

# SRSC 24/25

## PA2 – Project Assignment 2

### **SHP**

**A Secure Handshake Protocol for Authentication and Agreement of  
Cryptographic Configurations and Related Security Association Parameters**

# SHP (PA2) in sequence of PA1

- **Initially, PA1 (Project Assignment No. 1)** focused on the design, implementation, and experimental evaluation/observation of a Secure Channel Abstraction supported by a Datagram-Based Secure Transport Protocol.
- As a generic solution supporting different cryptographic parameterizations, it enables support for UDP-based applications (which are not initially supported for secure communication).

The generic support is provided by a '**Datagram-Based Secure Transport Protocol**'.

- **Goal:**

To offer a solution for transparent support of the initially provided client/server applications: Streaming Service, TFTP, and a testing application based on IP Multicasting.

- **Limitation:**

- Static configuration of cryptographic parameters and related security settings (including keys) for the cryptographic solutions used.
- Manual sharing of parameter configurations between endpoints in clients and servers."

# SHP (PA2) in sequence of PA1

- The goal of PA2 followed PA1, aiming to develop an improved solution that overcomes the limitations of the PA1 implementation.
- **Context:** implementation of a Secure Handshake Protocol (SHP) to be integrated with the initial solution developed in PA1.
- **Goal:** to provide authentication and agreement on cryptographic configurations for the dynamic establishment of related security association parameters (including cryptographic configurations).
- To achieve this goal, SHP must incorporate complementary cryptographic solutions, including:

- Password-based encryption methods
- Public-key cryptography
- Digital signatures

SHP-PHASE 1  
Context

SHP-PHASE 2  
Context

- Diffie-Hellman key agreement (for dynamic establishment of keys and/or security association parameters)

# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# Discussion: PA2 Requirements

## Outline

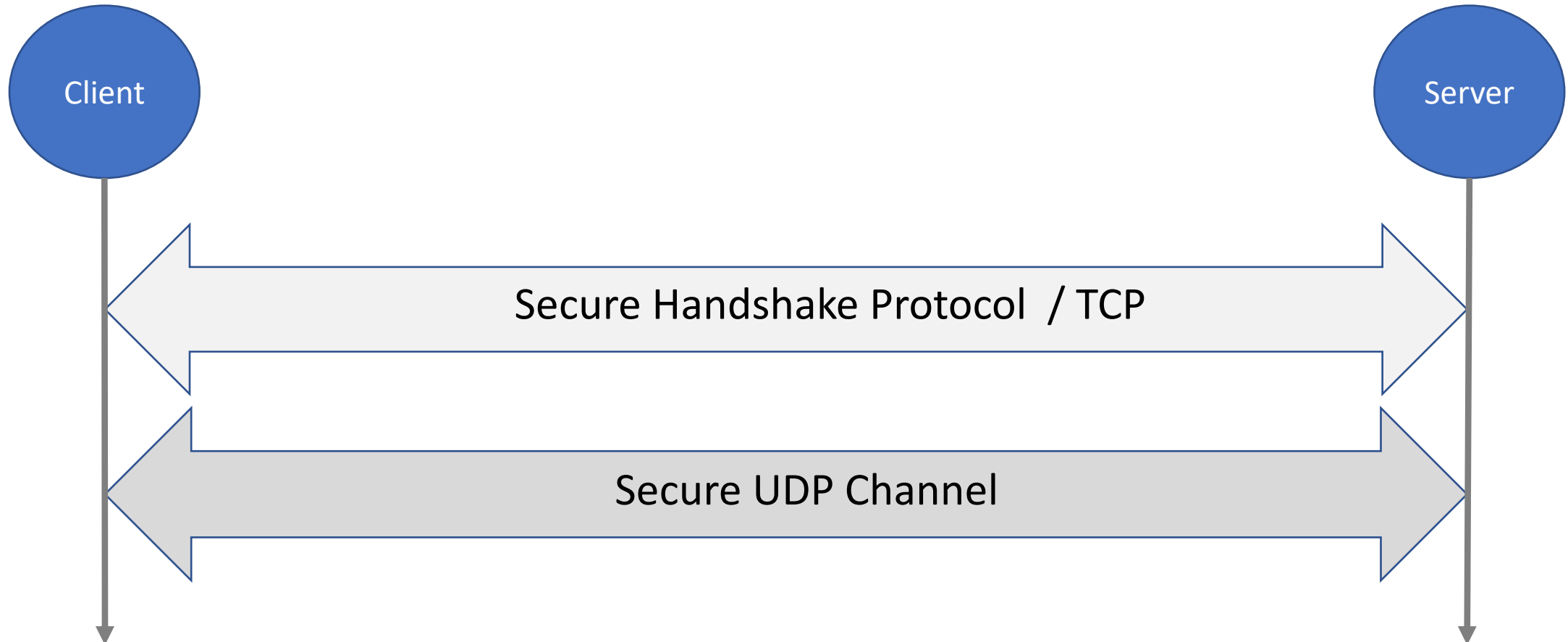
- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# Generic Model for the SHP Protocol

- SHP follow a Client/Server Approach
  - As the previous PA1 Solution
- SHP will be designed to be combined with a PA1 based solution
  - The idea is to reuse the PA1 development (with minimum modifications), to integrate it in a combined solution (integrating PA1 support with the SHP protocol to be developed in PA2)

