Integrated MSc Course on Informatics Engineering, DI/FCT/UNL
Computer Networks and Systems Security / Semester 1, 2019-2020

Practical Evaluation: Work Assignment #1

A Peer-Group Oriented Chat using Secure Multicast Communication Channels

Submission deadline: 31/October/2019, 23h59
(Evaluation criteria and Submission Process in Annex)

Summary

The objective of the project assignment is the design and implementation of a Secure Peer-Group Oriented Chat System, supported by Secure IP Multicasting Communication Channels. The work involves: the design analysis; development; and the preparation of a deployment package for demonstration. The system will be addressed from a provided Peer-Group Multicast Chat-System (with an initial implementation in Java language), not supporting the intended secure guarantees.

1. Project Description

The system implementation requires the development and setup of the following protocols, as two main requirements addressed in two project phases (with the evaluation criteria in the Annex 1, and with the work deliver process in Annex 2):

- **Phase 1: SMCP - Secure Multicast Communication Protocol.** The SMCP security services will protect message-flows and message-exchanges in chat sessions, implementing secure IP multicast channels used by a Secure Peer-Group Chat application (**SecureMChatClient**). The SMCP and related cryptographic constructions, parameterizations and setup configuration options, will provide a set of required security properties, defined from the OSI X.800 Security Framework conceptual reference.
- **Phase 2: SAAHP - Session Authentication, Authorization and Handshaking Protocol and SAAH Server. The** SAAHP and the SAAHPServer will provide services for authentication and access control of peers (users) participating in Chat sessions, as well as, support for dynamic installation and setup of session configuration data and security association parameters. The phase 2 is addressed as an evolution of the **SecureMChatClient** initially developed in the Phase 1, providing dynamic cryptographic parameterizations and configurations for a smooth integration with the SMCP initially developed.

2. System Model, Architecture and Components

The system model for the project is composed by the following components and related entities: (i) an initial application used by peers (user-entities) supporting the Peer-Group Oriented Chat Sessions (***SecureMChatClient***) implementing the Secure Multicast Communication Protocol (*SMCP*); (ii) and the extension of the initial SecureMChatClient application to implement also the Session Authentication, Authorization and Handshaking Protocol (SAAHP)

The ***SecureMchatClient*** will be implemented in Java language, using the JCA/JCE framework and leveraging from an initial Java application implementation (***MChatClient***) provided as starting material for the project. The provided ***MChatClient*** component already implements a base Peer-Group Chat application, supported on IP Multicasting, without any protection for secure communication services as targeted in the project. The system model for the project in phase 1 and phase 2 is represented in Figure 1.

In the project (phase 1) the ***SecureMChatClient*** must be implemented by extending the UDP Multicasting Sockets (prototyped as ***SMCPSockets***), to support the SMCP protocol encapsulated in IP Multicast

Packets, with the desired security requirements implemented for the SMCP protocol. In this development the key-design criteria must be oriented for the minimization of changes (as much as possible) in the initial code of the provided ***MChatClient,*** In this way, the new application will be protected for using the SMCP protocol, with minimal changes, by the replacement of UDP Multicast Sockets by ***SMCPSockets***. To participate in Chat sessions, users have their own static configurations and cryptographic parameterizations statically defined, with shared contents with all the users involved.

