A Method for Password-free Authentication

Mirko Oleszuk

Institut für Informatik
Heinrich-Heine-Universität Düsseldorf

19. December 2013

Introduction

- Passwords are ...!

Introduction

- Passwords are ...!
  - short, easy and used multiple times

Introduction

- Passwords are ...!
  - short, easy and used multiple times
- remembering multiple passwords is not comfortable

Introduction

- Passwords are ...!
  - short, easy and used multiple times
- remembering multiple passwords is not comfortable
- Passwords are stored in plaintext (sometimes)

Introduction

- Passwords are ...!
  - short, easy and used multiple times
- remembering multiple passwords is not comfortable
- Passwords are stored in plaintext (sometimes)
- ⇒ We need a solution without passwords!

Introduction

- Passwords are ...!
  - short, easy and used multiple times
- remembering multiple passwords is not comfortable
- Passwords are stored in plaintext (sometimes)
- ⇒ We need a solution without passwords!
  - free, easy, secure

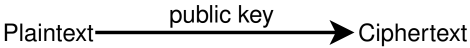
The RSA-Cryptosystem

- every user has a key-pair
  - public key
  - private key

The RSA-Cryptosystem

- every user has a key-pair
  - public key
  - private key

Encryption:



Plaintext

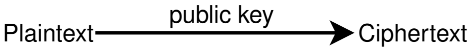
public key

Ciphertext

The RSA-Cryptosystem

- every user has a key-pair
  - public key
  - private key

Encryption:

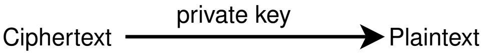


Plaintext

public key

Ciphertext

Decryption:



Ciphertext

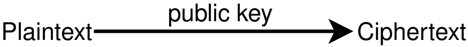
private key

Plaintext

The RSA-Cryptosystem

- every user has a key-pair
  - public key
  - private key

Encryption:

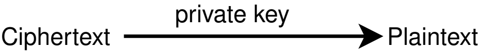


Plaintext

public key

Ciphertext

Decryption:



Ciphertext

private key

Plaintext

- asymmetric

The RSA-Cryptosystem

- every user has a key-pair
  - public key
  - private key

Signing:

Plaintext $\xrightarrow{\text{private key}}$ „Ciphertext“ (signed plaintext)

Verification:

„Ciphertext“ $\xrightarrow{\text{public key}}$ Plaintext

The RSA-Cryptosystem

- every user has a key-pair
  - public key
  - private key

Signing:

Plaintext $\xrightarrow{\text{private key}}$ „Ciphertext“ (signed plaintext)

Verification:

„Ciphertext“ $\xrightarrow{\text{public key}}$ Plaintext

- asymmetric $\equiv$ Public-Key-Cryptography

Overview & Preparation

Idea

- signed data identifys the unique user

Overview & Preparation

Idea

- signed data identifys the unique user

The Client

- has a private key
- has a user name

Overview & Preparation

Idea

- signed data identifys the unique user

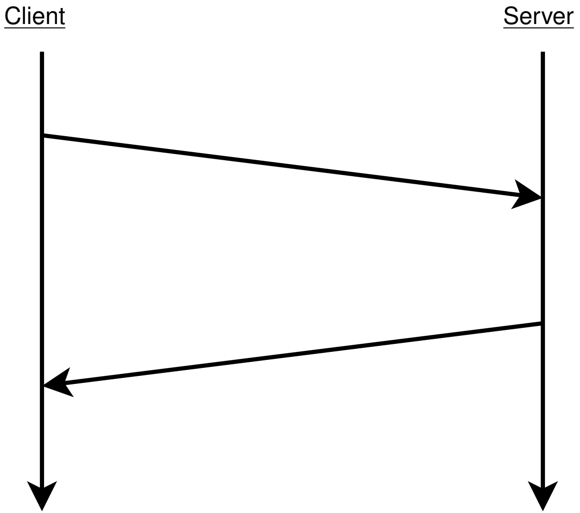
The Client

- has a private key
- has a user name

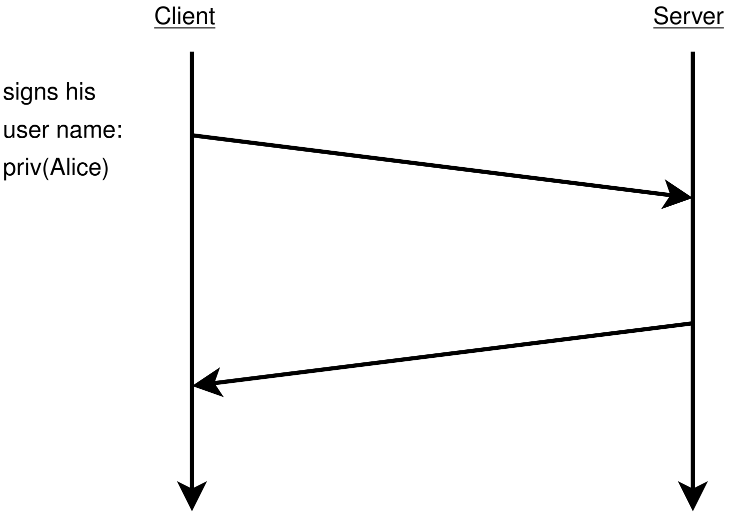
The Server

- has the public key of the client
- and it is associated with the user account

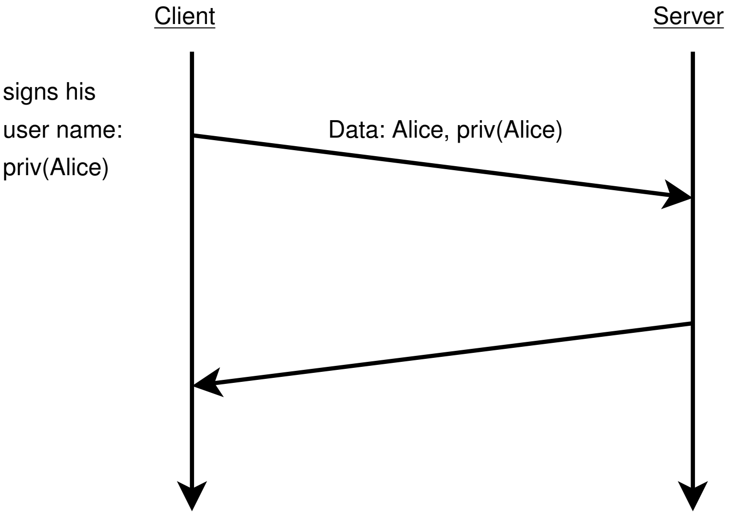
PKWL 1.0 - Signed Messages



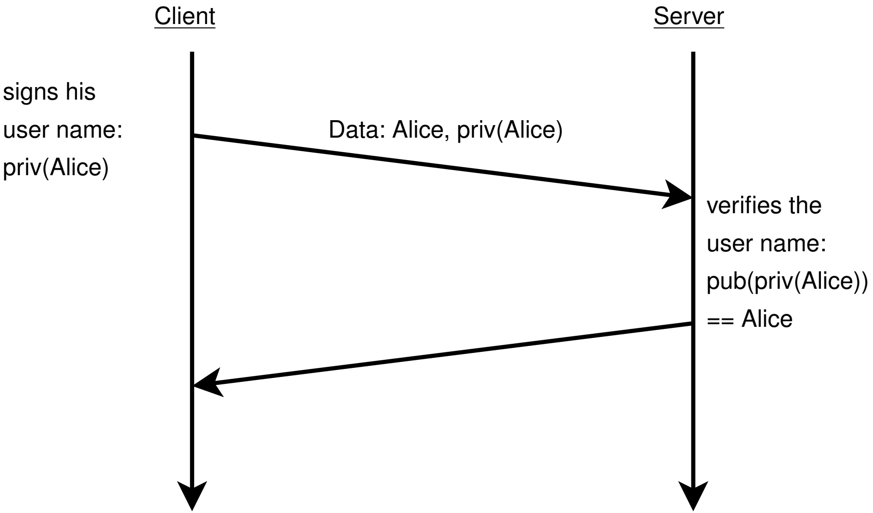
PKWL 1.0 - Signed Messages



PKWL 1.0 - Signed Messages

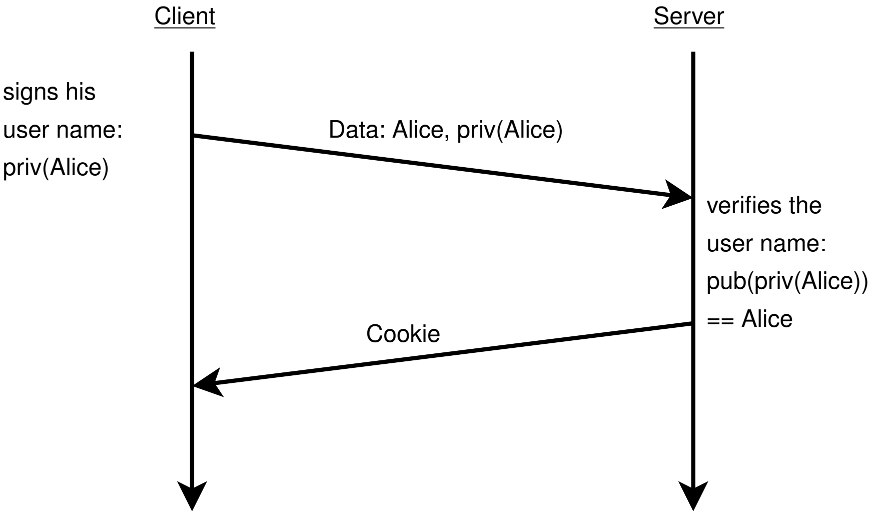


PKWL 1.0 - Signed Messages


Client
Server
signs his user name: priv(Alice)
Data: Alice, priv(Alice)
verifies the user name: pub(priv(Alice)) == Alice

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

PKWL 1.0 - Signed Messages



Client

Server

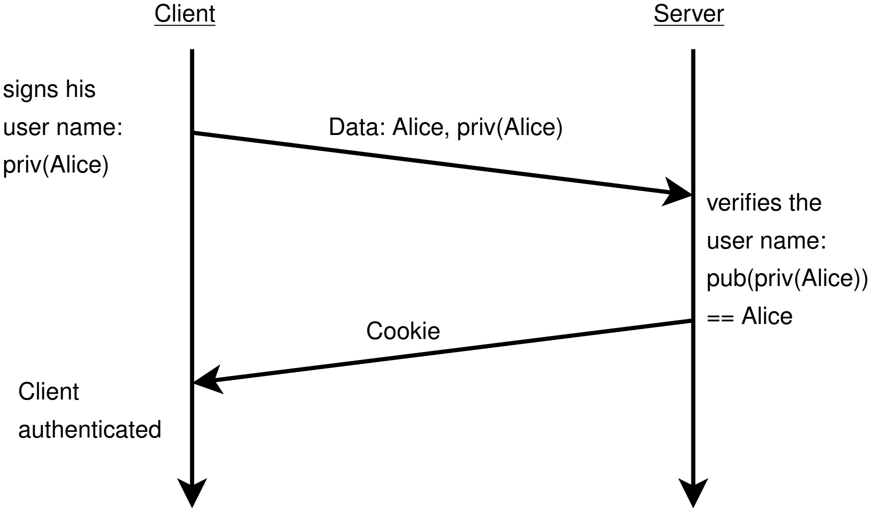
signs his user name: priv(Alice)

Data: Alice, priv(Alice)

verifies the user name: pub(priv(Alice)) == Alice

Cookie

PKWL 1.0 - Signed Messages



Client

Server

signs his user name: priv(Alice)

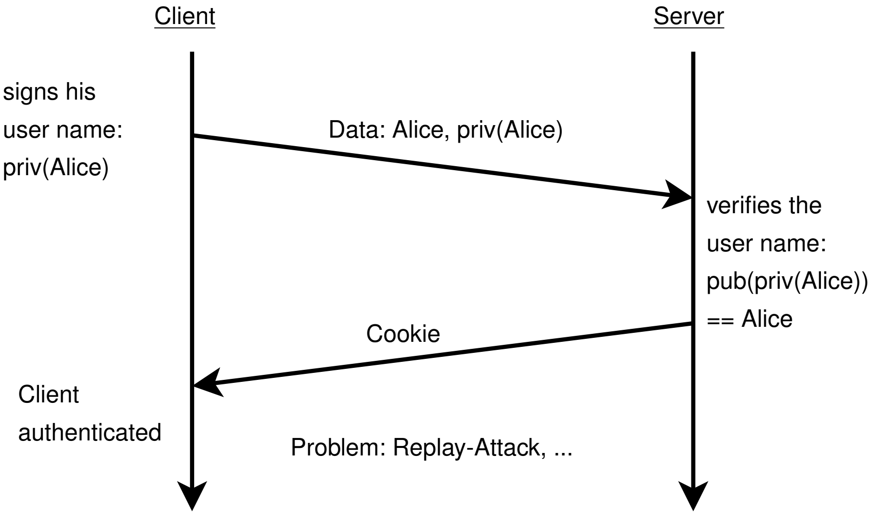
Data: Alice, priv(Alice)

verifies the user name: pub(priv(Alice)) == Alice

Cookie

Client authenticated

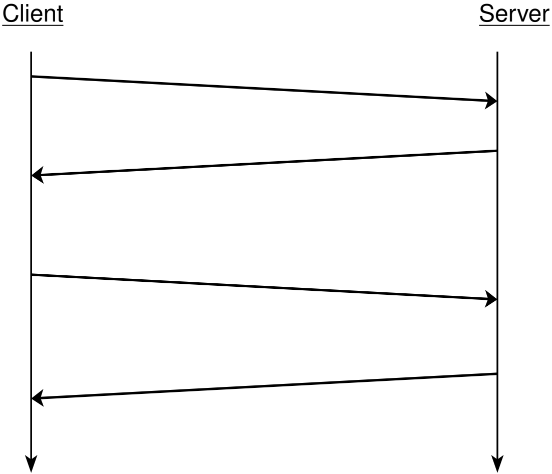
PKWL 1.0 - Signed Messages



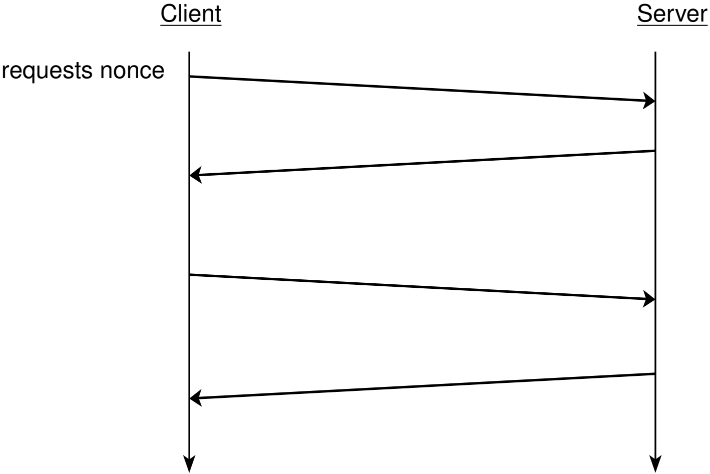
Client
Server
signs his user name: priv(Alice)
Data: Alice, priv(Alice)
verifies the user name: pub(priv(Alice)) == Alice
Cookie
Client authenticated

Problem: Replay-Attack, ...