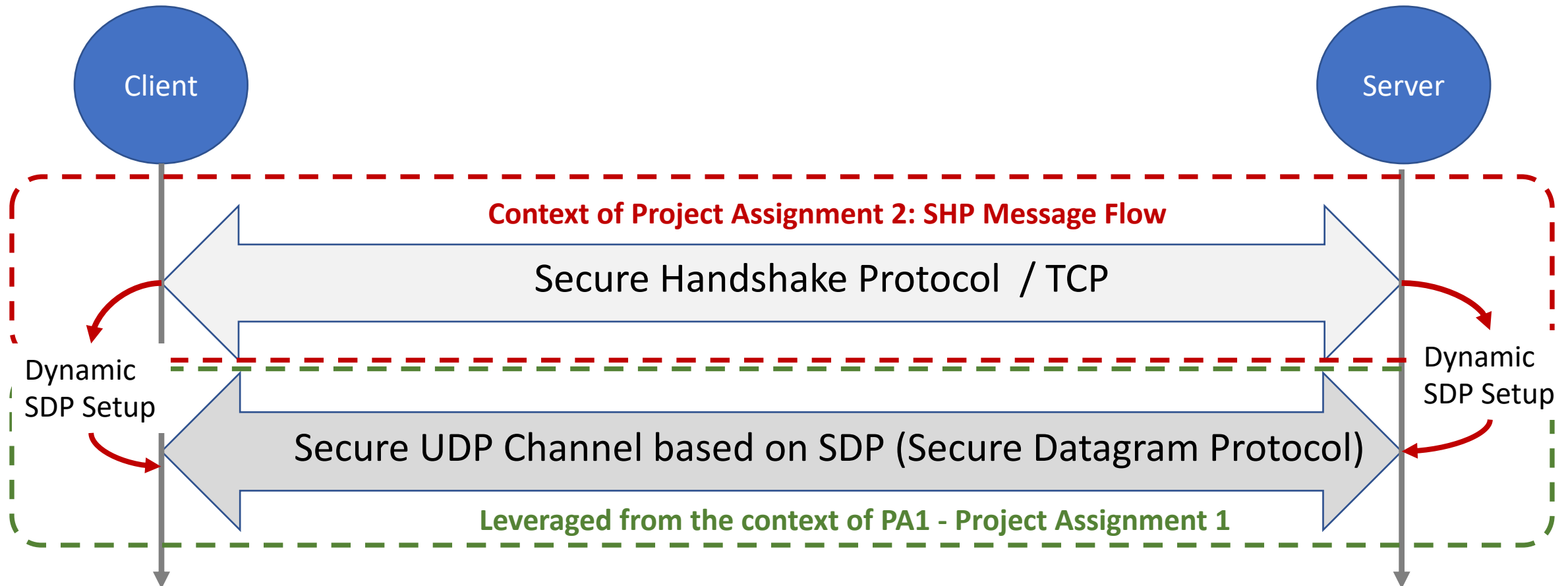
# Generic Approach of System Model

- Base: Client/Server Approach for SHP



# Generic Approach of System Model


- Base: Client/Server Approach for SHP





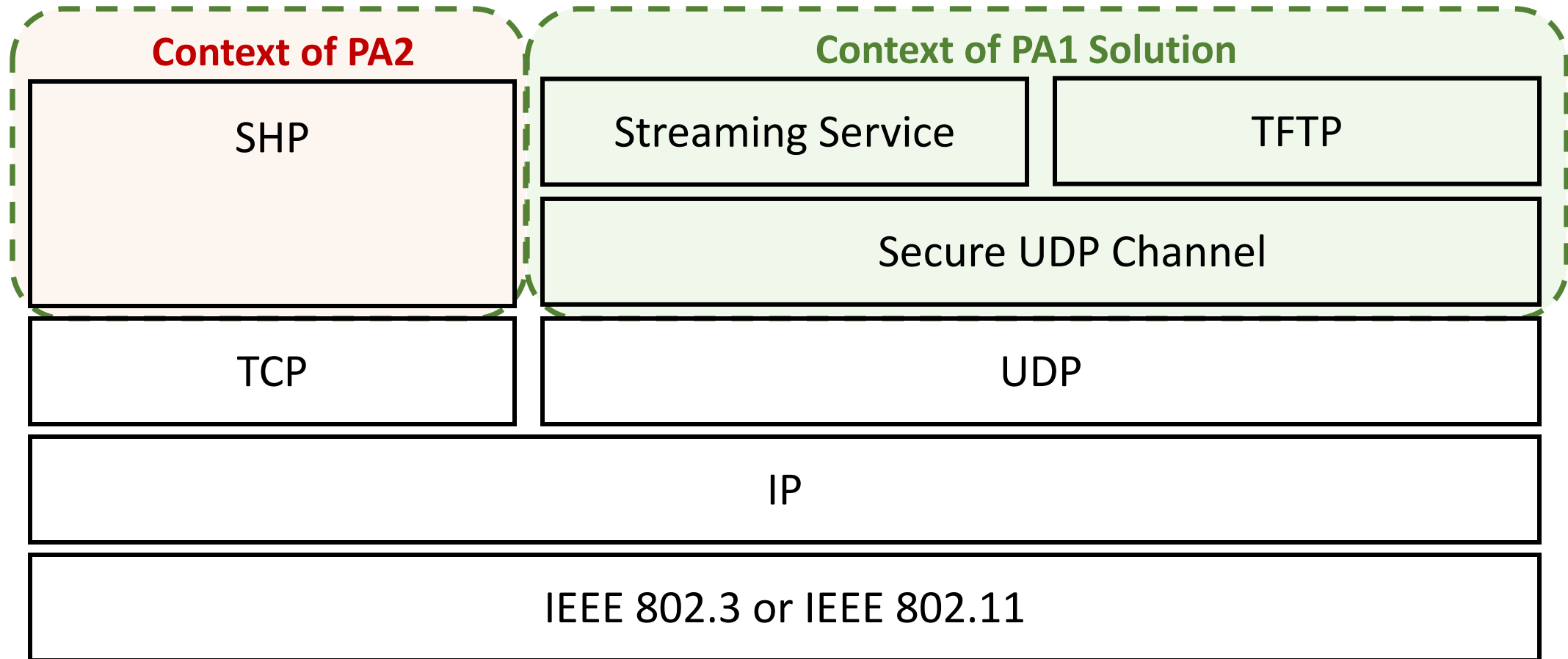
# Discussion: PA2 Requirements

## Outline

- 
- Context and Motivation
  - SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
  - Implementation phases
  - SHP Generic Protocol Model and Architecture
  - SHP Requirements in Phase 1 and Phase 1 Setup
    - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
  - SHP Requirements in Phase 2 and Phase 2 Setup
    - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
  - Improvements and qualitative factors
  - Evaluation criteria
  - Important dates
  - Delivery process

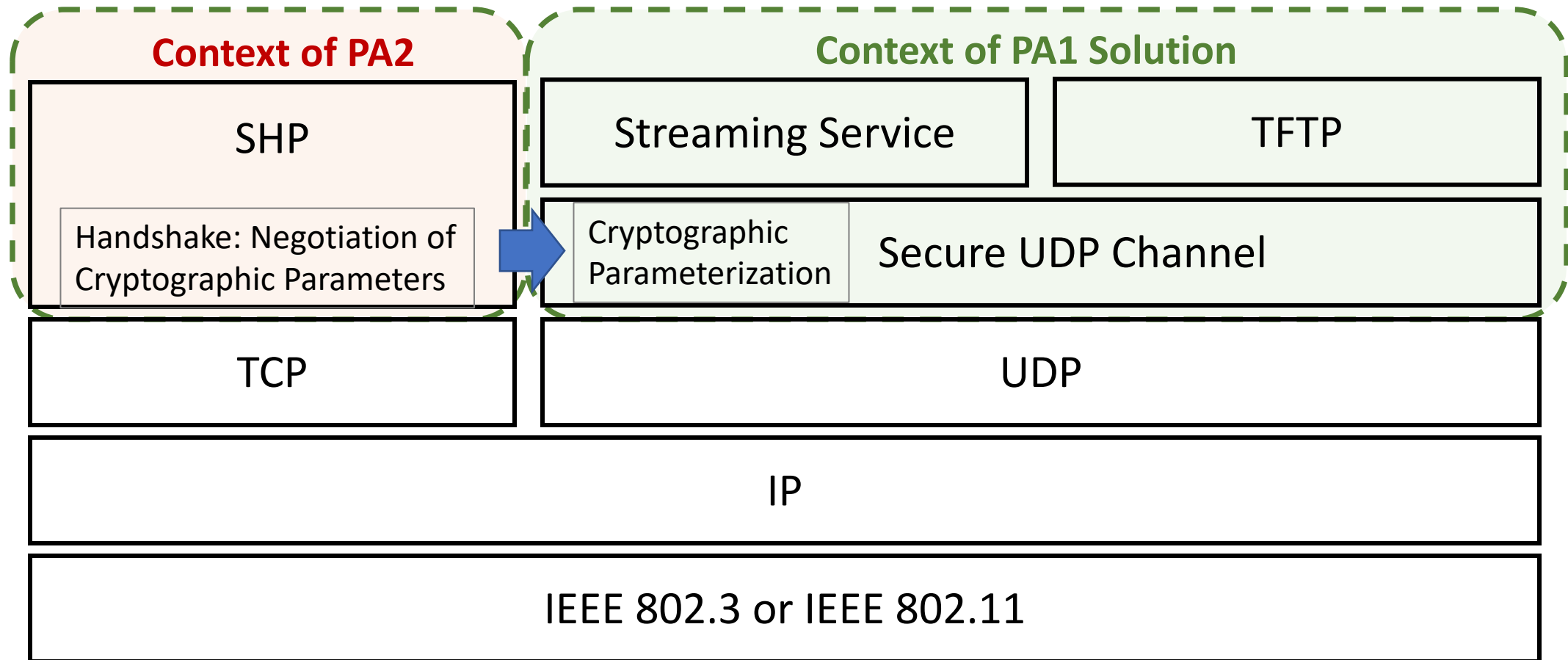
# Implementation Stack: SHP and integration with PA1

- PA2: SHP implementation stack in combination with the previous PA1 implementation Stack




# Implementation Stack: SHP and integration with PA1

- PA2: SHP implementation stack in combination with the previous PA1 implementation Stack



# Discussion: PA2 Requirements

## Outline

- 
- Context and Motivation
  - SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
  - Implementation phases
  - SHP Generic Protocol Model and Architecture
  - SHP Requirements in Phase 1 and Phase 1 Setup
    - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
  - SHP Requirements in Phase 2 and Phase 2 Setup
    - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
  - Improvements and qualitative factors
  - Evaluation criteria
  - Important dates
  - Delivery process

# Implementation Phases

This is just a suggestion

- Can implement PA2 in two successive versions of the SHP Protocol (as a possible development methodology), as sequential STEPS
  - **SHP Phase 1 (minimum, mandatory): \***
    - Use of Password Based Encryption (LAB 3 Materials), Digital Signatures and HMAC as basic Cryptographic Constructions for Phase 1
  - **SHP Phase 2 (complete work) \***
    - Use of Password Based Encryption (LAB 3 Materials), Digital Signatures, Diffie-Hellman Agreements and HMAC as required Cryptographic Constructions for Phase 2
    - Complementary improvements
- But you can choose to implement phase 2 right away (i.e., the full version), without need to implement phase 1: it is your decision

\*) *See the evaluation criteria (slides 62, 63 and 64)*

# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# SHP Protocol Model and Architecture

