Network Security - Week 3

Manuel E. Correia

DCC/FCUP

2025

Authentication Protocols - Examples

Enter the NSA

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader
2. Enter PIN

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader
2. Enter PIN
3. Is the pin correct?

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader
2. Enter PIN
3. Is the pin correct?
   - **YES** - Open the door

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader
2. Enter PIN
3. Is the pin correct?
   - **YES** - Open the door
   - **NO** - Get immediately shot by guard

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader
2. Enter PIN
3. Is the pin correct?
    - **YES** - Open the door
    - **NO** - Get immediately shot by guard

ATM Machine Protocol

1. Insert ATM card

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader
2. Enter PIN
3. Is the pin correct?
    - **YES** - Open the door
    - **NO** - Get immediately shot by guard

ATM Machine Protocol

1. Insert ATM card
2. Enter PIN

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader
2. Enter PIN
3. Is the pin correct?
    - **YES** - Open the door
    - **NO** - Get immediately shot by guard

ATM Machine Protocol

1. Insert ATM card
2. Enter PIN
3. Is the pin correct?

Authentication Protocols - Examples

Enter the NSA

1. Insert badge into reader
2. Enter PIN
3. Is the pin correct?
    - **YES** - Open the door
    - **NO** - Get immediately shot by guard

ATM Machine Protocol

1. Insert ATM card
2. Enter PIN
3. Is the pin correct?
    - **YES** - Perform transaction

Authentication Protocols - Examples

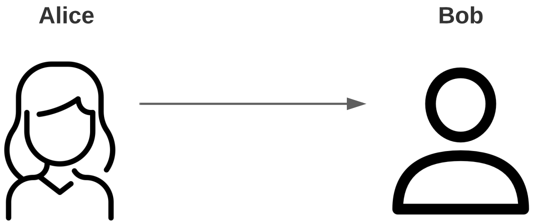
Enter the NSA

1. Insert badge into reader
2. Enter PIN
3. Is the pin correct?
   - **YES** - Open the door
   - **NO** - Get immediately shot by guard

ATM Machine Protocol

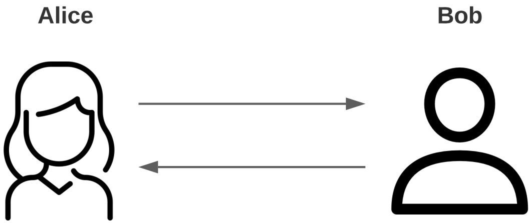
1. Insert ATM card
2. Enter PIN
3. Is the pin correct?
   - **YES** - Perform transaction
   - **NO** - Machine eats your card (eventually)

Authentication Protocols - For real now



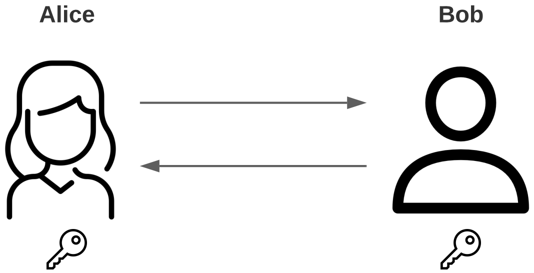
- Alice must prove her identity to Bob
    - They can be humans or computers

Authentication Protocols - For real now



- Alice must prove her identity to Bob
  - They can be humans or computers
- May also require Bob to prove its identity to Alice
  - A.k.a. mutual authentication

Authentication Protocols - For real now



- Alice must prove her identity to Bob
  - They can be humans or computers
- May also require Bob to prove its identity to Alice
  - A.k.a. mutual authentication
- Often entails establishing a session key
  - For cryptographic purposes

Authentication Protocols - Challenges

Stand-alone computer

Relatively simple

Authentication Protocols - Challenges

Stand-alone computer

Relatively simple

- Hash a password with a salt

Authentication Protocols - Challenges

Stand-alone computer

Relatively simple

- Hash a password with a salt
- (relatively reduced) attack surface

Authentication Protocols - Challenges

Stand-alone computer

Relatively simple

- Hash a password with a salt
- (relatively reduced) attack surface
- Careful with keystroke logging!

Authentication Protocols - Challenges

Stand-alone computer

Relatively simple

- Hash a password with a salt
- (relatively reduced) attack surface
- Careful with keystroke logging!

Over the network

Much more challenging

Authentication Protocols - Challenges

Stand-alone computer

Relatively simple

- Hash a password with a salt
- (relatively reduced) attack surface
- Careful with keystroke logging!

Over the network

Much more challenging

- Adversary can passively observe messages

Authentication Protocols - Challenges

Stand-alone computer

Relatively simple

- Hash a password with a salt
- (relatively reduced) attack surface
- Careful with keystroke logging!

