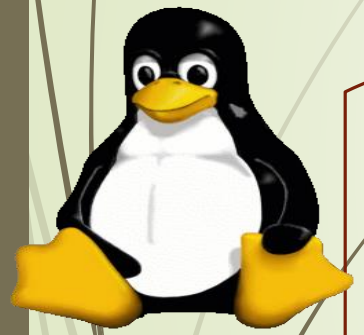
Digital Signatures and Authentication Protocols

Lecture Outline

1. Digital signatures
2. Key Distribution and Key Agreement

Introduction

- ≈ Suppose Bob wants to communicate with Alice over a channel in which Eve can intercept and transmit messages.
- ≈ This far, we have already considered the problem of message security - how Bob can encrypt a message for Alice such that Eve cannot decipher it in polynomial time with non-negligible probability.

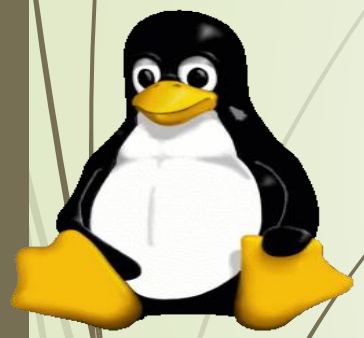


Now we consider a different problem:

- If Bob receives a message purportedly from Alice, how can he be certain it isn't a forgery by Eve?
  - ≈ This is particularly true if Bob has published a public key, and thus anyone can send him encrypted messages.

Solution: Have Alice "sign" the message in some way that Bob will recognize, and that Eve will be unable to forge.

Introduction



- There are a number of properties we would ideally like for such a signature:

1. Alice can efficiently sign any message, for some reasonable limit on the message size.
2. Given any document D that Alice has not signed, nobody can efficiently forge Alice's signature on D.
3. Given a document D and a signature, anyone (not just Bob!) can efficiently tell whether the signature is valid for D.

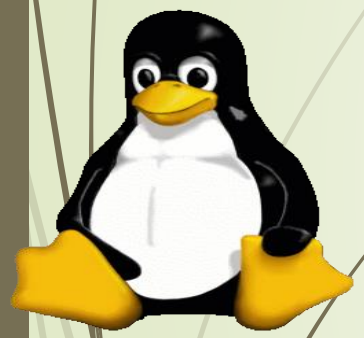
- We introduce digital signature schemes as a way of accomplishing this.

Digital Signatures: Used to provide

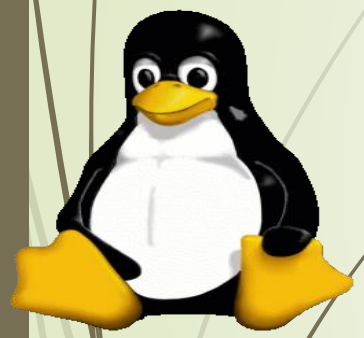
1. Data integrity

- The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.

- A digital signature provides message integrity



Digital Signatures: Used to provide



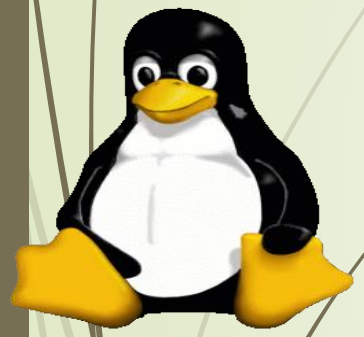
2. Message authentication

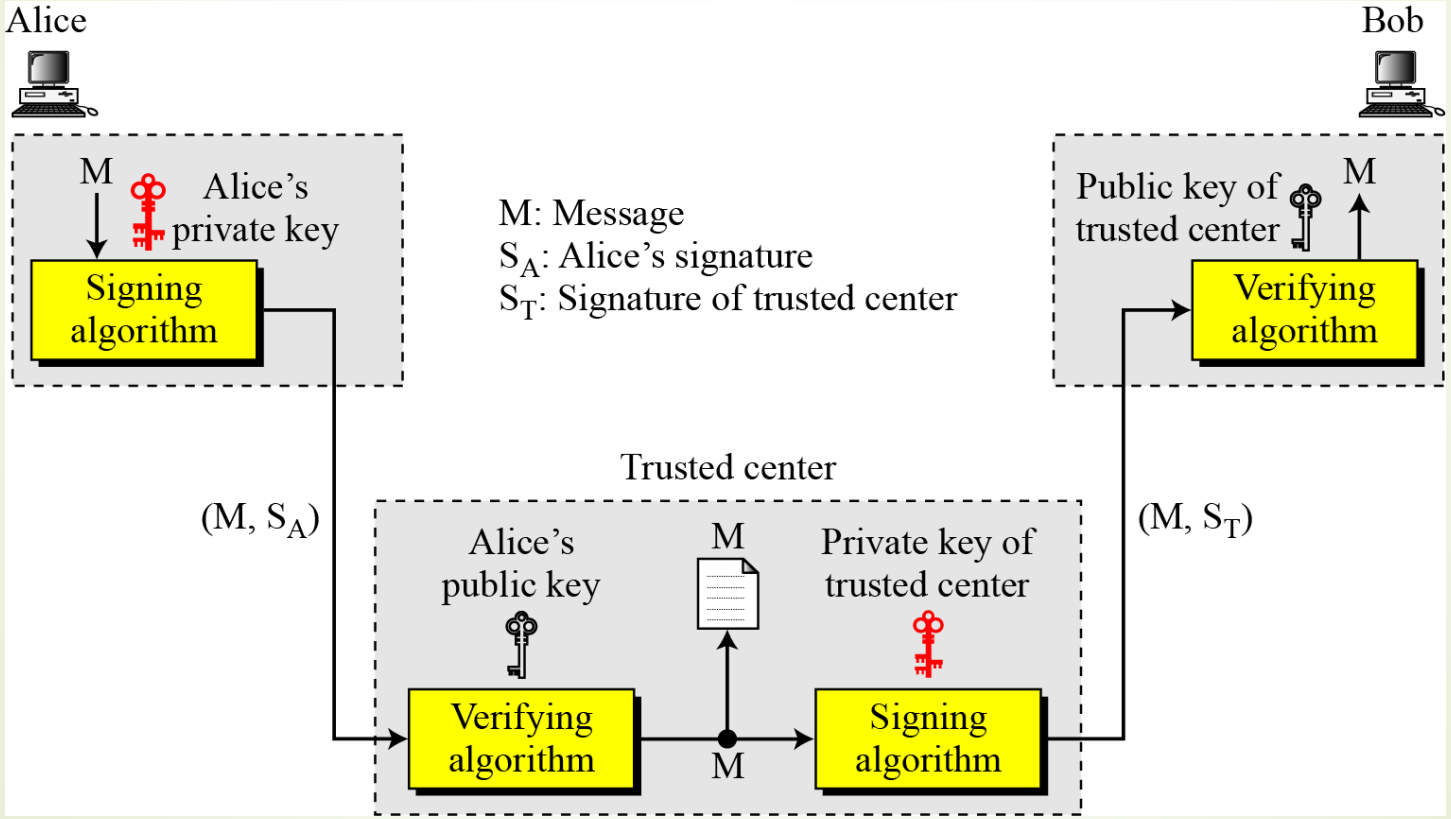
- A secure digital signature scheme, like a secure conventional signature can provide message authentication.
- A digital signature provides message authentication.

