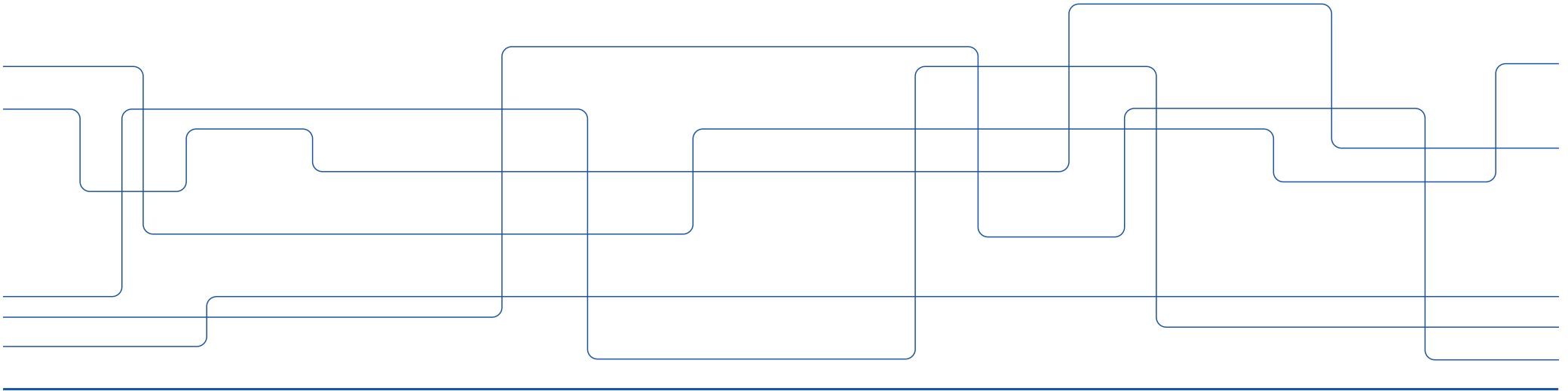# Project Introduction

Peter Sjödin

psj@kth.se

Overview

- Background

- Software support modules

- Assignment organization

NetPipe

- Java implementation of "netcat" (or "nc")
  - See https://netcat.sourceforge.net
- Read data from system input, write to TCP socket
- Read data from TCP socket, write to system output

How to Run NetPipe Client and Server

- Base version, without security

```
$ java NetPipeClient --host=<host name> --port=<port number>


$ java NetPipeServer --port=<port number>
```

NetPipe in Terminal Window

```
$ java NetPipeClient --host=serverhost --port=6789
Hello are you there?
Yes, I'm here.
```

Server terminal

```
$ java NetPipeServer --port=6789
Hello are you there?
Yes, I'm here.
```

File Transfer with NetCat

Client terminal

```
$ java NetPipeClient --host=serverhost --port=6789 <infile.txt
```

Server terminal

```
$ java NetPipeServer --port=6789 >outfile.txt
```

Copy "infile.txt" on client host to "outfile.txt" on server host

Use as Client for Existing Server

```
$ java NetPipeClient --host=mx.kth.se --port=25
220 mx-3.sys.kth.se ESMTP Postfix
```

Connect to incoming mailserver on port 25 (SMTP)

