Modern Secure Channel on Certified Smart Cards Using Ephemeral ECDH Keys

Emerald Password Manager for Smart Cards

Martin Vondracek, Elena Solodkova, Oldrich Tristan Florian

1. Introduction

The project aims to create a JavaCard applet and a PC application for secure channel communication. The user is provided with a smart card (Java Card) with pre-personalized 4-digit PIN and a paper with this PIN printed. If the user wants to use the card, they have to put it into the reader and type in the PIN.

Before any session, both card and a user (via implemented PC application) need to be mutually authenticated, and all subsequent data exchange between them needs to be protected by a secure channel. The PIN is never transmitted to the card or back. Therefore, they use it to establish an initial secret for the secure channel using a key exchange over an elliptic curve, which can then be used to derive session keys.

2. Password-Authenticated Key Exchange by Juggling

Password-Authenticated Key Exchange (PAKE) is a technique to establish secure communication between two parties solely based on their shared password [1, p. 2]. We wanted to use the elliptic curve cryptography, because since it saves time, cost, and space, its usage for a smart card seems the most appropriate [2, p. 12]. One of the PAKE techniques that is designed to work with elliptic curves is Password-Authenticated Key Exchange by Juggling (J-PAKE) [1, p. 2], which we decided to implement.

Generally, there are two variants of J-PAKE. The first is the two-round J-PAKE protocol, which is entirely symmetric [1, p. 8]. The second is a three-pass variant, in which the protocol is completed in three passes instead of two rounds without losing security [1, p. 8]. The three-pass variant is considered a more practical one [1, p. 8], we have decided to use it.

The three-pass J-PAKE over elliptic curve works as follows [1, p. 7–9]:

- Three passes
    1. Alice –> Bob
        - *G1 = G x [x1]*
            - *G* is a generator (a point on an elliptic curve)
            - *G x [x1]* is a multiplication of a point *G* with a scalar *x1*
        - *G2 = G x [x2]*
        - ZKPs for *x1* and *x2*
            - *x1* and *x2* are private keys chosen randomly from *[1, n – 1]*

2. Bob –> Alice
    - *G3 = G x [x3]*
    - *G4 = G x [x4]*
    - *B = (G1 + G2 + G3) x [x4*s]*
    - ZKPs for *x3*, *x4*, and *x4*s*
        - *s* is a secret value derived from PIN shared between Alice and Bob
3. Alice –> Bob
    - *A = (G1 + G3 + G4) x [x2 * s]*
    - ZKP for *x2*s*

- Session keys computation
    1. Alice computes *Ka = (B – (G4 x [x2 * s])) x [x2]*
    2. Bob computes *Kb = (A – (G2 x [x4 * s])) x [x4]*
    3. *Ka* equals to *Kb*, which equals to *G x [(x1 + x3) * (x2 * x4 * s)]*

3. Schnorr Non-Interactive Zero-Knowledge Proof

Schnorr zero-knowledge proof allows one to prove the knowledge of a discrete logarithm without revealing its value [3, p. 2]. The Schnorr zero-knowledge proof can be implemented over a finite field or an elliptic curve [3, p. 3], and since we are using J-PAKE over an elliptic curve, we have decided for the latter. Also, as to the interactions of the prover and the verifier, there are two variants of the Schnorr zero-knowledge proof. One is an interactive variant that requires the verifier to issue a challenge, while the other is a non-interactive variant that does not require any interaction with the verifier for getting the result [3, pp. 7–8]. We have used the second variant since it is also recommended in the J-PAKE RFC [1, p. 5].

The non-interactive variant of the protocol works in the following steps [3, pp. 7–8]:

- Alice (the prover)
    1. Alice computes her public key *A = G x [a]*
        - *a* is the private key chosen randomly from *[1, n – 1]*
    2. Alice chooses a number *v* randomly from [1, n – 1] and computes *V = G x [v]*.
    3. Alice issues the challenge by using a secure cryptographic hash function. The challenge *c* is defined as *c = H(G || V || A || UserID)*.
        - *UserID* is a unique identifier for the prover
        - We have used ALG_SHA_512 as the hash function
    4. Alice computes the result *r = v* – (*a * c*) *mod n*
        - *n* is the prime order of the generator
    5. Alice sends the *A*, *V,* and *r* to Bob.
- Bob (the verifier)
    1. Bob checks, if A is a valid point on the curve
    2. Bob checks, if A x [h] is not the point at infinity
        - *h* is the cofactor of the generated subgroup
    3. Bob checks, if V is equal to *G x [r] + A x [c]*