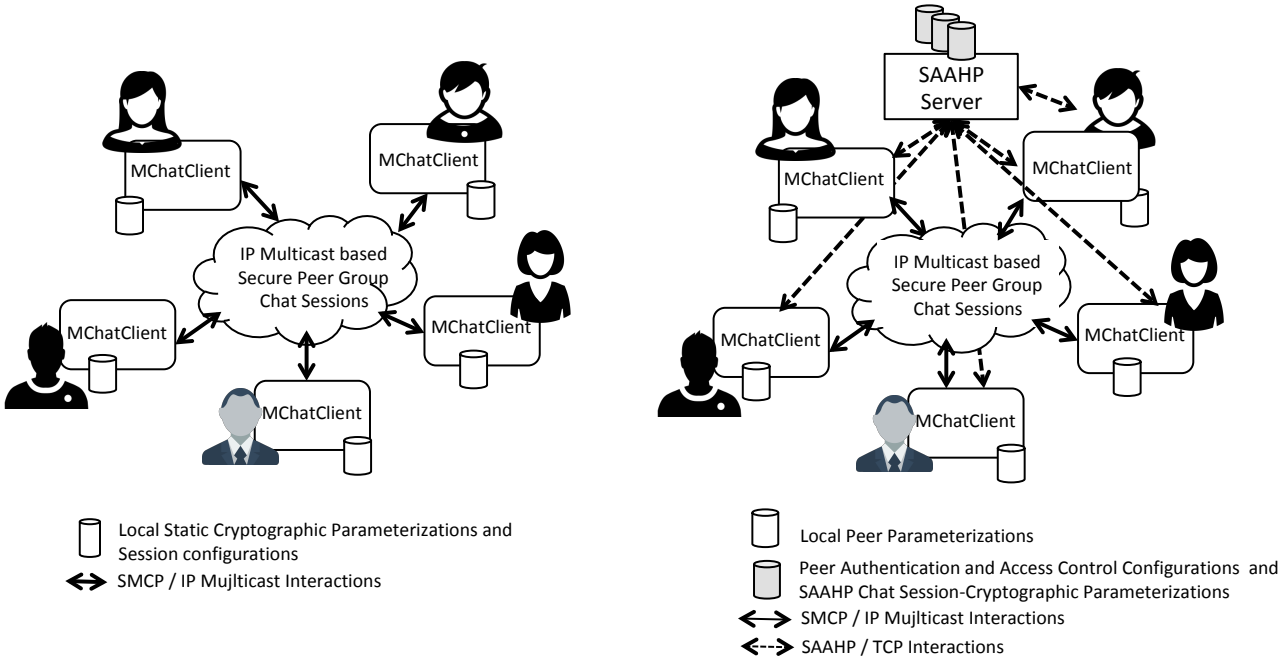

Fig.1 System Model for the Peer-Group Chat in Phase 1 (left) and Phase 2 (right)

In the phase 2 the MChatClient application will support the SAAHP/TCP protocol, as an handshaking protocol supported by TCP for the interaction with the SAAHP Server. This server provides services for authentication and access control of peers (users) interested in joining the Chat sessions. The SAAHP protocol also supports the dynamic setup of session configuration data, cryptographic parameterizations and security association parameters for users authorized to join the Chat sessions. In this phase, the ***SecureMChatClient*** will be extended from the first phase implementation, to use the SAAHP security services,

The implementation of the **SecureMchatClient** component in phases 1 and 2 must meet the requirements presented in the next sections, namely: the adversary model assumptions (section 3); the compliance and integration with the SMCP requirements and parameterizations (section 4), the compliance with the SAAHP specifications and parameterizations (section 5). For the implementation of those phases it is suggested to follow a sequential methodology (section 6).

3. Adversary Model Considerations

The adversary model for the project follows the typology of communication attacks against the underlaying channels supporting the SCMP and SAAHP, respectively IP Multicast channels and TCP/IP channels, as defined in the OSI X.800 framework (studied in lectures). Then, SCMP and SAAHP will include protection for passive and active communication attacks, requiring the following protections:

- Message confidentiality avoiding release of message contents (based on connectionless confidentiality arguments for SCMP and connection-oriented confidentiality for SAAHP);
- Message integrity (based on connectionless integrity assumptions for SCMP and connection-oriented integrity for SAAHP), protecting from message tampering, as well as, from traffic-flow tampering, including disordering attacks on the respective message flows);
- Authentication control services for message authentication controls
- Message replay protection;
- Protection against masquerading of endpoints or identity spoofing of communicating peers (given the peer names or authenticated digital identifiers)
- Non-repudiation guarantees: each peer will maintain a local log of messages observed in each session (with time-stamping controls and integrity proofs of all observed messages, as well as, the ordering observed integrity guarantees.

The following attacks are out-of-scope of the adversary model definition for the project requirements.

- Traffic analysis with reconnaissance of supported protocols;
- Denial of service attacks causing message-suppressions in the communication channel, channel unavailability by communication disruptions, or unavailability of communicating peer-endpoints and their client applications;

Being out-of-scope of mandatory requirements, students are invited in designing and implemented improved security mechanisms, thought to mitigate (or minimize) the impact of those attacks.

The adversary model assumptions are focused on the protection of multicast communication channels and endpoints processing the SMCP and SAAHP protocols. Intrusion attacks against the computers running the **SecureMChatClient** causing incorrect modifications on the runtime environment, or originated by the injection of malicious code changing the correct behavior of the software execution environment, are not considered in the adversary model definition.