Digital Signatures: Used to provide

3. Non-repudiation

Using a trusted center to provide non-repudiation



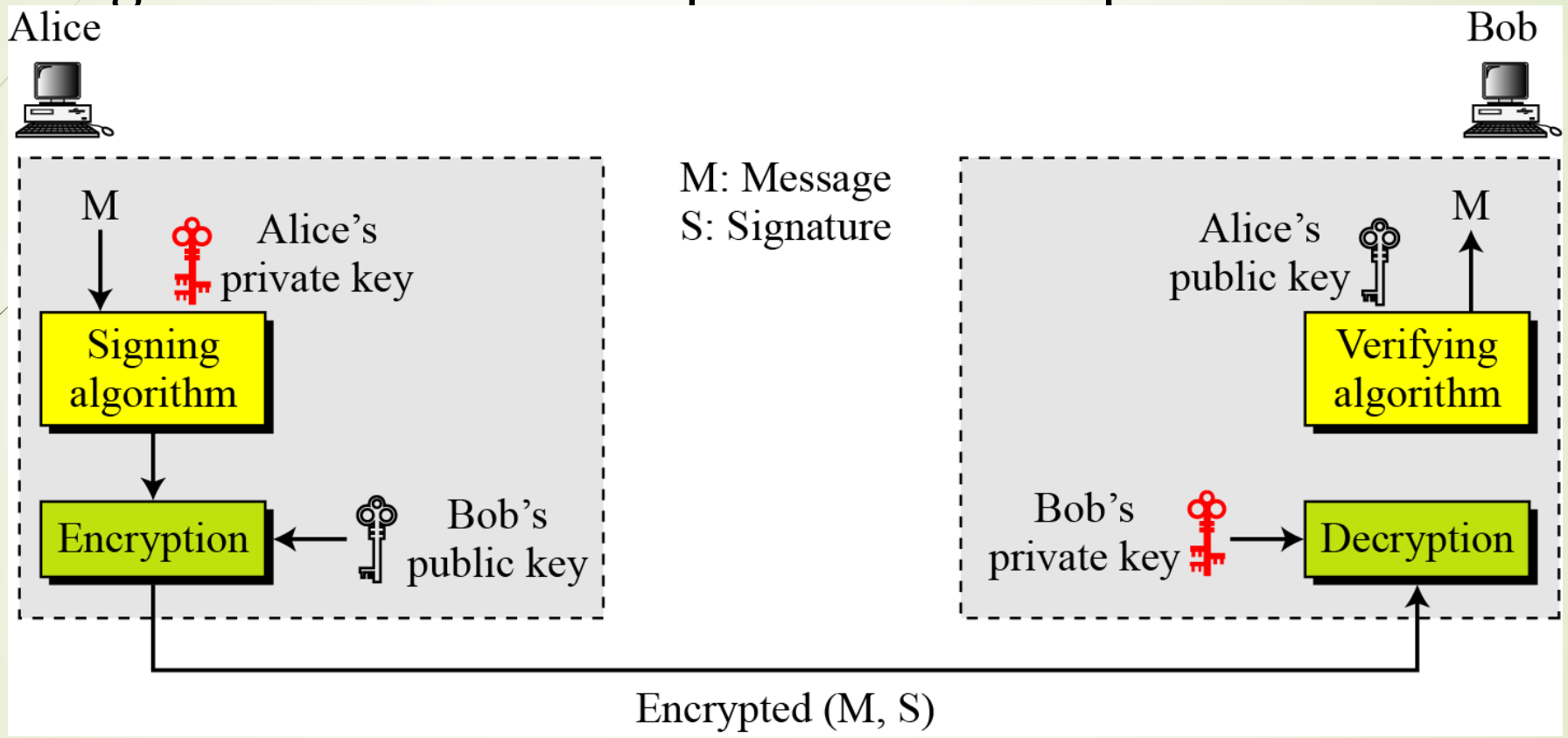


Non-repudiation can be provided using a trusted party

Digital Signatures: Used to provide

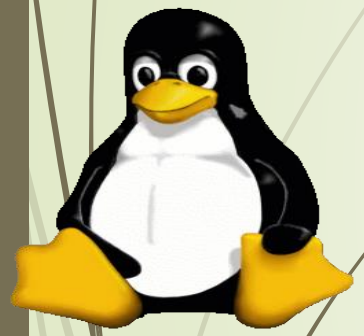
4. Confidentiality

Using a trusted center to provide non-repudiation



A digital signature does not provide privacy. If there is a need for privacy, another layer of encryption/decryption must be applied.

Introduction



Difference between a MAC and digital signatures

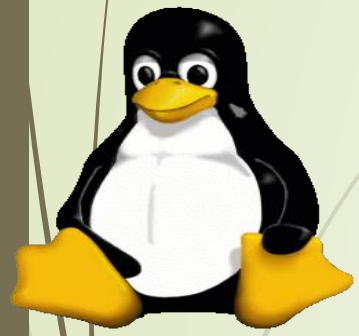
1. To prove the validity of a MAC to a third party, you need to reveal the key

2. If you can verify a MAC, you can also create it

3. MAC does not allow a distinction to be made between the parties sharing the key

4. Computing a MAC is (usually) much faster than computing a digital signature
   - Important for devices with low computing power

Properties of Digital Signatures



- Similar to handwritten signatures, digital signatures must fulfill the following:

1. Must not be forgeable
2. Recipients must be able to verify them
3. Signers must not be able to repudiate them later

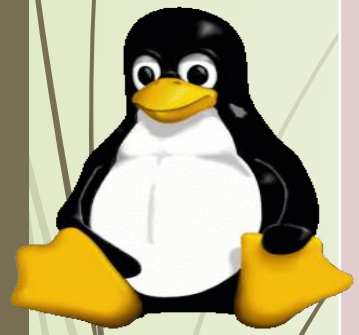
- Digital signatures cannot be constant and must be a function of the entire document it signs

- Differences between Conventional and Digital signatures

	Conventional Signatures	Digital Signatures
1. Inclusion: Signing documents	• A signature is physically part of the document being signed	• Is not attached physically to the message that is signed, so the algorithm that is used must somehow "bind" the signature to the message • The signature is sent as a separate document

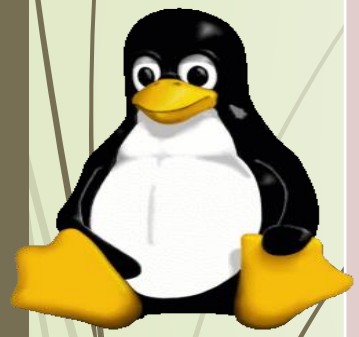
- Differences between Conventional and Digital signatures

	Conventional Signatures	Digital Signatures
2. Verification	• Verified by comparing it to other, authentic signatures. For example, when someone signs a credit card purchase, the salesperson is supposed to compare the signature on the sales slip to the signature on the back of the credit card in order to verify the signature • Easy to forge.	• Can be verified using a publicly known verification algorithm. Thus, "anyone" can verify a digital signature. • The use of a secure signature scheme will prevent the possibility of forgeries



- Differences between Conventional and Digital signatures

	Conventional Signatures	Digital Signatures
3. Duplicity	• A copy of a signed paper document can usually be distinguished from an original	• A "copy" of a signed digital message is identical to the original • There is no such distinction unless there is a factor of time on the document.

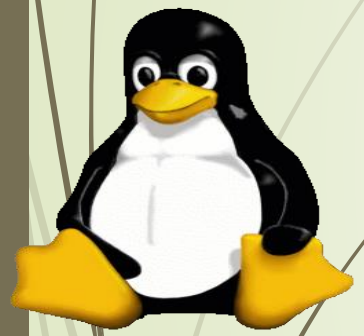


Components of a Signature Scheme

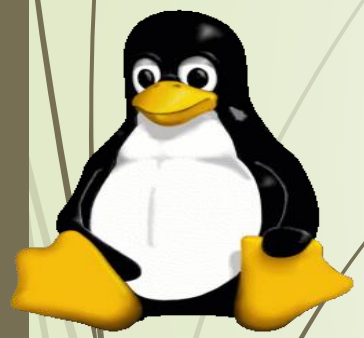
- A signature scheme consists of two components:

1. A signing algorithm
2. A verification algorithm.

- Bob can sign a message x using a (secret) signing algorithm sig.
- The resulting signature $sign(x)$ can subsequently be verified using a public verification algorithm ver.
- Given a pair (x, y), the verification algorithm returns an answer "$true$" or "$false$" depending on whether the signature is authentic or not.



Basic Algorithms of Digital Schemes



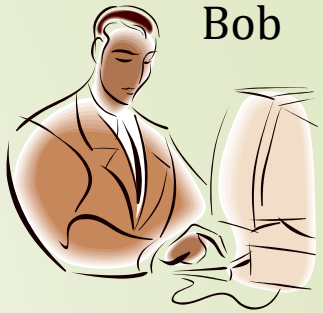
1. A key generation algorithm: To randomly select a signing key pair.

2. A signature algorithm: Takes message + private key as input and generates a signature for the message as output.

3. A signature verification algorithm: Takes signature + public key as input and generates information bit according to whether signature is consistent as output.

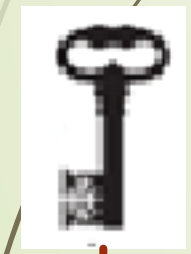
Components of a Signature Scheme

Generic Model of Signature Process



Bob

Alice's private key



Message M

Digital Signature Generation Algorithm

Alice's signature for M

σ

Message M

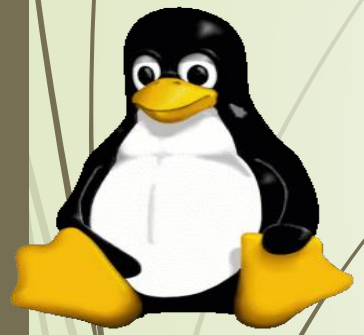
σ

Alice's public key

Digital Signature Verification Algorithm

Return Signature True/False

Formal Definition of a Signature Scheme



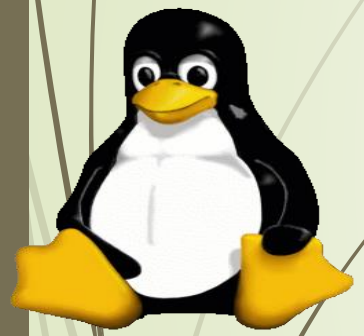
A digital signature scheme is a triple of poly-time computable algorithms $(KeyGen, Sign, Verify)$ over a message space M that satisfy the following conditions:

1. $KeyGen(1^n, R)$: Is a probabilistic (with coin flips R) poly-time algorithm that outputs a public key and a secret key pair, (PK, SK).

2. $Sign(D, PK, SK, R)$: Is a probabilistic(with coin flips R) poly-time algorithm that signs a document $D \in M$ with a signature $\sigma(D)$.
   Note: $|\sigma(D)|$ should be polynomially related to $|D|$.

3. $Verify\ (PK, D, s)$: Is a (possibly probabilistic) poly-time algorithm that outputs an element of $\{Yes, No\}$. It returns Yes (with negligible error) if s is a valid signature of D, i.e. $s = \sigma(D)$.

Signature Scheme



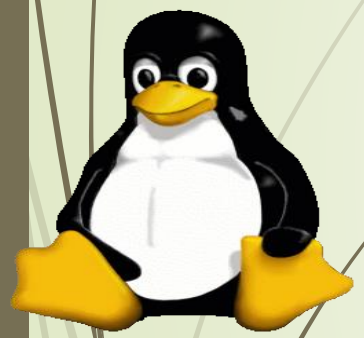
- Given such a scheme, Alice can set up her document signer.

1. First she generates $(PK, SK) \leftarrow KeyGen(1^n, R)$ and publishes PK while keeping SK secret.

2. Then when she wants to sign a document, D, she can run the signing algorithm $\sigma(D) \leftarrow Sign(D, PK, SK, R)$ and sends the pair $(D, \sigma(D))$ to Bob.