- Independently of the phases (phase 1 or phase2) the SHP protocol will be supported by TCP, following a Client/Server Model
- The implementation must be approached in a way that the SHP protocol will be **supported by a new primitive** (depending on the phases) to be executed by the Client and by the Server

client\_shp\_phase1

server\_shp\_phase1

client\_shp\_phase2

server\_shp\_phase2

**Goal:** to minimize the porting requirements (impact in code and lines-of-code changes) for applications (ex., Streaming App and TFTP App from Project Assignment 1 demonstrators) to adopt the new support solution, provided after the Project Assignment 2

# SHP Integration with the initial solution in PA1

- The Client/Server SHP model is regarded as an extension for the clients and servers in the following testing applications of PA1
- Streaming Service
  - Client is the Proxy
  - Server is the StreamingServer
- TFTP application
  - Client is the TFTPClient
  - Server is the TFTPServer

The SHP Implementation and Integration must be used to support both testing applications



# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# SHP – Phase 1 Specification Approach

Base Specification for SHP in Phase 1:

Students must complete the basic specification (maintaining the base requirements) for concrete implementations

# SHP Phase 1 Requirements:

## Authentication Requirements

- SHP Phase 1 must provide a strong two-factor authentication solution for users (using client-side applications) in the interaction with server-side applications
  - Client-Side Authentication Mechanisms
    - User Authentication, based on Passwords and Password-Based Encryption Construction
    - User Peer-Authentication Guarantees based on the use of Digital Signatures using Public-Key Cryptography (based on ECDSA Digital Signatures)
  - Server-Side Authentication Mechanisms
    - Server-Side Peer-Authentication Guarantees based on the use of Digital Signatures using Public-Key Cryptography (based on ECDSA Digital Signatures)
  - Client/Server Mutual Authentication
    - With the above solutions, SHP must provide mutual peer-authentication guarantees, that can be controlled by clients and servers

# SHP Phase 1 Requirements: Confidentiality Requirements

- SHP Phase 1 must provide connection-oriented confidentiality
  - During Clint/Server SHP message exchanges, the critical information that must be exchanged with confidentiality guarantees will be encrypted using Password-Based Encryption

# SHP Phase 1 Requirements:

## Integrity Requirements

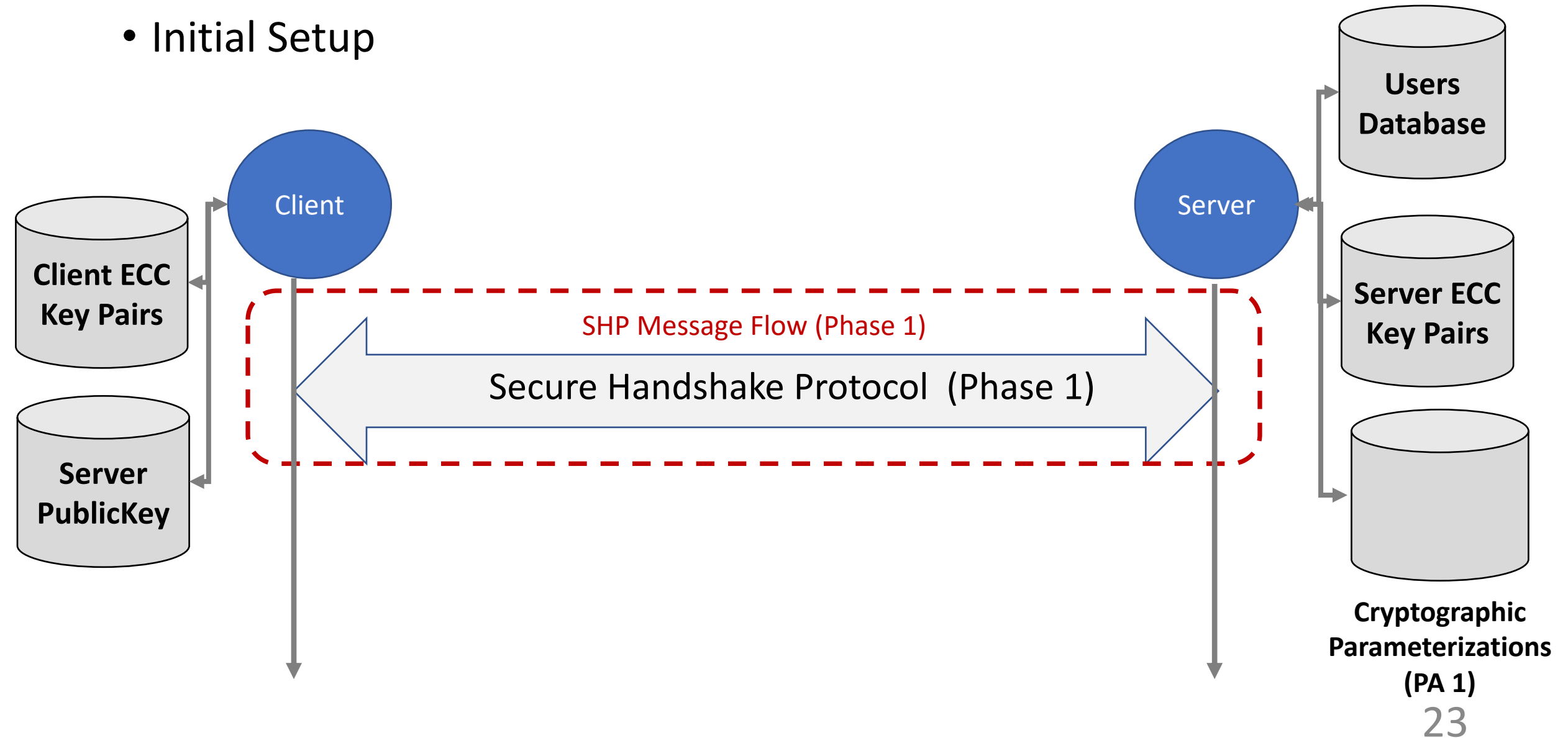
- SHP Phase 1 must provide integrity, with the following guarantees in considered critical SHP messages
  - Connection oriented message integrity (with no recovery), combining:
    - Use of HMACs for fast message authentication and integrity checking
    - Implicit integrity in messages where digital signatures are used
  - Flow integrity is ensured with the use of protected nonces, which are managed as random, used-use challenges/responses throughout the message exchanges

# SHP Phase 1 Requirements: Message-Replay Protection

- SHP Phase 1 must provide anti-message-reply guarantees:
  - Message-replaying countermeasures are based with the required control of uniqueness of critical messages, with the use of implicit mechanisms such as nonces, which are managed as random, single-use challenges/responses during message exchanges, as well as by controlling the uniqueness of HMACs in the protocol's operation.
  - In complementarity, message-exchanges in TCP sessions must be ordered, by controlling the message-types that will be processed (only-once) in the correct sequence of message exchanges.

# SHP – Phase 1: Setup

- Initial Setup



# Server-Side (Setup)

- User Database: A TXT File with the following entries (lines)

## userdatabase.txt

```
Userid : H(password) : salt : KpubClient
```

**Userid:** can use strings as email addresses (ex: [alice@gmail.com](mailto:alice@gmail.com))

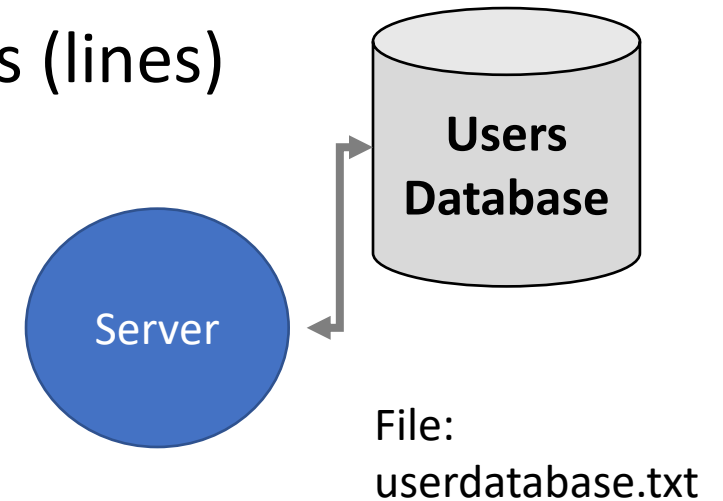
**password:** a strong password registered for the user

**H(password):** SHA256(password)

**salt:** random number registered for each user (128 bits)

**KpubClient:** a ECC public key registered for each user

Note: Can use Base64 or Hexadecimal representations for the binary objects in the database entries





# Server-Side (Setup)

- Cryptographic Parameterizations

## **ServerECCKeyPair.sec**

Is a text file that stores a pre-generated ECC keypair (public key and private key), to be used for ECDSA Digital signatures.