4. Secure Channel Protocol Design

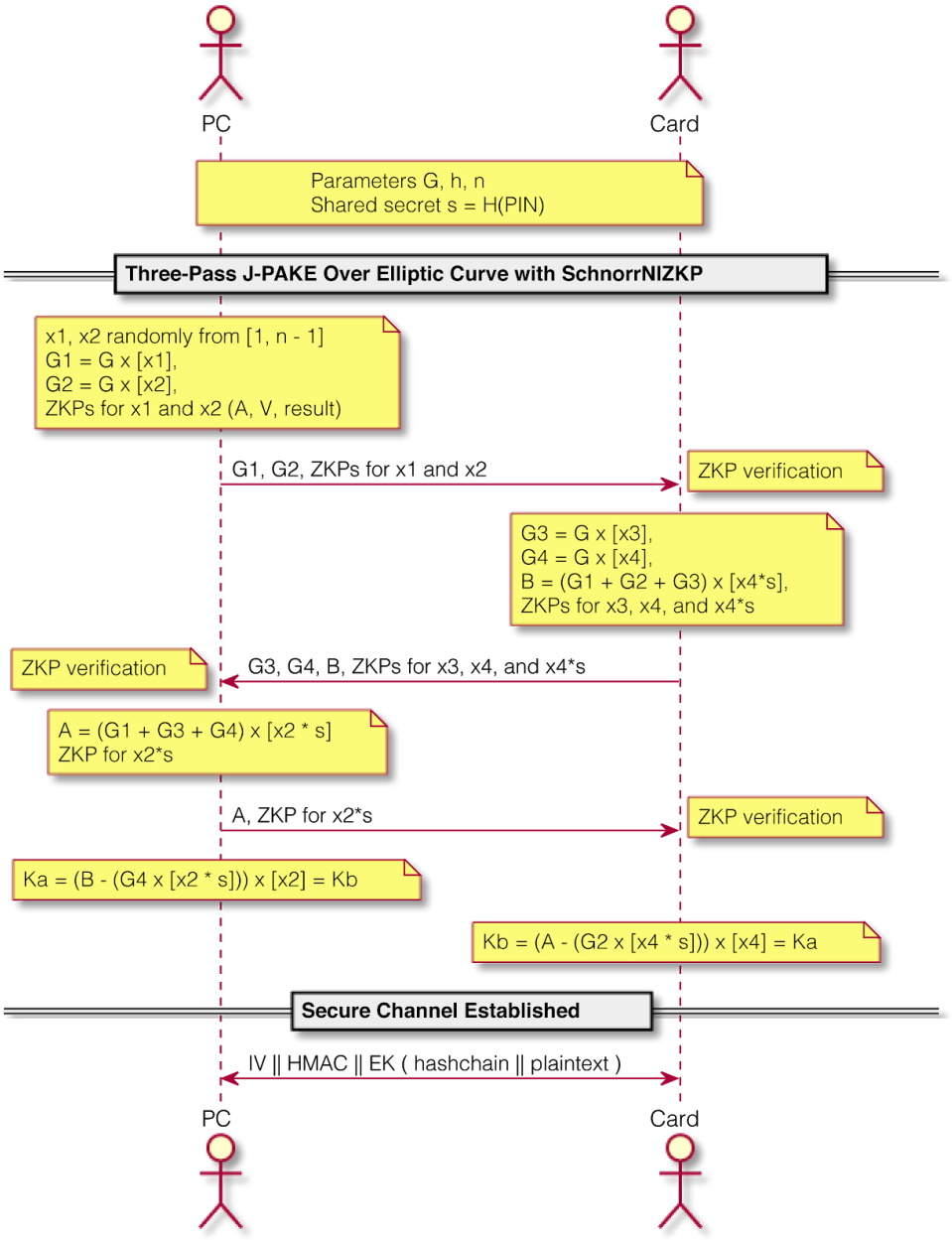


Figure 1 – Secure Channel Protocol Design

J-PAKE provides us **authenticity** and **key agreement**, while Schnorr Non-Interactive Zero-Knowledge Proof allows us authentication without revealing the shared secret. Together, they provide **protection against brute-force attacks**. Generally, a secure PAKE protocol should provide offline and online dictionary attack resistance, forward secrecy, and known-key security [1, p. 11]. J-PAKE satisfies all of these requirements [1, p. 12].

Once the session key is successfully derived as a result of J-PAKE, secure channel can be established. This channel uses following scheme, where || symbol denotes concatenation:

```
IV || HMAC || EK(hashchain || plaintext)
```

For the symmetric encryption (*EK*), AES-CBC with 256-bit key and 128-bit block size with padding according to ISO 9797 method 2 [4]. AES-CBC encryption is used with Initialization Vectors (*IVs*) which are generated with **cryptographically secure random number generator** on card. AES generally provides us **confidentiality**. **Integrity** of messages is achieved with HMAC based on SHA-256 [5]. This keyed-hash message authentication code ensures that value of HMAC was generated by a legitimate party with correct HMAC key. **Protection against replay attacks** is delivered by use of hash chains implemented with SHA-256.

Initially, we wanted to use AES-GCM, but it seems that jCardSim does not support it, although it claims support for Java Card Platform Classic Edition 3.0.5 [6].

As already mentioned, J-PAKE establishes shared session key (Ka = Kb). Our protocol then performs key derivation, implemented as SHA-256, in order to obtain **separate keys for encryption and HMAC**. Also, since we generate new keys in each session, it guarantees us **forward secrecy**. Therefore, we consider this protocol safe to use for a secure channel.

4.1. Protocol Design at APDU Level

Implemented applet supports communication via APDUs and Extended Length APDUs. The applet supports **plaintext messages**, **key agreement messages**, and **messages protected by our secure channel**. These three categories are identified by their APDU class byte values. As key agreement protocol requires sending two messages from PC to card, they are identified by value of APDU instruction byte.

All sensitive data exchanged between PC and card are protected by our secure channel. Furthermore, Messages of implemented *Emerald Password Manager for Smart Cards* are **protected including their metadata**.

5. Implementation

Our project is divided into three modules written in Java language as follows:

- `applet`: Java Card applet managing communication over our secure channel and providing messages to Sub-Applet *Emerald Password Manager for Smart Cards*. The top-level applet serves as a secure layver for generic Sub-Applets.
- `emApplication`: PC application for communication with *Emerald Password Manager for Smart Cards* on Java Card in smart card reader over our secure channel.
- `emCardTools`: Tools for communication with smart card reader used in by the PC application and during end-to-end testing of the applet. These tools are integrated from [7] which was published under MIT license.

We utilized Gradle system for build process, dependency management, testing, and also easy execution. Gradle configuration for Java Card project was based on [7] (published under MIT license), but was extended to better fit needs of our team. We have also found a bug in the

Gradle template and created a pull request to the upstream [8]. Also, we aimed at using JCMathLib for low-level EC operations [9], but we have encountered a bug and its integration was not successful. This bugs will be reported to upstream.

Development in our team was organized based on simplified *Git Flow* methodology. All code integrated to development (and later to master) branch had to go through pull requests with code review, code testing and Static Application Security Testing (SAST).

Source code of all implemented modules is available on project's repository [10].

