Protection in Networked Systems – Trust, Resilience, and Privacy

Tasks for Privacy Block

TECHNISCHE UNIVERSITÄT DARMSTADT

Telecooperation Lab

Deadline: January 2nd 2018
Group submission: you can submit in **groups of up to 3**
Discuss your problems with us/me: Daubert, Jörg (daubert@tk.tu-darmstadt.de)
via Forum in Moodle

DC-nets

The goal of this task sheet is the implementation of a real DC-network. This implementation is structured in stages, where every archived stage is awarded with points (10 points in total).

- You can choose a common (!) programming language of your choice. A Python framework will be provided. NO Whitespace, Brainfuck, etc.
- Each individual stage should be conveniently executable. Examples: via parameter, separate program, function (to commented in or out in main function).
- For simplicity, the communication paradigm is a star-topology. That means, all cryptographers (C) communicate with a central point, the Single Point of Failure (SPoF). The SPoF listens on a TCP socket and forwards all messages (text, line-based) to all connected Cs, except for the sending one.
- Stick to "Clean Code" principles for documentation. Meaning, use good code structure and telling names instead of extensive documentation. Only document where documentation is really necessary for understanding the code.
- Upload your solution to Moodle as ZIP file.

Task-1: Basic DC-net

Implement a DC-net with the SPoF and 3 Cs. You can define pre-shared keys for the 3 Cs. One C should be able to read a message (you can fix the length > 1) from command line. The C then "encrypts" and sends the message. Upon reception of the message, the other 2 Cs send their (empty) message. After every C received the other two messages, every C "decrypts" and displays the message. (3 P)

Task-2 Timed DC-net

Rather than waiting for the first message, add the capability to every C to send a message every X seconds. The message should be either the last entered message or the empty message respectively. (1 P).

Task-3: Timed conflicting DC-net

Extend the time DC-net such that Cs send their name/ID + time/increment at random intervals (multiple of the X seconds interval). (1 P)

What issue can occur here with DC-nets? How can that issue be detected by the Cs? How would you resolve this issue (theoretically)? (1 P)

Task-4: Larger DC-net

Extend the basic DC-net such that number of Cs is a parameter, i.e., the number of Cs (>=3) can be defined upon start. Devise a method that automatically generates the pairwise keys. Avoid communication; use the potentially bad option of using the name/ID of the Cs for that purpose. (2 P)

Task-5: Benchmarking DC-nets

Extend the larger DC-net with a ring topology to replace the pairwise keys. Here, as a metaphor, all Cs are placed in a ring. Every C, therefore, just has two neighbors and consequently knows two keys.

Now, compare this ring DC-net with the larger DC-net. For that, select a sufficiently high number of Cs, and send one message after the start of the program. The program terminates after every C obtained this message. Measure the total time using a method of your choice, e.g., within your code or with the time command. (2 P).