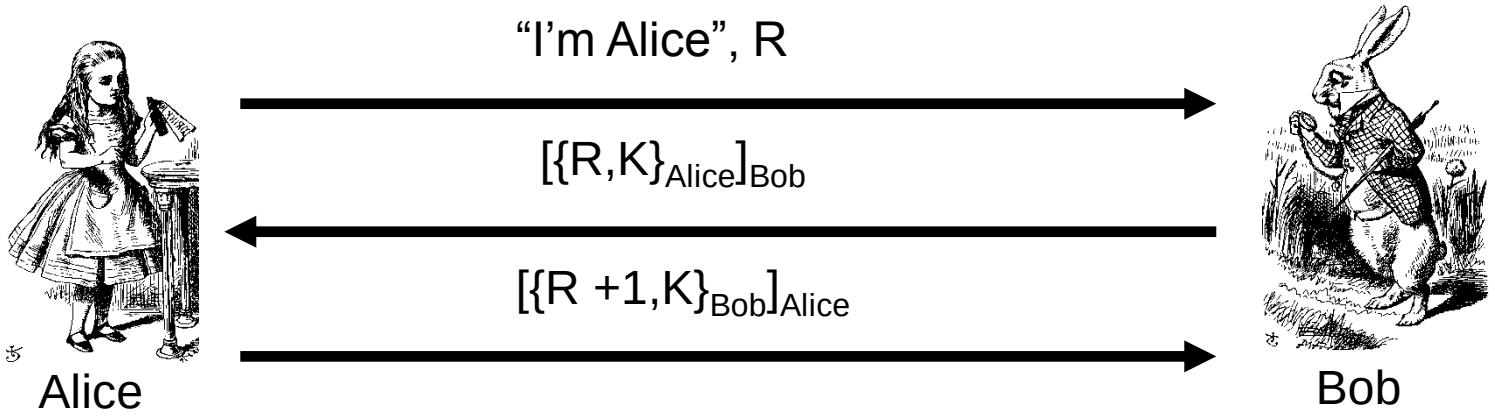
Public Key Authentication and Session Key

- Is this secure?
- Seems to be OK
- Mutual authentication and session key!



Public Key Authentication and Session Key

- s this secure?
- Seems to be OK
  - Anyone can see $\{R,K\}_{Alice}$ and $\{R+1,K\}_{Bob}$

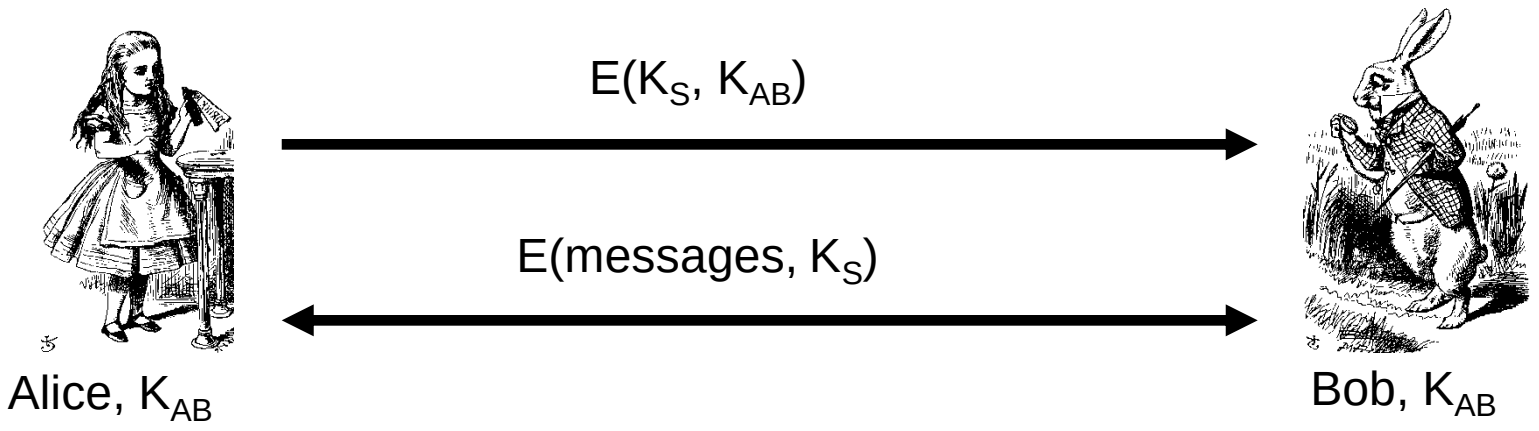


Perfect Forward Secrecy

- The concern…
  - Alice encrypts message with shared key K_{AB} and sends ciphertext to Bob
  - Trudy records ciphertext and later attacks Alice's (or Bob's) computer to find K_{AB}
  - Then Trudy decrypts recorded messages
- **Perfect forward secrecy (PFS):** Trudy cannot later decrypt recorded ciphertext
  - Even if Trudy gets key K_{AB} or other secret(s)
- Is PFS possible?
- Suppose Alice and Bob share key K_{AB}
- For perfect forward secrecy, Alice and Bob cannot use K_{AB} to encrypt
- Instead they must use a **session key** K_S and forget it after it's used
- Problem: How can Alice and Bob agree on session key K_S and ensure PFS?