5.1. Secure Channel Protocol

J-PAKE functionality is implemented inside `jpake` package. Since an asymmetric version of the protocol was chosen, the functionality is divided between two classes `jpakeActiveActor` (initiates the handshake) and `jpakePassiveActor` (for the follower). Classes `jpakeActiveFirstPayload`, `jpakeActiveSecondPayload`, and `jpakePassivePayload` provide methods for constructing payloads sent by each side and supporting functionality like getters and serialization/deserialization methods. For elliptic curve we choose **Curve25519**.

As for the Schnorr non-interactive zero-knowledge proof, the prover's part is done in the constructor of the `SchnorrZKP` class that has getters for *A*, *V*, and *r*. To allow the verifier to verify the result of the ZKP, he uses a static `ZKPUtil` class. This class has methods for the computations necessary to be done both by PC and card, and a verification method for the verifier.

5.2. Smart Card Applet

The top-level applet is implemented as an extension of standard Java Card applet class. During installation, it requires PIN as installation parameter. The applet stores important data in persistent memory and properly clears session data during `select` and `deselect` events. This applet also allows basic plaintext communication. Key agreement and secure channel management is delegated to our implemented Secure Channel Manager. Decrypted messages are forwarded to Sub-Applet and responses are encrypted and sent as APDUs to PC.

In case of any violations of our secure channel, the applet signals security alert. After three security alerts, the applet blocks itself and permanently refuses any communication. This is a **protection against PIN brute-force attack**.

5.3. Password Manager Sub-Applet

In order to demonstrate feasibility of our secure channel, we have implemented *Emerald Password Manager for Smart Cards*. This applet is able to process messages similar to APDUs. We have designed following protocol:

```
MESSAGE_TYPE || PASSWORD_SLOT_ID || PASSWORD_LENGTH || PASSWORD_VALUE
```

Where MESSAGE_TYPE is one of following:

- `MESSAGE_SET_PASSWORD`
- `MESSAGE_OK_SET`
- `MESSAGE_GET_PASSWORD`
- `MESSAGE_OK_GET`

This applet allows a user to securely store passwords in persistent memory of Smart Card and later retrieve them.

5.4. Computer Application

The computer application offers a simple Command Line Interface (CLI) for access to the *Password Manager for Smart Cards* applet on smart card. It can also connect to smart card in simulator for demonstration purposes. The application asks user for PIN. PIN is used only for subsequent key agreement. When session key is successfully created, entered PIN is discarded. This offers protection **against memory dump attack** after the PIN is discarded.

6. Testing

Our solution is tested with **unit tests** and **end-to-end tests** with APDUs. We have utilized Continuous Integration (**Continuous Testing**) via TravisCI. Code was also continuously checked with **SAST** tools from Code Climate. Final codebase has **92 % test coverage**.

7. Conclusion

We have created a Java Card applet and a PC application for secure channel communication based on a shared 4-digit PIN. Furthermore, we have created a password manager as the basic functionality of the card. For authenticity, protection against brute-force attacks, offline and online dictionary attack resistance, forward secrecy, known-key security, and authentication without revealing the shared secret, we used J-PAKE with Schnorr NIZKP over elliptic curves. Confidentiality is provided by AES-CBC with IVs generated with cryptographically secure random number generator, integrity of messages is achieved with HMAC based on SHA-256, and protection against replay attacks is delivered by use of hash chains implemented with SHA-256. Protocol performs a key derivation, implemented as SHA-256, to obtain separate keys for encryption and HMAC. Forward secrecy is guaranteed by generating new keys in each session.

8. Bibliography

[1] F. Hao, "J-PAKE: Password-Authenticated Key Exchange by Juggling," *IETF Tools*, Sep. 2017. https://tools.ietf.org/html/rfc8236 (accessed Apr. 20, 2020).

[2] A. Kayali, "Elliptic Curve Cryptography and Smart Cards," SANS Institute, Feb. 2004. Available: https://www.sans.org/reading-room/whitepapers/vpns/elliptic-curve-cryptography-sm

art-cards-1378.
[3] F. Hao, “Schnorr Non-interactive Zero-Knowledge Proof,” *IETF Tools*, Sep. 2017. https://tools.ietf.org/html/rfc8235 (accessed Apr. 20, 2020).
[4] “Cipher (Java Card API, Classic Edition),” *Oracle Help Center*, 2015. https://docs.oracle.com/javacard/3.0.5/api/javacardx/crypto/Cipher.html#ALG_AES_CBC_ISO9797_M2 (accessed Apr. 23, 2020).
[5] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-Hashing for Message Authentication,” *IETF Tools*, Feb. 1997. https://tools.ietf.org/html/rfc2104 (accessed Apr. 20, 2020).
[6] “AES-GCM (ALG_AES_GCM) is not supported. · Issue #153 · licel/jcardsim,” *GitHub*. https://github.com/licel/jcardsim/issues/153 (accessed Apr. 23, 2020).
[7] CRoCS, “JavaCard Template project with Gradle,” *GitHub*, Apr. 19, 2020. https://github.com/crocs-muni/javacard-gradle-template-edu (accessed Apr. 23, 2020).
[8] “fix: JUnit test annotations for running tests with current gradle configuration by mvondracek · Pull Request #4 · crocs-muni/javacard-gradle-template-edu,” *GitHub*. https://github.com/crocs-muni/javacard-gradle-template-edu/pull/4 (accessed Apr. 23, 2020).
[9] OpenCrypto Project, “JCMathLib,” *GitHub*, Aug. 03, 2019. https://github.com/OpenCryptoProject/JCMathLib (accessed Apr. 23, 2020).
[10] Team Emerald, “Modern Secure Channel on Certified Smartcards Using Ephemeral ECDH Keys,” *GitHub*, Apr. 23, 2020. https://github.com/mvondracek/PV204_smartcards_Emerald (accessed Apr. 23, 2020).