Can represent the keys in Base64 or Hexadecimal representation format  
For the implementation and demonstration students can choose any Curve (as seen in Labs)

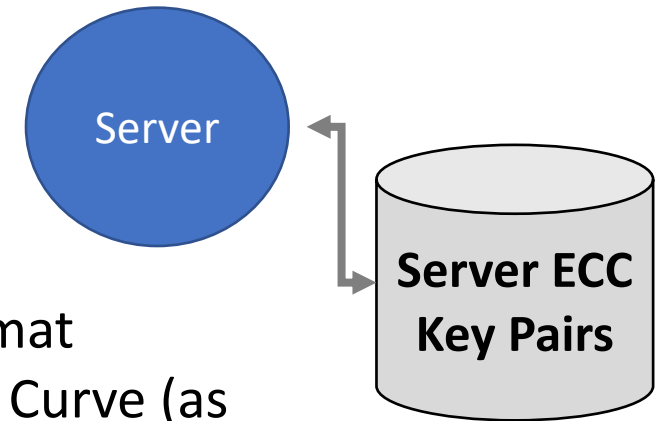
Format:

Curve: EllipticCurve // ex., secp256

PrivateKey: xxxxxxxxxxxx....xxxx

PublicKey: xxxxxxxxxxxx .... xxxxxx

Note: Can use Base64 or Hexadecimal representations for the binary objects in the database entries



# Server-Side (Setup)

- Cryptographic Parameterizations

## **ciphersuite.conf**

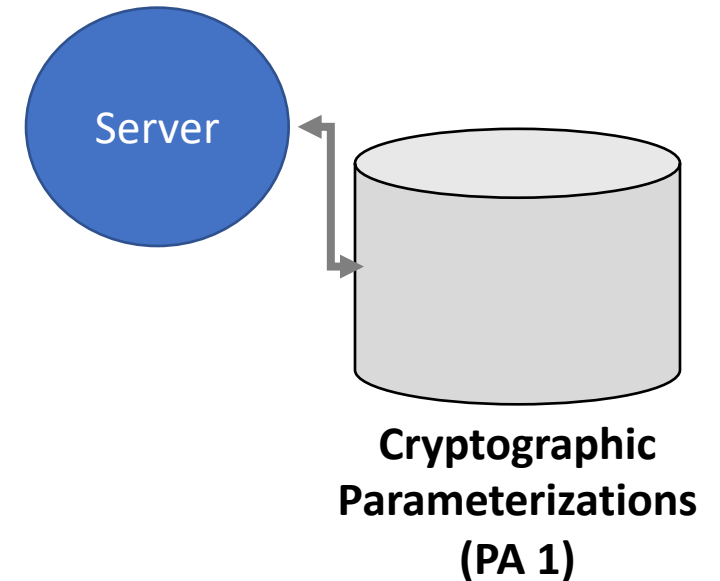
Text file

Contains the cryptographic parameterizations to be established

With the client, after the conclusion of the SHP with success

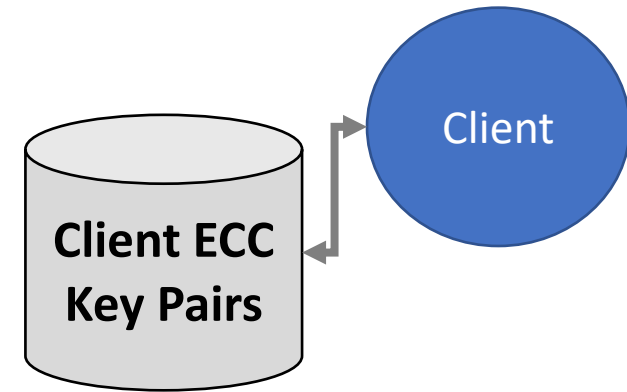
The ciphersuite.conf file has the same contents as specified for Project Assignment 1

Setup with possible different cryptographic parameterizations  
(as in Project Assignment 1)



# Client-Side (Setup)

- Cryptographic Parameterizations



File:  
ClientECCKeypair.sec

## **ClientECCKeypair.sec**

Is a text file that stores a pre-generated ECC keypair (public key and private key)

To be used for ECDSA Digital signatures.

Can represent the keys in Base64 or Hexadecimal representation format  
For the implementation and demonstration students can choose any Curve (as seen in Labs)

Format:

Curve: EllipticCurve // ex., secp256

PrivateKey: xxxxxxxxxxxx....xxxx

PublicKey: xxxxxxxxxxxx .... xxxxxx

# Client-Side (Setup)

- Cryptographic Parameterizations

## **ServerECCPubKey.txt**

Is a text file that stores a pre-generated ECC keypair (public key and private key)

To be used for ECDSA Digital signatures.

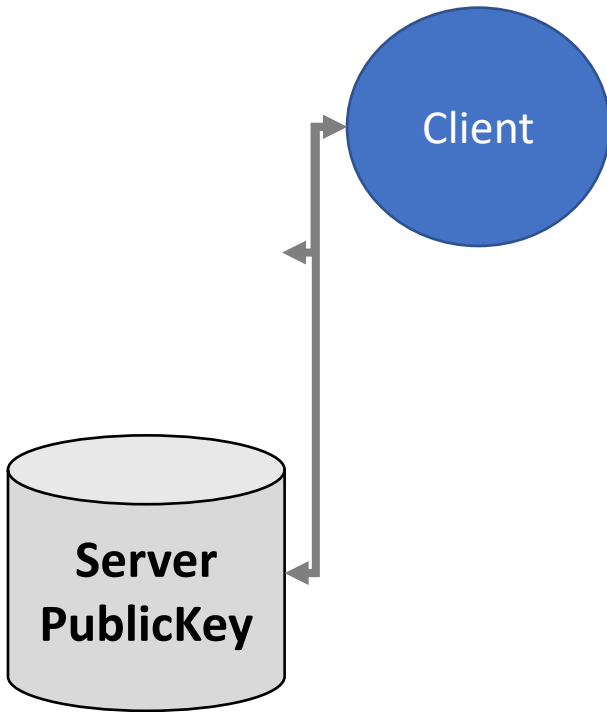
Can represent the keys in Base64 or Hexadecimal representation format  
For the implementation and demonstration students can choose any Curve (as seen in Labs)

Format:

Curve: EllipticCurve // ex., secp256

PrivateKey: xxxxxxxxxxxx....xxxx

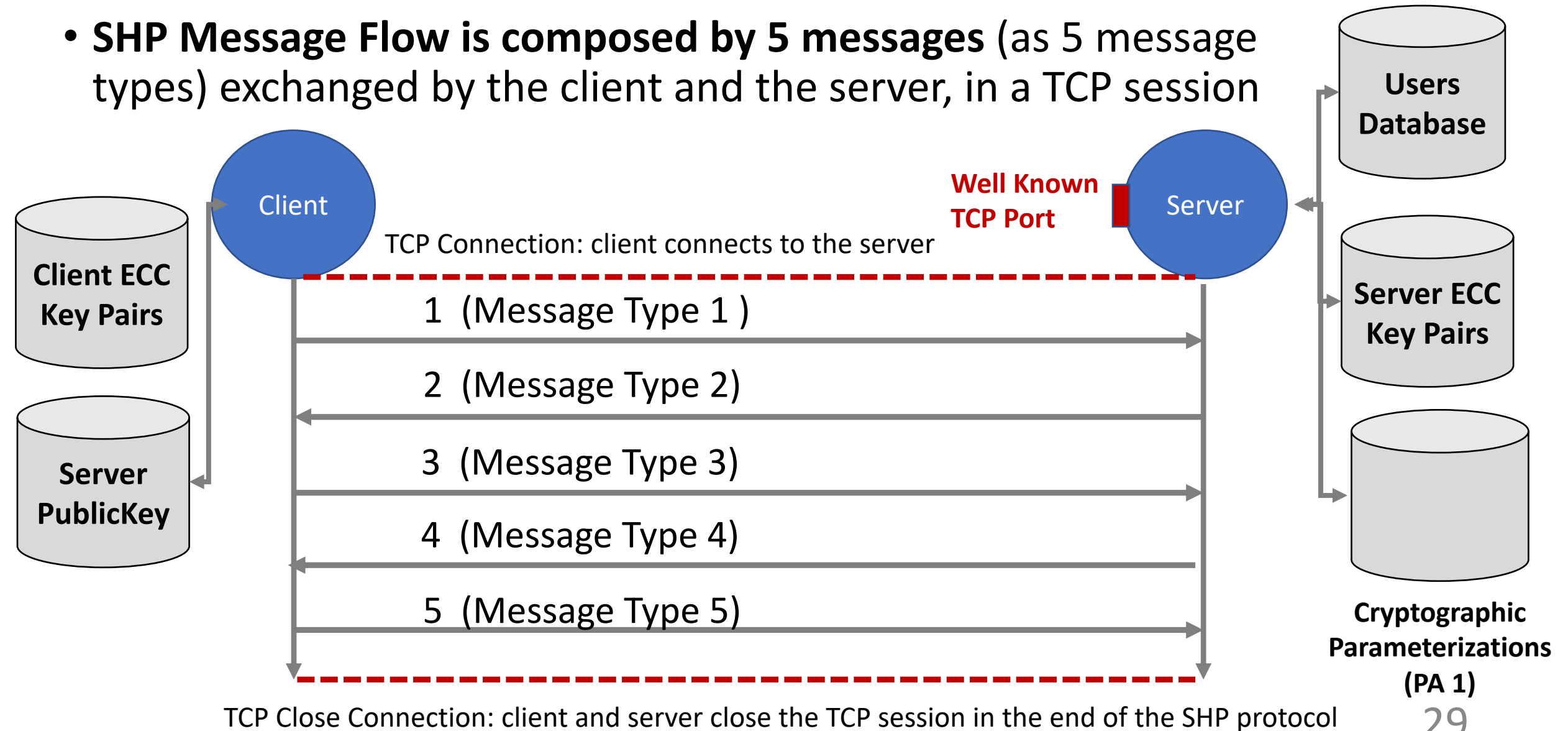
PublicKey: xxxxxxxxxxxx .... xxxxxx



File.  
ServerECCPubKey.txt

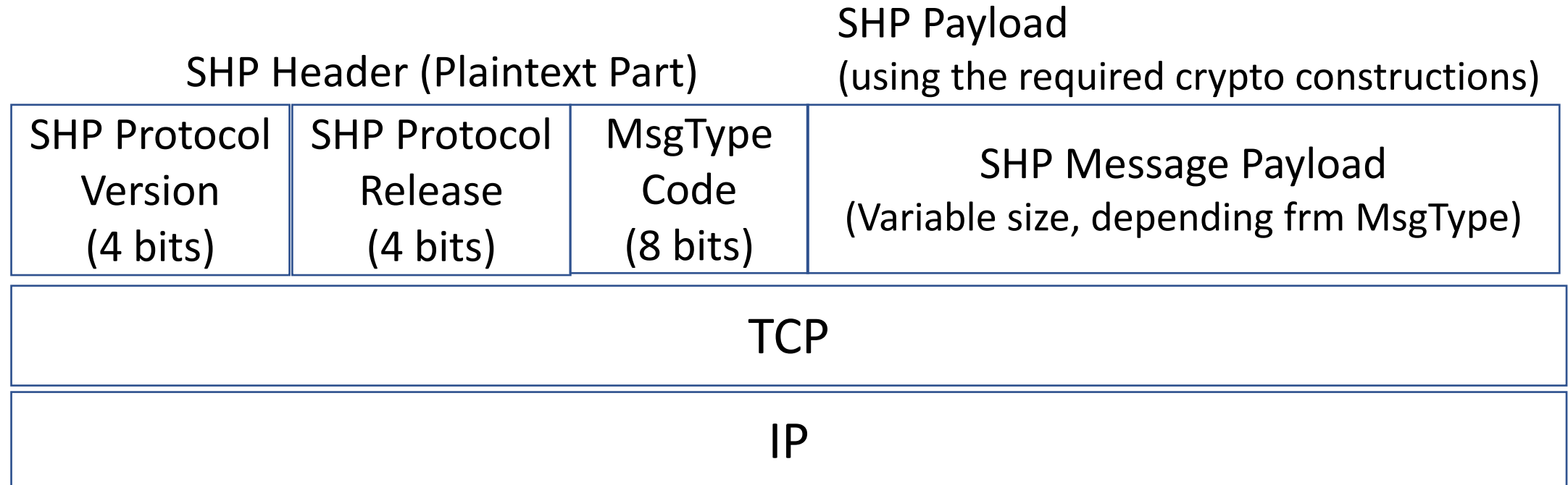
# SHP Phase 1: Message Flow

- **SHP Message Flow is composed by 5 messages** (as 5 message types) exchanged by the client and the server, in a TCP session



# SHP Phase 1

- SHP Messages as message types must be formatted in a generic format, as follows, supported by TCP based communication

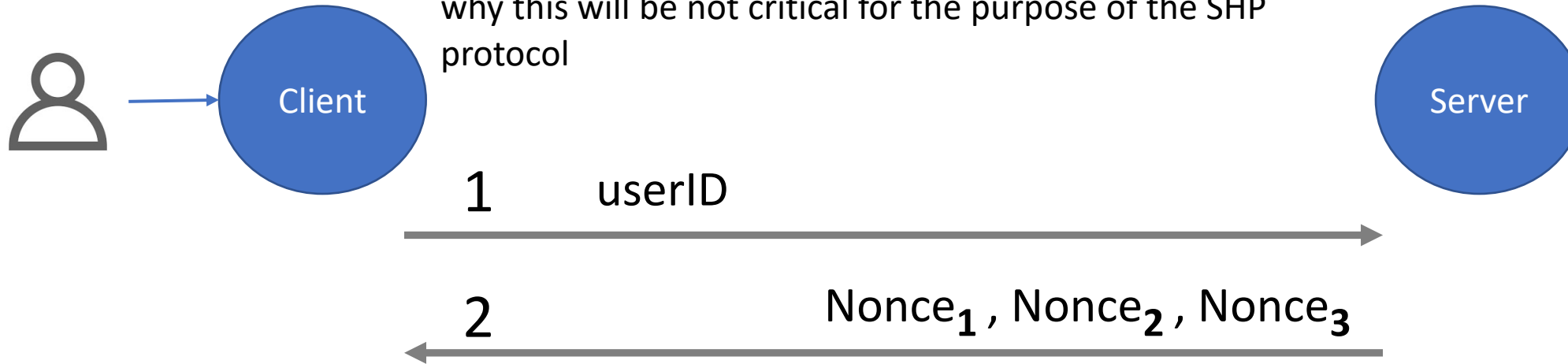


You can extend the SHP Message format adding control fields, depending on the concrete implementation. For example, you can add an initial field in the header (ex., with 2 bytes) to express the size of the entire packet (in bytes). Another example is to include a field with two bytes (between the MsgType and the SHP payload, expressing the size of carried payload).

# SHP Message Flow: SHP – Phase 1

- Messages: Payloads for Message types 1 and 2

This messages are passed in clear, and you must understand why this will be not critical for the purpose of the SHP protocol



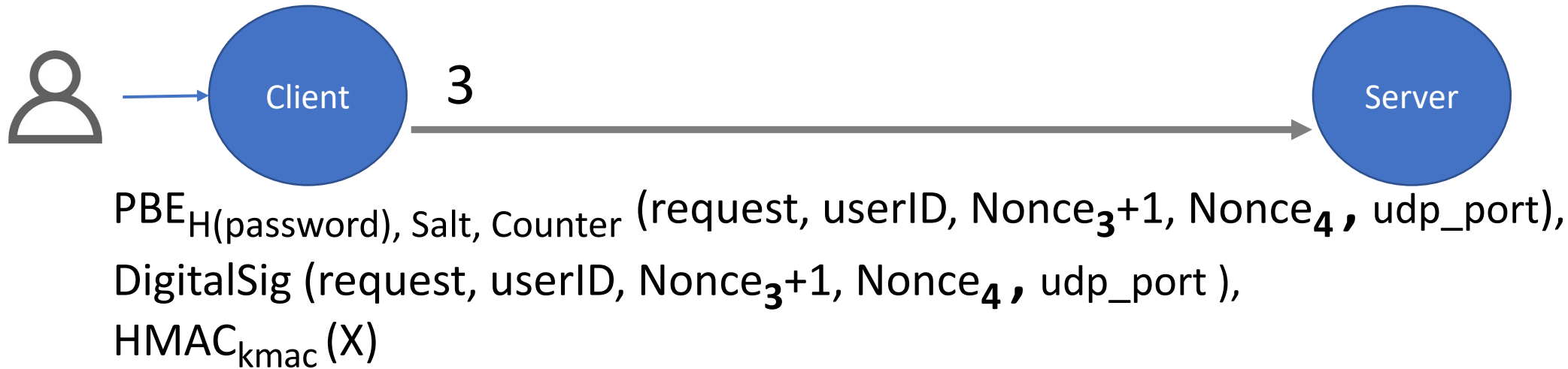
MsgType 1: 320 bytes max.  
userID, String (until 320 bytes)