4. Phase 1 – Secure Multicast Communication Protocol (SMCP)

Phase 1 implementation is mandatory for projects submitted for evaluation. In this phase the objective is to develop the **SMCP** and its implementation support in the SecureMChatClient application. To better understand the requirements students must be aware of the implementation initially provided in the MChatClient application in the use of IP Multicasting. The phase 1 is composed by the analysis and implementation of the SMCP message-format, the implementation of SMCP in SMCPSockets extending the IP Multicast Sockets and the use of SMCP Sockets in the SecureMChatClient application, with the required cryptographic parameterizations and CHAT-Session configurations.

4.1 SMCP Message Format

Messages in chat sessions are protected in the following way, as overlaying exchanged messages and content-parts encapsulated in base IP-Multicast datagrams:

vId || *sID* || SMCPmsgType || SAttributes || *SizeOfSecurePayload* || *SecurePayload* || *FastSecureMCheck*

- *vID*: byte identifying a version number of the protocol
- *sID*: an unique identifier for the Chat session. It can correspond (for example) to a string in the form "IP-Multicast-Address:Port" corresponding to the multicast address and port of the session, or a string with the equivalent dot-notation address used in each session
- *SMCPmsgType*: a byte corresponding to a specific message type (ex., 0x01 for the case of SMCP messages format.

- *SAttributes*: it is the hash value (ex., SHA-256) of a list of session attributes (as session meta data). Suggested fields are the following:
  - *sID*
  - *SessionName*: String name (external designation)
  - *SEA* - Symmetric Encryption Algorithm: String
  - *MODE* – Mode used for symmetric encryption operations; String
  - PADDING: padding method: String
  - H: HashFunction used to compute the hashed *IntegrityControl value contained* inside the *SecurePayload*)
  - MAC: MAC (HMAC or CMAC construction) used to compute the *FastSecurePayloadCheck* component.
- *SizeOfPayload*: an explicit control of the length of the SecurePayload part (number of bits)
- *SecurePayload*: An opaque byte-stream, encrypted with a symmetric encryption algorithm and a session-shared key Ks (encrypting the chat message-data and other fields), forming an encrypted payload with the following reference format:

  $\{FromPeerID \parallel SeqNr \parallel RandomNonce \parallel Message \parallel IntegrityControl\}_{Ks}$

  FromPeerID: The ID of the sender (it can be a String representing a userID or user nickname)
  SeqNr: A sequence number of the message as ordered by the sender
  RandomNonce: A randomly generated nonce that can be checked if needed by the receivers
  Message: the original plaintext message-data sent by the sender participant to the chat participants. It can be a message of anytype, but in the implementation we can address only text-messages (Strings or byte-arrays).
  IntegrityControl: The Hash of the plaintext message

- *FastSecurePayloadCheck;* A value computed over the entire SMCP message, using an HMAC or CMAC function (as expressed in the *SAttributes*), expressed in a byte-*array*.

4.2 Cryptographic Parameterizations and Configurations of SMCP Endpoints

For the requirements of Phase 1, the SMCP endpoints (beyond the implementation of SMCPSockets) require the following parameterizations and configuration files:

- A KeyStore (type JCEKS or PKCS12) in a file named ***SMCPKeystore.jecks*** with all the sensitive information for symmetric keys and MAC keys, according to the security association parameters for each session. A good idea is to use IP Multicast addresses associated to Chat sessions as index-key-identifiers for entries in the *KeyStore*
- A text-based configuration file (manually configured), named ***SMCP.conf***, with the following format.

```
<IPMC:PORT>
  <SID>String: Session Identifier</SID)
  <SEA>String: Name of the Symmetric Block-Encryption</SEA>
  <SEAKS>Integer: Key size for the SEA Algorithm</SEAKS>
  <MODE>String: Block Mode for Symmetric Encryption</MODE>
   <PADDING>String: Padding Standard Identification</PADDING>
   <INTHASH>String; Name of Secure Hash Integrity Function</INT-HASH>
<MAC>String: Name of the MAC Construction</MAC>
<MAKKS>Integer: used Key size for the MAC Construction</MAKKS>
</IPMC:PORT>
```

The ***SMCP.conf*** file can have multiple entries for more than one Chat session, using different parameters, expressing the required parameterizations for each session. A configuration example with the proposed format, with entries for three sessions, is provided in annex (Annex 3).

5. Phase 2 – Authentication and Dynamic Establishment of Security Association Parameters

The Phase 2 implementation is not mandatory for work submission but its valorization is part of the work assessment (see Annex 1). For the implementation of the Phase 2, there are only initial recommendations for design and implementation criteria (Annex 4). The complete Phase 2 specifications must consider the initial criteria. Students must design their own specifications for Phase 2, considering the initial requirements. The phase 2 involves two main components (that can be addressed as two separated objectives, that can be developed and delivered individually):

- The specification proposal for the SAAHP Protocol, formalized by the students in the final report, with the related setup, configurations and cryptographic parameterizations required by the SAAHP Server and SecureMChatClient applications to run the protocol in the Phase 2.
- The implementation of the designed SAAHP protocol, SecureMChatClient and SAAHServer components, as the implementation materials of the Phase 2.