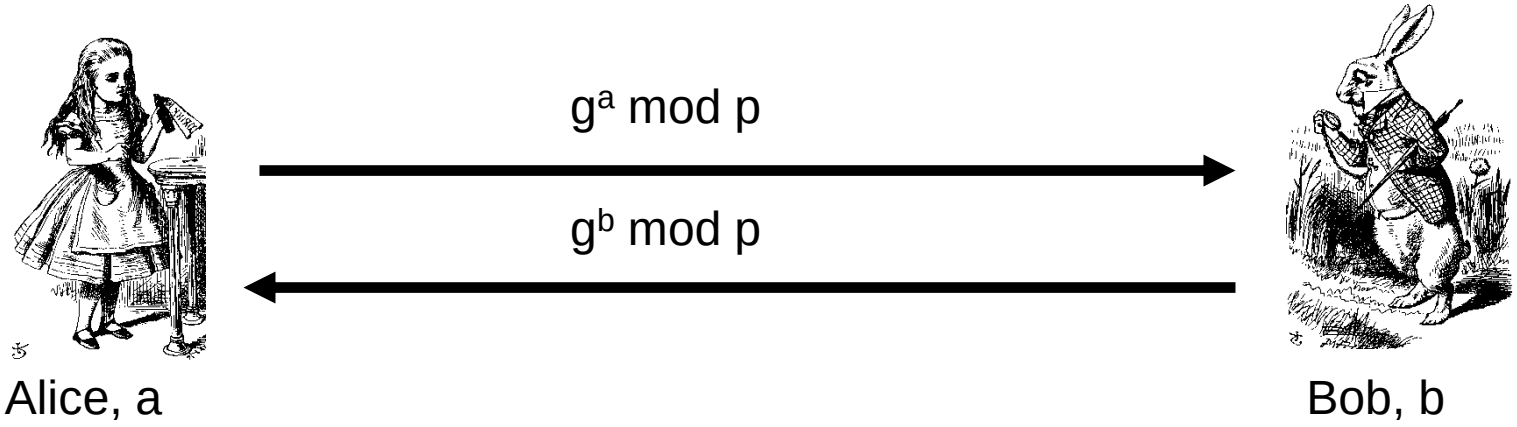
Naïve Session Key Protocol

- Trudy could also record $E(K_S,K_{AB})$
- If Trudy gets K_{AB}, she gets K_S



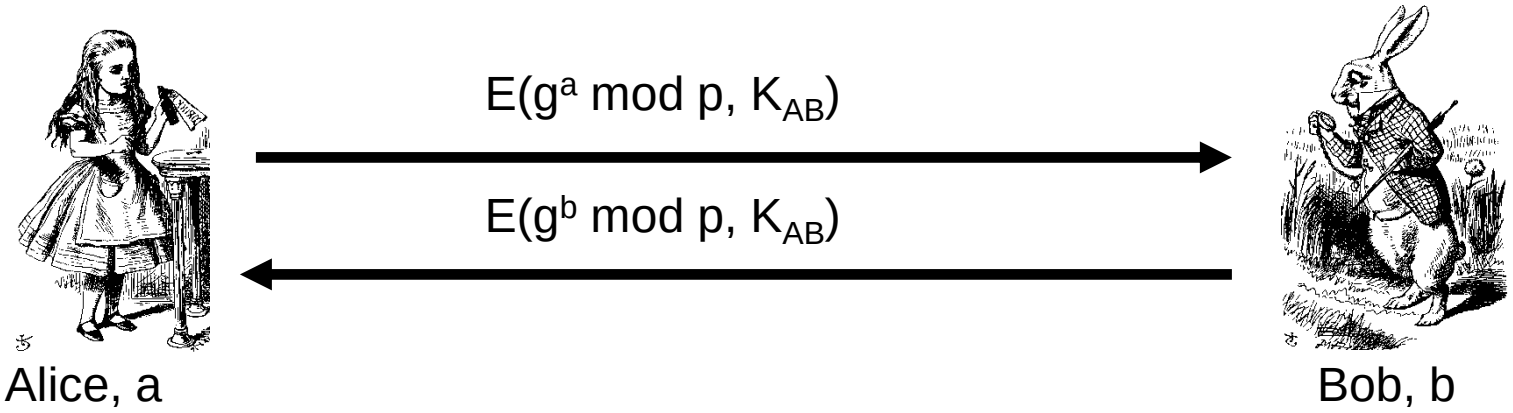
Perfect Forward Secrecy

- Can use **Diffie-Hellman** for PFS
- Recall Diffie-Hellman: public g and p
- But Diffie-Hellman is subject to MiM
- How to get PFS and prevent MiM?



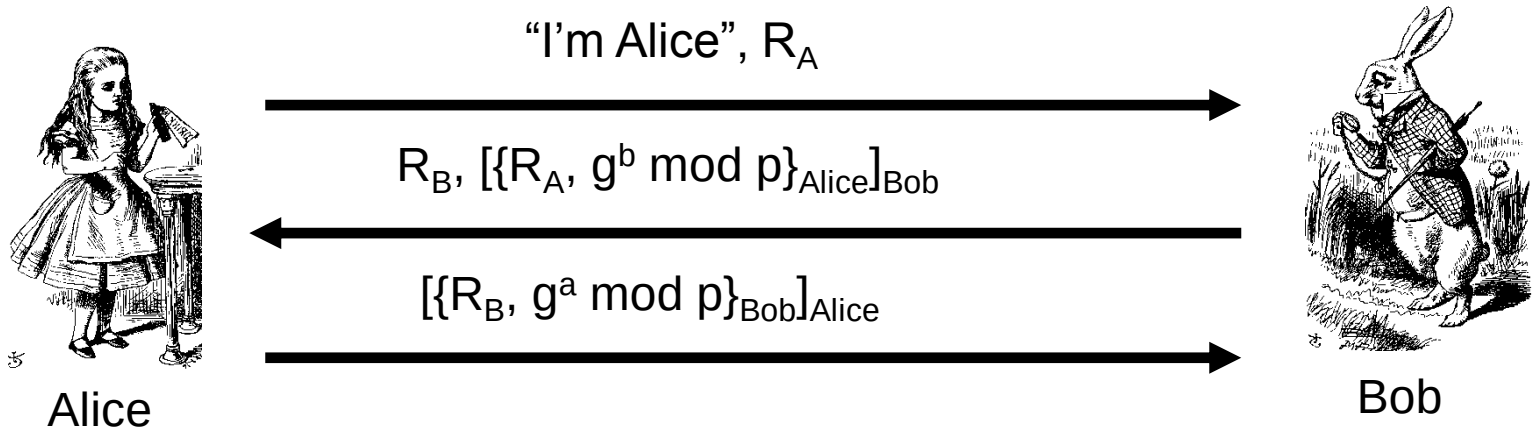
Perfect Forward Secrecy

- Session key $K_S = g^{ab} \bmod p$
- Alice forgets a, Bob forgets b
- **Ephemeral Diffie-Hellman**
- Not even Alice and Bob can later recover K_S
- Other ways to do PFS?



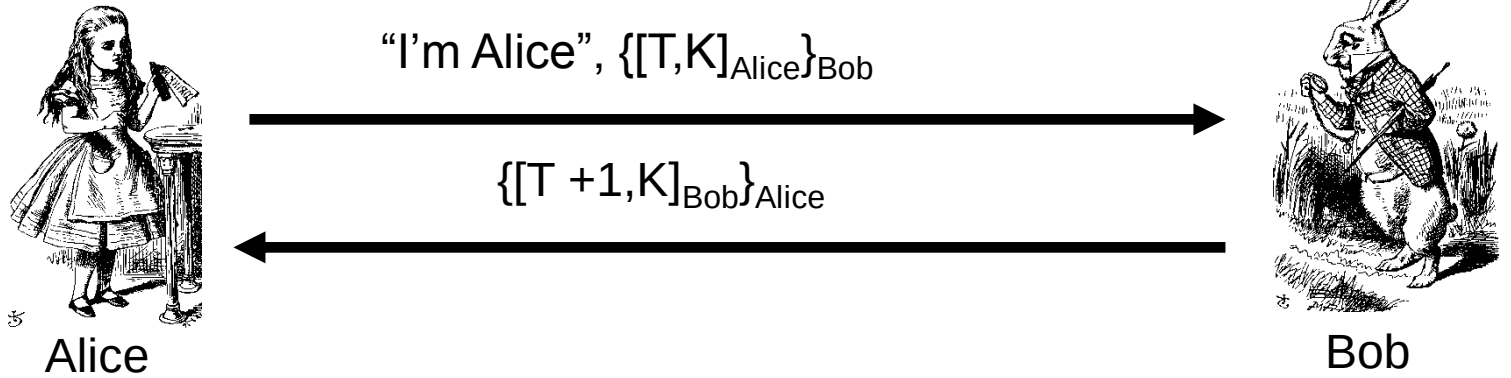
Mutual Authentication, Session Key and PFS

- Session key is $K = g^{ab} \bmod p$
- Alice forgets a and Bob forgets b
- If Trudy later gets Bob's and Alice's secrets, she cannot recover session key K



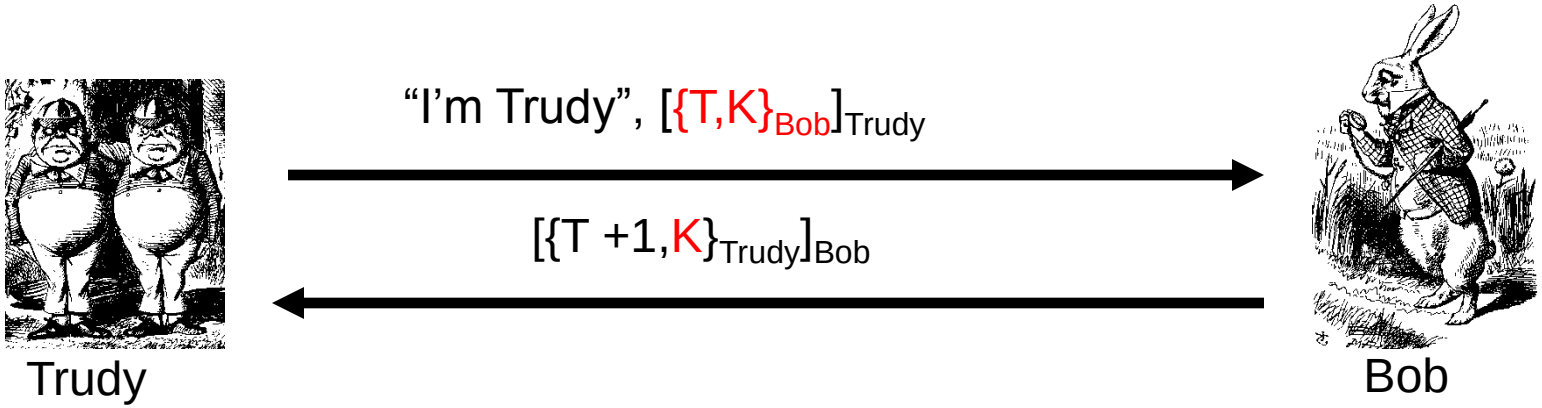
Public Key Authentication with Timestamp T

- A timestamp T is the current time
- Timestamps used in many security protocols (Kerberos, for example)
- Timestamps reduce number of messages
  - Like a nonce that both sides know in advance
- But, use of timestamps implies that time is a security-critical parameter
- Clocks never exactly the same, so must allow for **clock skew** ⇒ risk of replay
- How much clock skew is enough?