3. Bob can then verify the signature by running $Verify(PK, D, \sigma(D))$.

Properties of a Signature Scheme



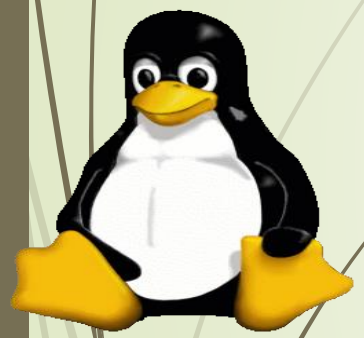
Two important properties of a digital signature scheme are

1. Correctness
2. Privacy.

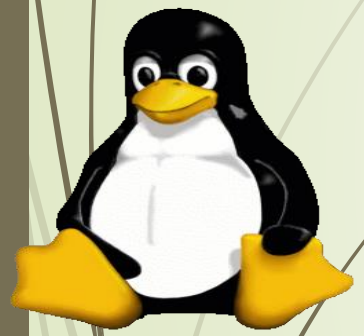
- By correctness, we mean that if PK and SK are generated according to $KeyGen(\,)$, then for all $D \in M$, $Verify\big(PK, D, \sigma(D)\big) = Yes$. Hence, a legally signed message can never be rejected.
- We also say that Eve forges a signature if she can produce a D and $\sigma(D)$ (that was not signed by Alice) such that $Verify\big(PK, D, \sigma(D)\big) = Yes$ with non-negligible probability.

Properties of a Signature Scheme

- It must verify the author and the date and time of the signature.
- It must authenticate the contents at the time of the signature.
- It must be verifiable by third parties, to resolve disputes.



Attacks on Signature Schemes



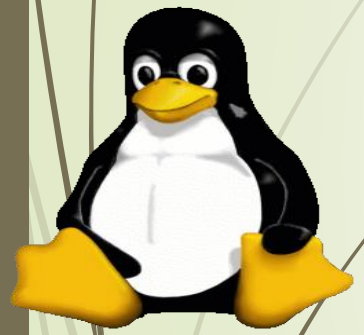
1. Key-only Attack: Eve only knows the public key.

2. Known Message Attack: Eve has seen a set of messages $\{m_1, \dots, m_k\}$ with their corresponding signatures $\{\sigma_1, \dots, \sigma_k\}$. The set of messages is given to her but not chosen by her.

3. Generic Chosen Message Attack: Eve chooses a fixed set of messages $\{m_1, \dots, m_k\}$ (there are two cases, where the messages are chosen independently of the public key or not) and gets to see the signatures of those messages $\{\sigma_1, \dots, \sigma_k\}$.

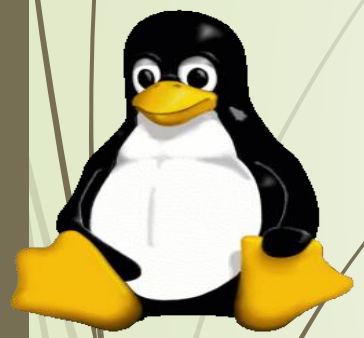
Attacks on Signature Schemes

4. Directed Chosen Message Attack: Similar to the generic attack, except that the list of messages to be signed is chosen after Eve knows A's public key but before any signatures are seen.



5. Adaptive chosen message attack: Eve is allowed to use A as an "oracle." This means the A may request signatures of messages that depend on previously obtained message–signature pairs.

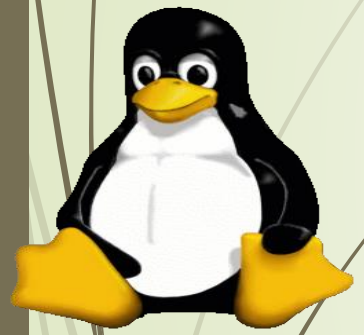
Security of a Signature Scheme



Q: Once this information is given to her, what does it mean for the signature scheme to be broken by Eve?

1. Total Break: Eve computes the secret key. This is as bad as it gets because now Eve can sign any message she wants.
2. Universal Forgery: Eve computes a poly-time algorithm that can forge a signature for any message.
3. Selective Forgery: Eve can forge a signature for a particular message of her choice.
4. Existential forgery: Eve forges a signature for at least one message. Eve has no control over the message. Consequently, this forgery may only be a minor nuisance to Alice.

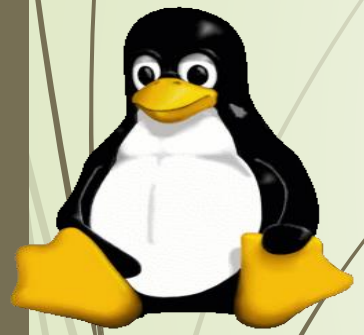
Requirements of a good Signature Scheme



- On the basis of the properties and attacks, the following requirements for a digital signature are formulated.

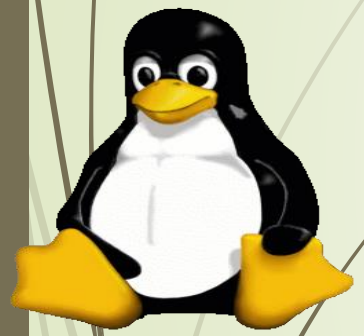
1. The signature must be a bit pattern that depends on the message being signed.
2. The signature must use some information unique to the sender to prevent both forgery and denial.
3. It must be relatively easy to produce the digital signature.
4. It must be relatively easy to recognize and verify the digital signature.
5. It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be practical to retain a copy of the digital signature in storage.

Security of a Signature Scheme



- When we talk about security for a digital signature scheme, we consider an adversary, Eve, who attempts to send a message to Bob and tries to forge Alice's signature.
- What possible information does Eve have access to before attacking the system?
- Here are some reasonable assumptions that have been proposed:

The ElGamal Signature Scheme

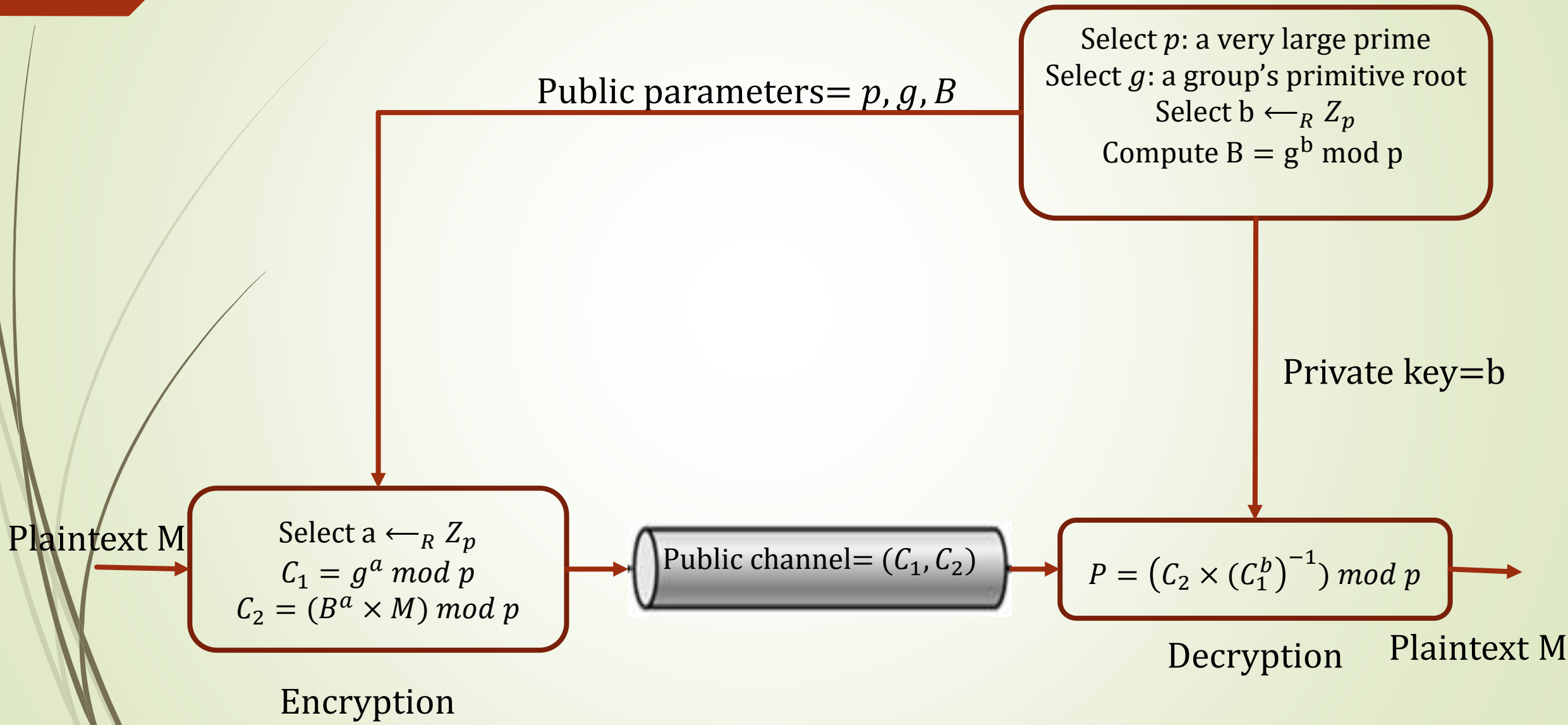


- Designed specifically for the purpose of signatures
- The ElGamal Signature Scheme is non-deterministic: This means that there are many valid signatures for any given message.
- The verification algorithm must be able to accept any of the valid signatures as authentic.