Use as Server for Existing Client

```
$ java NetPipeServer --port=8080
GET / HTTP/1.1
Host: serverhost:8080
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.4 Safari/605.1.15
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Visit server "serverhost:8080" with web browser

Project Assignment

- You are provided with an implementation of NetPipe without security

- Your job is to extend NetPipe with security
  - Handshake phase
  - Encrypted session

Secure NetPipe through Handshake

Authenticate client and server
- Exchange client and server certificates

↓

Establish Session Key
- Session key and IV for AES in CTR mode
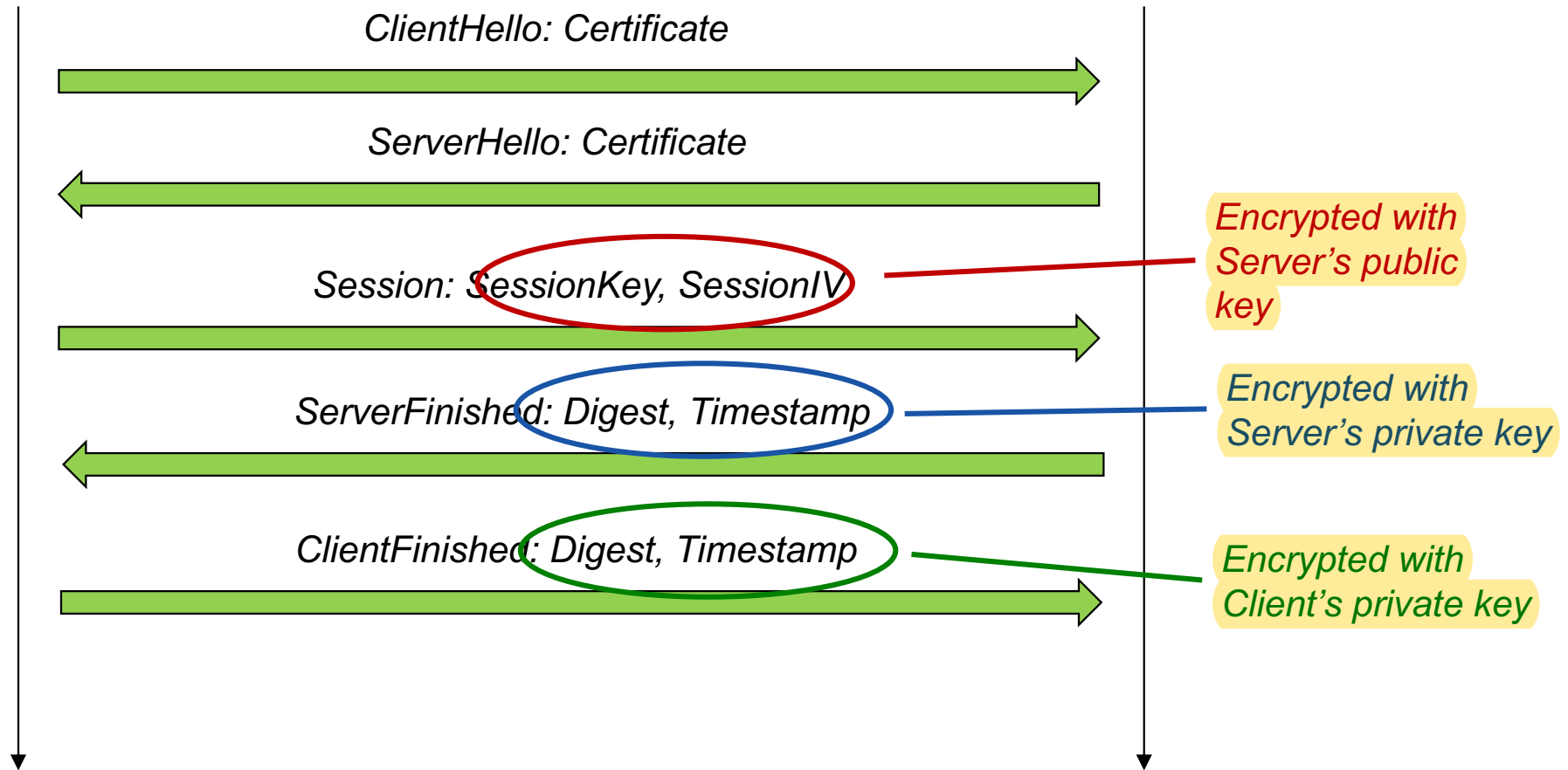
↓

Verify integrity of handshake
- Digests of messages

↓

Switch to session mode

Handshake Protocol

Client   Server

ClientHello: Certificate →

ServerHello: Certificate ←

Session: *SessionKey, SessionIV* → — *Encrypted with Server's public key*

ServerFinished: *Digest, Timestamp* ← — *Encrypted with Server's private key*

ClientFinished: *Digest, Timestamp* → — *Encrypted with Client's private key*

HandShakeMessage class