- Is this secure?
- Trudy can use Alice's public key to find
  - {T,K}Bob and then…

Public Key Authentication with Timestamp T

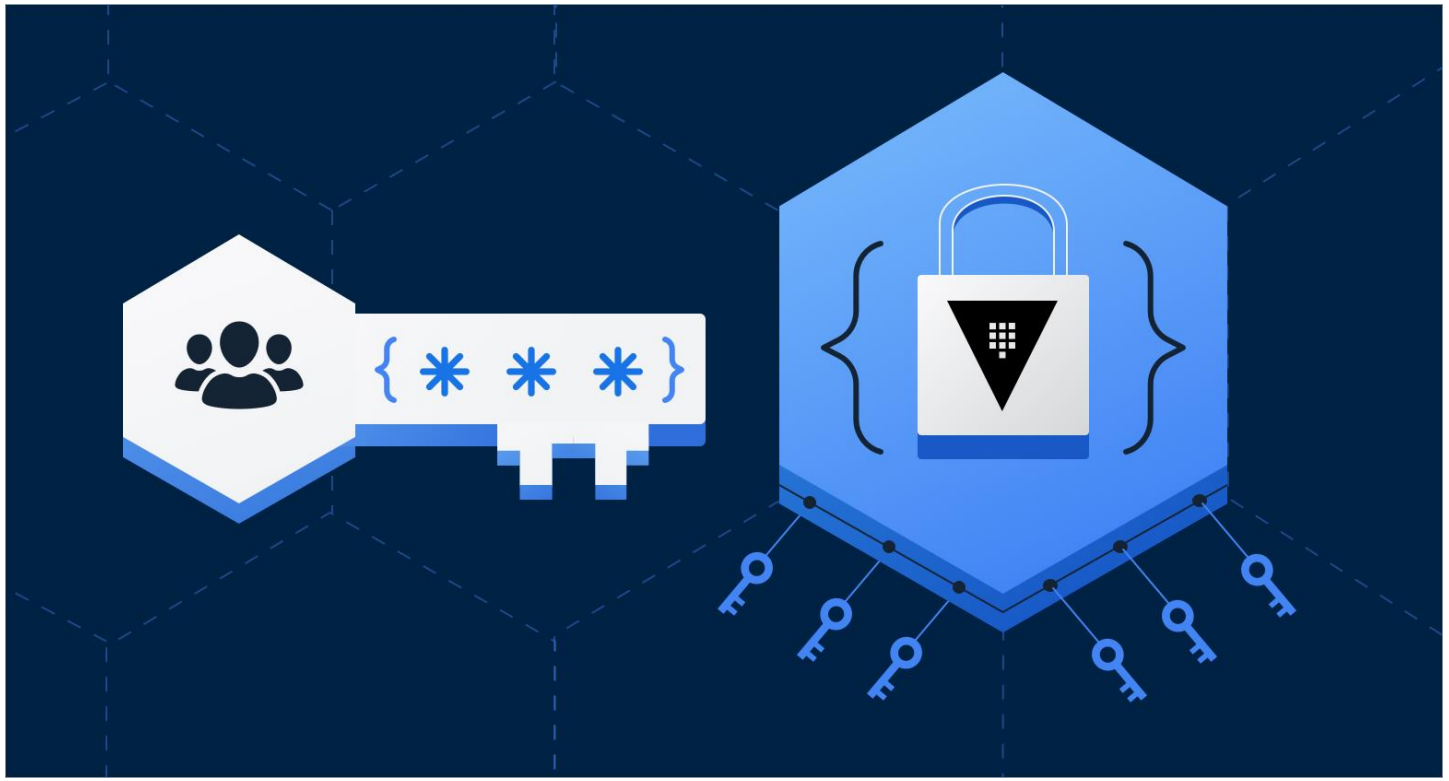


- Trudy obtains Alice-Bob session key K
- Note: Trudy must act within clock skew

Public Key Authentication

- Sign and encrypt with nonce…
  - Secure
- Encrypt and sign with nonce…
  - Secure
- Sign and encrypt with timestamp…
  - Secure
- Encrypt and sign with timestamp…
  - Insecure
- Protocols can be subtle!

Key Management

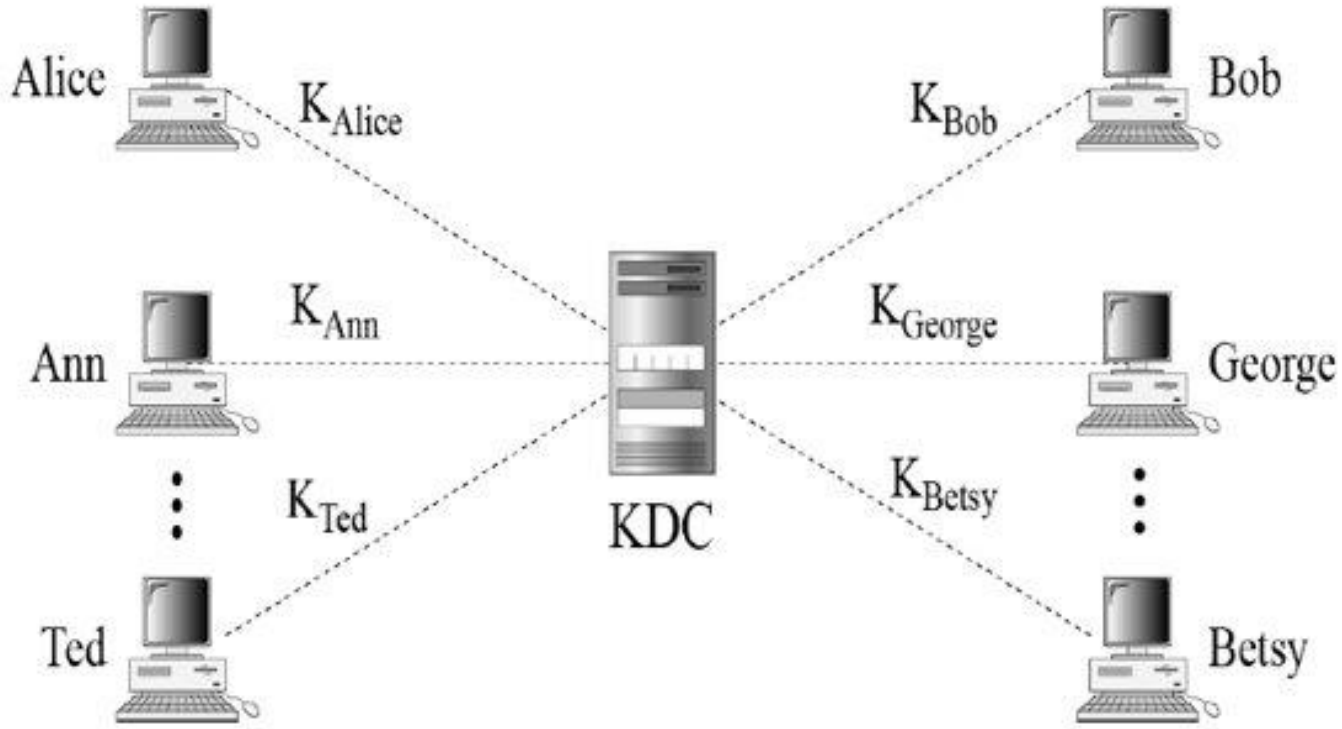


Key management

- Secret keys in symmetric key cryptography and public keys in Asymmetric key cryptography need to be distributed and maintained.
- **Symmetric Key Distribution:** *Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages. Symmetric-key cryptography, however, needs a shared secret key between two parties. The distribution of keys is another problem.*
- **Distribution of symmetric keys done by:**
  - **Using a TTP***(Kerberos)*
  - **Without using TTP***(Diffie Hellman)*
  - **Using KDC***(Needham Schroder)*
  - **Using CA's**
  - **PKI**

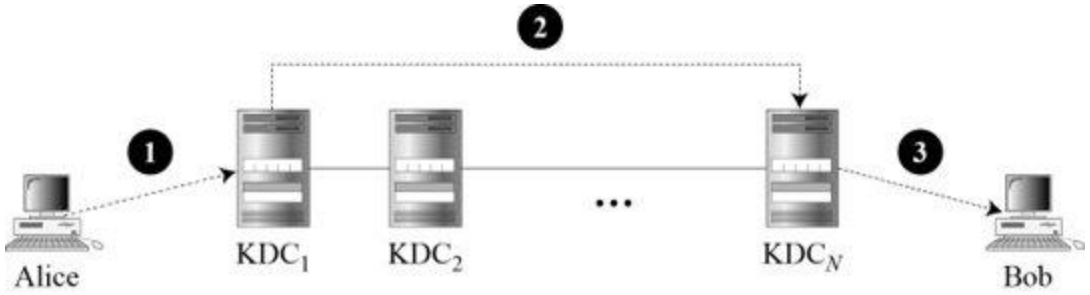
Key-Distribution Center: KDC

- A ***KDC creates a secret key for each member***. *This* ***secret key(K_{AS}, K_{BS})*** *can be used only between the member and the KDC, not between two members.*
- A **session symmetric key(K_{AB})** is established with KDC with Bob's(server) agreement between two parties is **used only once.**

