Integrated MSc. Course on Informatics Engineering, DI@FCT NOVA
Computer Networks and Systems Security - Semester 1, 2019/2020
WORK-ASSIGNMENT #1 REPORT for Evaluation

A Peer-Group Oriented Chat using Secure Multicast Communication Channels

Authors:
Eduardo Brás Silva (emf.silva@campus.fct.unl.pt),
Rúben André Barreiro (r.barreiro@campus.fct.unl.pt)

Summary/Abstract

Abstract. As we see the recent ***increase in the number of users of social networks and texting/chat applications*** in recent years, the protection of data and messages exchanged between these users is becoming increasingly essential. ***New varieties of attacks and new system failures are discovered daily***, so ***it's crucial design and improve new security protocols***, in order to ***protect and mitigate the attack surfaces of our adversaries and threat agents*** (i.e., ***hackers*** or ***malicious agents***). Our proposal, the ***Secure Multicast Peer Group-Oriented Chat***, offers a proposal of a secure protocol, initially implemented in ***Java*** programming language, in order to protect this kind of applications for adversary models following the typology of communication attacks against ***IP Multicast channels*** and ***TCP/IP channels***, as defined in ***OSI X.800***. It was made a ***design analysis***, ***development*** and the ***preparation*** of an ***initial version of a deployment package for demonstration***, to serve this purpose.

Summary table of the TP1 implementation submitted for evaluation (fill with X), according to the Google Submission Form

Coverage of the performed work TP1	YES	NO	Tested it works well	Tested, doesn't work well	Doesn't work
The work only addressed the implementation of the PHASE 1	X		X		
The work addressed the implementation of the PHASE 1 and in this report we present the design and specification of the SAAHP protocol for the Phase 2	X		N/A	N/A	N/A
The work addressed the implementation of Phase 1 and Phase 2, bit the phases were not integrated		X		X	
The work involved the development and the integration of Phase 1 and Phase 2		X			X
URL of the GitHub Repo Project shared with **henriquejoaolopesdomingos**:	https://github.com/rubenandrebarreiro/secure-multicast-peer-group-oriented-chat				

1. Introduction

It was ***implemented completely the phase 1***, addressing the requested requirements in specification of the project. Our proposal addresses the needed countermeasures for the adversary model defined in the specification of the project, which will be discussed in detail later.

In the specification, initially will exist only a core element, which will be the client. Obviously, as it's being addressed a system of a ***Multicast Peer-Group Chat*** service, it means that more than a ***Client*** can use this service simultaneously.

The ***Multicast Peer-Group Chat*** service offers a set of ***chat sessions*** (i.e., ***groups*** or ***"rooms" of conversations*** between ***Clients*** or ***Users*** of this service, ***exchanging messages between them***). As obvious, it's pretended that the ***Clients*** pretend that ***these conversations are kept in privately*** in the ***chat session***, i.e., only the ***Clients*** or ***Users*** present and currently active (***Online Clients/Users***) in this ***chat session***, can have access to these conversations.

These conversations are made performing an one-to-many communication, where a sender sends a message to all the other active ***Clients/Users*** in the Chat Session, which will be ***continually waiting to listen to new messages*** from any ***Client/User***, also present in the ***Chat Session***. It's pretended to be kept in any ***Chat Session***, the ***log of the exchanged messages*** of it. Any ***Client/User*** will be able to access this log, but only for the exchanged messages after it entered the ***Chat Session***, i.e., joined the ***Multicast IP Address*** corresponding to that ***Chat Session*** and providing the necessary ***parameters*** to be able to use that ***Chat Session***'s ***configurations***. This ***configurations*** and ***parameters*** are kept by the ***Clients/Users***, statically, in a local storage.

Each ***Chat Session*** can be addressed by an ***IP Multicast Address*** which will have some security configurations to be accessed and to guarantee that the messages are exchanged assuming some ***security and privacy principles***, using ***Symmetric Encryption*** each ***Client/User*** that pretend to participate in this ***Chat Session*** will be restricted to use this ***parameters of security***, if want to use this ***Chat Session***.

As in this environment it's assumed a communication channel not secure, the messages exchanged in a ***Chat Session*** will be needed to be layered, using some kind of ***security constructions*** and/or ***combinations of many security constructions***. In other words, each message exchanged in a ***Chat Session***, through the communication channel, will be secured from the moment that a ***Client/User*** sends it to the moment that the other ***Clients/Users*** received it.

It was ***addressed a second designing and initial implementation***, addressing the phase 2, as specified in the project's description. This second implementation can be extended from the first one and only differs in the addition of a second element, which will be a ***Server***. This ***Server*** will be responsible to keep the ***Peer Authentication*** and ***Access Control Configurations*** for each ***Client/User*** which can use this ***Secure Multicast Peer Group-Oriented Chat*** service.

In order to address this new setup, it will be also needed communication between a ***Client/User*** that pretend to join a conversation in a ***Chat's Session*** and the ***Server***, in order to obtain the necessary configurations to access a ***Chat's Session***. This communication assumes the same principle of a communication channel not secure, which also must be needed to be layered, using some kind of ***security constructions*** and ***combinations of many security constructions***, similarly to proposed to in the first implementation. But this secure communications will be made using ***Asymmetric Encryption***, supported also by ***Digital Signatures***.