Note:

- The Elgamal Encryption Scheme is designed to enable encryption by a user's public key with decryption by the user's private key.
- The Elgamal Signature Scheme involves the use of the private key for encryption and the public key for decryption

The ElGamal Encryption Scheme



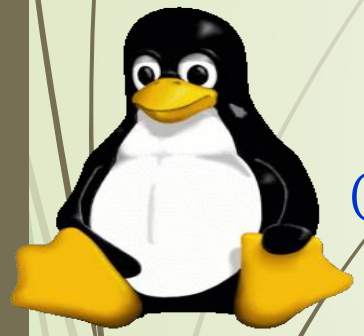
The ElGamal Signature Scheme: General Idea

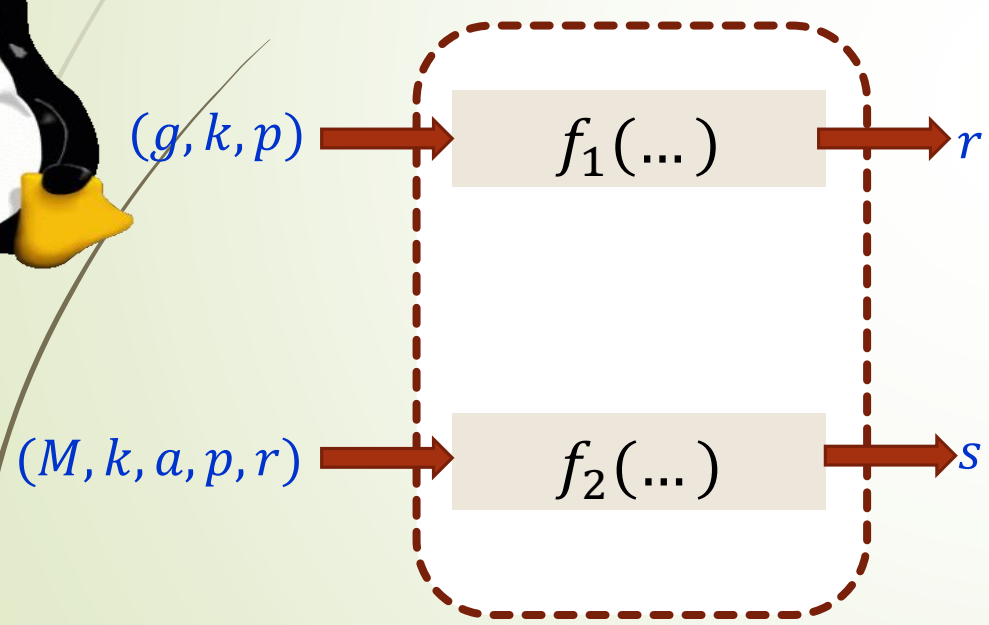
Signatures (r, s)
Message M
Public key $(p, g, A = g^a)$

Private key a
Random secret k

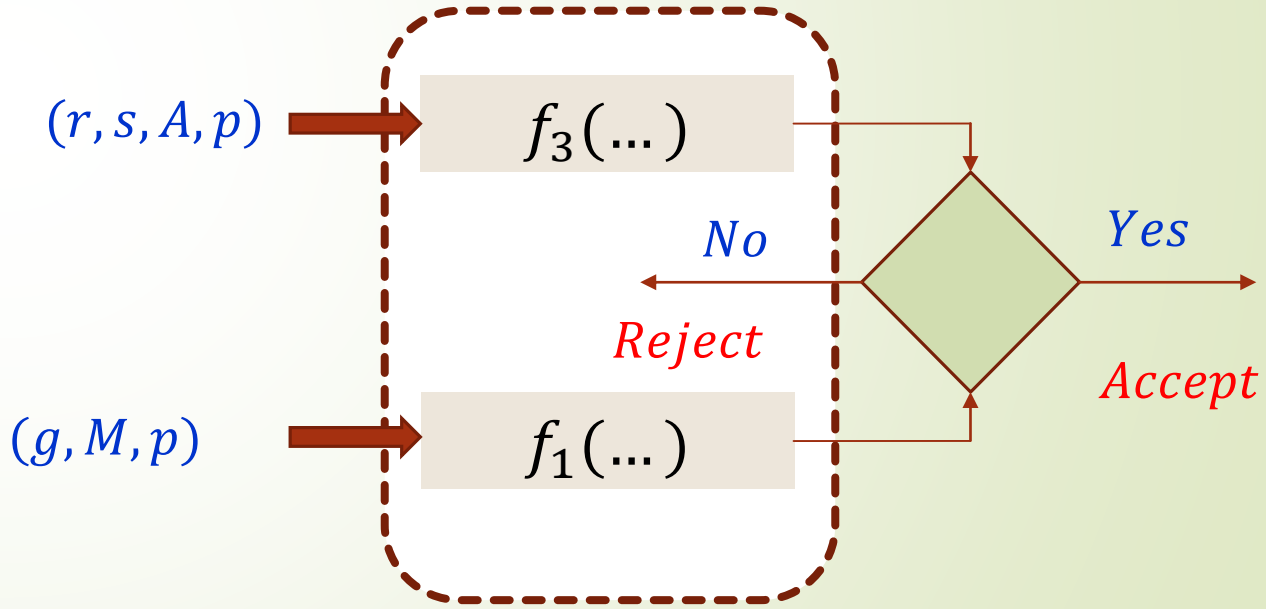
Signatures (r, s)

$$r = g^k mod\ p$$
$$s = k^{-1}(h(x - ar) mod\ p - 1$$



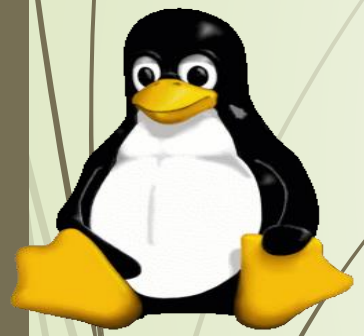


Signing Function



Veryfying Function

The ElGamal Signature Scheme



KeyGen

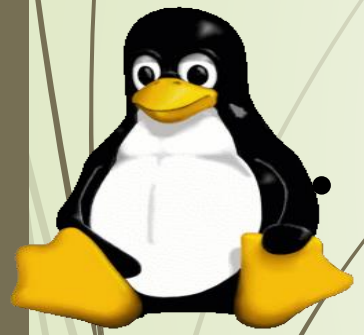
Key generation is the same as for the ElGamal encryption system

1. Alice generates a large random prime p and a primitive root $g \ mod \ p$.
2. She also chooses a randomly in the set $\{1,2, \dots, p-2\}$ and computes

$$A = g^a \ mod \ p$$

1. Her private key is a.
2. Her public key is (p, g, A).

The ElGamal Signature Scheme



Sign

- Alice signs a document $x \in \{0,1\}^*$.
- She uses the publicly known collision resistant hash function

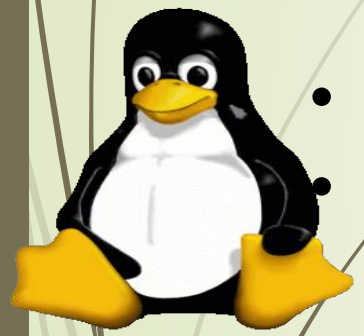
$$h(x): \{0,1\}^* \rightarrow \{1,2, \dots, p-2\}$$

- Alice also chooses a random number $k \in \{1,2, \dots, p-2\}$ which is coprime to $p-1$ i.e . $\gcd(\mathrm{k}, \mathrm{p}-1) = 1$
- She computes

$$r = g^k \bmod p,$$
$$s = k^{-1}(h(x) - ar) \bmod p - 1$$

- The signature of x is the pair (r, s).

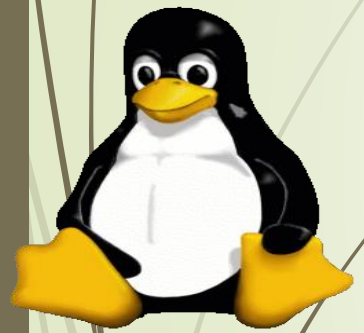
The ElGamal Signature Scheme



$Verify$

- Bob, the verifier, uses Alice's public key (p, g, A).
- First, he has to convince himself of the authenticity of this public key.
- He verifies that $1 \leq r < p - 1$.
- If this condition is not satisfied, then he rejects the signature; otherwise, he checks the congruence $A^r r^s \equiv g^{h(x)} \, mod \, p$.
- He accepts the signature if this congruence holds; otherwise, he rejects it.

The ElGamal Signature Scheme



Corectness

We show that the verification works.

$$A^r r^s \equiv g^{h(x)} \bmod p$$

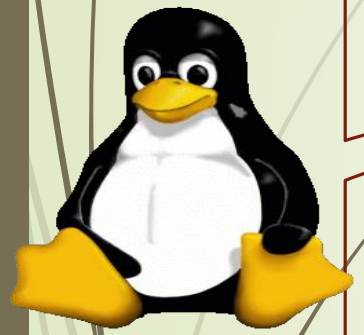
$$A^r r^s \equiv g^{ar} g^{ks}$$

$$\equiv g^{ar} g^{kk^{-1}(h(x)-ar)}$$

$$\equiv g^{ar+(h(x)-ar)}$$

$$\equiv g^{h(x)} \bmod p$$

The ElGamal Signature Scheme



Example 1

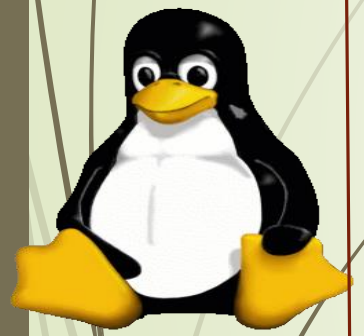
KeyGen

- Alice chooses $p = 23, g = 7, a = 6$ and computes $A = g^a \ mod \ p$.
- Her public key is $(\mathrm{p} = 23, \mathrm{g} = 7, \mathrm{A} = 4)$
- Her private key is $a = 6$.

Sign

- Alice wants to sign the document x, which has value $h(x) = 7$.
- She chooses $k = 5$ and obtains $r = 17$.
- The inverse of $k \ mod \ (p - 1 = 22)$ is $k^{-1} = 9$.
- Therefore, $s = k^{-1}(h(x) - ar) mod \ (p - 1) = 9 \times \big(7 - 6(17)\big) \ mod \ 22 = 3$
- The signature is (17,3)

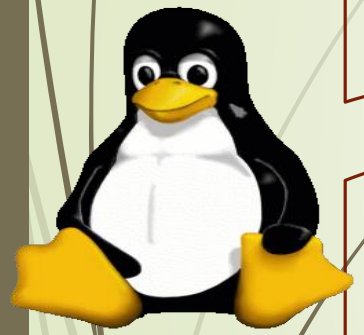
The ElGamal Signature Scheme



Verify

- Bob wants to verify this signature.
- He computes $A^r r^s \ mod \ p = 4^{17} \times 17^3 \ mod \ 23 = 5$.
- He also computes $g^{h(x)} \ mod \ p = 7^7 \ mod \ 23 = 5$, so the signature is verified.