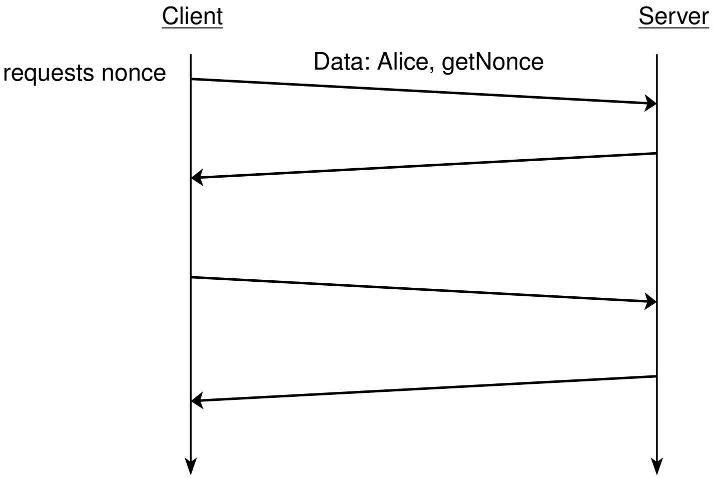
PKWL 1.1 - Signed Nonces



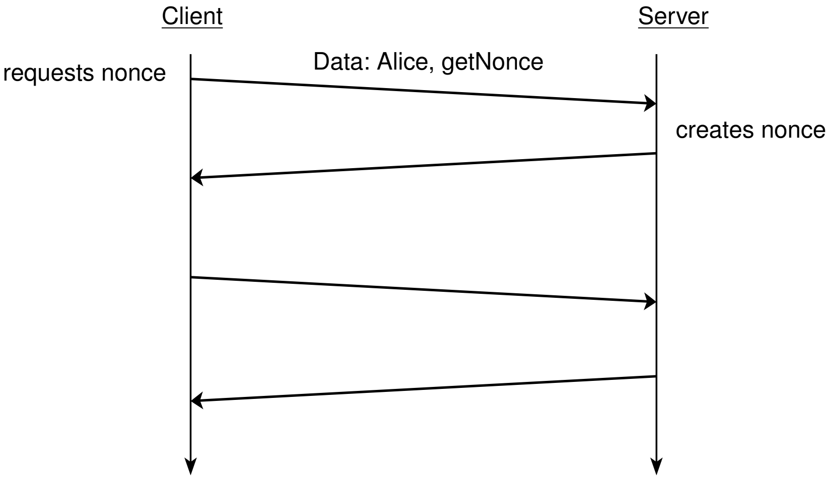
PKWL 1.1 - Signed Nonces



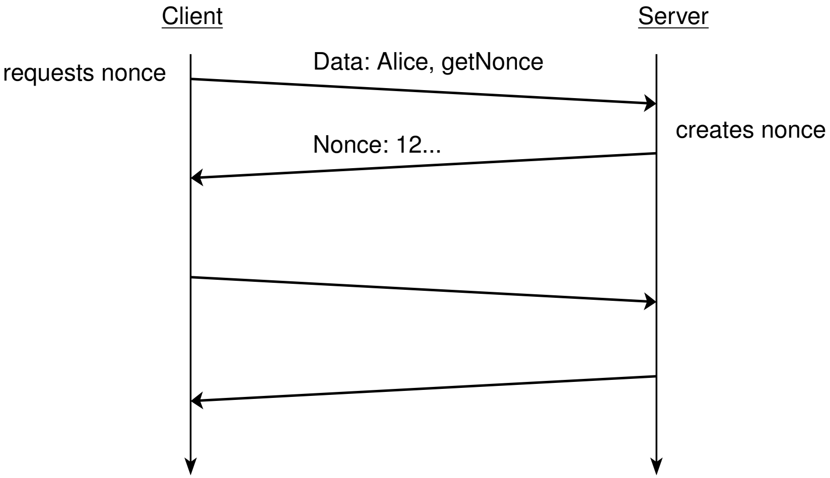
PKWL 1.1 - Signed Nonces



PKWL 1.1 - Signed Nonces



PKWL 1.1 - Signed Nonces



Client

Server

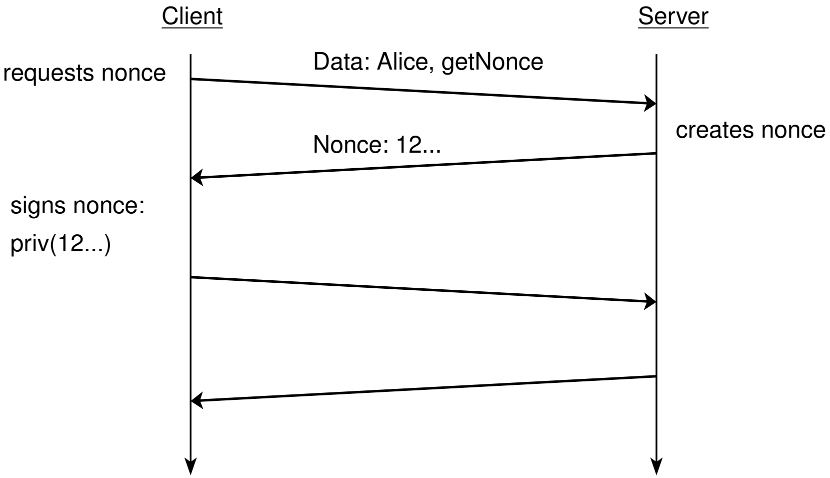
requests nonce

Data: Alice, getNonce

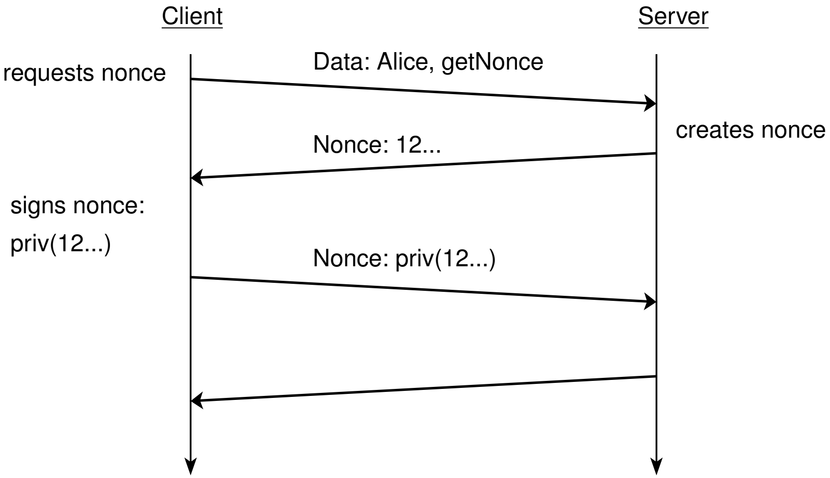
creates nonce

Nonce: 12...

PKWL 1.1 - Signed Nonces



PKWL 1.1 - Signed Nonces



Client

Server

requests nonce

Data: Alice, getNonce

creates nonce

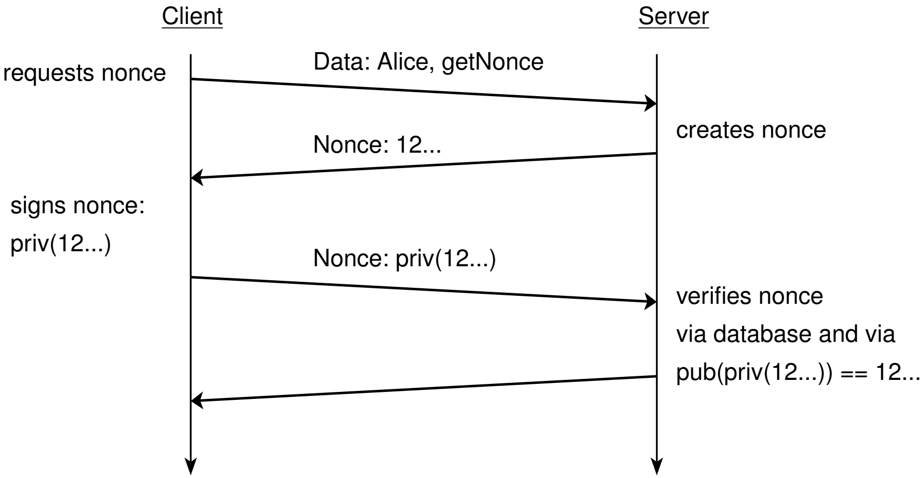
Nonce: 12...

signs nonce:

priv(12...)

Nonce: priv(12...)

PKWL 1.1 - Signed Nonces



Client

Server

requests nonce

Data: Alice, getNonce

creates nonce

Nonce: 12...

signs nonce:

priv(12...)