Appendix A: Application Output with APDU Traces

```
     Emerald Password Manager for Smartcards

  __
 (`/\
 `=\/\ __...--~~~~~~-._   _.-~~~~~~--...__
  `=\/\               \ /               \\
   `=\/                V                 \\
   //_\___--~~~~~~-._  |  _.-~~~~~~--...__\\
  //  ) (..----~~~~._\ | /_.~~~~~----.....__\\
 ===(     )==========\\|//====================
     \___/            `---`

PC    : Selected communication with EmeraldApplet in a simulator.
PC    : Simulated applet was installed with PIN `1 2 3 4`.
PC ... SC: Established connection to smartcard.
PC    : Enter 4 digit PIN.
1234
PC    : Using PIN `[1, 2, 3, 4]`
##################################################
Begin: demo plaintext
PC --> SC: Sending example plaintext message to card.
--> 0100000000
--> [0100000000] 5
<-- 00010203040506070809 9000 (10)
<-- 00010203040506070809 9000 (10) [13 ms]
PC <== SC: Received response:[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
End: demo plaintext
################################################
PC <-> SC: Key agreement started.
-->
AA0100000001D708434F4D505554455241040EC2251ECF0DE5AF74972D5F5C4C0A
4DC547D1E679A15C71FFB90B4E1454EDE633CC39DBC2CB55BAD614E503AA125BB5
E6AE993A5F5E5CFC7821C2B66F81EFA241047D5A804E1D1F555DA8FF4ED8C52DF9
48F397DA15A6B929AF0BBDC19B97DE7302208B21A4A0003D31C190CC6C52737406
C5A99CE21B07BA40C67AF901EBCE26BC41040EC2251ECF0DE5AF74972D5F5C4C0A
4DC547D1E679A15C71FFB90B4E1454EDE633CC39DBC2CB55BAD614E503AA125BB5
E6AE993A5F5E5CFC7821C2B66F81EFA241045B6FC83ECC7F4D05ED23415FD66131
BF176BDC4535B9E33D9729F20D7BCE22252B24CD053348CBD23E3E05ECCDC18657
779DB553AB61A4EFD25B5528EC3AAB392000E32A7C4B25FC9D7D2AF263D044BCB8
5631C1F4DB6F8700FA80016AC1DE0C0341047D5A804E1D1F555DA8FF4ED8C52DF9
48F397DA15A6B929AF0BBDC19B97DE7302208B21A4A0003D31C190CC6C52737406
C5A99CE21B07BA40C67AF901EBCE26BC41041CD05F0CC75961C5A4BF39A1ECA1E5
836A4FEA1E485F04CAD9FE76027B734D947D4E376DA7B458E080BAABA0518D44E1
A33D633A03B369C7DE73B3B88BA8213C20FEF1BEF71FD0189A08427177CA6935E2
355E41EE8BC301E0C75368BDB798203E
-->
[AA0100000001D708434F4D505554455241040EC2251ECF0DE5AF74972D5F5C4C0
A4DC547D1E679A15C71FFB90B4E1454EDE633CC39DBC2CB55BAD614E503AA125BB
5E6AE993A5F5E5CFC7821C2B66F81EFA241047D5A804E1D1F555DA8FF4ED8C52DF
948F397DA15A6B929AF0BBDC19B97DE7302208B21A4A0003D31C190CC6C5273740
6C5A99CE21B07BA40C67AF901EBCE26BC41040EC2251ECF0DE5AF74972D5F5C4C0
A4DC547D1E679A15C71FFB90B4E1454EDE633CC39DBC2CB55BAD614E503AA125BB
5E6AE993A5F5E5CFC7821C2B66F81EFA241045B6FC83ECC7F4D05ED23415FD6613
1BF176BDC4535B9E33D9729F20D7BCE22252B24CD053348CBD23E3E05ECCDC1865
7779DB553AB61A4EFD25B5528EC3AAB392000E32A7C4B25FC9D7D2AF263D044BCB
85631C1F4DB6F8700FA80016AC1DE0C0341047D5A804E1D1F555DA8FF4ED8C52DF
948F397DA15A6B929AF0BBDC19B97DE7302208B21A4A0003D31C190CC6C5273740
6C5A99CE21B07BA40C67AF901EBCE26BC41041CD05F0CC75961C5A4BF39A1ECA1E
5836A4FEA1E485F04CAD9FE76027B734D947D4E376DA7B458E080BAABA0518D44E
1A33D633A03B369C7DE73B3B88BA8213C20FEF1BEF71FD0189A08427177CA6935E
2355E41EE8BC301E0C75368BDB798203E] 478
<--
04434152444104 28D02BD086FBB7C4BCA64D5EF179CC5BC3716956B3B0673548B6
080CB5779C5F210077A52DEC726876F3AD052A698A581802B8A1BD7C6183F6AD69
52AF419DD041040E14EDFE34D75E7587AD35320074BF3C36B3183490F876194A30
EFB5196AE7482F2402FEDDEC88EAD67B1D5338B056DA1E86FFED46E9FAF62C5FB1
594200F1F0410477ACE17A362AA6256AA4E2A6FF97A7CAC67C7014A4371E3C3053
1864BC86DD9716D79380053E8D7FAD101AEA3DD202B76F89299BFDF6BB3BB8B3E1
D090A63433410428D02BD086FBB7C4BCA64D5EF179CC5BC3716956B3B0673548B6
080CB5779C5F210077A52DEC726876F3AD052A698A581802B8A1BD7C6183F6AD69
52AF419DD041046E8EF19C89BB491D3681D28F754112A8259FDDD67C435C72E49A
6B06AF231E5C01C79C403DCB9D8AC332B53552BA80920DBC2B2F2F1D2D09A0787D
5FE9E0DB222001F9705AD86ADA084C573B7389FBEAEFE8BAA3F4057123088CCDC0
4503C98E7241040E14EDFE34D75E7587AD35320074BF3C36B3183490F876194A30
EFB5196AE7482F2402FEDDEC88EAD67B1D5338B056DA1E86FFED46E9FAF62C5FB1
594200F1F0410431DD7485937CB4078C834EB0E5A16DE55E7ACDD8A925538C5042
5B70A52EE356264AD7303F8C22D4EF4ADC770D3E1906A2312AC818E7DB5053732F
```

```
BFC1C4A5B22001014454DCD672E0B329148EB8EB4A59660E78FFC5119483D03FE5
063C1D6E2541041C3D47BD209B9CC1AAA4B3D49D0C766290751D5C06755407115E
09BF75CDDAC561CE12F206711531D4B7C85DAE9D47430C635757321F2B74DC5A5B
E406EDB8B641047706187EC68A2445592E0DAF73F4B26F973709715C2CAF11D1FD
4380B19F4615156EC8AE05CF413E2813194314265DC271AF319D57672011BE0621
6063ACC35C2001AC93F627606A83823D61EC412204BAC19455E04133884A6324C9
9EBE9DF294 9000 (698)
<--
0443415244410428D02BD086FBB7C4BCA64D5EF179CC5BC3716956B3B0673548B6
080CB5779C5F210077A52DEC726876F3AD052A698A581802B8A1BD7C6183F6AD69