MsgType 2: 48 bytes

Nonces: Secure Randomly Generated by the Server, 128 bits each one

# SHP Message Flow: SHP – Phase 1

- Message Flow: Message type 3



**MsgType 3 size:** Size depending on used cryptographic constructions in message components

**request:** the request, according to the application (ex., movie or files to transfer)

**PBE()** : Must choose a secure PasswordBasedEncryption scheme

**DigitlSig()** : an ECDSA Signature, made with the client ECC private key (with a selected curve)

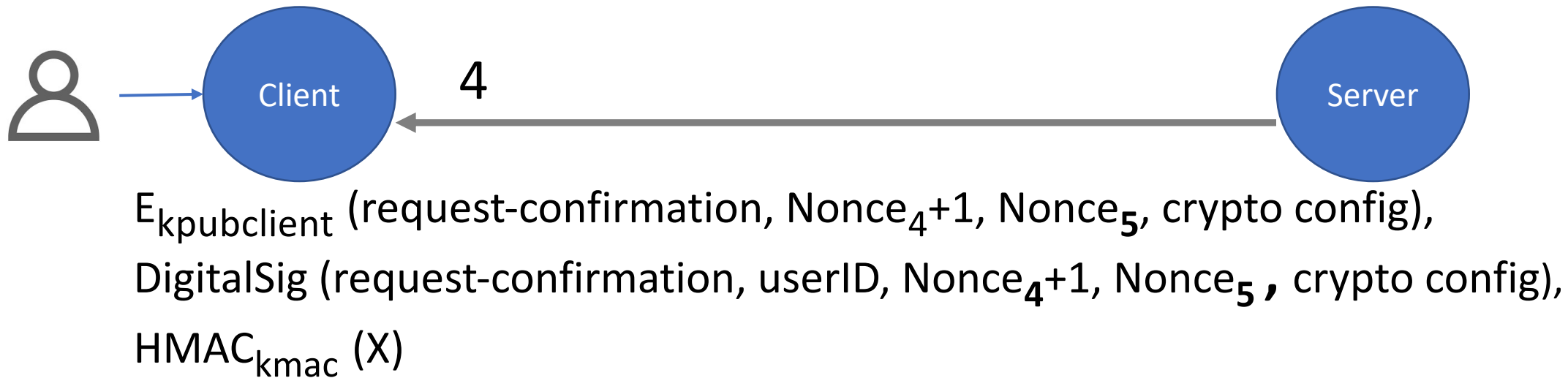
**HMAC()**: Must choose a secure HMAC construction, with the kmac derived from the password

**X:** the content of all (encrypted and signed) components in the message, to allow a fast message authenticity and integrity check



# SHP Message Flow: SHP – Phase 1

- Message Flow; Message type 4



MsgType 4 size: size depending on used cryptographic constructions

Request-confirmation: confirmation of client request (message type 3)

DigitalSig: an ECDSA Signature, made with the client ECC private key (with a selected curve)

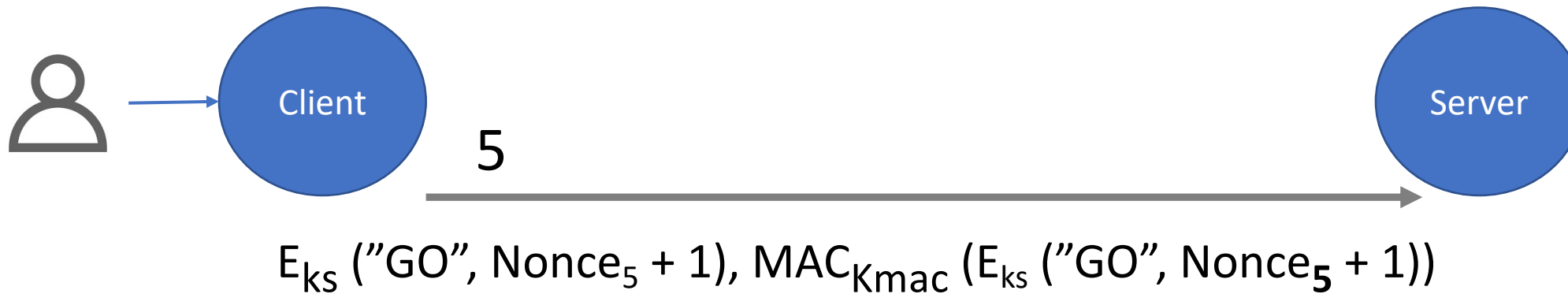
Must choose a secure HMAC construction, with the  $k_{mac}$  as used in MsgType3

Crypto config: datatype to send the Crypto configurations (**ciphersuite.conf**)

**X**: the content of all (encrypted and signed) components in the message, to allow a fast message authenticity and integrity check

# SHP Message Flow: SHP – Phase 1

- Message Flow; Message type 5



**MsgType 5:** is just a synchronization message, the Keys for  $E()$  and  $\text{MAC}()$  are those received in crypto config (in ciphersuite.conf) sent from the server in MsgType 4).

**MsgType 5 size:** depends on the used cryptographic constructions, with  $K_s$  and  $K_{\text{mac}}$  depending on the configurations in ciphersuite.conf

In this message, the client informs the server that it will be able to use the established cryptographic configurations (and included keys), as well as proving knowledge of the cryptographic keys to use with the symmetric cryptographic constructions, the symmetric cryptographic algorithms, and the MAC constructions (HMAC or SHA) that have been established.

# SHP Message Flow: SHP – Phase 1

## Summary of cryptographic constructions

- PBE (): can use any PBE construction (from Native Crypto Libraries or from Bouncy Castle Crypto Provider)
- Note that the “password” used for PBE construction is in fact a Secure Hash (using SHA-256) of the original password provided as user input
  - Note that this is required, because the server only has in the user database the Hash of Passwords in entries corresponding to each user
- For Digital Signatures, you must use ECDSA (selecting a secure curve)
- For HMACs, you must use a secure construction with SHA-256 (from Native Crypto Libraries or from Bouncy Castle Crypto Provider)
- Note that compared with PA1, there are no cryptographic parameterizations (*corresponding to ciphersuite configurations*) in the client side. These configurations only exist in the setup on the server side !
  - These cryptographic parameterizations will be established in the client, after the successful termination of the Client/Server SHP protocol

# SHP Message Flow: SHP – Phase 1 (cont)

## Summary of cryptographic constructions (cont.)

- $E_{k_{pub}} ( )$  : A Secure envelope using the public key  $K_{pub}$ , using ECC-ECIES
  - Note: this is used in MsgType 4, where the server uses the client Public key to send the confidential content, so only the correct client will be able to receive the included cryptographic configurations and related keys

# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# SHP Integration with the previous PA1