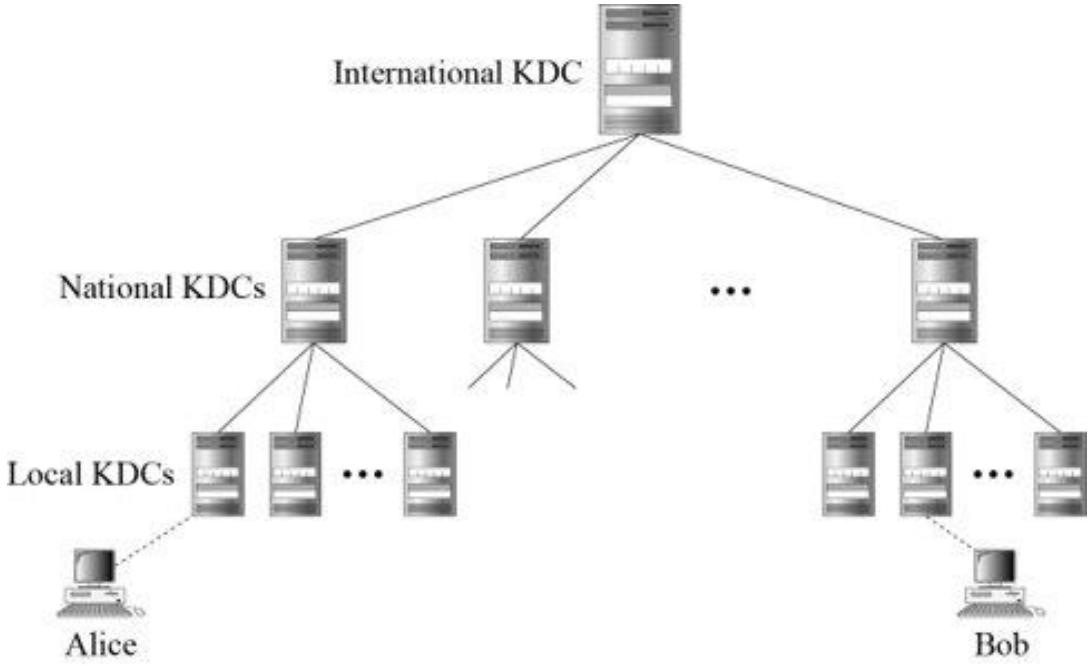


KDC Types

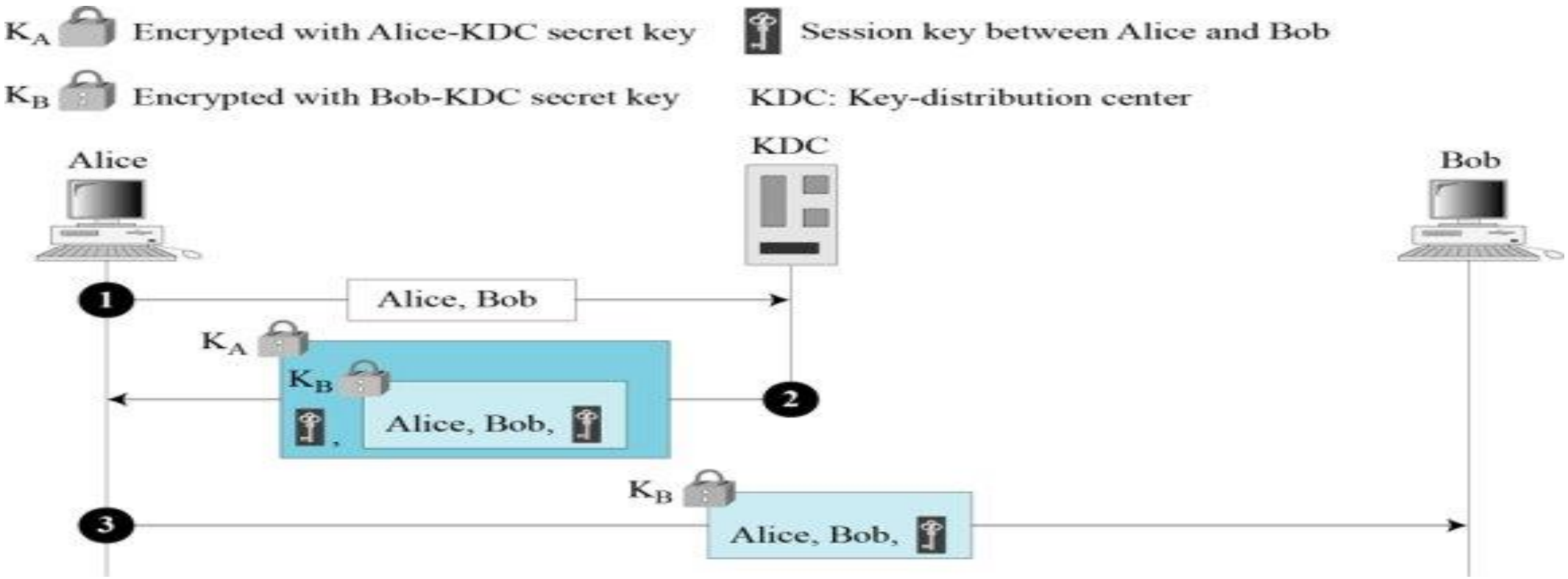
Flat Multiple KDCs



Hierarchical Multiple KDCs



A Simple protocol using KDC



Needham-Schroeder Protocol

- **Needham–Schroeder protocol** is one of the two key transport protocols intended for use over an insecure network, proposed by Roger Needham and Michael Schroeder
- This protocol is intended to provide **mutual authentication** between two parties communicating on a network, but in its proposed form is insecure.
- N-S is protocol **aims to establish a session key** between two parties on a network, typically to protect further communication.
- There are **two types** of Needham-Schroeder protocol.
  - **Needham-Schroeder protocol with symmetric key**
  - **Needham-Schroeder protocol with asymmetric key**
- The *Needham–Schroeder Symmetric Key Protocol*, based on a **symmetric encryption algorithm** forms the basis for the **Kerberos protocol.**
- Needham-Schroeder protocol allows to **prove the identity of the end users** communicating, and also **prevents a middle man from eavesdropping.**

Protocol Identities

- Alice (A) initiates the communication to Bob (B). K_{AB} is a server trusted by both parties. In the communication:
  - **A** and **B** are identities of Alice and Bob respectively
  - **K_{AS}** is a symmetric key known only to A and S
  - **K_{BS}** is a symmetric key known only to B and S
  - It is assumed that A and B already have secure symmetric communication with S using keys K_{AS} and K_{BS}
  - **N_A** and **N_B** are nonces generated by A and B respectively.
  - **K_{AB}** is a symmetric key, which will be the session key generated for the session between A and B