52AF419DD041040E14EDFE34D75E7587AD35320074BF3C36B3183490F876194A30
EFB5196AE7482F2402FEDDEC88EAD67B1D5338B056DA1E86FFED46E9FAF62C5FB1
594200F1F0410477ACE17A362AA6256AA4E2A6FF97A7CAC67C7014A4371E3C3053
1864BC86DD9716D79380053E8D7FAD101AEA3DD202B76F89299BFDF6BB3BB8B3E1
D090A63433410428D02BD086FBB7C4BCA64D5EF179CC5BC3716956B3B0673548B6
080CB5779C5F210077A52DEC726876F3AD052A698A581802B8A1BD7C6183F6AD69
52AF419DD041046E8EF19C89BB491D3681D28F754112A8259FDDD67C435C72E49A
6B06AF231E5C01C79C403DCB9D8AC332B53552BA80920DBC2B2F2F1D2D09A0787D
5FE9E0DB222001F9705AD86ADA084C573B7389FBEAEFE8BAA3F4057123088CCDC0
4503C98E7241040E14EDFE34D75E7587AD35320074BF3C36B3183490F876194A30
EFB5196AE7482F2402FEDDEC88EAD67B1D5338B056DA1E86FFED46E9FAF62C5FB1
594200F1F0410431DD7485937CB4078C834EB0E5A16DE55E7ACDD8A925538C5042
5B70A52EE356264AD7303F8C22D4EF4ADC770D3E1906A2312AC818E7DB5053732F
BFC1C4A5B22001014454DCD672E0B329148EB8EB4A59660E78FFC5119483D03FE5
063C1D6E2541041C3D47BD209B9CC1AAA4B3D49D0C766290751D5C06755407115E
09BF75CDDAC561CE12F206711531D4B7C85DAE9D47430C635757321F2B74DC5A5B
E406EDB8B641047706187EC68A2445592E0DAF73F4B26F973709715C2CAF11D1FD
4380B19F4615156EC8AE05CF413E2813194314265DC271AF319D57672011BE0621
6063ACC35C2001AC93F627606A83823D61EC412204BAC19455E04133884A6324C9
9EBE9DF294 9000 (698) [156 ms]
-->
AAEE0000F008434F4D5055544552410445FD031F7C00927439729B903BC697E04E
CD33BF7D92EF6F5CCD63E681F5824F16E60889ABA1DD23B6E5AB861C03E7CD27EB
FA2B28BA3EB4B631FE27C8FA30D241040FB3003EAD31FEC0948E18EEA46ABC5B75
93BBC11F0CEFF210569B8E3845618E1207E186CBF81C22E8670ED0FDDAE5ACDF1B
28242FE2FB259C02AA6286A74A2A410434CB36E30B96EE6AEFAFEA1F9ABAC03E5A
E9087D68F824161FC809A29A2254D025457AF8966D20ADBCCB8441D167742FC4C3
2676795A0558FB3485CEE830CEED20F5B30B06174A3FF2990FDAD9083AF899AEB4
99E6212AB6ED62B074438D7DB6CA
-->
[AAEE0000F008434F4D5055544552410445FD031F7C00927439729B903BC697E04
ECD33BF7D92EF6F5CCD63E681F5824F16E60889ABA1DD23B6E5AB861C03E7CD27E
BFA2B28BA3EB4B631FE27C8FA30D241040FB3003EAD31FEC0948E18EEA46ABC5B7
593BBC11F0CEFF210569B8E3845618E1207E186CBF81C22E8670ED0FDDAE5ACDF1
B28242FE2FB259C02AA6286A74A2A410434CB36E30B96EE6AEFAFEA1F9ABAC03E5
AE9087D68F824161FC809A29A2254D025457AF8966D20ADBCCB8441D167742FC4C
32676795A0558FB3485CEE830CEED20F5B30B06174A3FF2990FDAD9083AF899AEB
499E6212AB6ED62B074438D7DB6CA] 245
<-- 9000
<-- 9000 [58 ms]
PC   : New shared session key established.
```

```
PC <-> SC: Key agreement finished successfully.
################################################
Begin: demo password storage

PC --> SC: Set password `All your` to slot 0.
-->
EE000000806C5544797A91115DC3330EBD003851D2B5F4623E96E4D8374BE489F2
BBE6D432CB9960A85C1860E9D73AA57C61FE1F5EFDDC82F16A10530663614CEA2E
07E92CB84D240B2BE39175AA17A18771D4B925E6D6C86B6C9D329EA87A63DCAF13
B8027D31192AB1913C501686EE477CB8718CA355FAE2CC70625EFAB2637F4C632A
85
-->
[EE000000806C5544797A91115DC3330EBD003851D2B5F4623E96E4D8374BE489F
2BBE6D432CB9960A85C1860E9D73AA57C61FE1F5EFDDC82F16A10530663614CEA2
E07E92CB84D240B2BE39175AA17A18771D4B925E6D6C86B6C9D329EA87A63DCAF1
3B8027D31192AB1913C501686EE477CB8718CA355FAE2CC70625EFAB2637F4C632
A85] 133
<--
6C5544797A91115DC3330EBD003851D277EC4E1E4984E6E48625A29F743D04AA7D
AACEF16BC3FB49113C6EFBAF6F24E1FDDC82F16A10530663614CEA2E07E92CB84D
240B2BE39175AA17A18771D4B9258F9E9F0700A00EFD8B2954D607FF67474D3B2F
64B9DFE054798F20FDA09F79625A47EB3C3B73CACE279377B17AEAB147    9000
(128)
<--
6C5544797A91115DC3330EBD003851D277EC4E1E4984E6E48625A29F743D04AA7D
AACEF16BC3FB49113C6EFBAF6F24E1FDDC82F16A10530663614CEA2E07E92CB84D
240B2BE39175AA17A18771D4B9258F9E9F0700A00EFD8B2954D607FF67474D3B2F
64B9DFE054798F20FDA09F79625A47EB3C3B73CACE279377B17AEAB147    9000
(128) [9 ms]
PC <== SC: Password successfully set to slot 0.

PC --> SC: Set password `base are` to slot 1.
-->
EE000000802AA2AB70039C5510DDF06420ECEB88922B52E5279A555F80CB88C490
70560ADCFAE45A6F4BF00BB67CB1CAE5861E84DA8AB14DDA0173CB103AA8192ECE
6BA16CE2D93F9B455C1C664BB14ED2E9A9B45A4090621A7A8FC2A5C04A328A27FE
CAC2F5E94BA70CB2866E2639E388E12A4B76609B08FF67A998C8F1B62F9E61A8BE
7A
-->
[EE000000802AA2AB70039C5510DDF06420ECEB88922B52E5279A555F80CB88C49
070560ADCFAE45A6F4BF00BB67CB1CAE5861E84DA8AB14DDA0173CB103AA8192EC
E6BA16CE2D93F9B455C1C664BB14ED2E9A9B45A4090621A7A8FC2A5C04A328A27F
ECAC2F5E94BA70CB2866E2639E388E12A4B76609B08FF67A998C8F1B62F9E61A8B
E7A] 133
<--
2AA2AB70039C5510DDF06420ECEB889288BF305A6D2258F98F00D6B3521BE9BE10
F607E9242906C4252B3A9A48BE281A8AB14DDA0173CB103AA8192ECE6BA16CE2D9
3F9B455C1C664BB14ED2E9A9B45AFAE38E14A1D5714AD54E68F1DF27C095696754
D5FCB1BFE52B2B03E352D3994F47FC75832CD85FDE3D473609A35189BA    9000
(128)
```