# How to launch the “Client” and the “Server” for the Streaming Service (integration in the PA1 context)

The Client is the Proxy

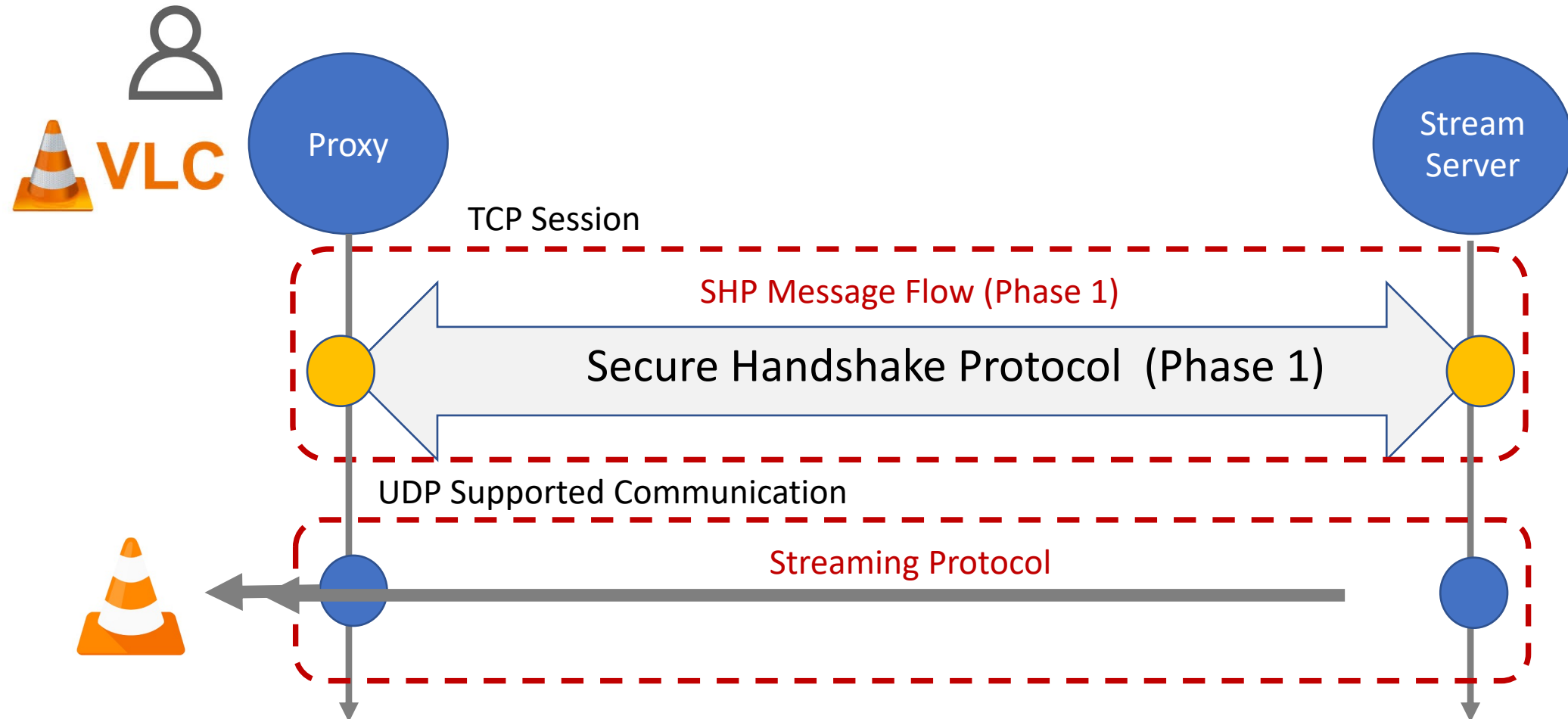
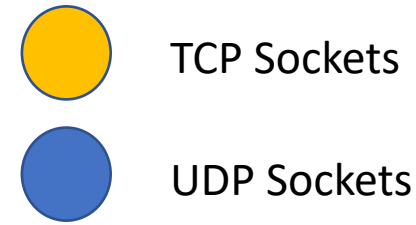
The Server is the Streaming Server

As you see in the next slides, there are some differences in the way you must launch the Proxy and the Streaming Server, for the integration with the SHP Protocol.

You must understand the differences, taking into consideration your initial solution for PA1

# SHP – Phase 1

- Integration with the Streaming Service





# How to launch the “Client” for the Streaming Service (from the PA1 context)

The Client is the Proxy

```
$ java Proxy <username> <password> <server> <tcp_port> <movie> <udp_port> <player_port>
```

**username:** username as registered in the user database, server side

**password:** user password

**server:** the server host machine (DNS name) or IP address

**tcp\_port:** the tcp port where the server is waiting

**movie:** the requested movie

**udp\_port:** the udp\_port where the client will receive the movie for real-time playing

**player\_port:** the udp\_port of the player that will play the streamed movie

# How to launch the “Server” for the Streaming Service (from the PA1 context)

The Server is the Streaming Server

```
$ java StreamServer
```

Note that now, it is not necessary to pass any argument to the Stream Server. You must know why.

# Important qualitative factor

- Independently of the required changes in the code of Proxy and Stream Server (compared with the solution on your PA1), due to the different input arguments, other changes in the code of Proxy and Stream Server must be minimized
- In the limit (best solution): for the integration of SHP, it will be required only to add 1 more line in the code:
  - A new primitive in the Proxy: **shp\_phase\_1\_client(... )**
  - A new primitive in the Streaming Server code, that will fire the execution of the SHP protocol: **shp\_phase\_1\_server( ... )**
- You must design the specification for these primitives (in terms of required input parameters), and you must implement these primitives in specialized classes

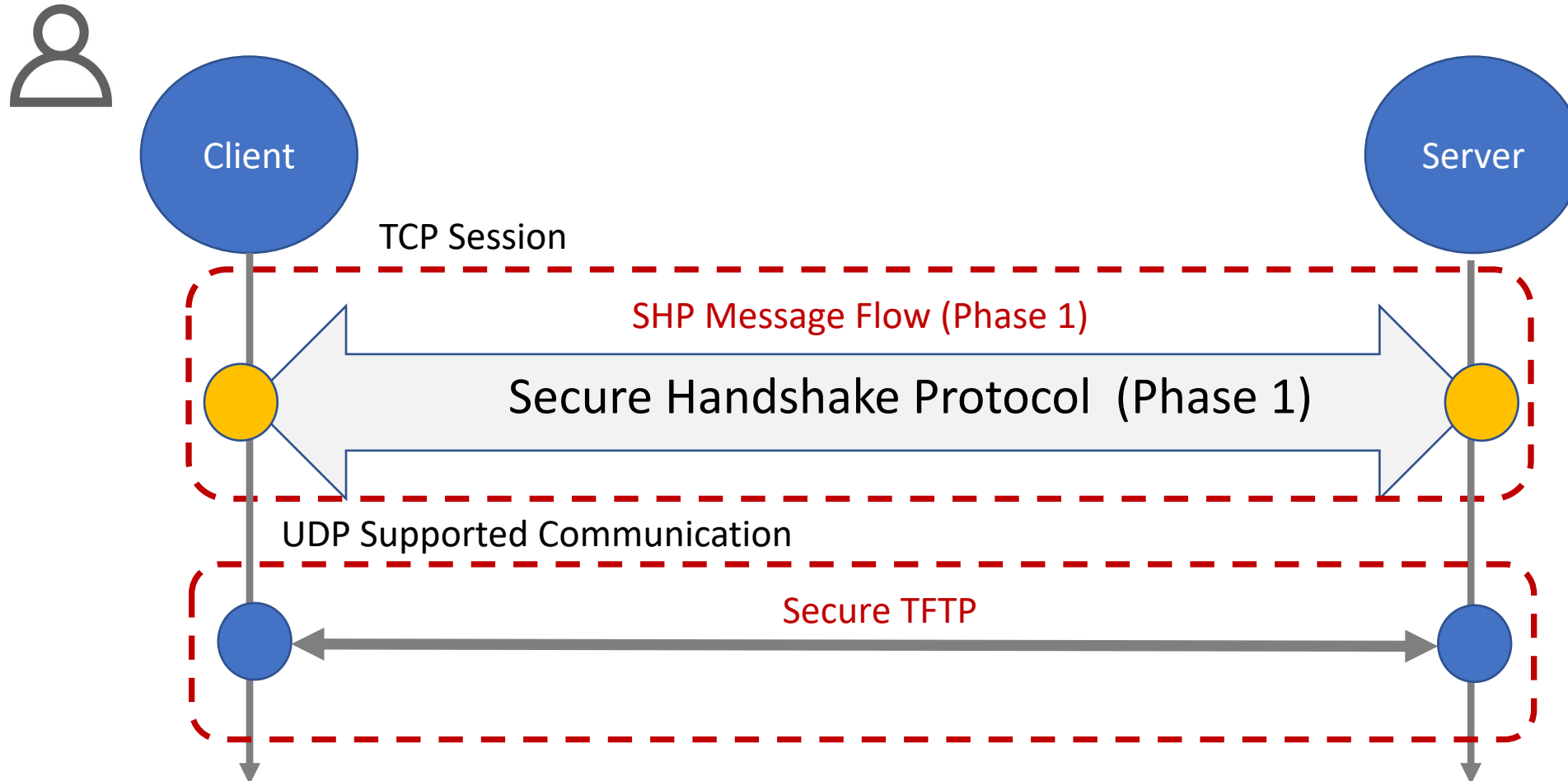
# How to launch the “Client” and the “Server” for TFTP

The Client is the TFTPClient

The Server is the TFTPServer

# SHP – Phase 1

- Integration with the TFTP Application



# How to launch the “Client” and the “Server” for the TFTP Application (integration in the PA1 context)

The Client is the TFTPClient

The Server is the TFTPServer

As you see in the next slides, there are some differences in the way you must launch the TFTPClient and the TFTPServer, for the integration with the SHP Protocol.