Over the network

Much more challenging

- Adversary can passively observe messages
- Adversary can replay messages

Authentication Protocols - Challenges

Stand-alone computer

Relatively simple

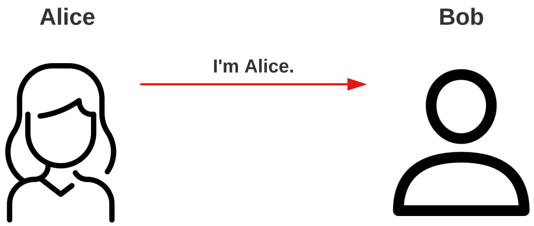
- Hash a password with a salt
- (relatively reduced) attack surface
- Careful with keystroke logging!

Over the network

Much more challenging

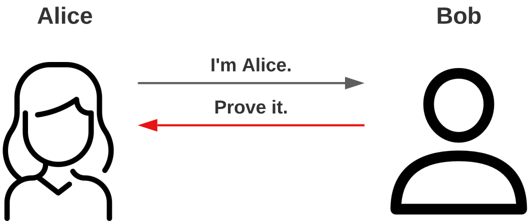
- Adversary can passively observe messages
- Adversary can replay messages
- Active attacks possible (insert, delete, change)

A Naive Authentication Protocol



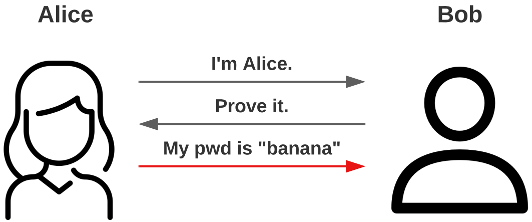
- Simple and (perhaps) OK for a stand-alone machine

A Naive Authentication Protocol



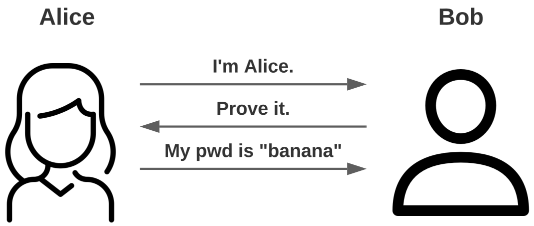
- Simple and (perhaps) OK for a stand-alone machine

A Naive Authentication Protocol



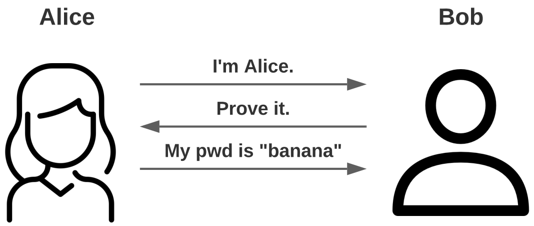
- Simple and (perhaps) OK for a stand-alone machine

A Naive Authentication Protocol



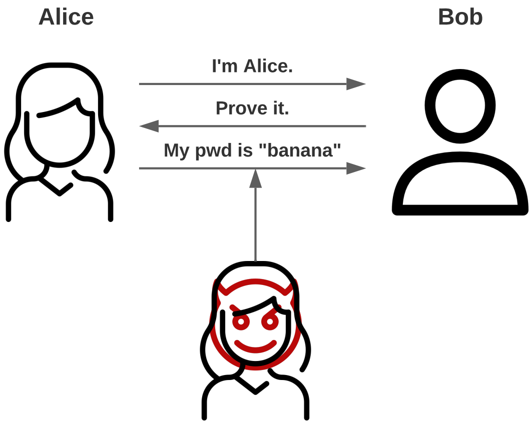
- Simple and (perhaps) OK for a stand-alone machine
- Highly insecure for a networked system

A Naive Authentication Protocol

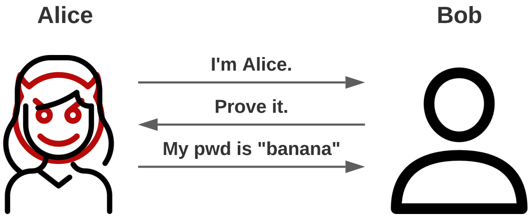


- Simple and (perhaps) OK for a stand-alone machine
- Highly insecure for a networked system
    - Subject to replay attacks
    - Bob must know Alice's password (explicitly)

The network is insecure!

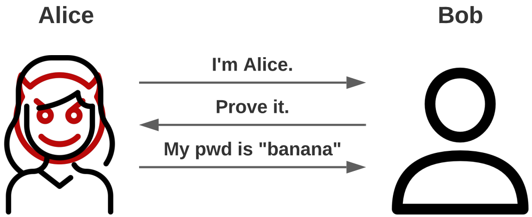


Authentication Attack



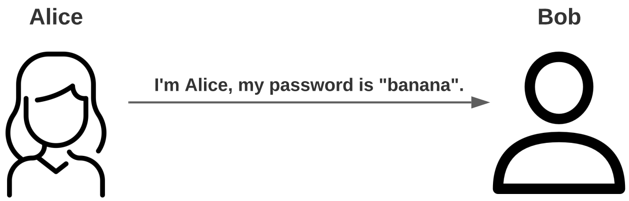
- This is an example of a **replay** attack
  - The adversary observed the interaction
  - Used the messages to repeat a communication pattern

Authentication Attack



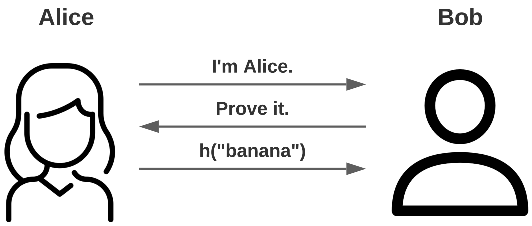
- This is an example of a **replay** attack
  - The adversary observed the interaction
  - Used the messages to repeat a communication pattern
- How can we prevent replay attacks?