```
HandshakeMessage(MessageType messageType)
```

```
public MessageType getType()

public void putParameter(String parameter, String value)

public String getParameter(String parameter)

public void send(Socket socket)

public static HandshakeMessage recv(Socket socket)

public byte[] getBytes()
```

HandShakeMessage Class Example

```
HandshakeMessage clienthello = new HandshakeMessage(HandshakeMessage.MessageType.CLIENTHELLO);

clienthello.putParameter("Certificate", encodeCertificate(clientcert));

clienthello.send(socket);
```

Receive a message

```
HandShakeMessage fromclient = HandshakeMessage.recv(socket);

if (fromclient.getType().equals(HandshakeMessage.MessageType.CLIENTHELLO)) {

    String encodedCertificate = fromclient.getParameter("Certificate"));

    …

}
```
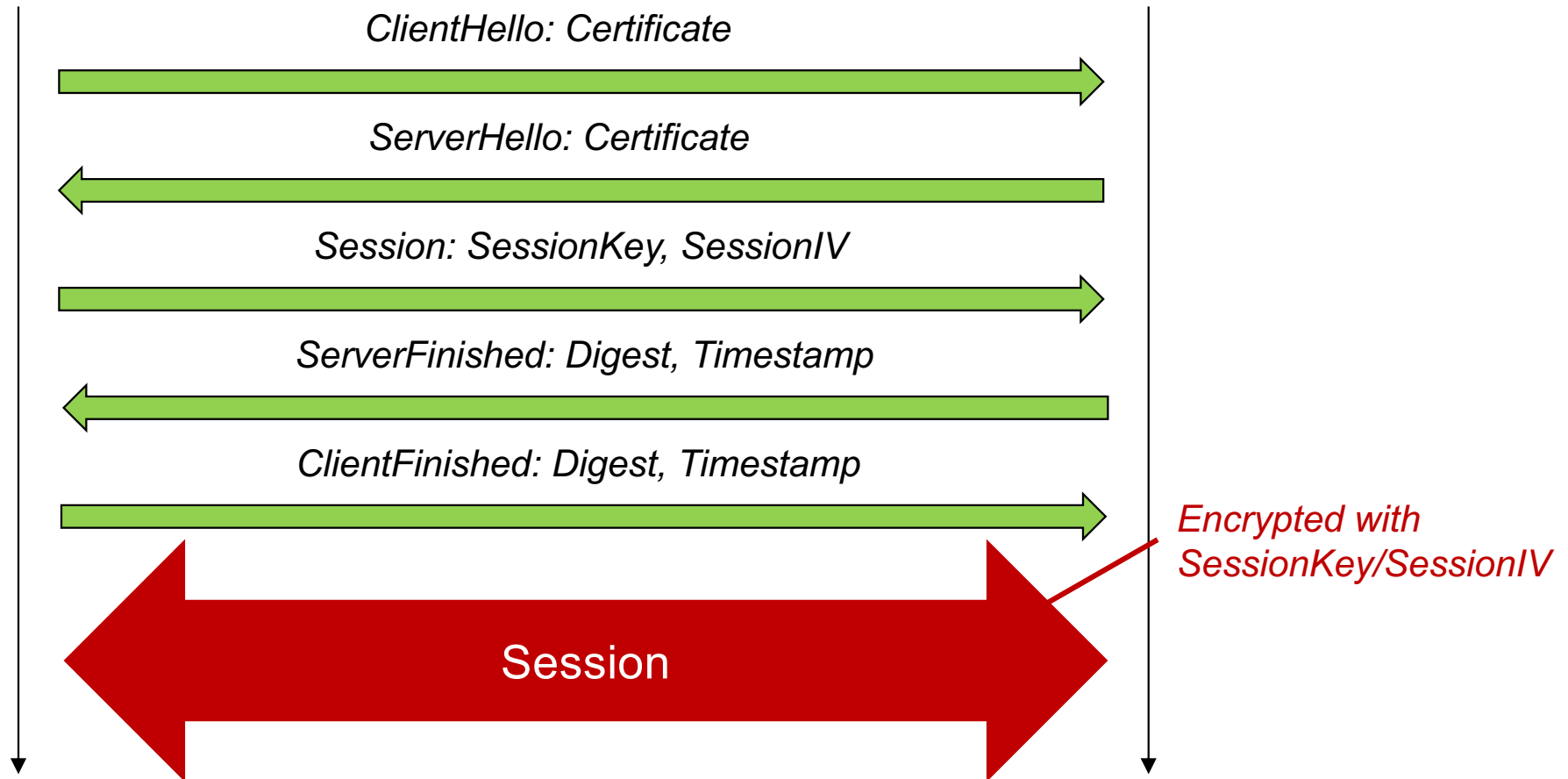
Handshake Messages Implementation

- Extension of Java "Properties" class
  - Use Properties methods for further inspection and debugging (if you need to)

- Use Java's ObjectOutputStream and ByteArrayOutputStream for serialization
  - Turn object into byte array

Session

Client   Server

ClientHello: Certificate

ServerHello: Certificate

Session: SessionKey, SessionIV

ServerFinished: Digest, Timestamp

ClientFinished: Digest, Timestamp

Session

Encrypted with SessionKey/SessionIV

How to Run NetPipe Client and Server

- With security