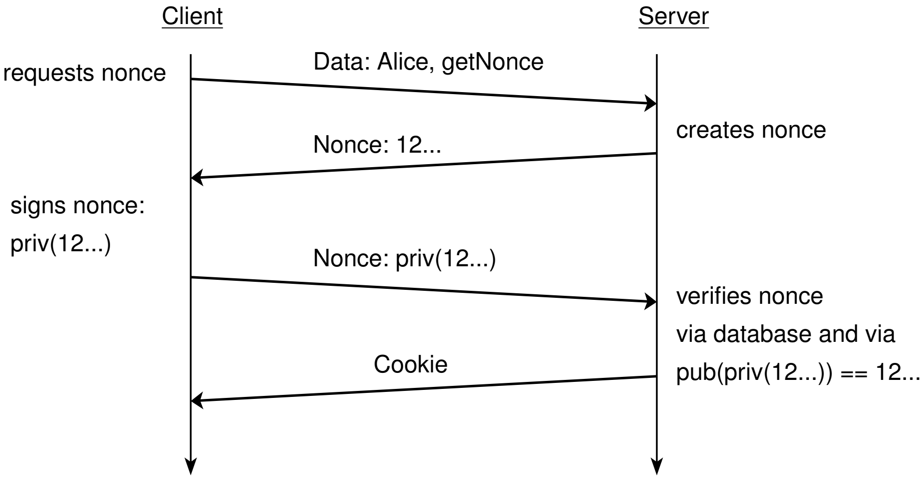
Nonce: priv(12...)

verifies nonce

via database and via

pub(priv(12...)) == 12...

PKWL 1.1 - Signed Nonces



Client

Server

requests nonce

Data: Alice, getNonce

creates nonce

Nonce: 12...

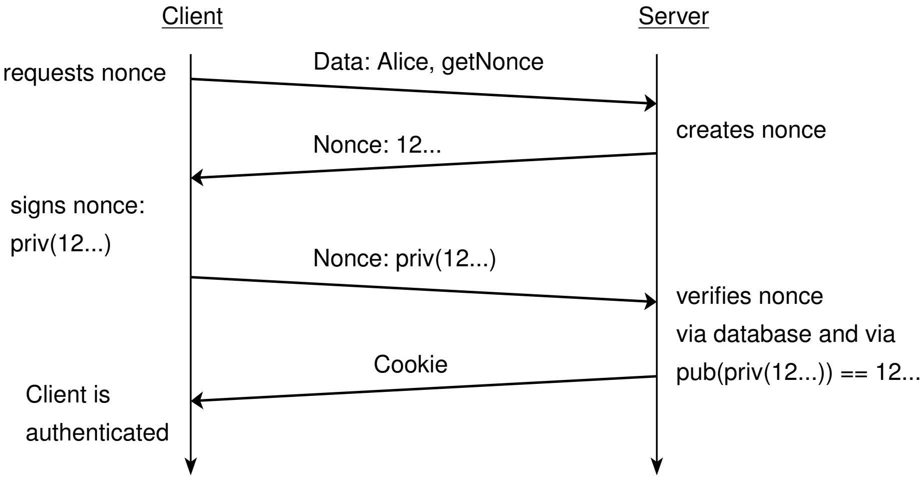
signs nonce:
priv(12...)

Nonce: priv(12...)

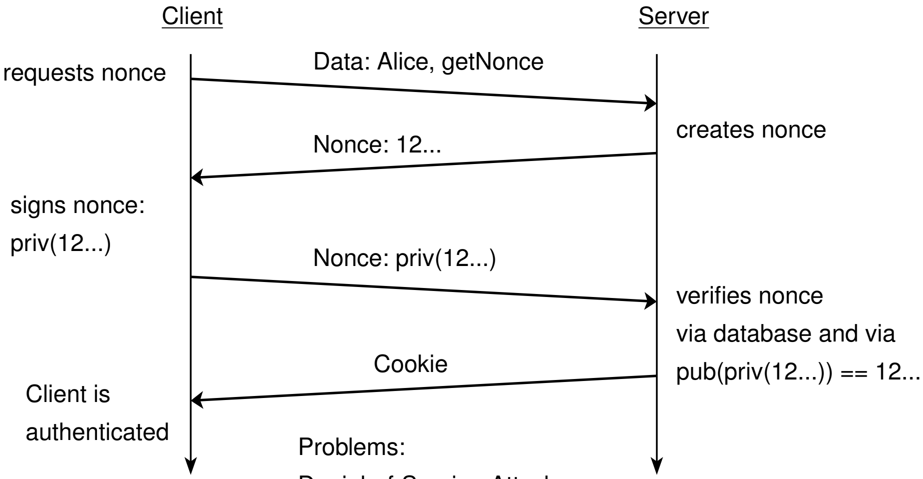
verifies nonce
via database and via
pub(priv(12...)) == 12...

Cookie

PKWL 1.1 - Signed Nonces

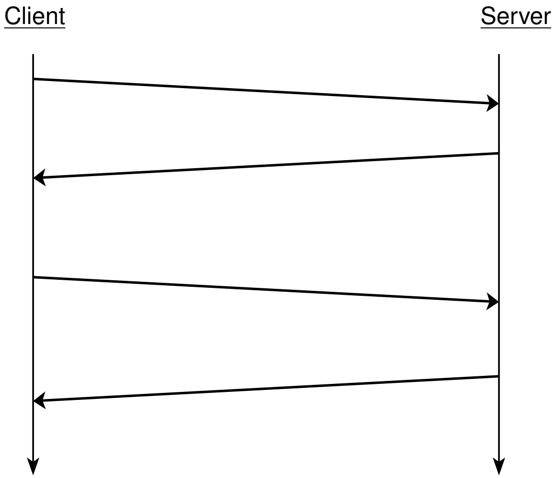


PKWL 1.1 - Signed Nonces

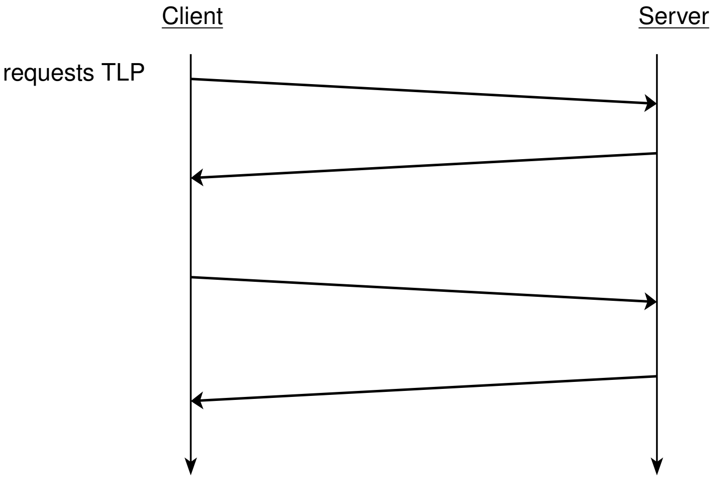


Problems:
Denial-of-Service-Attack,
malicious nonce, ...