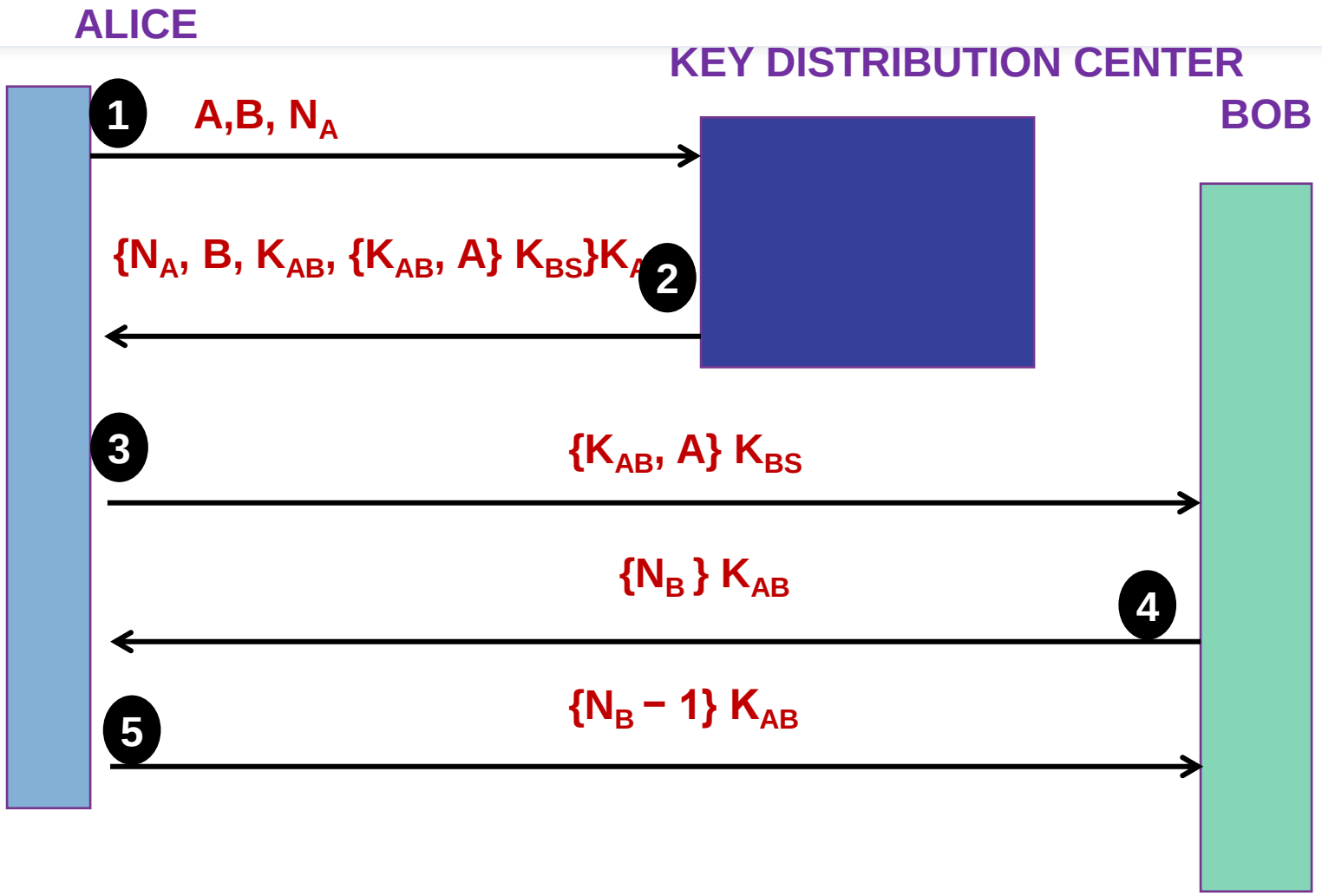
Nonces

- N-S protocol uses **nonces** (short for **“numbers used once”),** randomly generated values included in messages. This is used in encryption protocols to **prevent replay attack**.
  - For example if somebody captures a packet during the communication between me and a shopping website, he can resend the packet without decrypting it, and the server can accept the packet and do operations on it. To prevent this, nonce(the random value generated) is added to the data, so as the server can check if that nonce is valid, or expired.
- If a nonce is generated and sent by A in one step and returned by B in a later step, A knows that B’s message is fresh and not a replay from an earlier exchange.
- Note that a nonce is not a timestamp. The only assumption is that it has not been used in any earlier interchange, with high probability

NS PROTOCOL(with symmetric keys) STEPS

1. Alice sends a message to KDC that includes her identity, Bob's identity and her Nonce N_A.
   - $A \rightarrow KDC : A,B, N_A$
2. KDC generates K_{AB} the session key. The session key K_{AB} is appended with Alice's identity and encrypted with Bob's secret key K_{BS} which is known as **ticket to Bob**. The ticket along with session key K_{AB} , Bob's identity and Alice's nonce NA is encrypted with Alice's secrete key K_{AS} and sends to Alice. The nonce assures Alice that the message is fresh and that the server is replying to that particular message and the inclusion of Bob's name tells Alice who she is to share this key with.
   - $S \rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\} K_{BS}\}K_{AS}$
3. Alice forwards the Bob's ticket.
   - $A \rightarrow B : \{K_{AB}, A\} K_{BS}$
4. Bob can decrypt the ticket with the key K_{BS} he shares with KDC, thus authenticating the data. Bob sends Alice a nonce encrypted under K_{AB} to show or confirm that he has the symmetric key or session key provided by the middle man server K_{AB}.
   - $B \rightarrow A : \{N_B\} K_{AB}$
5. Alice performs a simple operation on the nonce provided by Bob, re-encrypts it and sends it back to Bob just to verify that she is still alive and that she holds the key.
   - $A \rightarrow B : \{N_B - 1\} K_{AB}$

Needham-Schroeder Protocol



KERBEROS

Motivation for KERBEROS

- Users wish to access services on servers.
- Three threats exist:
  1. User pretend to be another user.
  2. User alter the network address of a workstation.
  3. User eavesdrop on exchanges and use a replay attack.
- **Solution:** Authentication using public keys
  - N users⇒ N key pairs
- Authentication using symmetric keys
  - N users requires (on the order of) N^2 keys
- Symmetric key case **does not scale**
- Kerberos based on symmetric keys but only requires N keys for N users
  - Security depends on TTP (trusted third parties) & No PKI is needed

KERBEROS

- In Greek mythology, Kerberos is 3-headed dog with a snake for a tail that guards entrance to Hades
- Kerberos is an authentication protocol based on symmetric key crypto that provides mutual authentication
  - Originated at **MIT**
  - Based on **Needham and Schroeder protocol**
  - Relies on a **Trusted Third Party (TTP)**
  - uses **UDP port 88** by default.
  - Several versions of the protocol exist. Two versions in use: 4 & 5- **Version 4 makes use of DES**
  - Used by **Red Hat 7.2** and **Windows Server 2003** or higher
  - Kerberos has also become a standard for websites and **Single-Sign-On implementations** across platforms.
  - Kerberos Consortium maintains Kerberos as an open-source project.

KERBEROS Working Parties

- Kereberos has separated user verification from issuing of tickets. The main working parties are:
  - **Client Workstation(CWS):** Alice is the client workstation
  - **Authentication Server (AS):** Each user(client as well as server) register with AS and are granted with a user identity and a password. AS has a database with user identities and passwords. AS verifies user, issues a secret key to be used between Alice and TGS.
  - **Ticket-Granting Server (TGS):** TGS issues a ticket for real server Bob. It also provides session key KAB between Alice and Bob.
  - **Real Server:** The real server (Bob) provides services for the user (Alice).
- AS and TGS together constitutes KDC
- The security of the protocol relies heavily on participants maintaining loosely synchronized time and on short-lived assertions of authenticity called Kerberos tickets.
- In Kerberos KDC acts as the TTP. TTP is trusted, so it must not be compromised

KERBEROS Login Process

1. Alice enters her username which is received by the authentication server(AS) in plain text.
2. **AS** creates a package for user Alice which contains **a randomly generated session Keys(K_{A-TGSS} =SA(Mark stamp)), user id of Alice and expiration time** and the package is encrypted using **symmetric key(K_{AS-TGS} =KKDC) that AS & TGS shares**. The output is known as **Ticket Granting Ticket(TGT),** which can be opened only by TGS. **AS encrypts TGT and K_{A-TGS} using symmetric key (K_{A-AS}) derived from password of Alice.** TGT is issued when user logs in and acts as user credentials which is later on used to obtain tickets to access network resources. TGT ensures statelessness of Kerberos.

   Symmetric Key K_{A-AS} = h(Alice's password)

   TGT=(K_{A-TGS},"Alice",timestamp)K_{AS-TGS}

Alice does not know K_{A-AS} , but can be created from Alice's password and can extract session key(K_{A-TGSS}) and **TGT**. Client machine immediately destroys the password that Alice has entered from its memory to prevent its stealing by the attacker.

((TGT)K_{AS-TGS}, K_{A-TGS})K_{A-AS}

KERBEROS Login Process

3. Alice has successfully logged in and want to access Bob, the email server. To communicate with Bob Alice needs a ticket and sends TGT to TGS. Client machine creates a message for TGS, which includes following**: ID of server(BOB), Expiration time(curent time stamp) encrypted with session key** K_{A_TGS} which is known as **authenticator, TGT** which already it received. Timestamp **encrypted with session key** K_{A_TGS}/authenticator prevents replay attack by Eve.