The ElGamal Signature Scheme



Example 2

Assume a prime field GF(19). It has primitive roots {2,3,10,13,14,15}

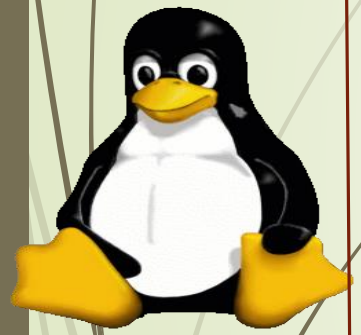
KeyGen

- Alice chooses $p = 19, g = 10, a = 16$ and computes $A = g^a \ mod \ p$.
- Her public key is $(\mathrm{p} = 23, \mathrm{g} = 10, \mathrm{A} = 4)$
- Her private key is $a = 16$.

Sign

- Alice wants to sign the document x, which has value $h(x) = 14$.
- She chooses $k = 5$ and obtains $\mathrm{g^k mod}\ 19 = 3$.
- The inverse of $k \ mod \ (p - 1 = 18)$ is $k^{-1} = 5^{-1} = 11$.
- Therefore, $s = (k^{-1}(h(x) - ar) mod \ (p - 1) = 11 \times (14 -$

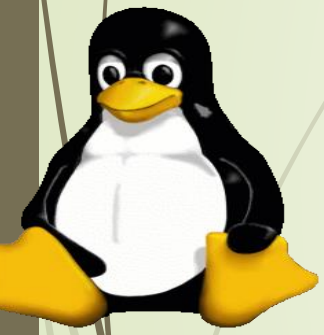
The ElGamal Signature Scheme



Verify

- Bob wants to verify this signature.
- He computes $A^r r^s \ mod \ p = 4^3 \times 3^4 \ mod \ 19 = 16$.
- He also computes $g^{h(x)} \ mod \ p = 10^{14} \ mod \ 19 = 5184 \ mod \ 19 = 16$
- Thus the signature is valid.

Example 3



Jane chooses $p = 3119, g = 2, d = 12\ \ 7$ and calculates $A = 2^{127}\ mod\ 3119 = 1702$. She also chooses r to be 307. She announces g, A, and p publicly; she keeps d secret.

1. Show how Jane can sign a message $M = 320$.
2. Show how Mary can verify the signature on the message

Suppose now Jane wants to send another message, M = 3000, to Tim. She chooses a new r, 107.

1. Show how Jane can sign the new message.
2. Show how Tim can verify the signature on the message

Example 3

Example 3

$M = 320$

$S_1 = e_1{}^r = 2^{307} = 2083 \bmod 3119$

$S_2 = (M - d \times S_1) \times r^{-1} = (320 - 127 \times 2083) \times 307^{-1} = 2105 \bmod 3118$

$V_1 = e_1{}^M = 2^{320} = 3006 \bmod 3119$

$V_2 = d^{S_1} \times S_1{}^{S_2} = 1702^{2083} \times 2083^{2105} = 3006 \bmod 3119$

$M = 3000$

$S_1 = e_1{}^r = 2^{107} = 2732 \bmod 3119$

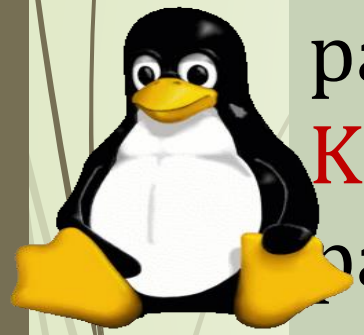
$S_2 = (M - d \times S_1)\, r^{-1} = (3000 - 127 \times 2083) \times 107^{-1} = 2526 \bmod 3118$

$V_1 = e_1{}^M = 2^{3000} = 704 \bmod 3119$

$V_2 = d^{S_1} \times S_1{}^S = 1702^{2732} \times 2083^{2526} = 704 \bmod 3119$

Key Distribution and Key Agreement

Introduction



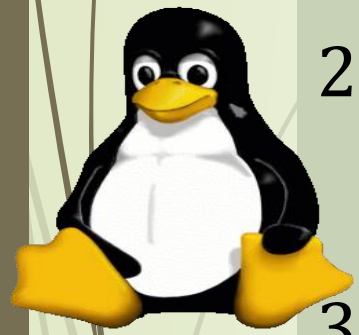
Definitions

Key distribution: A mechanism whereby one party chooses a secret key and then transmits it to another party or parties.

Key agreement: Denotes a protocol whereby two (or more) parties jointly establish a secret key by communicating over a public channel.

In a key agreement scheme, the value of the key is determined as a function of inputs provided by both parties.

Key Distribution

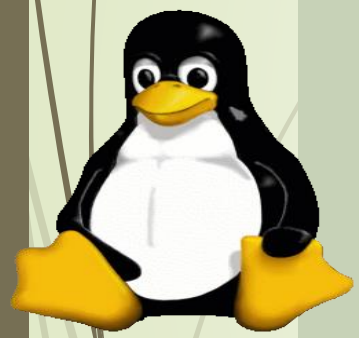


1. Symmetric encryption schemes require both parties to share a common secret key

   Issue is how to securely distribute this key without revealing it to an adversary
2. Many attacks are based on poor key management and distribution

   Rather than breaking the codes
3. This is, actually, the most difficult problem in developing secure systems

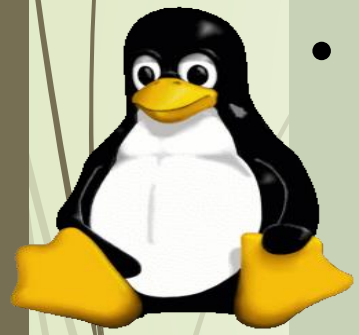
Key Distribution



Various key distribution alternatives exist for parties A and B:

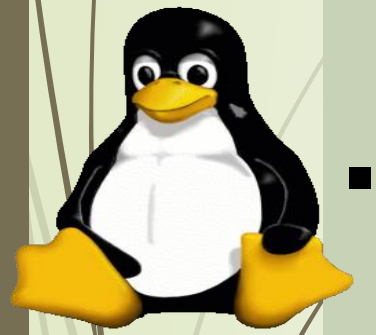
1. A can select key and physically deliver to B
   - Does not scale for a large and distributed group
   - How many keys do we need for N users?
2. Third party can select & physically deliver key to A & B
   - Similar comment as 1
   - Sometimes finding a “trusted” third party is another problem
3. If A & B have communicated previously, they can use previous key to encrypt a new key
   - Good option but initially several keys to be distributed
4. If A & B have secure communications with a third party C, C can relay key between A & B on demand
   - Only N master keys are enough

Key Distribution Facts



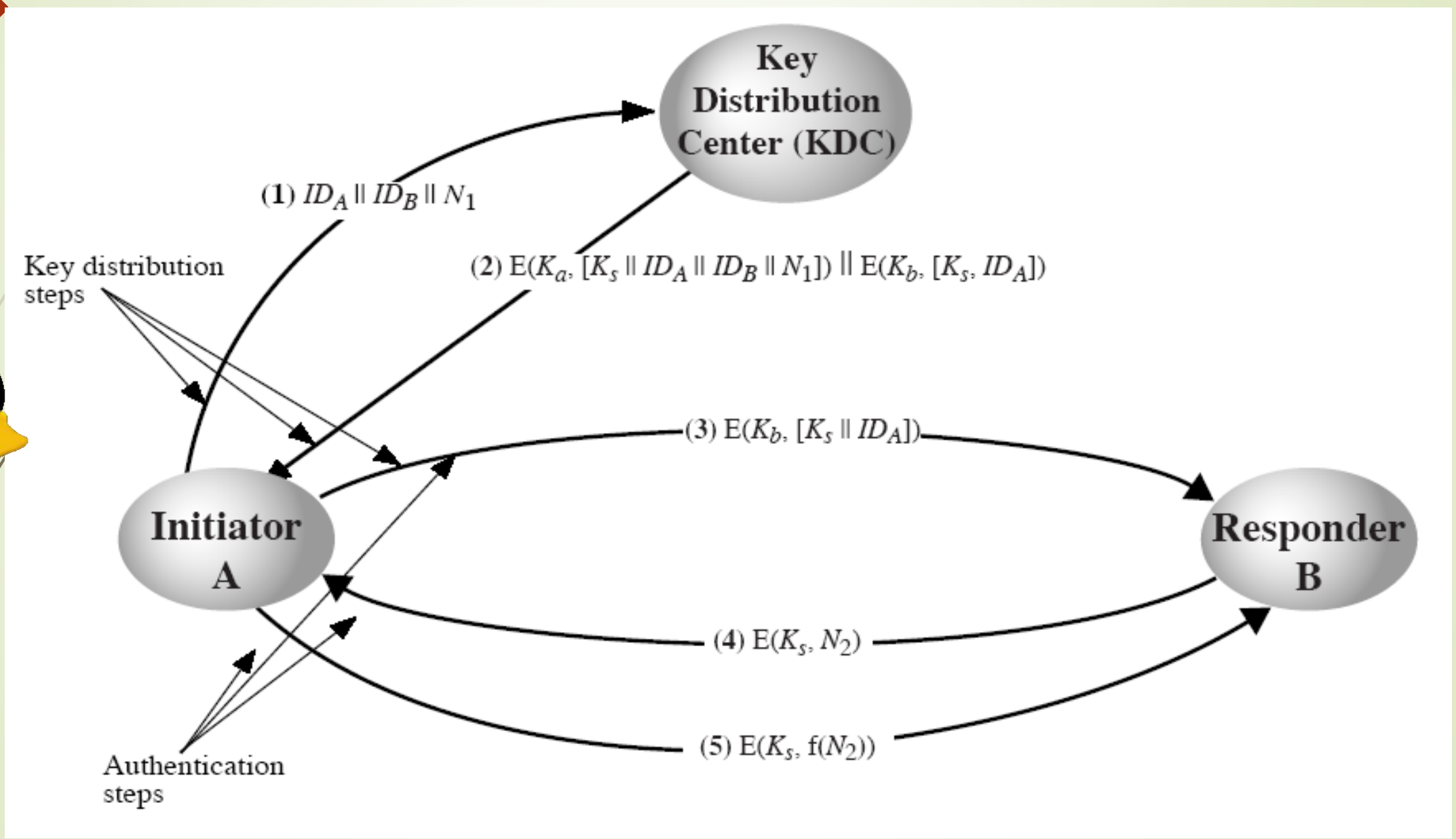
- "Conservation of trust" principle
  - A secure communication cannot be based on nothing; either there should be an initial direct contact or an indirect protocol to transfer trust
- Either physical delivery or a trusted third party
  - Physical delivery is the only option to avoid a third party
    - Most basic system is PIN entry
  - Otherwise regardless of symmetric or asymmetric encryption, you need a trusted third party