A (still pretty) Naive Authentication Protocol



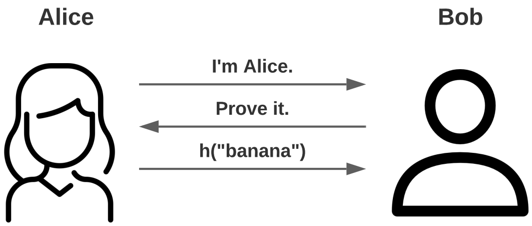
- More efficient
- But doesn't solve the replay attack problems!

Hiding the Exact Password



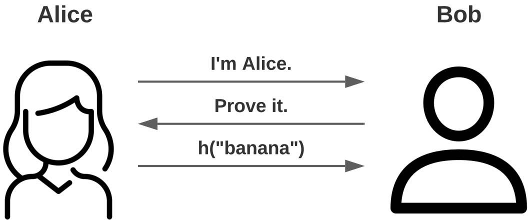
- This approach hides Alice's password

Hiding the Exact Password



- This approach hides Alice's password
  - From both Bob, and the adversary!

Hiding the Exact Password



- This approach hides Alice’s password
  - From both Bob, and the adversary!
- But it’s still subject to replay attacks...

Freshness: challenge-response

To prevent replay, we leverage a technique called **challenge-response**

Freshness: challenge-response

To prevent replay, we leverage a technique called **challenge-response**

- Suppose Bob wants to authenticate Alice (our setting)
- Bob sends a *challenge* to Alice
- Alice must respond to the *challenge* according to its password

Freshness: challenge-response

To prevent replay, we leverage a technique called **challenge-response**

- Suppose Bob wants to authenticate Alice (our setting)
- Bob sends a *challenge* to Alice
- Alice must respond to the *challenge* according to its password

Challenge

Challenge is chosen such that...

- Replay is not possible
- Only Alice can provide the correct response
- Bob can (efficiently) verify the response

Freshness: nonce

Nonce: A **n**umber that is only used **once.**

Freshness: nonce

Nonce: A **n**umber that is only used **once.**

Design choices

- How can we choose the nonce?

Freshness: nonce

Nonce: A **n**umber that is only used **once.**

Design choices

- How can we choose the nonce?
- How should Alice work with the nonce?

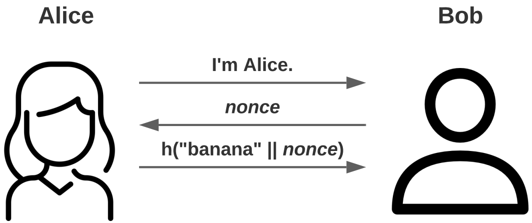
Freshness: nonce

Nonce: A **n**umber that is only used **once.**

Design choices

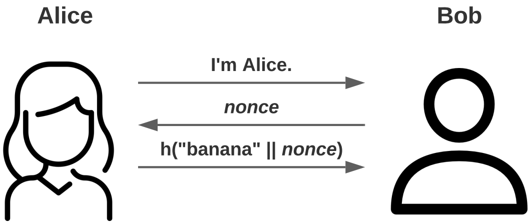
- How can we choose the nonce?
- How should Alice work with the nonce?
- How can Bob verify the nonce-enhanced response?

Using Nonces



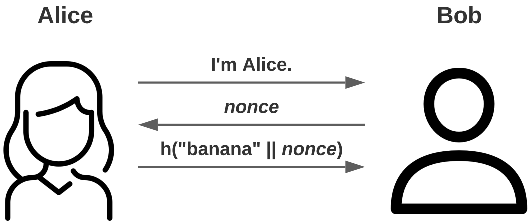
- Nonce is the challenge
  - Every request for authentication must use a different nonce

Using Nonces



- Nonce is the challenge
  - Every request for authentication must use a different nonce
- The hash is the response
  - The message used for the first authentication will not work for any of the following ones
  - Collision-resistant hashes!

Using Nonces



- Nonce is the challenge
  - Every request for authentication must use a different nonce
- The hash is the response
  - The message used for the first authentication will not work for any of the following ones
  - Collision-resistant hashes!
- Bob must know Alice's pwd to verify.