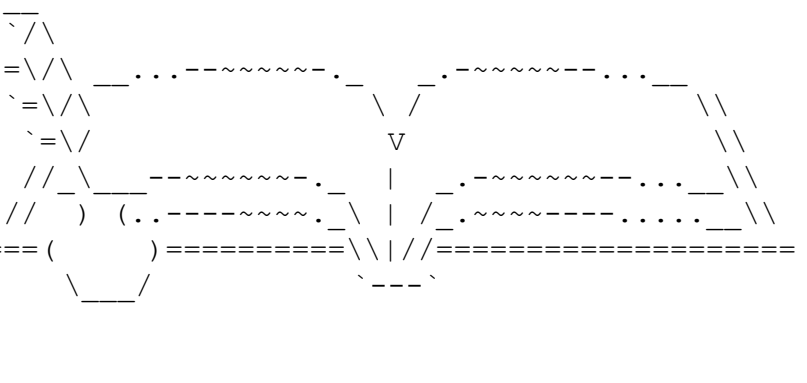
```
<--
2AA2AB70039C5510DDF06420ECEB889288BF305A6D2258F98F00D6B3521BE9BE10
F607E9242906C4252B3A9A48BE281A8AB14DDA0173CB103AA8192ECE6BA16CE2D9
3F9B455C1C664BB14ED2E9A9B45AFAE38E14A1D5714AD54E68F1DF27C095696754
D5FCB1BFE52B2B03E352D3994F47FC75832CD85FDE3D473609A35189BA     9000
(128) [16 ms]
PC <== SC: Password successfully set to slot 1.

PC --> SC: Set password `belong to us` to slot 2.
-->
EE0000008051373E8B6FDEC284DB569204CA13D2CADE730CAF8CDB8C74BDCCC764
5A490C4064160499C9E582F7A731EC05993E43CEC58B75D5E0881EA907F5B0F300
743647A7C670F9179CF458DE2CB868ED2F1E32C0CC4D053CD4EE826D294B190E2B
B8D3E6AE868957CD5EE446557C28A4821590F4DDA1C1255A5E50813DB1EBDF65D0
93
-->
[EE0000008051373E8B6FDEC284DB569204CA13D2CADE730CAF8CDB8C74BDCCC76
45A490C4064160499C9E582F7A731EC05993E43CEC58B75D5E0881EA907F5B0F30
0743647A7C670F9179CF458DE2CB868ED2F1E32C0CC4D053CD4EE826D294B190E2
BB8D3E6AE868957CD5EE446557C28A4821590F4DDA1C1255A5E50813DB1EBDF65D
093] 133
<--
51373E8B6FDEC284DB569204CA13D2CAD489256E41C77BA3A2B38AC15407406C4D
4CAE18B5E550213F476E947ABC5A3BC58B75D5E0881EA907F5B0F300743647A7C6
70F9179CF458DE2CB868ED2F1E323B6D6A9958C6893FA9ACB99AA701A3A7DAB828
81701228BD81646A0A8150EACD14D8BFDAC4090621E5AA769B159A6974     9000
(128)
<--
51373E8B6FDEC284DB569204CA13D2CAD489256E41C77BA3A2B38AC15407406C4D
4CAE18B5E550213F476E947ABC5A3BC58B75D5E0881EA907F5B0F300743647A7C6
70F9179CF458DE2CB868ED2F1E323B6D6A9958C6893FA9ACB99AA701A3A7DAB828
81701228BD81646A0A8150EACD14D8BFDAC4090621E5AA769B159A6974     9000
(128) [32 ms]
PC <== SC: Password successfully set to slot 2.

PC --> SC: Get password from slot 2.
-->
EE000000805DCAB02A0E3D50461E73F1BB7A9549A3499CA2A76951EB500F19537F
FE8038DFAEDCC6D59390A5E4798FD34D56EACB38F08E733F9E2FF6B3516A8605F0
9D7188F4ADD16AC4E89F00A6B5DE6014757AA4000B9497503B3F80553E2C76F9FD
8ECC06A078785318CBDE0A3656662171DB9E85E79565131E300B47CCCC674F773B
63
-->
[EE000000805DCAB02A0E3D50461E73F1BB7A9549A3499CA2A76951EB500F19537
FFE8038DFAEDCC6D59390A5E4798FD34D56EACB38F08E733F9E2FF6B3516A8605F
09D7188F4ADD16AC4E89F00A6B5DE6014757AA4000B9497503B3F80553E2C76F9F
D8ECC06A078785318CBDE0A3656662171DB9E85E79565131E300B47CCCC674F773
B63] 133
<--
5DCAB02A0E3D50461E73F1BB7A9549A360B68484253509DD2331ECF30915D925A8
BF106A4A7BACE834163F4A3BB0125AF08E733F9E2FF6B3516A8605F09D7188F4AD
```

```
D16AC4E89F00A6B5DE6014757AA4419B93208E18598015C81823E6A7F137939B77
5475C1AFC6EAA68A686320321F1EA65DCE306A4EAB76E7C2D690FFABBD    9000
(128)
<--
5DCAB02A0E3D50461E73F1BB7A9549A360B68484253509DD2331ECF30915D925A8
BF106A4A7BACE834163F4A3BB0125AF08E733F9E2FF6B3516A8605F09D7188F4AD
D16AC4E89F00A6B5DE6014757AA4419B93208E18598015C81823E6A7F137939B77