You must understand the differences, taking into consideration your initial solution for PA1

# How to launch the “Client” for TFTP

The Client is the TFTPClient

```
$ TFTPClient <username> <password> <server_host> <tcp_port> <type_r_w> <filename> <mode>
```

**username:** username as registered in the user database, server side

**password:** user password

**server\_host:** the hostname that runs the TFTPServer

**tcp\_port:** the tcp port where the server is waiting to execute the SHP protocol

**type\_r\_w filename, mode:** the same arguments as used in Project Assignment 1

# How to launch the “Server” for TFTP

The Server is the TFTPServer

```
$ TFTPServer <tcp_port>
```

**tcp\_port:** the tcp port where the server is waiting to execute the SHP protocol



# Important qualitative factor in the SHP Phase 1 integration with the previous PA1 applications

- Independently of the required changes in the code of Proxy and Stream Server (compared with the solution on your PA1), due to the different input arguments, other changes in the code of Proxy and Stream Server must be minimized
- In the limit (best solution): for the integration of SHP, it will be required only to add 1 more line in the code:
  - A new primitive in the Proxy: **shp\_phase\_1\_client(... )**
  - A new primitive in the Streaming Server code, that will fire the execution of the SHP protocol: **shp\_phase\_1\_server( ... )**
- You must design the specification for these primitives (in terms of required input parameters), and you must implement these primitives in specialized classes

# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 2 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# SHP – Phase 2 Specification

# Differences and similarities from PA2 Phase 1 to PA2 Phase 2

- PA2 Phase 2 is an extended version of SHP – Phase 1
- The security properties and guarantees as initially provided by the SSH-Phase 1 are maintained, but confidentiality will be implemented in a different way
  - **User Authentication / Peer-Authentication:** SHP Phase 2 must provide the same guarantees as provided by SHP Phase 1
  - **Integrity Requirements:** Same guarantees as in SHP – Phase 1
  - **Message-Replaying:** same guarantees as in SHP – Phase 1
  - **Confidentiality guarantees** implemented differently, using a Diffie-Hellman agreement and using ECC-ECIES Constructions
- Modifications in message exchanges

# SHP Message Flow: SHP – Phase 2


- Message Flow: Message Types 1 and 2: same as in SHP Phase 1



# SHP Message Flow: SHP – Phase 2

- Message Flow: Message Type 3 (see comparing the content with Phase 1)

3



$PBE_{H(\text{password}), \text{Salt}, \text{Counter}}(\text{request}, \text{userID}, \text{Nonce}_3+1, \text{Nonce}_4, \text{UDP\_PORT}),$   
 $\text{ydh-client},$   
 $\text{DigitalSignature}_{K_{\text{privClient}}}(\text{request}, \text{userID}, \text{Nonce}_3+1, \text{Nonce}_4, \text{UDP\_PORT}, \text{ydh-client}),$   
 $\text{HMAC}_{K_{\text{mac}}}(\text{X})$

Ydh-client: public Diffie-hellman number, dynamically generated by the client for a Diffie-Hellman agreement with the Server

Note: it will be crucial to include the public Diffie-Hellman number ydh-client as one of the elements in the Digital Signature, for the security of the Diffie Hellman agreement.

The other cryptographic constructions are the same as in SHP-Phase 1

# SHP Message Flow: SHP – Phase 1

- Message Flow: Message Type 4 (se, comparing the content with Phase 1)

4



$E_{k_{pubclient}}$  (request-confirmation,  $Nonce_4+1$ , Nonce 5, crypto config),

Ydh-server,

$DigitalSignature_{K_{privServer}}$  (request-confirmation, userID,  $Nonce_4+1$ , crypto config, Ydh-server)

$HMAC_{k_{mac}}$  (X)

Ydh-server: public Diffie-hellman number, dynamically generated by the server for a Diffie-Hellman agreement with the Client

The other cryptographic constructions are the same as in SHP-Phase 1

Note that ydh-server is one of the elements in the digital signature from the server

Note that now, in crypto config information, the server will not include any keys or IVs for the sent parameterizations. These values will be generated from the Diffie-Hellman agreement

# SHP – Phase 2: Cryptographic constructions

- Password-Based Encryption using SHA256
  - Similar, as in SHP Phase 1 specification
- Digital Signatures: ECDSA
- HMAC: use HMAC with a SHA 256 construction
- Diffie Hellman Agreement
  - Private and public numbers generated by the Client and the Server must have 2048 bits
  - Use pre-defined / Pres-shared primitive root and prime number in Client and Server Setup, for the Diffie Hellman Agreement
  - After Message-Type 4, Client and Server must establish a secret  $K_s$ , from which all keys (symmetric keys or HMAC keys) must be derived



# Start the SHP handshake in application code

- Must implement the SHP to start **with one only additional instruction** you can add in the code of the two applications:

Client side:

`client_shp_phase2` (.... with the required arguments ...)

Server side:

`server_shp_phase2` (... with the required arguments ...)

# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 2 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# How to launch the “Client” and “Server” for the Streaming Service and for the TFTP Application (SHP Phase 2 Integration)

In the same way as in SHP Phase 1:

Slides: 35 to 48

# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 2 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# Quality factors of the implementation

- Generic solution, to be used with minimal modifications of Streaming Service or TFTP applications already used in Project Assignment 1
  - The minimum required modifications, the best is the Project Assignment 2 solution
- Correctness, demonstration and experimental observation of the SHP protocol, for the support of both application-demonstrators: Streaming Service and TFTP
- The solution must be able to operate with different cryptographic configurations (cyphersuites.conf) inherited (and already supported) from the PA1 implementation

# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 2 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# Evaluation Criteria: SHP Phase 1 can achieve 16/20 Points

- SHP-Phase 1 correctness with demonstration of operation (in discussion/demonstration sessions with each student or group)
  - Until 10 Points
- Integration demonstration with correct operation of SHP-Phase 1, to support the Streaming Service (Application)
  - Until 2 Points
- Integration demonstration with correct operation of SHP-Phase 1, to support the TFTP (Application)
  - Until 2 Points
- Qualitative factors of SHP-Phase 1 Implementation
  - Until 2 Points

# Evaluation Criteria: SHP Phase 2 can achieve 20/20 Points

- SHP-Phase 2 correctness with demonstration of operation (in discussion/demonstration sessions with each student or group)
  - Until 14 Points
- Integration demonstration with correct operation of SHP-Phase 2, to support the Streaming Service (Application)
  - Until 2 Points
- Integration demonstration with correct operation of SHP-Phase 2, to support the TFTP (Application)
  - Until 2 Points
- Qualitative factors of SHP-Phase 1 Implementation
  - Until 2 Points



# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 2 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# Important Dates

- Period for Project Assignment Development, Tests and Experimental Observation of Correctness and Operation
  - **From 26/November/2024 to 11/December/2024**
  - **Note: students/groups must plan and schedule the implementation process for the best agenda in each case**
- Delivery / Submission period
  - **From 4 to 11 December/2024**
  - **Deadline date: 11/December/2024, 23h59 (Lisbon time)**
- Slots will be available for student's demonstrations and for discussion of developed solution in the following period
  - From 12 to 21 December
  - Possible complementary period: From 3 to 8 January/2025

# Discussion: PA2 Requirements

## Outline

- Context and Motivation
- SHP Protocol ad integration with PA1 Solution: Integrated Communication Support Stack
- Implementation phases
- SHP Generic Protocol Model and Architecture
- SHP Requirements in Phase 1 and Phase 1 Setup
  - Integration of SSH Phase 1 in the context inherited form the PA1 Solution
- SHP Requirements in Phase 2 and Phase 2 Setup
  - Integration of SSH Phase 2 in the context inherited form the PA1 Solution
- Improvements and qualitative factors
- Evaluation criteria
- Important dates
- Delivery process

# Delivery Process

- The same as in Project Assignment 1
  - GitHub Repository with the project implementation, shared with “henriquejoaolopesdomingos” in the delivery/submission moment
  - Google Form that must be filled to complete the submission
    - **The form will be available after 28/November**
    - **It will be open until 12/December/2024 (with no penalization)**