Password not Necessary

An alternative design

- Alice and Bob share symmetric key k

Password not Necessary

An alternative design

- Alice and Bob share symmetric key k
- *Assumption:* k is known only to Alice and Bob

Password not Necessary

An alternative design

- Alice and Bob share symmetric key k
- *Assumption:* k is known only to Alice and Bob
- Authentication requires proving knowledge of k

Password not Necessary

An alternative design

- Alice and Bob share symmetric key k
- *Assumption:* k is known only to Alice and Bob
- Authentication requires proving knowledge of k
- How can we accomplish this?
  - Key cannot be sent over the network (obviously)
  - Replay attacks cannot be successful

Password not Necessary

An alternative design

- Alice and Bob share symmetric key k
- *Assumption:* k is known only to Alice and Bob
- Authentication requires proving knowledge of k
- How can we accomplish this?
  - Key cannot be sent over the network (obviously)
  - Replay attacks cannot be successful

Recall crypto notation!

- Protect message m, using key k to produce ciphertext c

Password not Necessary

An alternative design

- Alice and Bob share symmetric key k
- *Assumption:* k is known only to Alice and Bob
- Authentication requires proving knowledge of k
- How can we accomplish this?
  - Key cannot be sent over the network (obviously)
  - Replay attacks cannot be successful

Recall crypto notation!

- Protect message m, using key k to produce ciphertext c
- Encryption: $c = \text{Encrypt}(k, m)$

Password not Necessary

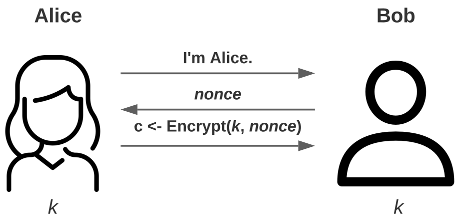
An alternative design

- Alice and Bob share symmetric key k
- *Assumption:* k is known only to Alice and Bob
- Authentication requires proving knowledge of k
- How can we accomplish this?
  - Key cannot be sent over the network (obviously)
  - Replay attacks cannot be successful

Recall crypto notation!

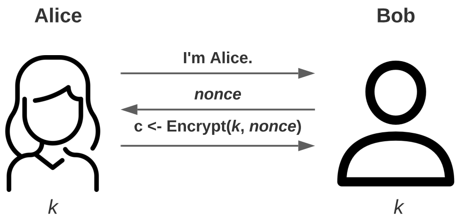
- Protect message m, using key k to produce ciphertext c
- Encryption: $c = \text{Encrypt}(k, m)$
- Decryption: $m = \text{Decrypt}(k, c)$

Using Symmetric Keys



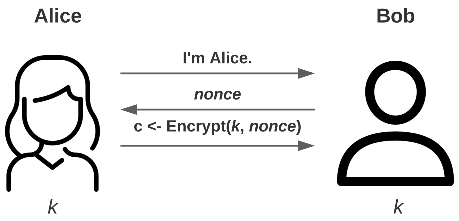
- Bob can authenticate Alice using k
  - Decrypt(k, c)
  - *Assumption:* Knowledge of k is necessary to create a valid c
  - If c decrypts to *nonce*, then we must be talking to Alice!

Using Symmetric Keys



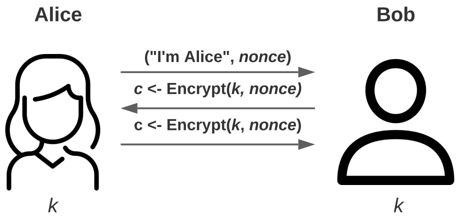
- Bob can authenticate Alice using k
  - Decrypt(k, c)
  - *Assumption:* Knowledge of k is necessary to create a valid c
  - If c decrypts to *nonce*, then we must be talking to Alice!
- Alice does not authenticate Bob...

Using Symmetric Keys



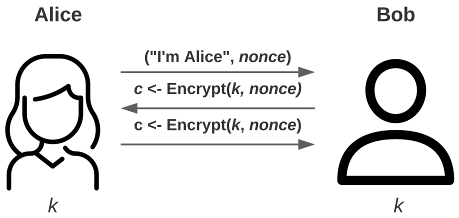
- Bob can authenticate Alice using k
  - Decrypt(k, c)
  - *Assumption:* Knowledge of k is necessary to create a valid c
  - If c decrypts to *nonce*, then we must be talking to Alice!
- Alice does not authenticate Bob...
- How can we get mutual authentication?

Using Symmetric Keys - Mutual Authentication



- Works... right?

Using Symmetric Keys - Mutual Authentication



- Works... right?
- Anyone can repeat *c* to “prove” they are Alice.

Mutual Authentication

We have a secure one-way authentication protocol!

Mutual Authentication

We have a secure one-way authentication protocol!

One-way to Mutual

So all we have to do is do the protocol twice:

- The first time is used for Bob to authenticate Alice
- The second is used for Alice to authenticate Bob

Mutual Authentication

We have a secure one-way authentication protocol!