6. Proposed Development Methodology

The suggested methodology addresses the following steps:

1) Students must develop initially the support for **Secure Multicast Communication** and parameterizations, for the first approach of the new ***SecureMChatClient*** application in the first phase. For this effort, the focus must be the implementation of the SCMP Protocol, in ***SMCPSockets***, extending the base functionality of Multicast UDP Sockets (initially used by the provided ***MChatClient*** application), with the sockets initialized with the SMTP Parameterizations in JECKS keystores and smtp.conf configuration files.

2) The first implementation students must test the implementation for different cryptographic configurations, to investigate the robustness and flexibility in the provided security configurations for the implemented SMCP Protocol. For the work delivering and proof-of-correctness, specific test configuration files will be provided.

Note that for steps 1 and 2 address the mandatory Phase 1. After the step 2 (Phase 1 finished) students can go for the Phase 2 (optional), with the following sub-steps:

3) Students must design the protocol specification and setup conditions for the SAAHP protocol (as used by the SecureMChatClient application and SAAHPServer). The SAAHP protocol must be formalized in order to be reported for work-delivering and this is independent of the provision of the associated impleentation.

4) From the previous specification, the Phase 2 can be implemented, involving a new version of the ***SecureMChatClient*** application, the implementation of the ESMCPSockets and the ***SAAHPServer***, and the new necessary setup configurations.

Annexes

Annex 1 - Evaluation Criteria
Annex 2 - Work deliverable, submission process and required materials
Annex 3 - Configuration files for SMCP endpoints
Annex 4 – Design criteria for the Phase 2

Annex 1 – Evaluation Criteria

Phase 1: 15/20 points, evaluated in the following way:

- Correctness and complete implementation of the SCMP, according to the initial specifications and valuable extensions introduced by students (until 8/20 points)
- Implementation of SCMPSockets and the minimal impact in necessary changes in the original code provided in the original MChatClient application (until 3/20 points)
- Implementation of the SecureMChatClient application, with the necessary SMCP.conf static configuration and defined cryptographic parameterizations (until 2/20 points)
- Flexibility, robustness and transparency to support different cryptographic parameterizations for the SecureMChatClient application (until 2/20 points)

Phase 2: 5/20 points

- Design and formal specification for the SAAHP protocol (in the work report) with the complete clarification of the specification and setup process including the specification of contents for configuration files and cryptographic parameterizations in the SAAHP server and SecureMChatClient applications (until 2/20 points)
- Correctness and completeness of an implementation with the required parameterizations for the SAAHP protocol, according to the proposed specification (until 3/20 points.

Annex 2 – Work deliverable, submission process and required materials

The work submission (until 31/Oct/2019), via Google Form filled for the submission, requires the following materials available and frozen on the submission date:

Mandatory pieces:

- **SRSC1920-TP1-Phase1**: A GitHub project repository with Source archive and development activity, related to the project implementation for Phase 1, containing the source code and all the required parameterizations and configuration files)
  - Project must be frozen until 31/Oct/2019
  - Project must be shared for cloning with; henriquejoaolopesdomingos, hj@fct.unl.pt
- **SecureMchatClient.jar**: a signed jar application for Phase 1 (and a respective keystore for signature verification) for the executable SecureMChatClient.jar
  - Must be in the a directory called "**SecureMChatClient-Application-Phase1**" in the project repository
- **Report.pdf**: A short report following a reference template that will be available
  - Must be in a directory called "TP1-Report" in the project repository

Optional pieces

- **SRSC1920-TP1-Phase2**: A GitHub project repository with Source archive and development activity, related to the project implementation for Phase 2, containing the source code and all the required parameterizations and configuration files)
- **SecureMchatClient.jar**: a signed jar application for Phase 2 (and a respective keystore for signature verification) for the executable SecureMChatClient.jar
  - Must be in the a directory called "**SecureMChatClient-Application-Phase2**" in the project repository
- **SAAHPServer.jar** a signed jar application for Phase 2 (and a respective keystore for signature verification) for the executable SecureMChatClient.jar
  - Must be in the a directory called "**SAAHP-Server-Phase2**" in the project repository
- **Report.pdf**: A short report, following a reference template that will be available