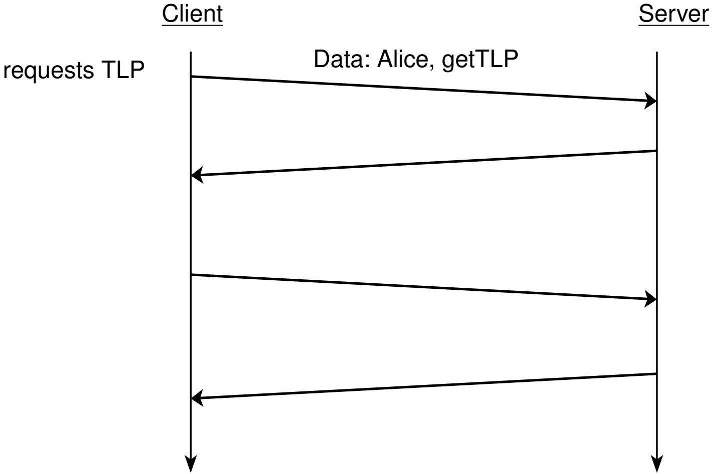
PKWL 2.0 - Proof-of-Work-System



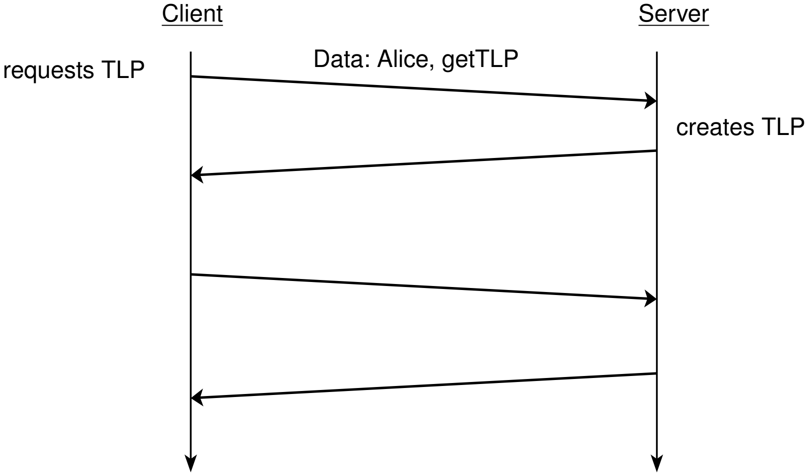
PKWL 2.0 - Proof-of-Work-System



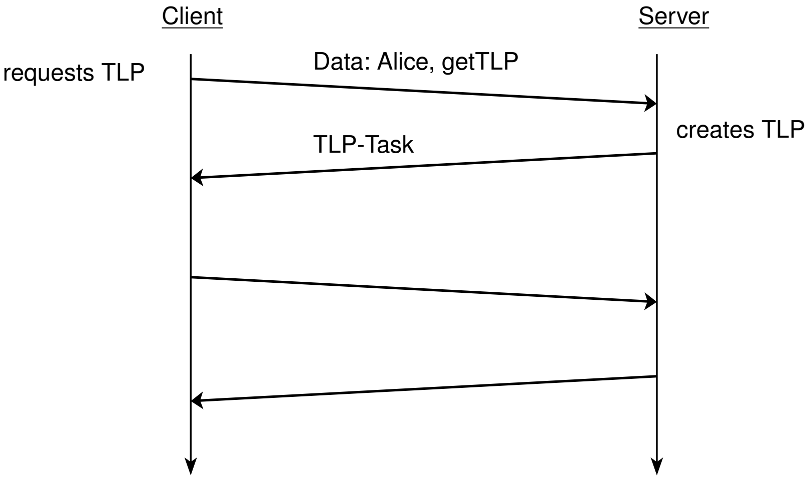
PKWL 2.0 - Proof-of-Work-System



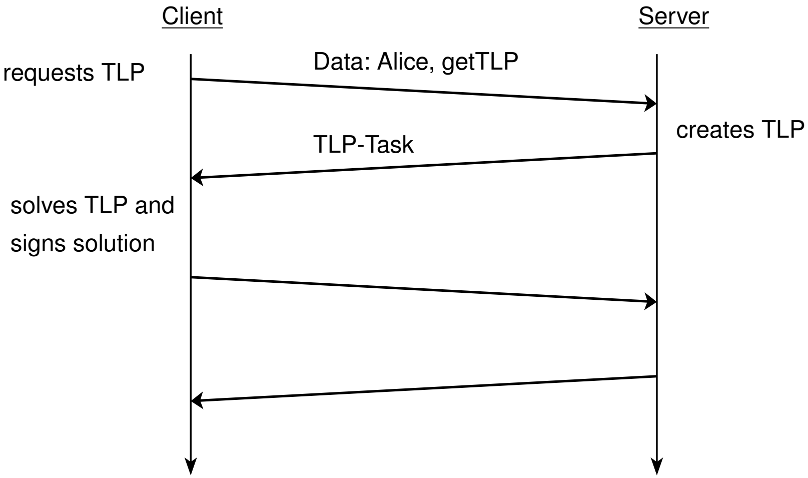
PKWL 2.0 - Proof-of-Work-System



PKWL 2.0 - Proof-of-Work-System



PKWL 2.0 - Proof-of-Work-System



Client

Server

requests TLP

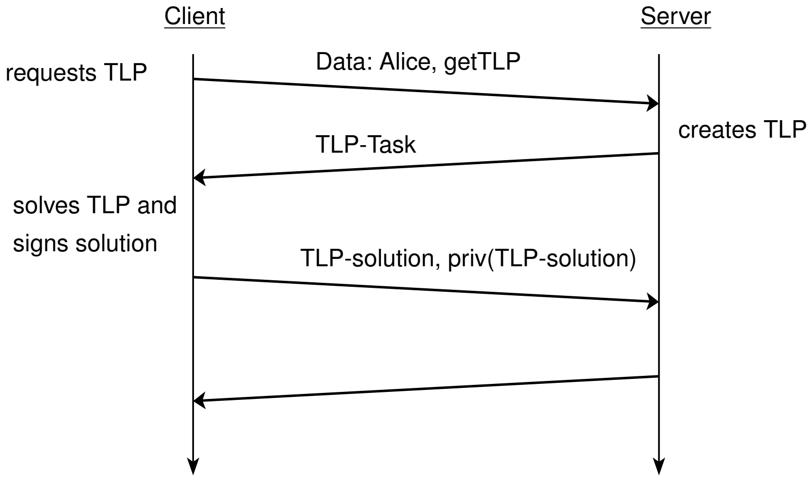
Data: Alice, getTLP

creates TLP

TLP-Task

solves TLP and signs solution

PKWL 2.0 - Proof-of-Work-System



Client

Server

requests TLP

Data: Alice, getTLP

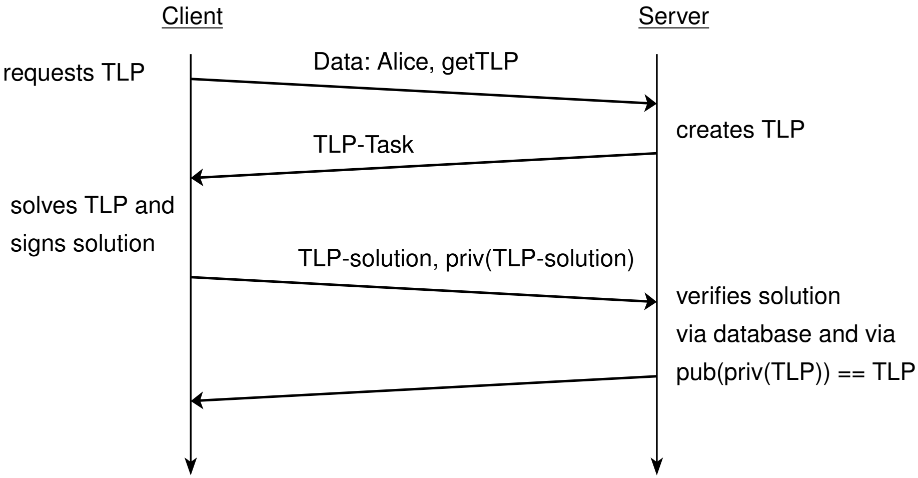
creates TLP

TLP-Task

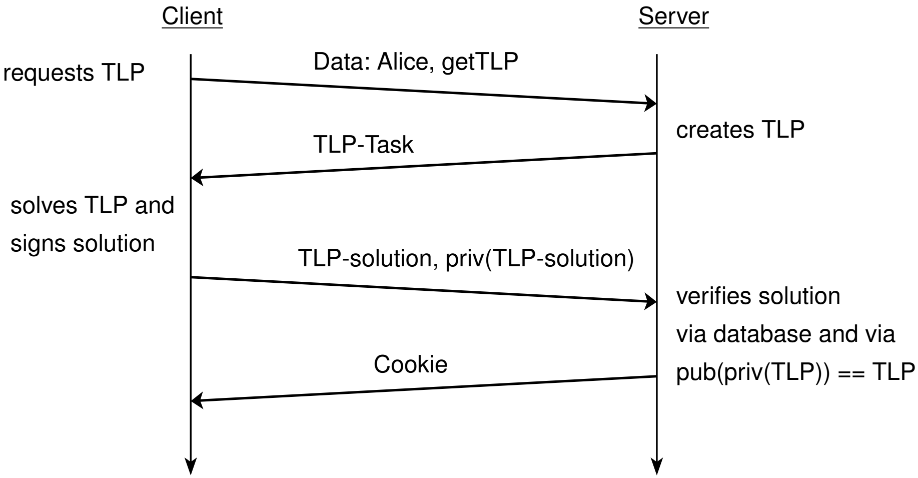
solves TLP and signs solution

TLP-solution, priv(TLP-solution)