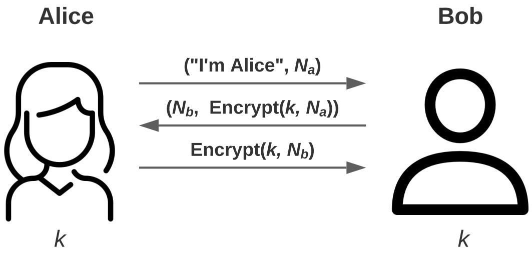
One-way to Mutual

So all we have to do is do the protocol twice:

- The first time is used for Bob to authenticate Alice
- The second is used for Alice to authenticate Bob

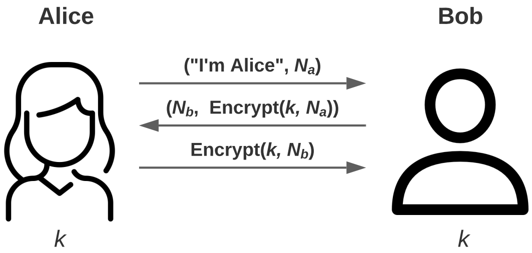
What could possibly go wrong?

Mutual Authentication - Naive Protocol



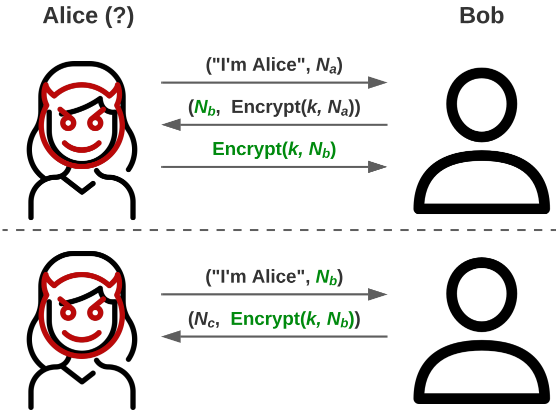
- Ok, now we have mutual authentication!

Mutual Authentication - Naive Protocol



- Ok, now we have mutual authentication!
- Actually...
  - Subject to **reflection** attack

Mutual Authentication - Reflection Attack



Mutual Authentication - Lessons Learned

One-way to Mutual

One-way authentication protocol is **not** secure for mutual auth

Mutual Authentication - Lessons Learned

One-way to Mutual

One-way authentication protocol is **not** secure for mutual auth

- Subtle errors (told you!)
- In this case, the obvious solution is not secure

Mutual Authentication - Lessons Learned

One-way to Mutual

One-way authentication protocol is **not** secure for mutual auth

- Subtle errors (told you!)
- In this case, the obvious solution is not secure

- Problem is related to the fact that the response is not bound to a specific protocol execution

Mutual Authentication - Lessons Learned

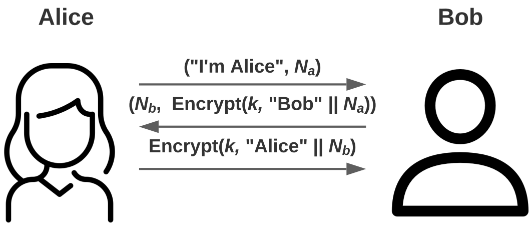
One-way to Mutual

One-way authentication protocol is **not** secure for mutual auth

- Subtle errors (told you!)
- In this case, the obvious solution is not secure

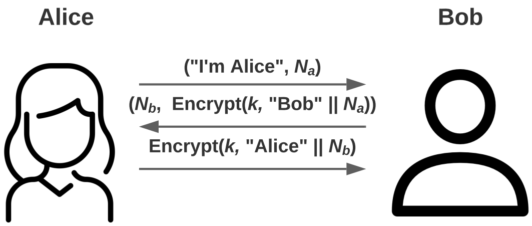
- Problem is related to the fact that the response is not bound to a specific protocol execution
- How can we enforce this one-to-one relation between responses, challenges and identities?

Mutual Authentication - Reflection-proof


Alice
Bob
("I'm Alice", N_a)
(N_b, Encrypt(k, "Bob" || N_a))
Encrypt(k, "Alice" || N_b)

- Responses are bound to their respective identities

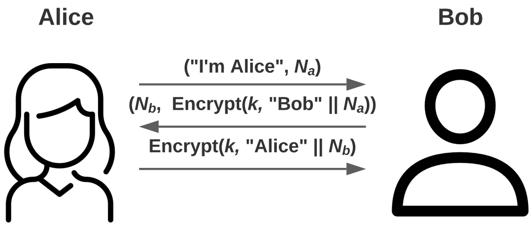
Mutual Authentication - Reflection-proof



Alice
Bob
("I'm Alice", N_a)
(N_b, Encrypt(k, "Bob" || N_a))
Encrypt(k, "Alice" || N_b)

- Responses are bound to their respective identities
- Does this help?

Mutual Authentication - Reflection-proof


Alice
Bob
("I'm Alice", N_a)
(N_b, Encrypt(k, "Bob" || N_a))
Encrypt(k, "Alice" || N_b)

- Responses are bound to their respective identities
- Does this help?
- Apparently so...
  - At least for that type of attacks!

Using Public-key Cryptography

What if we use asymmetric keys?

- Alice has secret key *sk* and Bob knows public key *pk*

Using Public-key Cryptography

What if we use asymmetric keys?

- Alice has secret key *sk* and Bob knows public key *pk*
- *Assumption: sk* is known only to Alice

Using Public-key Cryptography

What if we use asymmetric keys?