2. System Model, Architecture and Components

As mentioned previously, the system model considered to phase 1 of the project's specification, ***assumes only communication interactions between the Clients/User***. This ***Clients/Users*** must keep the ***parameters*** and the ***Chat's Sessions*** configurations, locally, in a static fashion. The parameters kept by the ***Users/Clients*** are the ***Cryptographic Parameterizations*** needed to be able to access the ***Chat's Sessions*** and also the ***configurations*** will be the necessary setup to access the ***Chat's Sessions*** which it have rights to do it. Thus, the components addressed to an initial setup of this implementation are the following:

- ***Client***, which will keep the following components:
    - A structure (i.e., a ***configuration file***) to keep the ***Configuration of the Chat's Sessions***, which have rights to access.
    - A structure to keep the ***Cryptographic Parameterization*** to be able to perform secure messages' exchange with the other ***Users/Clients***, which are currently using the ***Chat's Session*** pretended to join and participate.

For the phase 2 of the project's specification, as it assumes also communication interactions between the ***Clients/Users*** and a ***Server***, it's strictly necessary consider some assumptions about this second introduced element and a small and "cirurgical" redesigning of the setup of the phase 1. The function of this second element.

(IMCOMPLETE - TODO complete from this statement)

Use this section to characterize form the of the project system model and components presented in the initial statement, presenting specifically what is different from what is initially presented in the initial statement.
Adversary Model Considerations

3. Adversary model

3.3.1 Phase 1. Check how the adversary model is defined in the initial statement for the phase 1. If you want to mention something different then do so in this section. If not, and if you think the adversary model definition is the same, simply reiterate the conditions of the adversary model from the initial specification.

3.3.2 Phase 2. If your implementation addresses the phase 2, describe what you consider is different from the adversary model for phase 2. You must only consider this section of you implemented phase 2 specifications or specification and implementation. If you consider that the adversary model conditions are exactly the same, you must only say it.

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

4.1 Configuration file for MchatClient applications protected by SCMP

The configuration file format being used is just like the one specified. It is inside of the "res" folder and is named as SMCP.conf.

It requires a blank line to exist between the configuration of each IP:PORT combination.

```
<224.1.1.1:9876>
  <SID>FCTSpeak</SID>
  <SEA>DES</SEA>
  <SEAKS>64</SEAKS>
  <MODE>CBC</MODE>
  <PADDING>PKCS7Padding</PADDING>
  <INTHASH>SHA-256</INTHASH>
  <MAC>HMacSHA256</MAC>
  <MACKS>256</MACKS>
</224.1.1.1>

<224.1.1.2:1111>
  <SID>SRCS</SID>
  <SEA>AES</SEA>
  <SEAKS>128</SEAKS>
  <MODE>ECB</MODE>
  <PADDING>PKCS5Padding</PADDING>
  <INTHASH>SHA1</INTHASH>
  <MAC>HmacMD5</MAC>
  <MACKS>128</MACKS>
</224.1.1.2>
```

4.2 Keystores

For this assignment we are using two different keystores. Both use the same password: CSNS1920

The one inside the “res” folder, called SMCPKeystore.jecks , is the keystore in which we store all the keys used for the various cryptographic constructions, both for the symmetric encryption algorithm, using a IP:PORT identifier, and the for the MAC, using a IP:PORT:mac identifier.

The other keystore is on the root of the project called signKeystore. It is used to store the certificate to sign the generated Jar file.

4.3 Running the Client as a Standalone Application

In the SecureMChatClient-Application-Phase1 folder can be found a signed SecureMchatClient.jar file.

To run the program, go to the root of the project and run:

```
java -jar SecureMChatClient-Application-Phase1/SecureMchatClient.jar ReportUser 224.1.1.1 9876.
```
or
run the runJar.sh script found on the root of the folder.

The expected arguments are: username, multicastIP, port.

4.4 Tested Cryptographic Parameterizations

The tested cryptographic parameterization that worked were:

DES/CBC/PKCS7Padding
AES/ECB/PKCS5Padding
AES/CTR/NoPadding
Serpent/OFB/PKCS5Padding
Twofish/CFB/PKCS5Padding
Blowfish/CTR/PKCS7Padding
DESede/CBC/PKCS5Padding

A tested cryptographic parameterization tested that did not work:

AES/GCM/NoPadding

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

If you didn't address Phase 2, you dont need to write this section 5 and your section 5 will be the conclusion section. Otherwise you must start by describing the objectives and results achieved in addressing the Phase 2, according to your submitted work. After the initial summary please include the following sub-sections.

5.1 Generic specification of the SAAHP protocol

Try to present here an initial specification of your designed protocol, namely message requests and responses, according to the required rounds between a client and the SAAHP server (endpoint). A good idea would be the representation of the protocol in a sequence temporal diagram.

5.2 Formal specification of the SAAHP Protocol: SAAHP Message format

Present here a good formalization of the SAAHP message format specification, clarifying each part of the message format, the cryptographic contents and parameterizations, as well as, the security properties satisfied by the cryptographic constructions in your message format.

5.3 Detail of the protocol operation to support client JOINS and client LEAV

Present here if you have specific support to the JOIN and LEAVE operations from clients in chat sessions

5.4 Rekeying strategies

Discuss here if you considered a rekeying mechanism, in the support of JOINS and LEAVES from clients, in CHAT sessions.

5.5 Configuration files in the SAAHP Server

Present and exemplify the required setup and configuration files in the SAAHP server, namely for: user-authentcation, access-control, chat-session parameterizations, keystores and truststores.

5.6 Configuration files in the client side

Present and exemplify the required setup and configuration files required in the client side

5.7 Trust-Management Assumptions

Explain here how you addressed trust-management support, namely, what is required to setup initially, for the trust-ability-assumptions of clients and SAAHP server to manage public-keys and/or verification, assessment and validation conditions of public-keys and public-key certificates.

6 Highlights of the implementation

This optional section can be used if you want to highlight something you want to emphasize from your implementation, or if you want to present arguments for your design-options and implementation models and/or used techniques.

7 Conclusions and Final Remarks

Use this to summarize your main conclusions and to present your final remarks on the implementation of the TP#1.