

Security Technologies: Project Design

Adam Janovský, Marie William Gabriel Konan, Matěj Plch

1 Client side

Our goal is to modify open source program KeePassXC¹ to use Java Card. KeePassXC is a port of popular password manager KeePass². KeePass allows users to decrypt password database using password and/or key file. We will add a button to KeePassXC, which shows a prompt requesting PIN from the user. If the PIN is accepted by the card, then password for the database is loaded from the card and used by KeePassXC for unlocking the database.

2 Java Card side

Purpose of the card is to store user's password and provide it back only after inserting valid PIN. PIN has 4 digits and user has 3 attempts. After 3 unsuccessful attempts the card locks itself. The card will support APDUs for setting PIN and the password, for PIN attempt, and for requesting the password.

3 Secure channel

Let us note, that it is reasonable to stick with some standard solution instead of inventing the wheel. Next, let us discuss possible authentication options within the secure channel. We discuss the need of authentication and proceed with selection of the solution.

The authentication in secure channel should be mutual. Consider the case, when the smart card is somehow (physically or logically) replaced by the malicious card. Then this card would be able to receive user credentials (i.e. PIN) via established secure channel. Therefore, mutual authentication is needed. That implies usage of some pre-shared secret or public key cryptography. This case is somewhat complicated, therefore we further discuss both cases, i.e. mutual authentication and one-sided authentication (which make things much easier). Let us proceed with discussion on the pre-shared secret vs. public-key cryptography approach for secure channel.

The PC-Application binary is not secure place to store static keys. If we use asymmetric cryptography, again, we have to store the private key somewhere. Therefore, some tradeoff between security and usability is necessary. The corresponding (private/pre-shared) key could be stored on the server, on other javacard and so on, to increase the security.

¹<https://keepassxc.org/>

²<http://keepass.info/>

We have evaluated all these options as quite artificial, as they are either complicated or decrease usability a lot. There seems to be a reasonable path between - hide the hardcoded key carefully, though it of course can't be used in real life.

If we omit the model, when the card has to be authenticated, then asymmetric cryptography can be used nicely, based on TLS. In this case, we would go with some mixture of TLS and SCP'10. We would omit certification authorities for simplicity. Both keys would be stored on the card, the user would have certificate with himself. We still want to discuss the attacker's model with the teacher.

In case we will need mutual authentication and pre-shared symmetric static key, we can select reasonable standardized secure channel from GlobalPlatform. We will either use SCP'02 or SCP'03, as those have suitable API for java card and C library for our application. Both standards could be implemented according to the documentation available from GlobalPlatform³.

³<https://www.globalplatform.org/specificationscard.asp>