5475C1AFC6EAA68A686320321F1EA65DCE306A4EAB76E7C2D690FFABBD    9000
(128) [25 ms]
PC <== SC: Retrieved password `belong to us` from slot 2.

PC --> SC: Get password from slot 1.
-->
EE00000080F9C06213585289654996F0C40467B9A6D7F54954EFD75ED4AC1E443A
82253541460D0048FB1AC5FC5538A7C4E57EC1737B4917D289818FD9C5FD423D8A
804D948037E034C671779CDED673BE311B632A04B2CD21E1E41C5AE6DE388E7CF1
E2256EFFE7E477151716CBB8DFEBF79DDB1108BD87E47C34C2C150BF33CA138134
A5
-->
[EE00000080F9C06213585289654996F0C40467B9A6D7F54954EFD75ED4AC1E443
A82253541460D0048FB1AC5FC5538A7C4E57EC1737B4917D289818FD9C5FD423D8
A804D948037E034C671779CDED673BE311B632A04B2CD21E1E41C5AE6DE388E7CF
1E2256EFFE7E477151716CBB8DFEBF79DDB1108BD87E47C34C2C150BF33CA13813
4A5] 133
<--
F9C06213585289654996F0C40467B9A65AE28846AE75892A269FCADC08A19DDA86
238F00008E5E5396DD7BC446CCEA667B4917D289818FD9C5FD423D8A804D948037
E034C671779CDED673BE311B632A930775DE0549B1FB99E6971DC8703313FC294F
198B21613E5715CF03E1EA452E403469B8C881ED96FDCE47913BAD675A    9000
(128)
<--
F9C06213585289654996F0C40467B9A65AE28846AE75892A269FCADC08A19DDA86
238F00008E5E5396DD7BC446CCEA667B4917D289818FD9C5FD423D8A804D948037
E034C671779CDED673BE311B632A930775DE0549B1FB99E6971DC8703313FC294F
198B21613E5715CF03E1EA452E403469B8C881ED96FDCE47913BAD675A    9000
(128) [35 ms]
PC <== SC: Retrieved password `base are` from slot 1.

PC --> SC: Get password from slot 0.
-->
EE000000805B16A08C7A0C5F1570F966EF5FB5D40003AAF1A85495CA3900930685
9D882809FE4D26D6278C44215A0970F711F4937DC2F9466AEFE12B1AF0ED8ED518
52210F8CA585C5E3DB7ABC42E240AF87B5DE6DE1EEA0F7896C1C14B750A694891F
F1BAE861753672B15D514466F8E720D4B653FF4A237E164E6E8083438F387F83E7
22
-->
[EE000000805B16A08C7A0C5F1570F966EF5FB5D40003AAF1A85495CA390093068
59D882809FE4D26D6278C44215A0970F711F4937DC2F9466AEFE12B1AF0ED8ED51
852210F8CA585C5E3DB7ABC42E240AF87B5DE6DE1EEA0F7896C1C14B750A694891
FF1BAE861753672B15D514466F8E720D4B653FF4A237E164E6E8083438F387F83E
722] 133
```

```
<--
5B16A08C7A0C5F1570F966EF5FB5D400589EC4CBCA3CE72B83264DCA765397557B
74D7FA600EC7904C814138EB658187C2F9466AEFE12B1AF0ED8ED51852210F8CA5
85C5E3DB7ABC42E240AF87B5DE6DF98B807B7B2238E4AE81607ED0F55F38C27F0F
651C9C663D3CB0570542073FC80F351D5965D805E69D8306A50C620144     9000
(128)
<--
5B16A08C7A0C5F1570F966EF5FB5D400589EC4CBCA3CE72B83264DCA765397557B
74D7FA600EC7904C814138EB658187C2F9466AEFE12B1AF0ED8ED51852210F8CA5
85C5E3DB7ABC42E240AF87B5DE6DF98B807B7B2238E4AE81607ED0F55F38C27F0F
651C9C663D3CB0570542073FC80F351D5965D805E69D8306A50C620144     9000
(128) [55 ms]
PC <== SC: Retrieved password `All your` from slot 0.

End: demo password storage
##################################################
Done.
exit
```