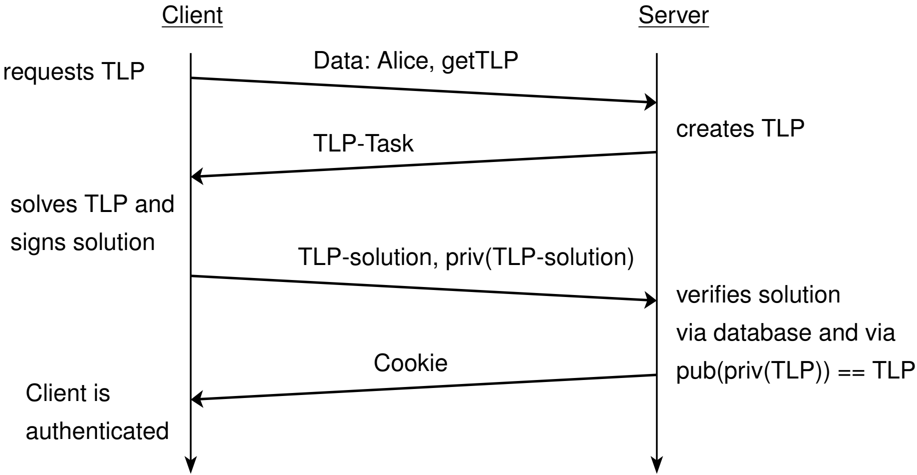
PKWL 2.0 - Proof-of-Work-System



PKWL 2.0 - Proof-of-Work-System


Client
Server
requests TLP
Data: Alice, getTLP
creates TLP
TLP-Task
solves TLP and signs solution
TLP-solution, priv(TLP-solution)
verifies solution via database and via pub(priv(TLP)) == TLP
Cookie

PKWL 2.0 - Proof-of-Work-System



Client

Server

requests TLP

Data: Alice, getTLP

creates TLP

TLP-Task

solves TLP and signs solution

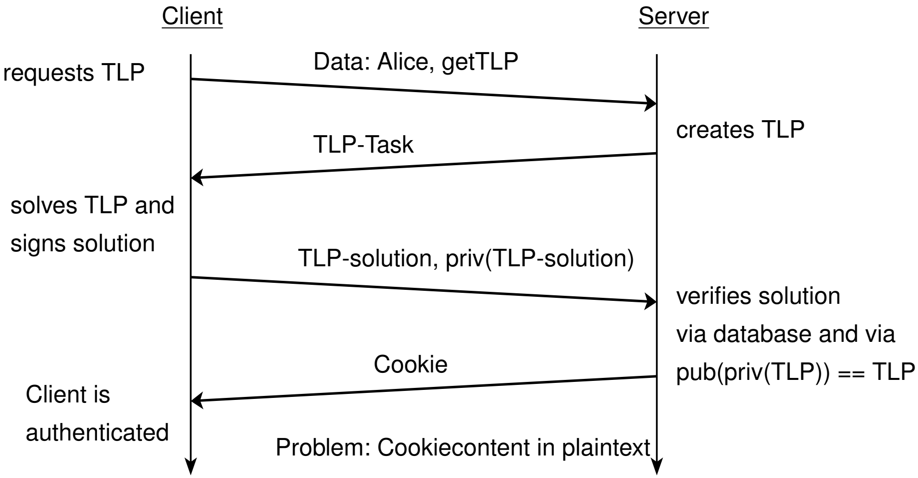
TLP-solution, priv(TLP-solution)

verifies solution via database and via pub(priv(TLP)) == TLP

Cookie

Client is authenticated

PKWL 2.0 - Proof-of-Work-System



Client

Server

requests TLP

Data: Alice, getTLP

creates TLP

TLP-Task

solves TLP and signs solution

TLP-solution, priv(TLP-solution)

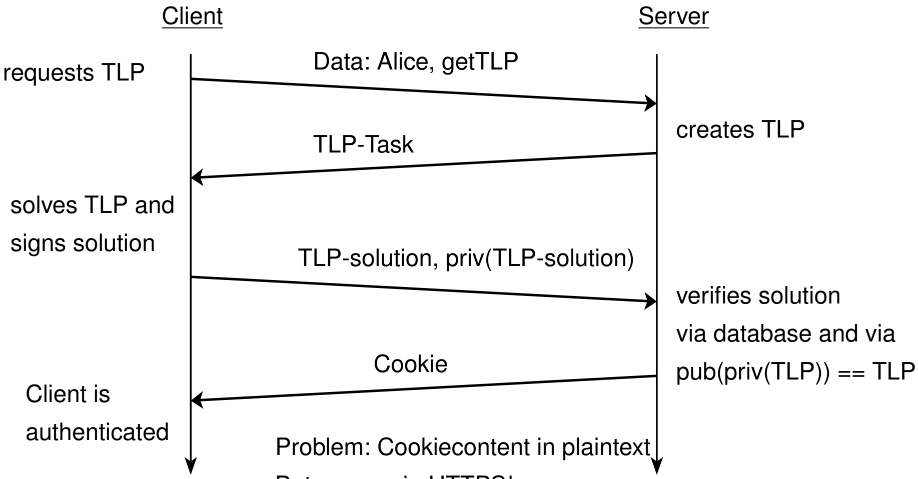
verifies solution via database and via pub(priv(TLP)) == TLP

Cookie

Client is authenticated

Problem: Cookiecontent in plaintext

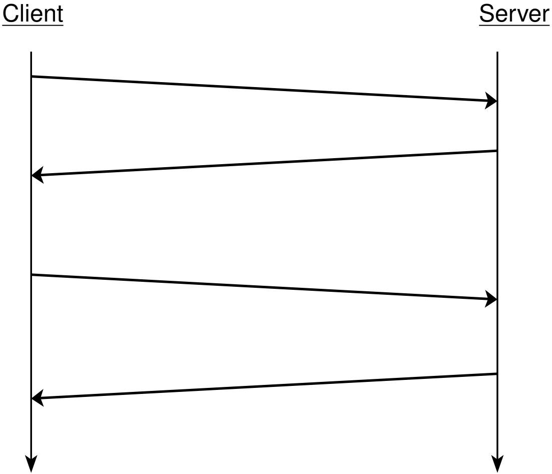
PKWL 2.0 - Proof-of-Work-System


Client
Server
requests TLP
Data: Alice, getTLP
creates TLP
TLP-Task
solves TLP and signs solution
TLP-solution, priv(TLP-solution)
verifies solution via database and via pub(priv(TLP)) == TLP
Cookie
Client is authenticated

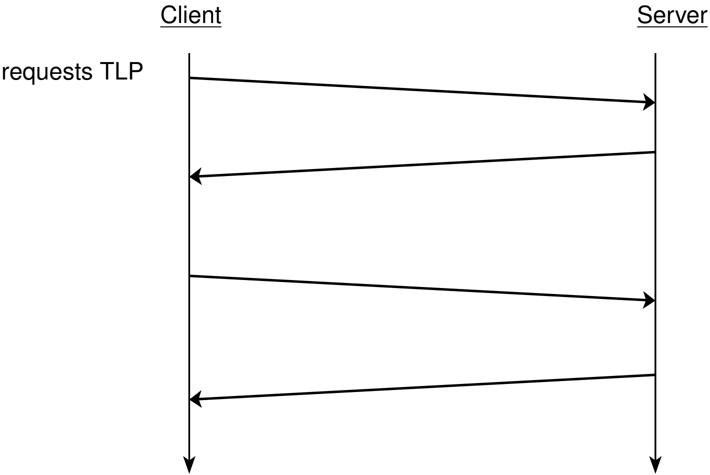
Problem: Cookiecontent in plaintext

But: secure in HTTPS!