- Alice has secret key *sk* and Bob knows public key *pk*
- *Assumption: sk* is known only to Alice
- Authentication requires proving knowledge of *sk*

Using Public-key Cryptography

What if we use asymmetric keys?

- Alice has secret key *sk* and Bob knows public key *pk*
- *Assumption: sk* is known only to Alice
- Authentication requires proving knowledge of *sk*
- How can we accomplish this?
  - Decrypt something with *sk*
  - Sign something with *sk*

Using Public-key Cryptography

What if we use asymmetric keys?

- Alice has secret key *sk* and Bob knows public key *pk*
- *Assumption: sk* is known only to Alice
- Authentication requires proving knowledge of *sk*
- How can we accomplish this?
  - Decrypt something with *sk*
  - Sign something with *sk*

Recall crypto notation!

- Encryption: $c = \text{Encrypt}(pk, m)$
- Decryption: $m = \text{Decrypt}(sk, c)$

Using Public-key Cryptography

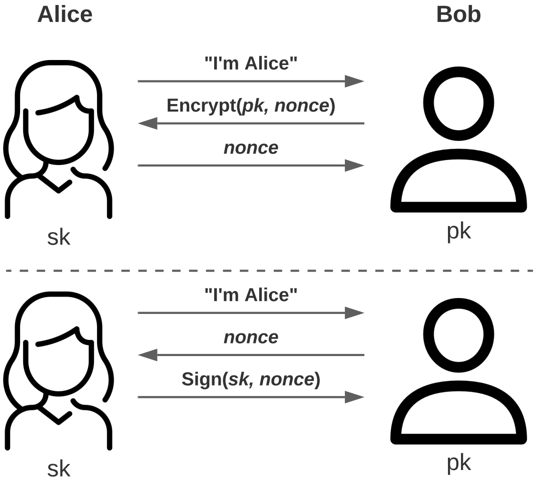
What if we use asymmetric keys?

- Alice has secret key *sk* and Bob knows public key *pk*
- *Assumption: sk* is known only to Alice
- Authentication requires proving knowledge of *sk*
- How can we accomplish this?
  - Decrypt something with *sk*
  - Sign something with *sk*

Recall crypto notation!

- Encryption: $c = \text{Encrypt}(pk, m)$
- Decryption: $m = \text{Decrypt}(sk, c)$
- Signature: $s = \text{Sign}(sk, m)$
- Verification: $T/F = \text{Verify}(pk, m, s)$

Mutual Authentication - PKC



Caution with keys...

It’s (generally) a bad idea to use the same key pair for encryption and signing.

- Keys get lost
- Keys get compromised

Caution with keys...

It's (generally) a bad idea to use the same key pair for encryption and signing.

- Keys get lost
- Keys get compromised

Good practices

- Use one key pair for encryption/decryption or signing/verification
- Use another for authentication

Establish a secure channel

Session key

- A symmetric key for each session

Establish a secure channel

Session key

- A symmetric key for each session
  - Ephemeral
  - Used for confidentiality and/or integrity

Establish a secure channel

Session key

- A symmetric key for each session
  - Ephemeral
  - Used for confidentiality and/or integrity

Purpose

- Alice and Bob use session key to encrypt/authenticate data

Establish a secure channel

Session key

- A symmetric key for each session
  - Ephemeral
  - Used for confidentiality and/or integrity

Purpose

- Alice and Bob use session key to encrypt/authenticate data
- The adversary cannot determine the session key

Establish a secure channel

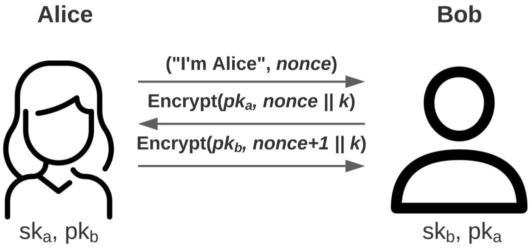
Session key

- A symmetric key for each session
  - Ephemeral
  - Used for confidentiality and/or integrity

Purpose

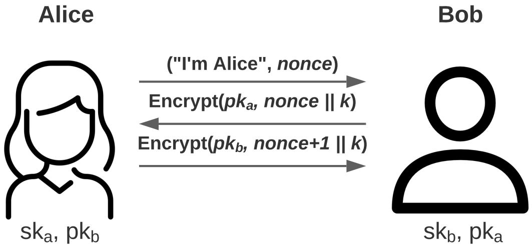
- Alice and Bob use session key to encrypt/authenticate data
- The adversary cannot determine the session key
- Limit amount of data encrypted with a given key
  - Limit damage (data loss) if one key is corrupted