Key Distribution Centre (KDC)



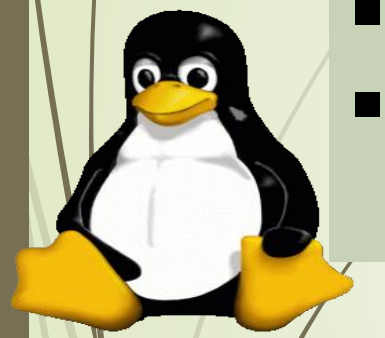
- The use of a KDC is based on the use of a hierarchy of keys. At minimum, two levels of keys are used (master and session key)
- Communication between two end systems is encrypted using a temporary key often referred to as a session key.
- Typically, the session key is used for the duration of a logical connection and then discarded
- The master key is shared between the KDC and an end system or user and is used to encrypt the session key.

Key Distribution Scenario

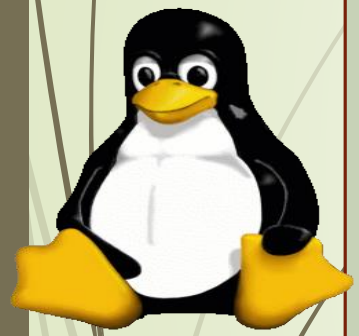


Key Distribution Scenario

- Let us assume that A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.
- A has a master key k_a known only to itself and the KDC
- Similarly, B shares the master key k_b with the KDC.



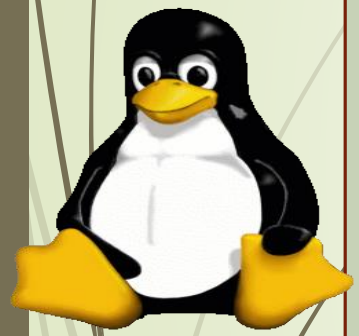
Key Distribution Scenario



Step 1

- A issues a request to the KDC for a session key to protect a logical connection to B.
- The message includes the identity of A and B and a unique identifier N_1 for this transaction referred to as a nonce.
- The nonce may be a timestamp, counter, or a random number. The minimum requirement is that it differs with each request.
- To prevent a masquerade, it should be difficult for an opponent to guess the nonce. Thus a random number is good for a nonce.

Key Distribution Scenario



Step 2

- The KDC responds with a message encrypted using k_a thus A is the only one who can successfully read the message and A knows that it originated from the KDC. The message includes two items intended for A,

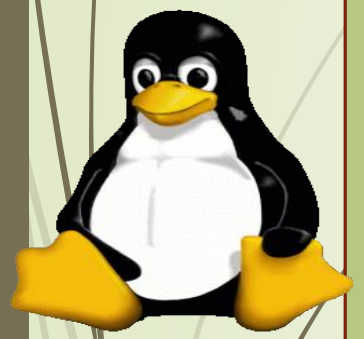
i. The one-time session key k_s intended for A
ii. The original request message including the nonce to enable A to match this request with the appropriate request

- Thus A can verify that its original request was not altered before reception by the KDC and because of the nonce, that it is not a replay of some previous request.
- The message also includes two items for B

i. The one-time session key k_s to be used for the session
ii. An identifier of A (e.g. its network address), IDA

- This two are encrypted with k_b and are sent to B to establish a connection and prove A's identity

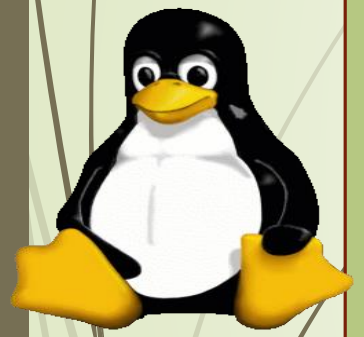
Key Distribution Scenario



Step 3

- A stores the session key for use in upcoming session and forwards to B the information that originated from the KDC namely $E(k_b, [k_s \parallel ID_A])$.
- Because this information is encrypted by k_b, its protected from eavesdropping.
- B now knows the session key k_s, knows the other party is A from ID_A and knows that the information originated from the KDC (because its encrypted by k_b)
- At this point the session key has been securely delivered to A and B and they may begin their protected exchange
- However two additional steps as desirable

Key Distribution Scenario



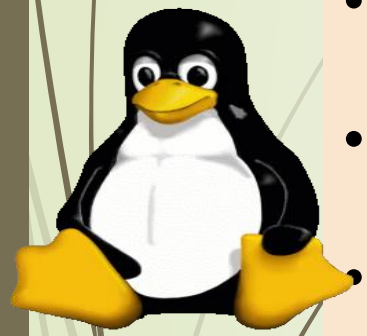
However two additional steps as desirable

Step 4

- Using the newly minted session key for encryption, B sends a nonce N_2 to A
- Also using K_s, A responds with $f(N_2)$where f is a function that performs some transformations on N_2(e.g. adding one)
- These steps assures B that the original message it received (step 3) was not a replay.

Note: The key distribution only involves step 1 to 3 while step 4 and 5 perform an authentication function.

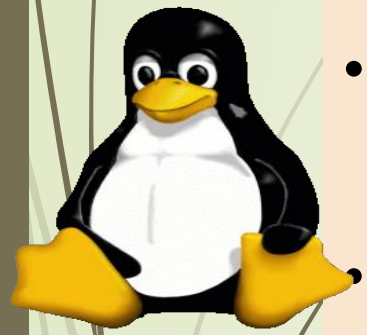
Major Issues with the KDC



1. Hierarchical key control

- Its not necessary to limit the key distribution function to a single KDC esp. for very large networks.
- As an alternative, a hierarchy of KDC's can be established
- For example, there can be local KDC's each responsible for a small domain of the overall internetwork.
- If two entities in different domains desire a shared key , then the corresponding local KDC's can communicate through the global KDC
- The hierarchical concept can extended to three or even more layers depending on the size of the user population and the geographic scope of the internetwork.
- A hierarchical scheme minimizes the effort involved in master key distribution because most master keys are those shared by the local KDC with its local entities

Major Issues with the KDC



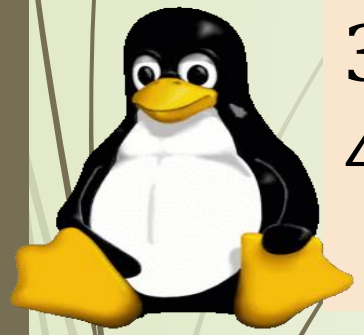
2. Session Key Life Time

- The distribution of session keys delays the start of any exchange and places a burden on network capacity
- A security manager must try to balance these competing considerations in determining the lifetime of a particular session key
- For connection oriented protocols, one obvious choice is to use the same session key for the length of time that the connection is open, using a new session key for each session
- If a logical connection has a very long lifetime, it would be prudent to change the session key periodically.
- For connectionless protocol, the is no connection initiation or termination, hence not obvious how often one needs to change the session keys. The most secure way is to use a new session key for each exchange
- Another better strategy is to use a given session key for a certain fixed only or for a certain number of transactions

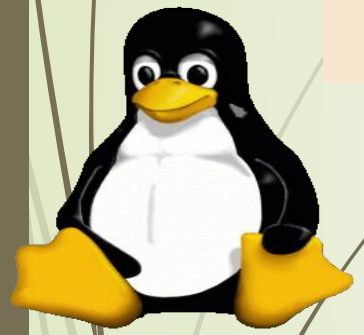
Techniques of distributing public keys

Some methods

1. Public Announcement
2. Publicly available databases/directories
3. Centralized Distribution: Public key Authority
4. Public key Certificates



