# PV204

# Phase 2 Report

#### UCO 456462 Jan Kubeša, UCO 502606 Amal Chukkinin, UCO 445508 Marek Hrašna

Requirements:

Card and user share value of authentication (PIN). PIN is set onto the card during installation via installation parameters and printed out to the user. When the user inserts the card to the reader, he needs to provide the PIN – it is used only to authenticate ECDH shared secret and is never transmitted to the card or back. Initial secret is then established via ECDH. Both the card and the user are authenticated before every session with challenge-response from PC and flow restrictions on card.

The established ECDH shared secret is used for derivation of the session keys.

Subsequent data exchanged between the card and the PC are then protected by the secure channel. The session can be finished by an explicit command from the PC or can be interrupted prematurely by sudden removal of the card from the reader. The data then gets erased, all temporary secrets are erased.

* After receiving the shared secret, the PC site calculates MD5 hash of the shared sec, sets it as an AES key and encrypts a random number, 16 byte long, generated using secure random number generator. The PC then sends the cipher to the card.
* The card decrypts the cipher using the MD5 hash key of the secret and gets the random number. Card reverses the obtained random number and encrypts it again with same key and sends back to the PC.
* PC decrypts the reversed random number, checks with the original one and if the match, both PC and Card are mutually authenticated.
* If the reverse of the random number do not match, we displays a ‘wrong pin’ message and ask for a next try. If the PIN tries number is exceeded, we block the card.
* Once a secure channel is established, pin guesses will not be detected and an error will be thrown out.

For our key agreement, we decided to use a SPAKE2 protocol. To implement it, we needed to use Diffie-Helman protocol for shared secret, AES as the product of a MD5 hash of the shared secret.

## Implementation and difficulties

Our first difficulty during implementation laid in choosing the protocol that would satisfy the security requirements. After a call we decided that SPAKE2 protocol would be our choice. The main idea is that we are basically doing ECDH, but protecting it by password and then unwrapping it back to ECDH on the other side to prevent man-in-the-middle attacks.

Our second challenge was to understand the maths and correctly implement it.

During the implementation we had to make other choices and research on what libraries to use. We used different libraries to implement the cryptography but finally we ended up using BouncyCastle on both sides, after difficulties with our previous libraries, possibly with the JCSim itself, which cost us many hours of debugging. We also used BigInteger library to ease our computations in the cryptographic operations.

At the end we used same libraries (BigInteger and BouncyCastle) on both the card and the PC side, so our implementation is quite uniform.

We decided that the way to handle PIN authentication was that the card will have it hardcoded and the PC will have it from the user, neither of them verify it on the same device.

### Attack Protection

-MITM - As mentioned before, we have prevented the man-in-the-middle-attack by password protecting the ECDH.

-Password guessing - We accounted for the password guessing by using a PIN counter.

-Replay attack - We are using counters on the individual messages.

-Integrity – The first 32 bytes of an encrypted message contain the MD5 hash of the rest of the message