PKC Mutual Authentication - Attempt #1



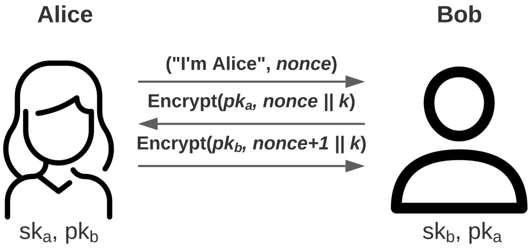
Alice
Bob
("I'm Alice", *nonce*)
Encrypt(*pk*$_a$, *nonce* || *k*)
Encrypt(*pk*$_b$, *nonce*+1 || *k*)
sk$_a$, pk$_b$
sk$_b$, pk$_a$

PKC Mutual Authentication - Attempt #1


Alice
Bob
("I'm Alice", *nonce*)
Encrypt(*pk$_a$*, *nonce* || *k*)
Encrypt(*pk$_b$*, *nonce*+1 || *k*)
sk$_a$, pk$_b$
sk$_b$, pk$_a$

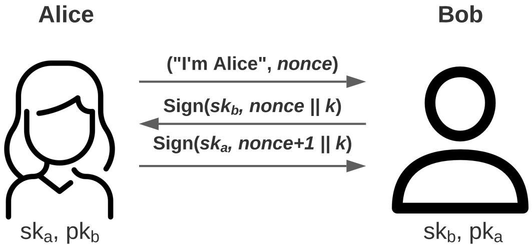
- Session key is secure

PKC Mutual Authentication - Attempt #1

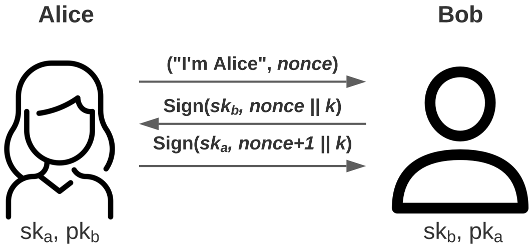

Alice
Bob
("I'm Alice", *nonce*)
Encrypt(pk_a, *nonce* || *k*)
Encrypt(pk_b, *nonce*+1 || *k*)
sk_a, pk_b
sk_b, pk_a

- Session key is secure
- Only Alice is authenticated...

PKC Mutual Authentication - Attempt #2

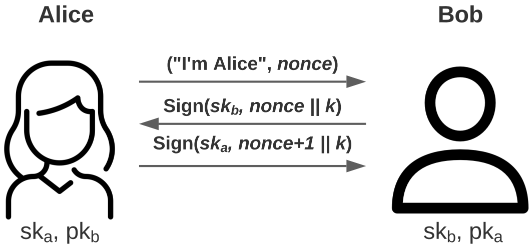


PKC Mutual Authentication - Attempt #2



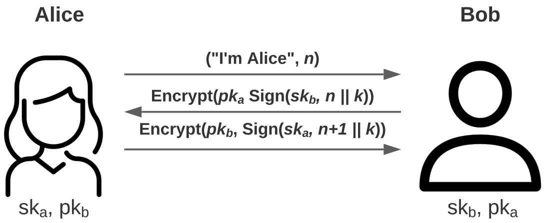
- Mutual authentication is ensured

PKC Mutual Authentication - Attempt #2

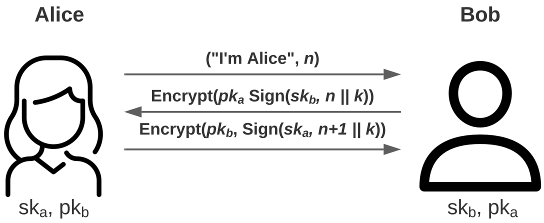


- Mutual authentication is ensured
- But the key is not protected at all!

PKC Mutual Authentication - Attempt #3

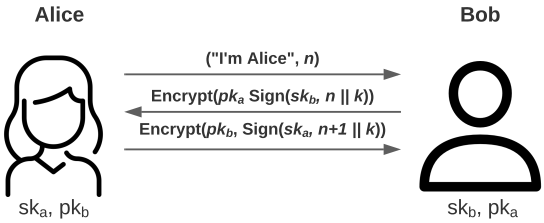

Alice
Bob
("I'm Alice", n)
Encrypt(pk_a Sign(sk_b, n || k))
Encrypt(pk_b, Sign(sk_a, $n+1$ || k))
sk_a, pk_b
sk_b, pk_a

PKC Mutual Authentication - Attempt #3


Alice
Bob
("I'm Alice", n)
Encrypt(pk_a Sign(sk_b, n || k))
Encrypt(pk_b, Sign(sk_a, $n+1$ || k))
sk_a, pk_b
sk_b, pk_a

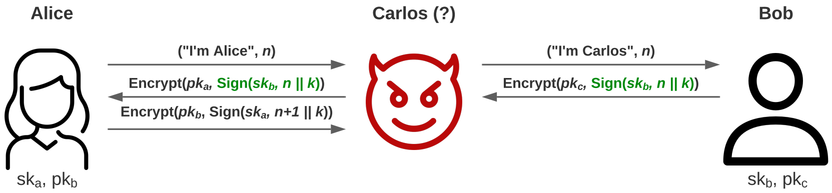
- Is this secure?