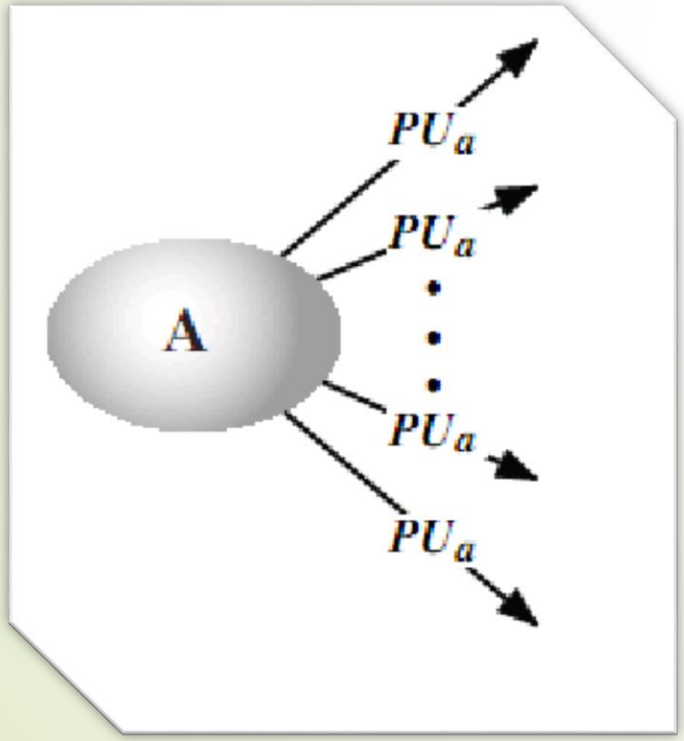
Techniques of distributing public keys

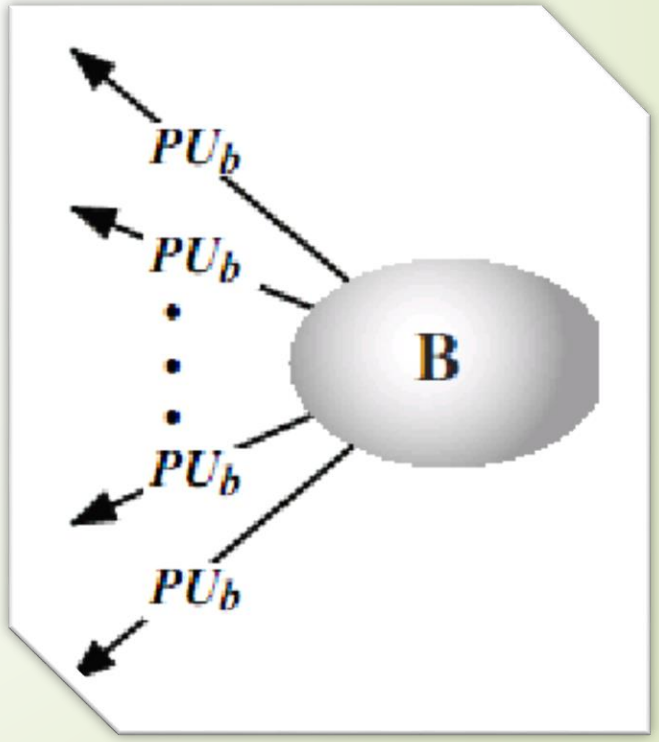


1. Public Announcement

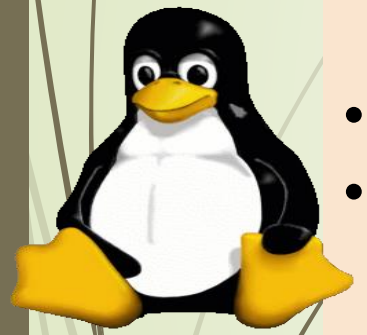
Broadcast your public key to the public

- Via newsgroups, mailing lists, from personal website, etc.
- Major weakness is anyone can easily pretend as yourself (forgery): so attacks are possible





Techniques of distributing public keys

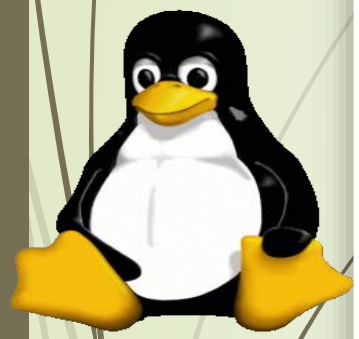


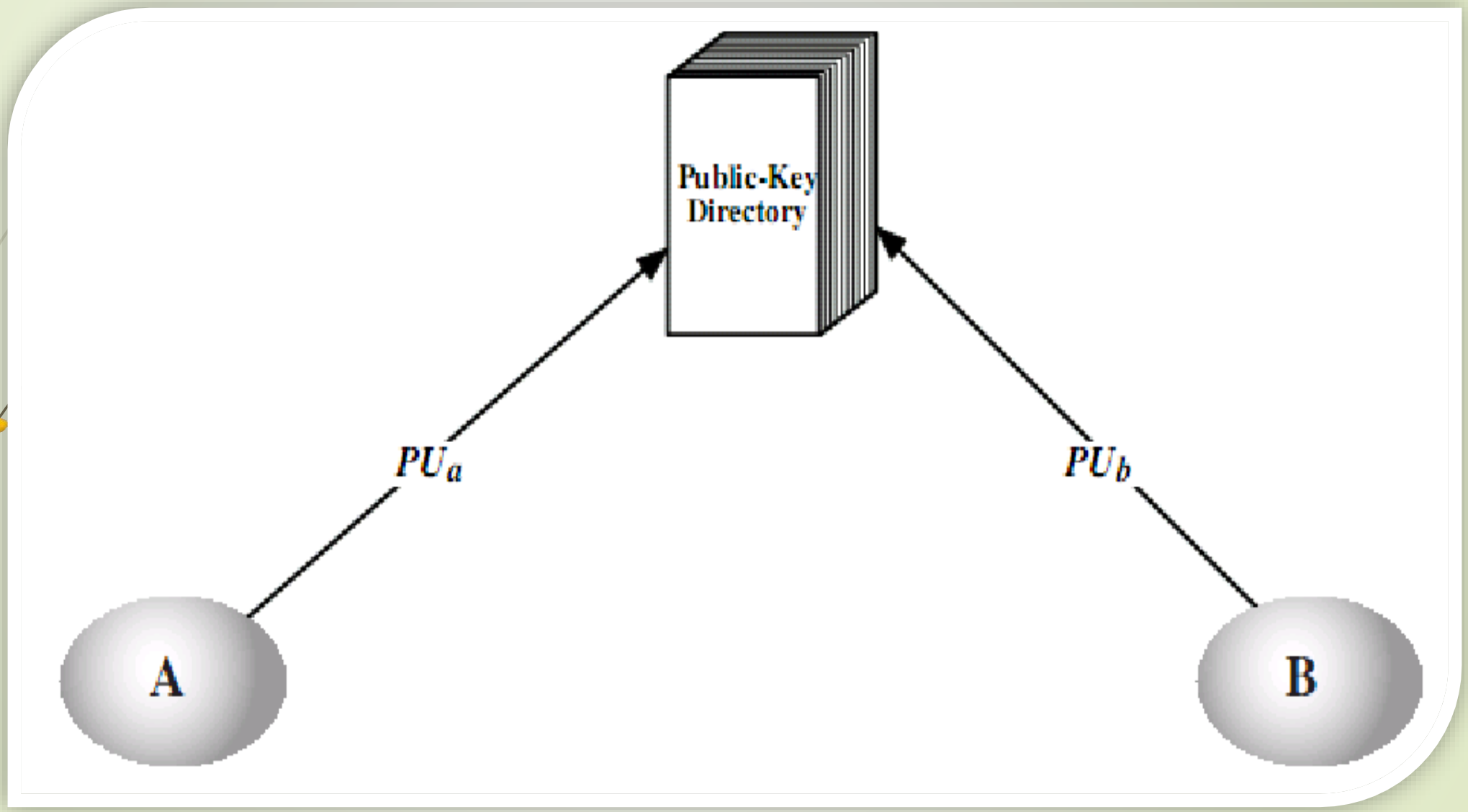
2. Publicly available databases/directories

We can obtain greater security by registering keys with a public directory
The directory must be trusted with the following properties

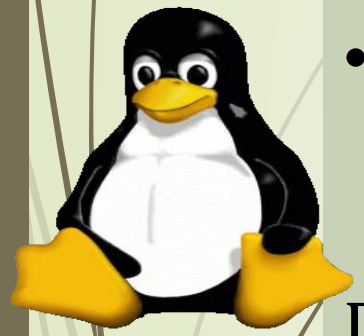
- The authority maintains a directory/database with {name, public key} pairs for each participant
- Each participant registers a public key with the directory authority
- A participant may replace the existing key with a new one at any time because the corresponding private key has been compromised in some way
- Participants could also access the directory electronically. For this purpose, secure authenticated communication from the authority to the participant is mandatory
- If administered thoroughly: - good
  - But a proper administration is difficult
    - Need secure mechanisms for registration, update, delete.

Techniques of distributing public keys





Techniques of distributing public keys



3. Centralized Distribution: Public key Authority

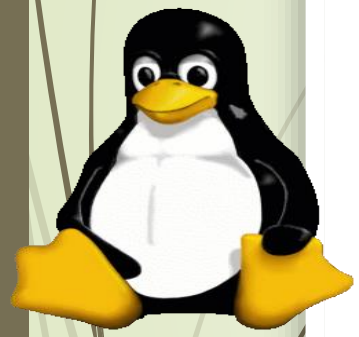
- Stronger security for public key distribution can be achieved by providing tighter control over the distribution of public keys from the directory
- It requires users to know the public key for the directory and that they interact with the directory in real-time to obtain any desired public key securely