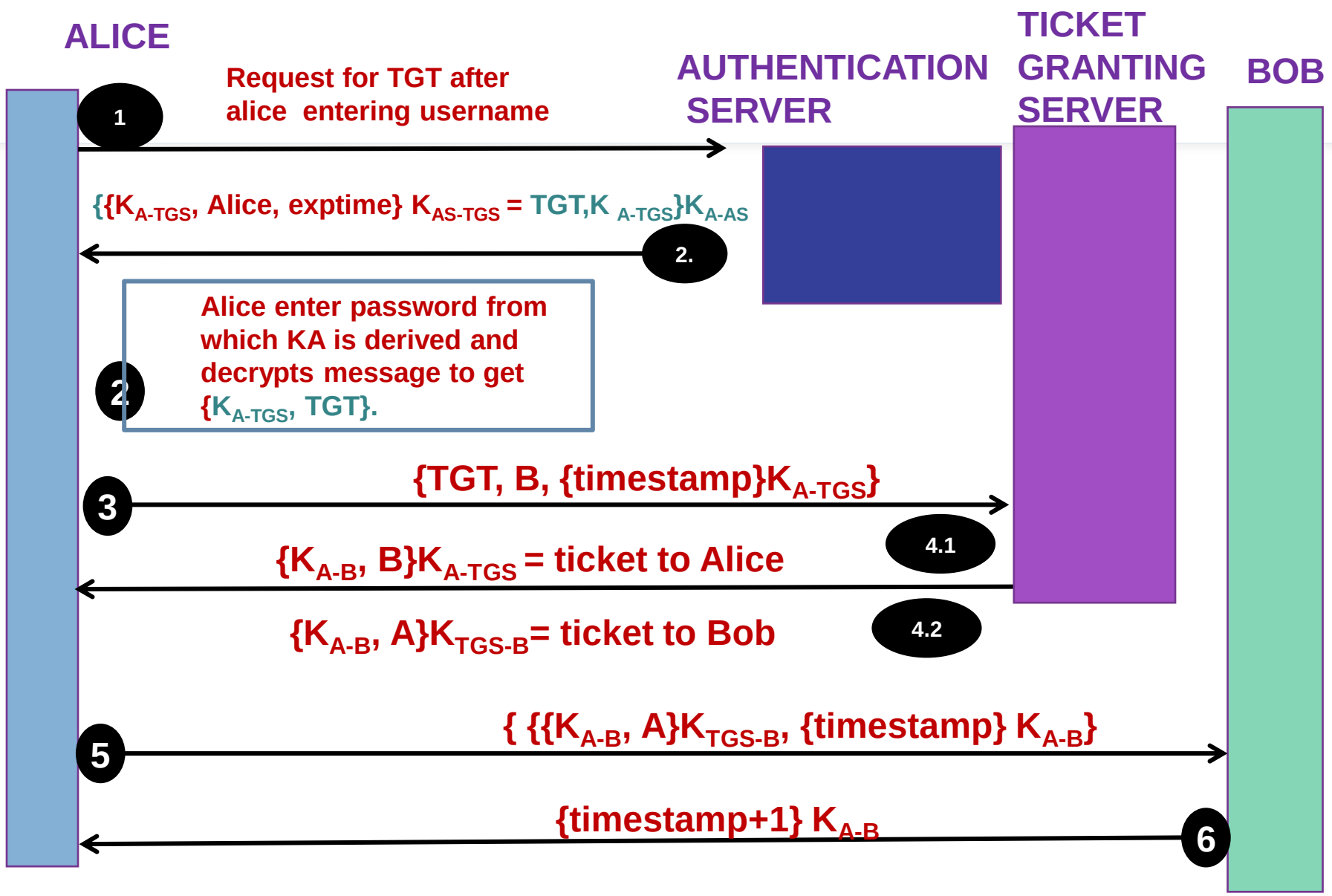
4. **TGS obtains** $K_{A\text{-}TGS}$ **from TGT**; since only TGS knows how to decrypt TGT using $K_{AS\text{-}TGS}$. TGS verifies TGT and timestamp. TGS creates a session key $K_{A\text{-}B}$ for Alice to have secure communication with BOB. TGS sends 2 tickets to Alice.

   1. $K_{A\text{-}B}$ combined with **Bob's id and encrypted with key** $K_{A\text{-}TGS}$ It is known as **ticket to Alice.**
   2. $K_{A\text{-}B}$ combined with **Alice's id and encrypted with BOB's secret Key(**$K_{TGS\text{-}B}$**).** It is known as **ticket to Bob.**

5. Alice forwards BOB's ticket $\{K_{A\text{-}B}, A\}K_{TGS\text{-}B}$ which she had received from TGS. To guard replay attacks Alice adds time stamp encrypted with $K_{A\text{-}B}$.

6. Bob generates KA-B and decrypts time stamp using KA-B. For acknowledgement BOB adds 1 to time stamp and encrypts the result with KA-B and send it to Alice.

KERBEROS PROTOCOL



Kerberos Security

ADVANTAGES

- Statelessness

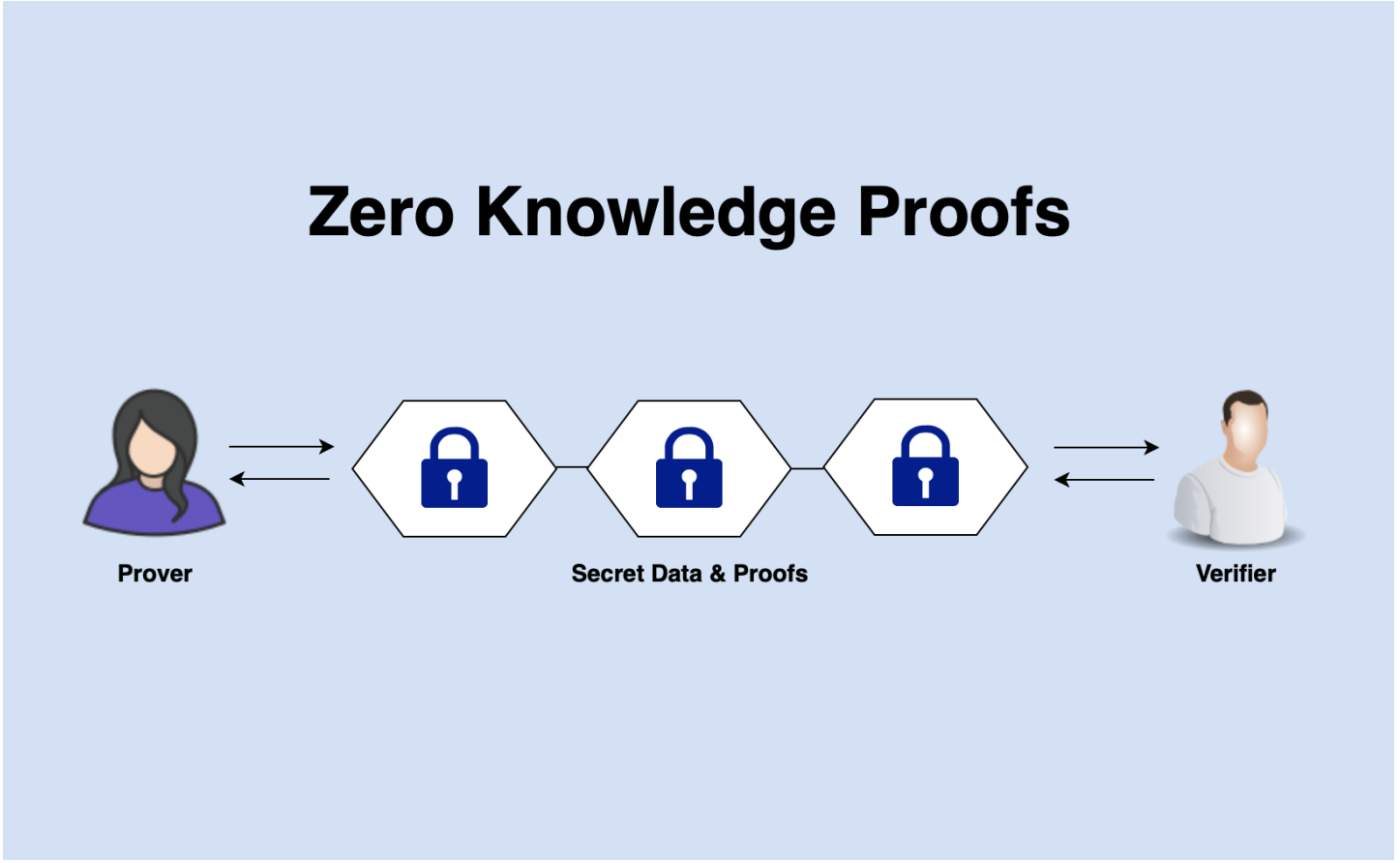
K_{AB} is sent from Alice to Bob. If it was done by KDC to Bob then Bob has to save key and maintain state for further communication.
Also $K_{AS\text{-}TGS}$ is generated by KDC(TTP) without intervention of Alice a nd Bob. So KDC remains stateless. Eliminates DOS attack.