PKC Mutual Authentication - Attempt #3


Alice
Bob
("I'm Alice", n)
Encrypt(pk_a Sign(sk_b, n || k))
Encrypt(pk_b, Sign(sk_a, $n+1$ || k))
sk_a, pk_b
sk_b, pk_a

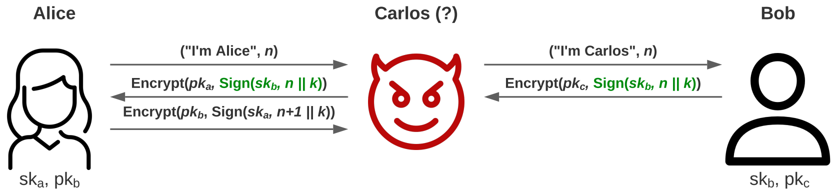
- Is this secure?
- Vulnerable to a MitM attack!

MitM attack



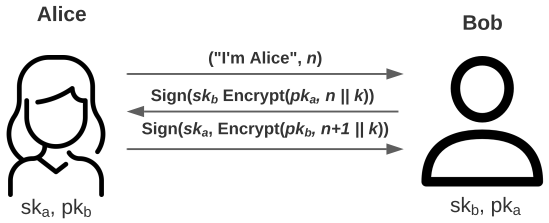
Alice
Carlos (?)
Bob
("I'm Alice", n)
Encrypt(pk_a, Sign(sk_b, n || k))
Encrypt(pk_b, Sign(sk_a, $n+1$ || k))
("I'm Carlos", n)
Encrypt(pk_c, Sign(sk_b, n || k))
sk_a, pk_b
sk_b, pk_c

MitM attack

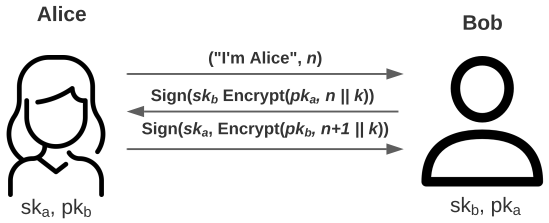

Alice
Carlos (?)
Bob
("I'm Alice", n)
Encrypt(pk_a, Sign(sk_b, n || k))
Encrypt(pk_b, Sign(sk_a, n+1 || k))
("I'm Carlos", n)
Encrypt(pk_c, Sign(sk_b, n || k))
sk_a, pk_b
sk_b, pk_c

- Carlos can get the session key in the clear
- Alice is convinced to use the same session key
- But she isn't talking to Bob!

PKC Mutual Authentication - Attempt #4

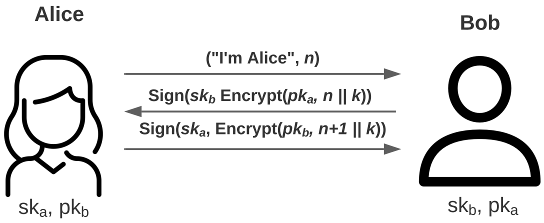

Alice
Bob
("I'm Alice", n)
Sign(sk_b Encrypt(pk_a, n || k))
Sign(sk_a, Encrypt(pk_b, $n+1$ || k))
sk_a, pk_b
sk_b, pk_a

PKC Mutual Authentication - Attempt #4


Alice
Bob
("I'm Alice", n)
Sign(sk_b Encrypt(pk_a, n || k))
Sign(sk_a, Encrypt(pk_b, $n+1$ || k))
sk_a, pk_b
sk_b, pk_a

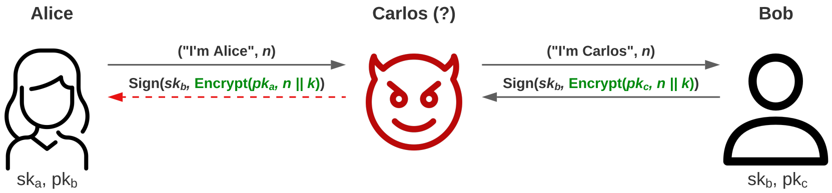
- Is this secure?

PKC Mutual Authentication - Attempt #4


Alice
Bob
("I'm Alice", n)
Sign(sk_b Encrypt(pk_a, n || k))
Sign(sk_a, Encrypt(pk_b, $n+1$ || k))
sk_a, pk_b
sk_b, pk_a

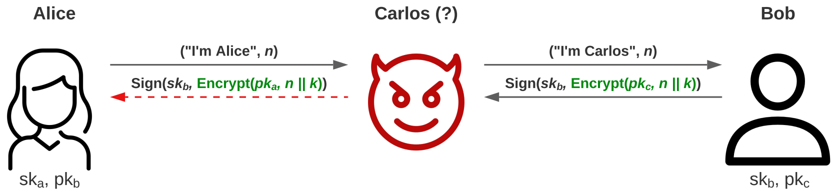
- Is this secure?
- All seems ok!

MitM attack?



- Carlos only has to send a valid signature to convince Alice

MitM attack?



- Carlos only has to send a valid signature to convince Alice
- ... but without knowledge of sk_b, it cannot produce it!

Network Security - Week 3

Manuel E. Correia

DCC/FCUP

2025