```
$ java NetPipeClient --host=<host name> --port=<port number>
    --usercert=<client certificate PEM file>
    --key=<client private key DER file>
    --cacert=<CA certificate PEM file>



$ java NetPipeServer --port=<port number>
    --usercert=<server certificate PEM file>
    --key=<server private key DER file>
    --cacert=<CA certificate PEM file>
```

Implementation

- Much code at your disposal – you job is to put it all together

- Results from preparatory tasks

  – SessionKey, SessionCrypto, HandshakeDigest, …

- NetPipe classes

  – Without encryption

  – TCP connection forwarding data between system input/output and TCP socket

    > *bidirectional forwarding*

- HandShakeMessage class

  – Encoding, decoding, and transmission of handshake messages

Organisation of Project Assignment

- Implement secure NetPipe client and server in Java

- Individual assignment
  - You may collaborate, but each student submits his/her own solution
  - Submit according to instructions
  - Graded Pass/Fail

- Supervision sessions next two weeks if you need help
  - Sign up in Canvas
  - If no students have signed up before 18:00 the day before, we will cancel

- Make-up opportunity
  - If you made a serious attempt at submitting before the due date, you will get feedback and a chance to improve
  - Due date after exam

Evaluation

- First, your implementation will be checked for basic functionality

- However, you are really implementing a communication protocol here

- Therefore, your implementation must interoperate with other implementations
  - We will test your code against a reference implementation
  - For example, your NetPipeClient against our reference NetPipeServer

- So, make sure to follow the instructions in detail
  - Exact spelling in handshake messages, for instance

Evaluation (2)

- Organize your submission according to instructions
  - If you use some other organization, for instance with extra packages, your code will not run and you fail
- If you fail, but have made a serious attempt to solve the assignment by the due date, you will get feedback and another chance

To Consider

- In the preparatory tasks, we have "micro-managed" you with given classes, methods, etc.

- Now it is up to you to organize your implementation

- Put the different pieces together into a running systems
  - Tasks
  - Handshake messages
  - Classes, methods,

- How you do this is up to you!
  - You are free to add classes, methods, etc

To Consider (2)

- Submit in time
  - If you don't submit before due date, you won't get a grade
  - It is the content of your KTH GitHub repository at the due date that counts
    - *Any changes pushed after that date will be ignored*

- Submit your own solution
  - We will check for plagiarism
    - *Really. We will.*

- Test before you submit

- Run with reference implementation
  - Your client with reference server
  - And vice versa

Good Luck!

Questions?