- Anonymity of Alice: TGS doesn't need to know who is making request to issue session key and to decrypt the message as it verifies key $K_{AS\text{-}TGS}$
- Replay Attack preventin
- Less symmetric keys
- Authentication assured

DISADVANTAGES

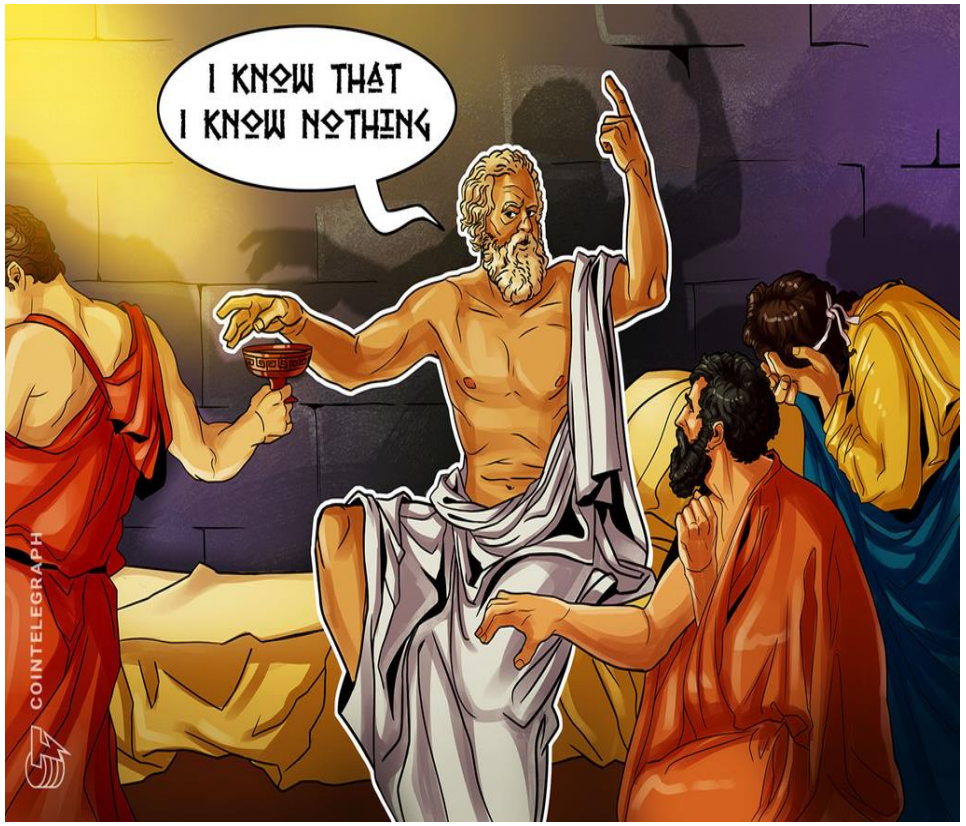
- Entire security depends on security of KDC/TTP. If its compromised entire security is compromised
- Kerberos uses timestamp for mutual authentication and replay attack prevention, the drawback is time becomes a critical security parameter. All clocks need to be synchronized and clock skew must be tolerated(typically 5 minutes)
- The administration protocol is not standardized and differs between server implementations.
- Single point of failure: It requires continuous availability of a central server. When the Kerberos server is down, no one can log in. This can be mitigated by using multiple Kerberos servers and fallback authentication mechanisms.

ZERO KNOWLEDGE PROOFS



ZERO KNOWLEDGE PROOFS

- An **authentication scheme** developed by **Fiege, Fiat** and **Shamir.** Usually run in several rounds and have high costs.
- In ZKP Alice wants to prove Bob that she knows a secret without revealing any information about secret to anyone, even to Bob.Bob must be able to verify that Alice knows secret, even though he gains no information on secret.
- *A zero-knowledge proof is a digital protocol that allows for data to be shared between two parties without the use of a password or any other information associated with the transaction.*



Relevance of ZKP in Authentication

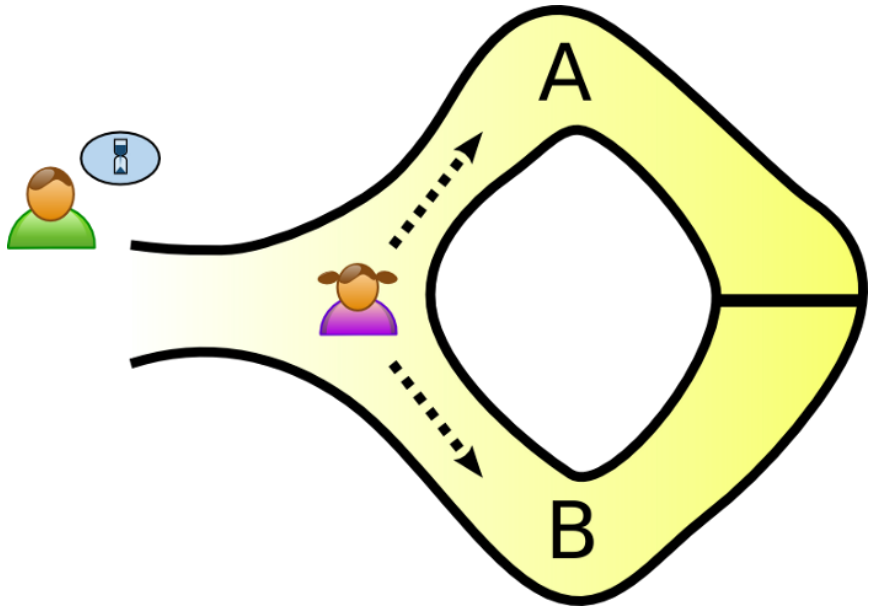
- In password authentication claimant needs to send her password to verifier. The isssues associated are
  - Evesdropping by Trudy
  - Dishonest verifier can reveal secret.
  - Use thepassowrd to impersonate user
- In zeroknowledge authentication, claimant doesnt reveal anything that endanger confidentiality of secret.
- From the interaction between claimant and verifier,the verifier knows claimant have secret or not.

Usecases

- *Zero-knowledge proofs offer a lot of benefits to blockchain systems help in making crypto transaction's extremely secure.* cryptocurrency transactions can be done without disclosing any data related to it — such as where the transactions originated from, where it went or how much money was transferred. **Eg:** Zcash, Quorum(a native blockchain ecosystem by JP Morgan Chase). ZoKrates using Solidity.
- Governments can also use ZKPs to determine the nuclear capabilities of various militaries without having to spy on or inspect their inventories. Eg: Defense Advanced Research Projects Agency, or DARPA, released a statement in which it claimed that it was working on a new project called SIEVE — i.e., Securing Information for Encrypted Verification and Evaluation — that makes use of ZKPs to determine the origin of highly secure data without the U.S. government having to reveal the way in which it was acquired.
- Digital identification mechanisms, a secure alternative to the fog of birth certificate photocopies and smartphone photos of passports. Those ID schemes could also allow people to prove that they meet a minimum age requirement without sharing their date of birth, or that they have a valid driver's license without handing over their number.

Alibaba Cave

- There is a cave which has two entries inside called A and B. Both entries are connected inside the cave by a door that can open if correct secret spell words known("open "sesame"). Peggy says Victors that she knows the secret spell to open the door. Victor wants to know whether Peggy knows the secret word; but Peggy, being a very private person, does not want to reveal her knowledge and also does not like to reveal which path she entered. How Peggy supposed to prove that she knows the secret spell without telling Victor the spell and without telling from which entrance she entered.