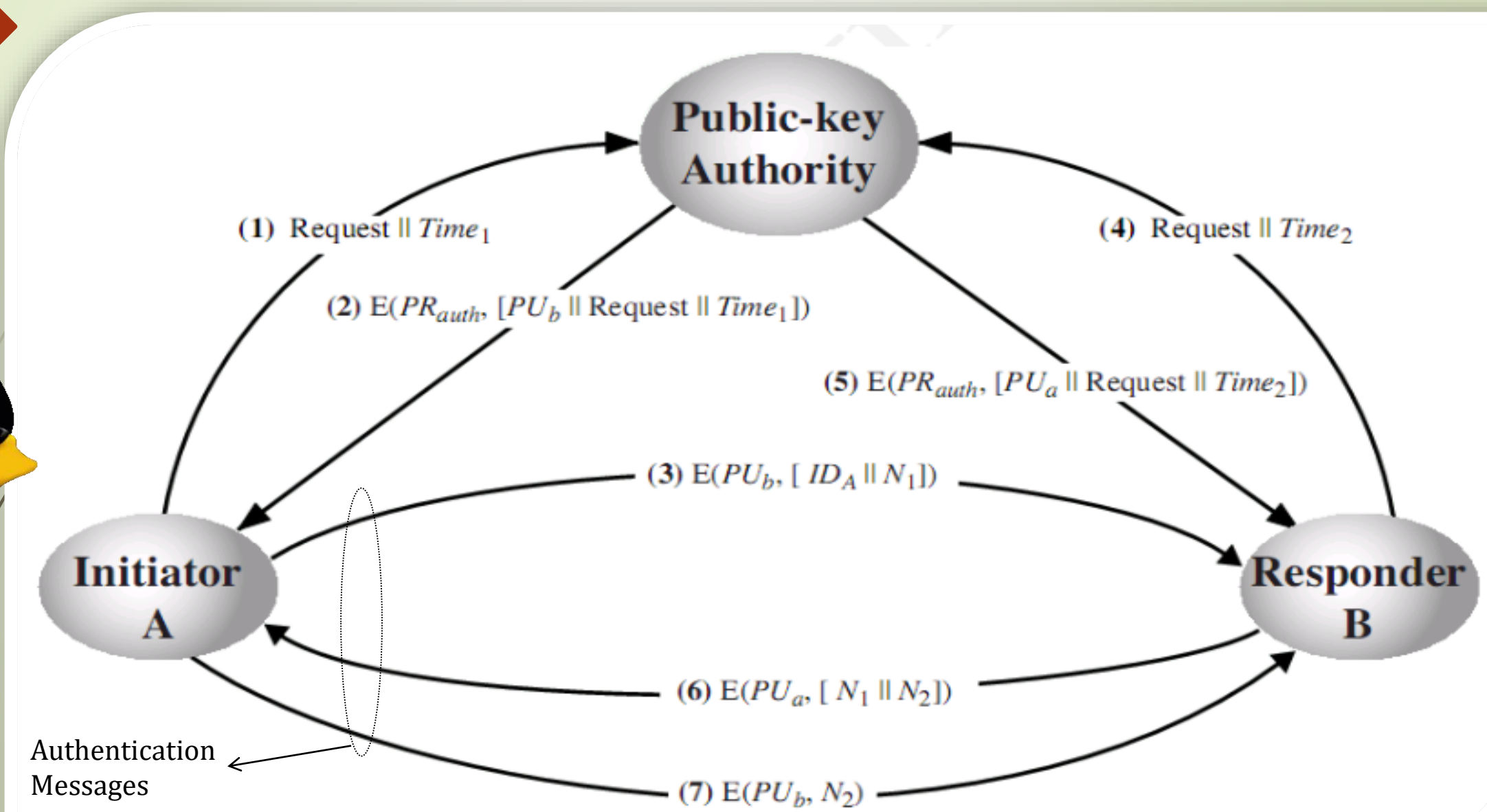
Disadvantages

a. Authority is an active entity and may create a performance bottleneck
b. Database should be kept secure to prevent unauthorized modification

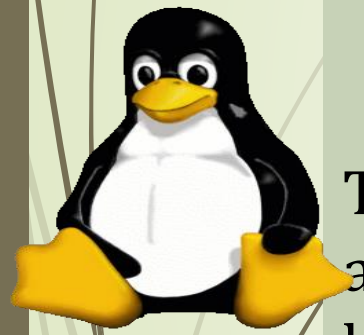
- Totally seven messages are required.

Techniques of distributing public keys





Techniques of distributing public keys

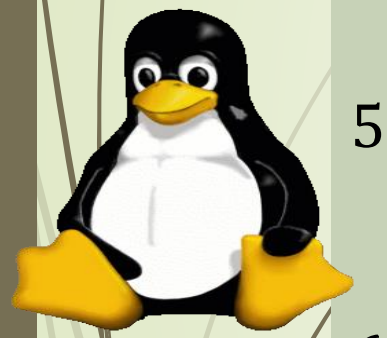


1. A sends a time stamped message to the public key authority containing a request for the current key of B.
2. The authority responds with a message that is encrypted using the authority's privates key PR_{auth}. Thus A is bale to decrypt the message using the authorities public key. Hence A is assured that the message originated with the authority.

The message includes the following

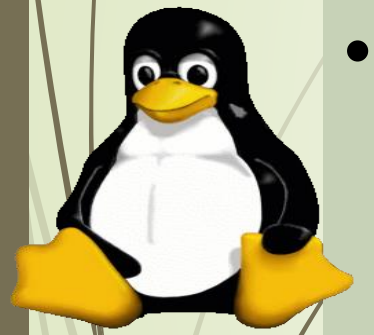
a. B's public key PU_b which A can use to decrypt messages destined for B
b. The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority.
c. The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key.

Techniques of distributing public keys



3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1) which is used to identify this transaction uniquely.
4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.
5. At this point, public keys have been securely delivered to A and B and they may begin there protected exchange. However two additional steps are desired.
6. B sends A a message encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) . Because only B could have decrypted message (3), the presence of (N_1) in message (6) assures A that the correspondent is B.
7. A returns (N_2) , encrypted using B's public key, to assure B that its correspondent is A.

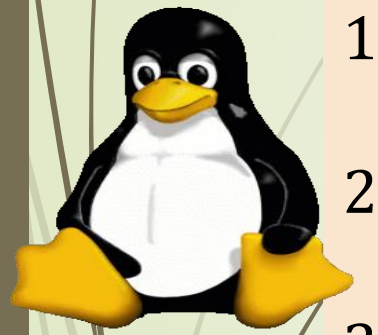
Techniques of distributing public keys



4. Public key Certificates

- PKC's can be used to exchange keys without contacting a public key authority
- A certificate binds an **identity** to **public keys** with all contents signed by a trustee public-key or Certificate Authority(CA)
- This can be verified by anyone who knows the public-key authorities public-key

Techniques of distributing public keys



4. Public key Certificates

A participant can also convey its key information to another by transmitting its certificate

Other participants can verify that the certificate was created by the authority,

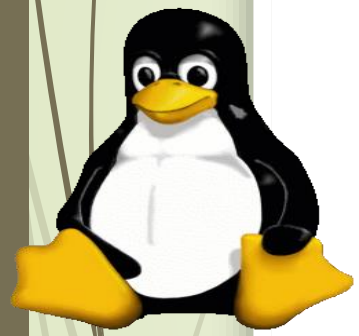
The following requirements can be placed on this scheme

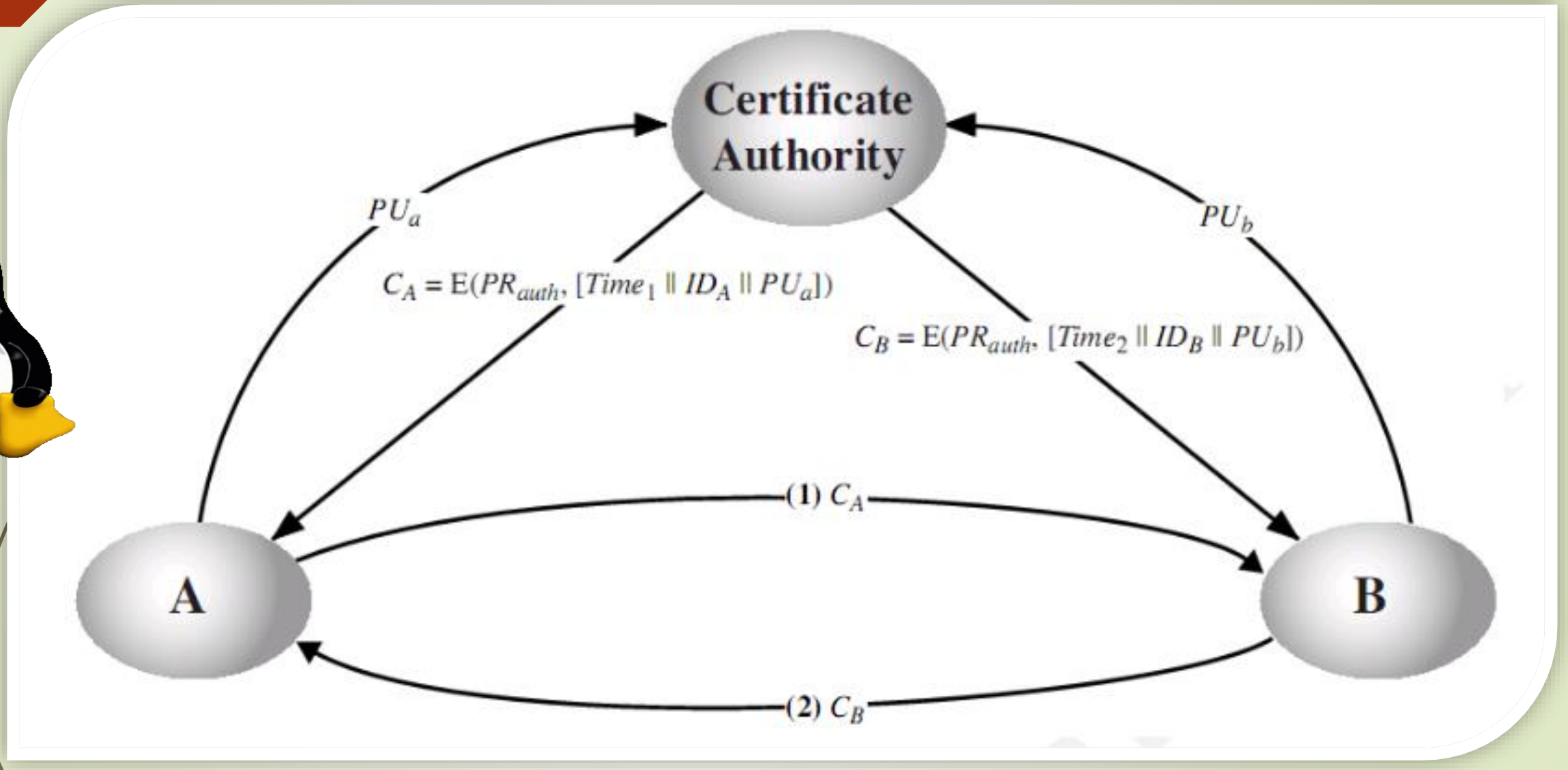
1. Any participant can read a certificate to determine the name and public key of the certificate's owner
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit
3. Only the certificate authority can create and update certificates
4. Any participant can verify the currency of the certificate

One such scheme has become universally accepted for formatting public-key certificates: the X.509 standards

X.509 certificates are used in most network security applications, including IP security, secure socket layer (SSL), secure electronic transactions (SET).

Techniques of distributing public keys





Assignment

Explain the working mechanism of the following signature schemes (15 marks)

1. RSA signature scheme (5 mark)
2. Digital Signature Standard (5 mark)
3. Schnorr Signature Scheme(5 mark)