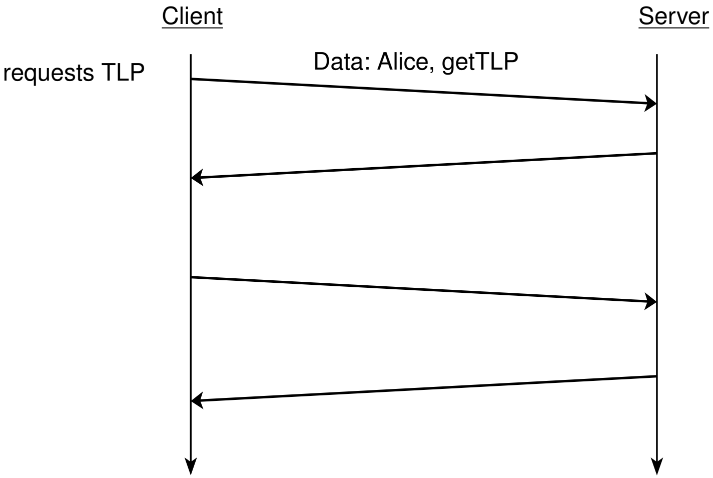
PKWL 3.0 - Encrypted Cookies



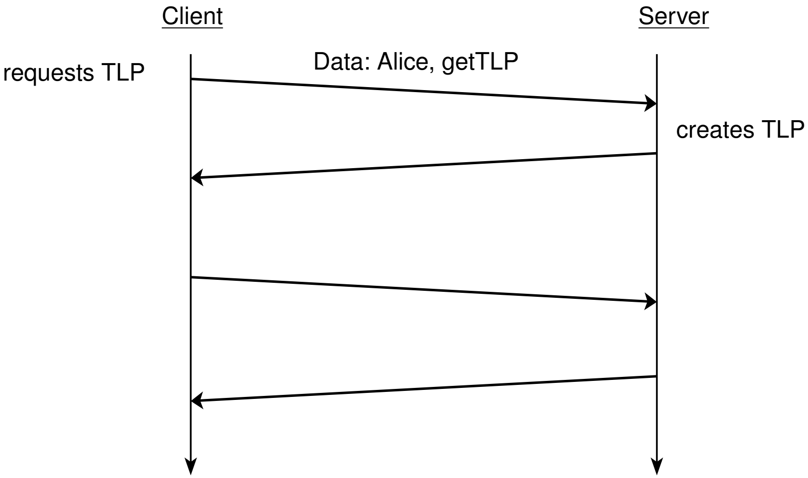
PKWL 3.0 - Encrypted Cookies



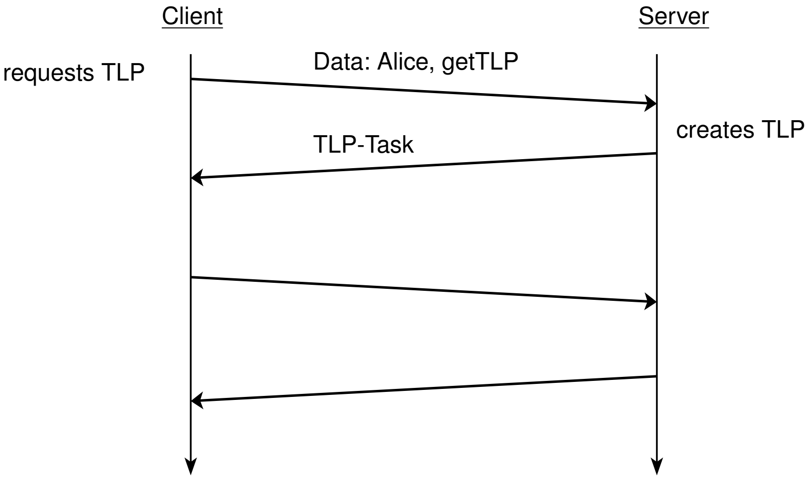
PKWL 3.0 - Encrypted Cookies



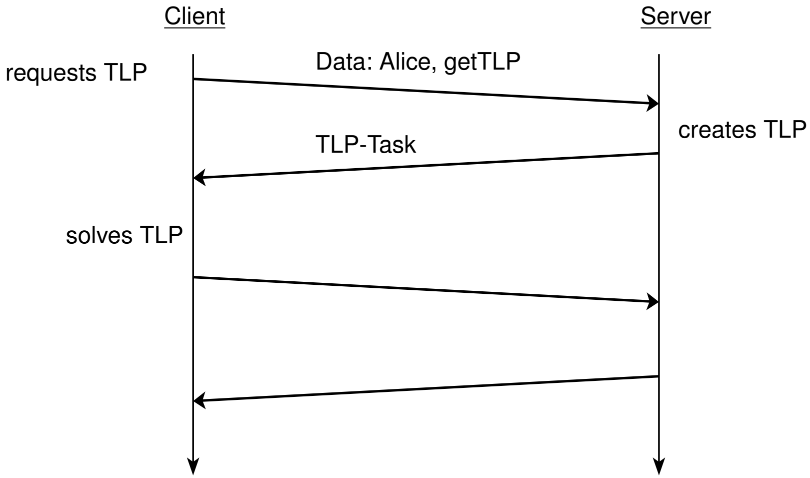
PKWL 3.0 - Encrypted Cookies



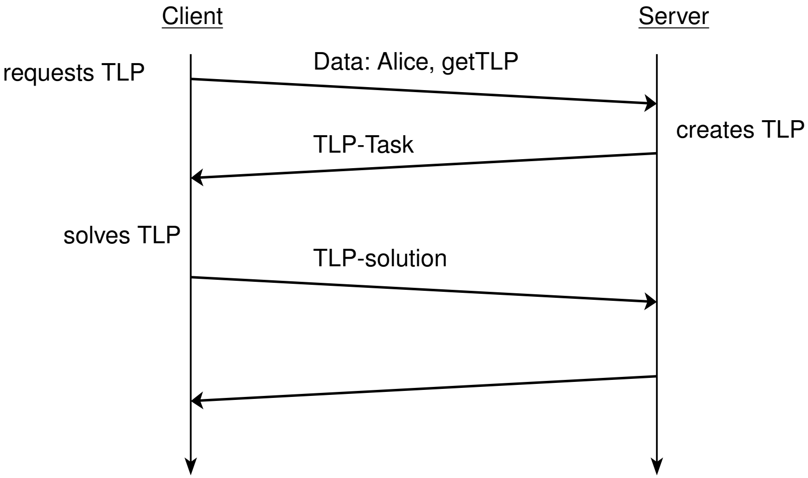
PKWL 3.0 - Encrypted Cookies



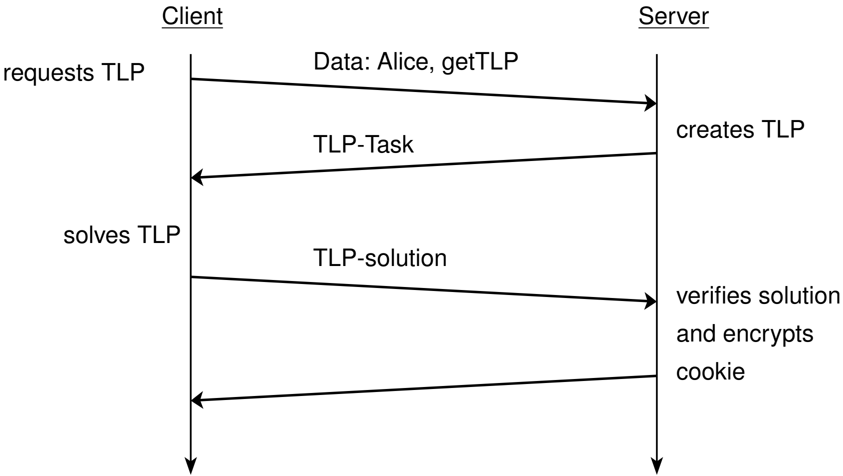
PKWL 3.0 - Encrypted Cookies



PKWL 3.0 - Encrypted Cookies



PKWL 3.0 - Encrypted Cookies



Client

Server

requests TLP

Data: Alice, getTLP

creates TLP

TLP-Task

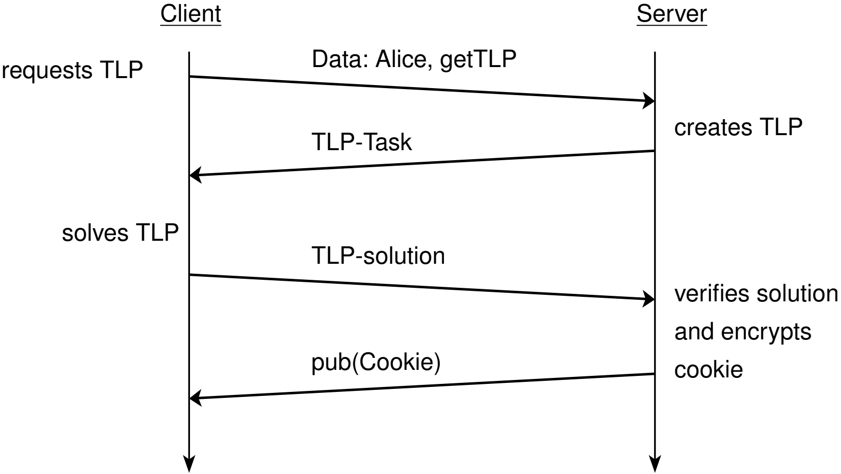
solves TLP

TLP-solution

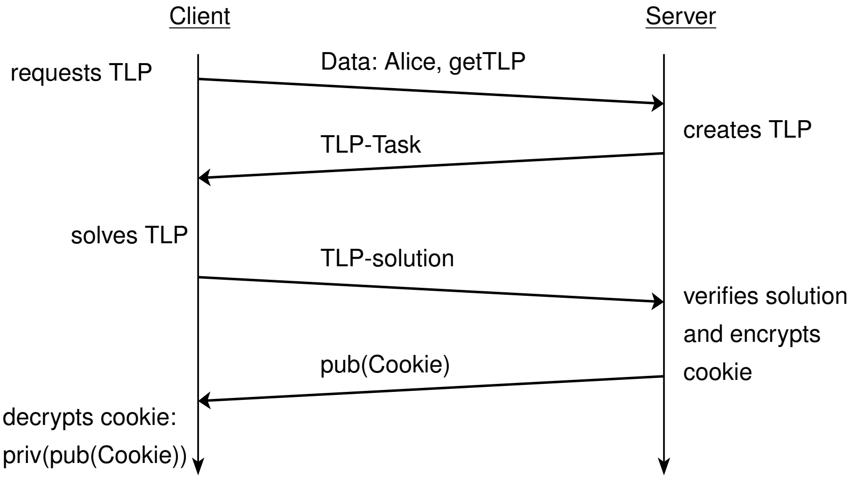
verifies solution and encrypts cookie

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

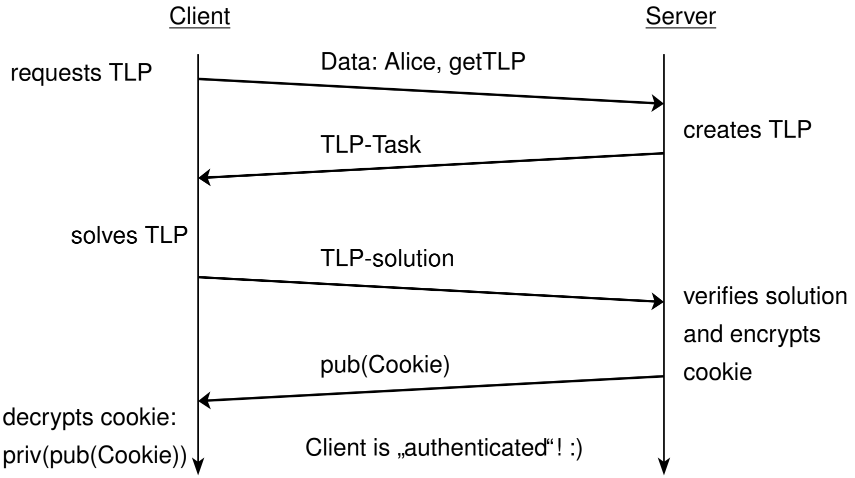
PKWL 3.0 - Encrypted Cookies



PKWL 3.0 - Encrypted Cookies



PKWL 3.0 - Encrypted Cookies



Client

Server

requests TLP

Data: Alice, getTLP

creates TLP

TLP-Task

solves TLP

TLP-solution

verifies solution and encrypts cookie

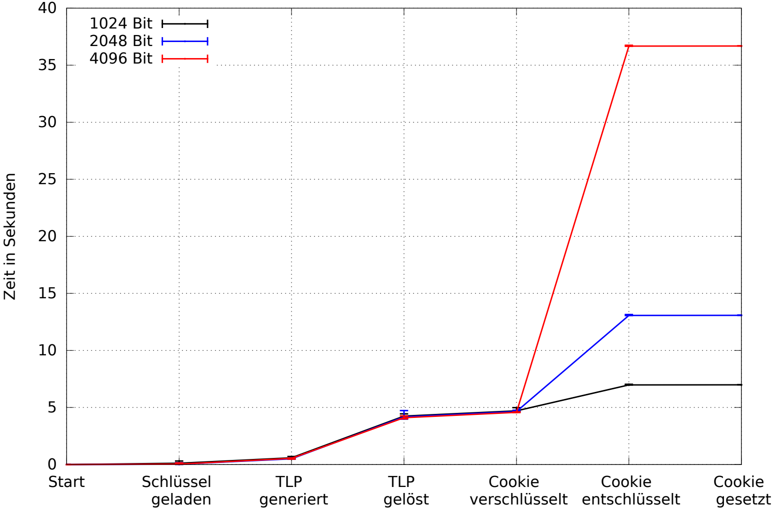
pub(Cookie)

decrypts cookie: priv(pub(Cookie))

Client is „authenticated“! :)

Demo

Evaluation


40
35
30
25
20
15
10
5
0
Zeit in Sekunden
1024 Bit
2048 Bit
4096 Bit
Start
Schlüssel geladen
TLP generiert
TLP gelöst
Cookie verschlüsselt
Cookie entschlüsselt
Cookie gesetzt

Evaluation

Evaluation

- Security via design
  - Proof-of-Work-System
  - RSA-Cryptosystem

Evaluation

- Security via design
  - Proof-of-Work-System
  - RSA-Cryptosystem
- Performance