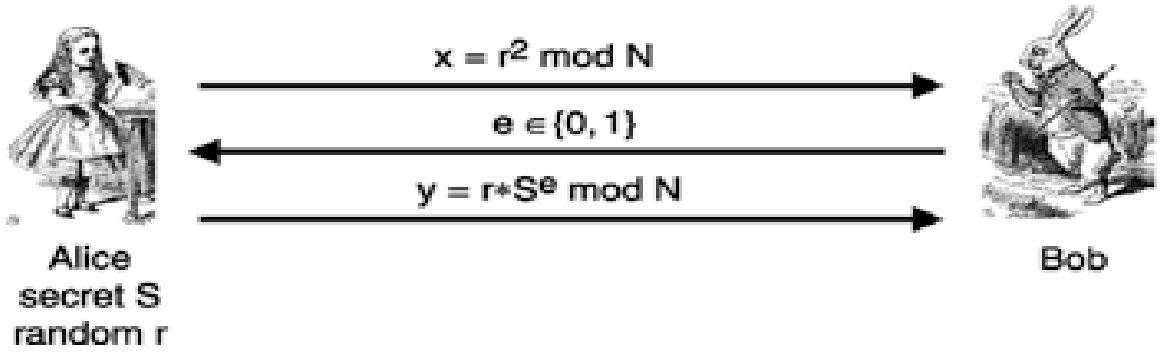


Alibaba Cave

- First, Victor waits outside the cave as Peggy goes in. Peggy takes either path A or B; Victor is not allowed to see which path she takes.
- Then Victor enters the cave and chooses an exit path for peggy. If she really does know the magic word, this is easy: she opens the door, if necessary, and returns along the desired path.
- suppose she did not know the word. Then, she would only be able to return by the named path if Victor were to give the name of the same path by which she had entered. Since Victor would choose A or B at random, she would have a 50% chance of guessing correctly. If they were to repeat this trick many times, say 20 times in a row, her chance of successfully anticipating all of Victor's requests would become vanishingly small (about one in a million).Probability that Peggy can trick Victor will be $(1/2^n)$

Achieving Cave Effect-Fiat Shamir Protocol

- To set up scheme and prove its secure on RSA, the trusted third paty or verifier or Bob choses **two very large prime numbers p** and **q** and calculate **N**, as **N=p*q**, where N is known to public and p and q are secret.
- Alice the claimant choses a secret **S** such that **S** is coprime to **N** and value of **S** is **1<=S<=N-1.** Alice computes **$v= S^2 \bmod N$**. Alice keeps **S** as her private key and **v** as her public key with TTP/Bob. Alice must convince Bob that she knows **S** without revealing any information on**S**. It can be done in 5 steps.

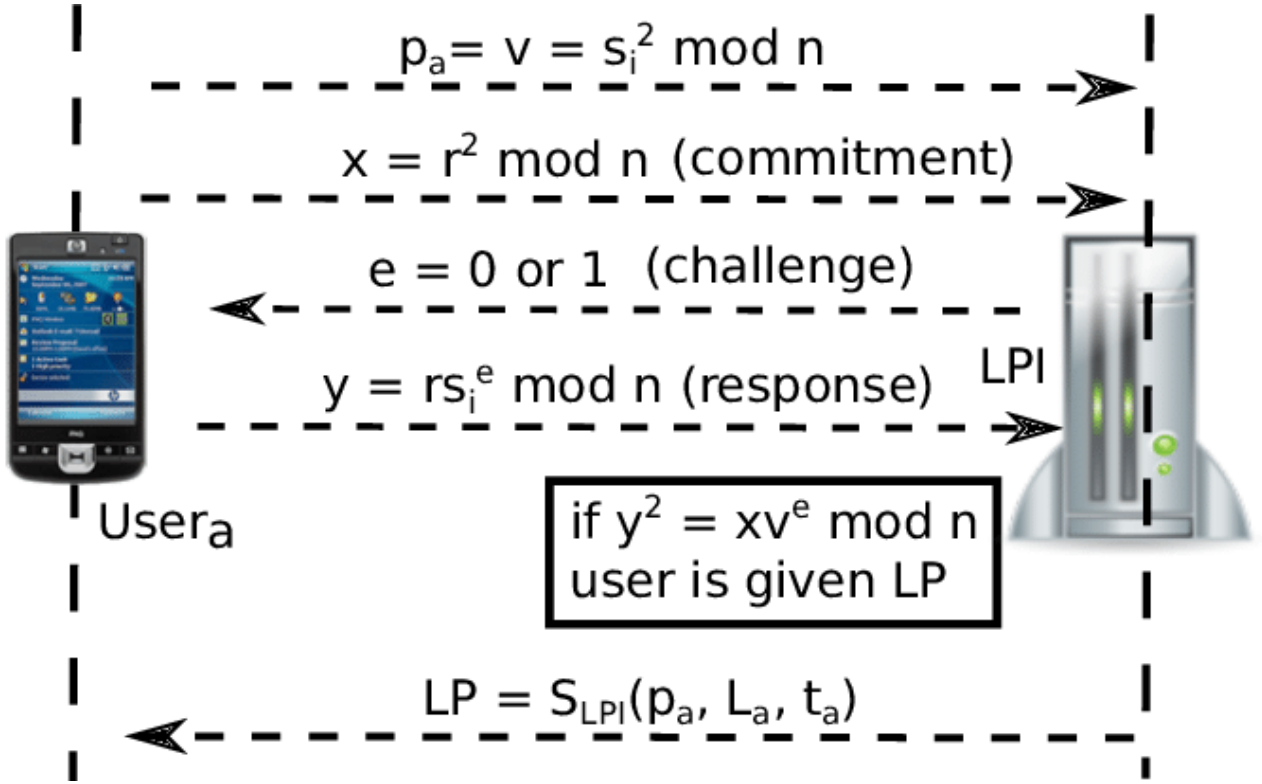


Fiat Shamir Protocol

1. Alice selcts a random number **r** between **0** and **N-1** and sends **$x = r^2 \bmod N$** to Bob. Here **x** is called as witness.
2. Alice sends **x** to Bob as witness.
3. Bob sends the challenge **e** to Alice**,** which is either **0** or **1**.
4. Alice calculates **$y = r * S^e \bmod N$** to Bob, where **r** is the random number selected by Alice in step 1 and**S** is her private key. Alice send **y** to Bob to show that she knows her private value **S.**
5. Bob calculates **y^2** and **$x * v^e \bmod N$**. If the 2 values are congruent, then Alice knows secret key **S**or she has calculated value of **y** in some other ways.

$y^2 = (r * S^e)^2 = r^2 s^{2e} = r^2(s^2)^e = xv^e \bmod N$

Fiat Shamir Protocol



Verification of Fiat Shamir Protocol

- **Verification** is repeated several times with value of **e** equal to 0 or 1.
- Alice must pass the test for every round even if she fails in one round she is not authenticated.
- If alice is dishonest 2 cases can happen:
  - Assume Bob has selected e as 1, then Alice must know secret S.
  - If **e=1** then Alice responds with **$y=r*S \bmod N$** in msg 4 and **$y^2= r^{2*}.s^2=x.v \bmod n$**. So **Alice must know secret S**
  - If **e=0** then Alice responds with **$y=r*S^0 \bmod N=r \bmod N$** in msg 4 and **$y^2= r^{2*}=x \bmod n$.** So **Alice must not know secret S**.
- If Trudy expects Bob to send e = 1, she can send $x = r^{2*}v^{-1}$ in msg 1 and $y = r$ in msg 3. If Bob chooses $e \in \{0,1\}$ at random, Trudy can only fool Bob with probability ½.

Fiat Shamir Facts

- Trudy can fool Bob with prob 1/2, but after n iterations, the probability that Trudy can fool Bob is only $(½)^n$
- Just like Bob's cave Bob's e ∈ {0,1} must be unpredictable
- Alice must use new r for each iteration or else
  - If e = 0, Alice sends r in message 4
  - If e = 1, Alice sends r*S in message 4
  - Anyone can find S given **both** r and r*S
- Security Benefits
- It allows authentication with anonymity. Both Alice and Bob knows public Key V but there is nothing in V that

Fiat-Shamir Zero Knowledge?

- **Is Fiat-shamir protocol really "zero knowledge"? Can Bob Learn anything about Alice's secret S?**
- Zero knowledge means that Bob learns **nothing** about the secret
- V= S^2 Mod N is public. Bob sees **r^2mod N** in message 1 and assuming **e=1** Bob sees **r * S mod N** in message 4. If Bob can find r from **r^2Mod N** then he can find **S** . Here security relies on finding a modulus square root which is computationally infeasible.
- **ZKP in the Real World? ZKP is not just fun and games for mathematicians!**
- If Public key certificates are used to identify users then No anonymity .
- ZKP is supported in Microsoft's Next Generation Secure Computing Base (NGSCB) used to authenticate software "without revealing machine identifying data"

Thank you

anooja@somaiya.edu