Annex 3: Configuration files for SMCP endpoints

```
<224.5.6.7:9000>
    <SID>Chat of Secret Oriental Culinary</SID>
    <SEA>AES</SEA>
    <SEAKS>256</SEAKS>
    <MODE>GCM</MODE>
    <PADDING>PKCS5Padding</PADDING>
    <INTHASH>SHA256</INTHASH>
    <MAC>HMacSHA256</MAC>
    <MAKKS>256</MAKKS>
</224.5.6.7>

<252.10.20.30:12224>
    <SID>Secret Chat of the Long Night of Horrors</SID>
    <SEA>RC6</SEA>
    <SEAKS>256</SEAKS>
    <MODE>CTR</MODE>
    <PADDING>NoPadding</PADDING>
    <INTHASH>SHA512</INTHASH>
    <MAC>HMacSHA512</MAC>
    <MAKKS>512</MAKKS>
</252.10.20.30>

<230.100.100.100:6666>
    <SID>Secret Chat of the Long Night of Horrors</SID>
    <SEA>Blowfish</SEA>
    <SEAKS>448</SEAKS>
    <MODE>CBC</MODE>
    <PADDING>PKC5Padding</PADDING>
    <INTHASH>SHA1</INTHASH>
    <MAC>DES</MAC>
    <MAKKS>64</MAKKS>
</230.100.100.100>
```

Annex 4: Design criteria for Phase 2

- The SAAHP Protocol must be implemented as a request/response (SAAHP Request / SAAHP Response) protocol, supported on TCP (with the SAAHP Server implemented as a concurrent server to serve the ***SecureMCastClient*** applications (clients). Alternatively, the SAAHP Protocol can use HTTP support, for the interaction between ***SecureMCastClient*** applications (clients) and the SAAHP Server. The SAAHP server and the SAAHP protocol also can be implemented with a *SAAHP/ REST/HTTP* communication stack (or, if desired, *SAAHP/REST/TLS* or *SSAHP/REST/HTTPS* stack), with the SAAHP server implementing the respective SAAHP server REST endpoints.
- Students are free to choose the programming environment for SAAHP, with the only restriction that the SAAHP protocol and the used cryptographic constructions must be supported.
- The message format for SAAHP Requests / Responses must be designed by the students, using and combining constructions for digital signatures (asymmetric cryptography), confidential envelopes (using Asymmetric Cryptography) and Password-Based Encryption Constructions and MAC Constructions, to protect Peer-Authentication, Data-Confidentilality and Message Integrity guarantees.
- The design of the SAAHP protocol, SecureMCastClient and SAAHP server cannot assume any pre-shared secrets initially configured (including MAC keys or Symmetric Encryption Keys). The only initial configurations are the following:
    - In SecureMCastClients: pre-generated public-key client certificate in a local file, private keys in a local private keystore (for digital signatures) and trusted keystores (with certificates and public keys) of the SAAHP server
    - In the SAAHPServer: pre-generated public key server certificate in a local file, private keys in a local private keystore (for digital signatures) and a trusted keystore with the public key certificates of potential clients. It would be very interesting if the truststore in the server side can have only one root-certificate. In this way, the clients will send certification chains (with the root certificate in the root of the chain), avoiding the storage of many client-certificates in the truststore maintained in the server side.
- The SAAHP server must manage configuration files for User-Registration and Access-Control Lists (to control permissions of clients (users) to access pre-registered Chat Sessions, as well as, cryptographic parameterizations and security association parameters for all the registered Chat sessions, using proper keystores and configuration files for this purpose.
- The only static (pre-initialized) information in the client side must be:
    - Local files with public-key certificates of clients. The client certificates must be sent to the SAAHP server in the SAAHP Request message
    - Focal files for cryptographic parameterizations for the SAAHP protocol (SAAHP.conf), according to the cryptographic constructions that will be used in the SAAHP Request/Response messages
    - A keystore (used as a truststore) with the SAAHServer public key certificate
- No cryptographic parameterizations or security association parameters for the SCMP protocol will be statically managed, locally to the client. All the parameterizations for cryptographic constructions used in SCMP messages must be established, dynamically, distributed in a secure way from the SAAHServer to the SecureMchatClient applications, in the SAAHP Response messages, after the validation and processing of SAAHP Requests to join specific Chat Sesisons registered and configures in the SAAHP server.
- The SAAHP protocol must operate under mutual-authentication assumptions and guarantees for SecureMChatClient applciations (user-clients) and the SAAHServer