  - 2048 Bit-Keylength - ca. 13 seconds

Evaluation

- Security via design
  - Proof-of-Work-System
  - RSA-Cryptosystem
- Performance
  - 2048 Bit-Keylength - ca. 13 seconds
- Comfort
  - no configuration on server- and client-side
  - complete control via graphical user interface
  - daily routine: two mouse clicks for authentication

Conclusion

- Passwords are a thing of the past

Conclusion

- Passwords are a thing of the past
- PKWL offers security, performance und comfort

Conclusion

- Passwords are a thing of the past
- PKWL offers security, performance und comfort
- password-free authentication works!

Conclusion

- Passwords are a thing of the past
- PKWL offers security, performance und comfort
- password-free authentication works!

- Future Work

Conclusion

- Passwords are a thing of the past
- PKWL offers security, performance und comfort
- password-free authentication works!

- Future Work
  - dynamic time-adjustment of the Proof-of-Work-Systems

Conclusion

- Passwords are a thing of the past
- PKWL offers security, performance und comfort
- password-free authentication works!

- Future Work
  - dynamic time-adjustment of the Proof-of-Work-Systems
  - automatic authentication in the background

Conclusion

- Passwords are a thing of the past
- PKWL offers security, performance und comfort
- password-free authentication works!

- Future Work
  - dynamic time-adjustment of the Proof-of-Work-Systems
  - automatic authentication in the background
  - W3C plans a Web Cryptography API

Questions?

- Source-Code: `https://github.com/mirkoole/PKWL`