Appendix B: Application Output on Incorrect PIN

Emerald Password Manager for Smartcards



```
PC    : Selected communication with EmeraldApplet in a simulator.
PC    : Simulated applet was installed with PIN `1 2 3 4`.
PC ... SC: Established connection to smartcard.
PC    : Enter 4 digit PIN.
9876
PC    : Using PIN `[9, 8, 7, 6]`
##################################################
Begin: demo plaintext
PC --> SC: Sending example plaintext message to card.
--> 0100000000
--> [0100000000] 5
<-- 00010203040506070809 9000 (10)
<-- 00010203040506070809 9000 (10) [1 ms]
PC <== SC: Received response:[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
End: demo plaintext
```

```
################################################
PC <-> SC: Key agreement started.
-->
AA0100000001D708434F4D50555445524104623 5F24389C2A9F852FC9836F6ECC2
9BECB107B6EB5CEAB3E898E512C38B7C5E6823E61F7E77B344D2F0677C5A7ADAB3
963661C749F566175A69C527E6077DF841040B0BA6617BACA9E89708336F0A44E8
D7DD99FE4B998C443D1D8D493F5175AAE30AA1D0809E34B9E275B33C229DCE015E
6C53E0B075020E5C0C03CD105B988CF441046235F24389C2A9F852FC9836F6ECC2
9BECB107B6EB5CEAB3E898E512C38B7C5E6823E61F7E77B344D2F0677C5A7ADAB3
963661C749F566175A69C527E6077DF841040F15AC9506595C466B47C479643897
7C42C6A8663FC01572E72063E04BA592FB23EDEFB47101312A5BA2031F70D4FDFA
6C4E02074E370E930D95825F10238DB720F8D57A7421FE95BC354CDE4FA2E6B7B3
14EDEBFCDF312ACFE4E1D4A4D815638D41040B0BA6617BACA9E89708336F0A44E8
D7DD99FE4B998C443D1D8D493F5175AAE30AA1D0809E34B9E275B33C229DCE015E
6C53E0B075020E5C0C03CD105B988CF441046E86DDC60F1C65BC789C93A519EE5B
C541C1F29DA1E24DB88BBFBCB466DC3836082251688BD02906FDBD4285245D487D
00924E914858F2A401DE66218DBC144D20043C07E90A2A628B7CE2C2A7ED561EFD
FD7A2E0A70D247DBB3D0FA685D10F050
-->
[AA0100000001D708434F4D50555445524104623 5F24389C2A9F852FC9836F6ECC
29BECB107B6EB5CEAB3E898E512C38B7C5E6823E61F7E77B344D2F0677C5A7ADAB
3963661C749F566175A69C527E6077DF841040B0BA6617BACA9E89708336F0A44E
8D7DD99FE4B998C443D1D8D493F5175AAE30AA1D0809E34B9E275B33C229DCE015
E6C53E0B075020E5C0C03CD105B988CF441046235F24389C2A9F852FC9836F6ECC
29BECB107B6EB5CEAB3E898E512C38B7C5E6823E61F7E77B344D2F0677C5A7ADAB
3963661C749F566175A69C527E6077DF841040F15AC9506595C466B47C47964389
77C42C6A8663FC01572E72063E04BA592FB23EDEFB47101312A5BA2031F70D4FDF
A6C4E02074E370E930D95825F10238DB720F8D57A7421FE95BC354CDE4FA2E6B7B
314EDEBFCDF312ACFE4E1D4A4D815638D41040B0BA6617BACA9E89708336F0A44E
8D7DD99FE4B998C443D1D8D493F5175AAE30AA1D0809E34B9E275B33C229DCE015
E6C53E0B075020E5C0C03CD105B988CF441046E86DDC60F1C65BC789C93A519EE5
BC541C1F29DA1E24DB88BBFBCB466DC3836082251688BD02906FDBD4285245D487
D00924E914858F2A401DE66218DBC144D20043C07E90A2A628B7CE2C2A7ED561EF
DFD7A2E0A70D247DBB3D0FA685D10F050] 478
<--
044341524441041 20FD2D28BDB700F6A05492F1D68C2DE05AAC601A30FE6255F27
F7FD132D0DA24E537D75DA25CE1D9E041B3AB190D191011F1BA933B43AC8468615
A534D871C741040F632A00F9E4953CAA01957EF33828670F6AEC23A99629254B97
7AB2C10499557EA1480724340525C46E1BC808686DD541B4533A17DE6EC485ED17
CFF38BC08F41045647C54A98F24428AAE92DF267887E3F68E332F4719F07CA95B5
6751B4A5891513DF32C2E37AEABDB6A471B4F4862F6F03340C3E9073D5B3FB6823
9D543767741041 20FD2D28BDB700F6A05492F1D68C2DE05AAC601A30FE6255F27
F7FD132D0DA24E537D75DA25CE1D9E041B3AB190D191011F1BA933B43AC8468615
A534D871C7410432B9184B3A2231ED4B33B26721DA6A3E6CB3B9616853019273EE
71E46525684408436F1108BF69269543B1D90B6EB415C5F1A5A89E92F512D0A862
D164F815D020064AEF20845F88DCF30F5B76C4379EB11F50133602EE10FD0AD4CD
AC816E2CC441040F632A00F9E4953CAA01957EF33828670F6AEC23A99629254B97
7AB2C10499557EA1480724340525C46E1BC808686DD541B4533A17DE6EC485ED17
CFF38BC08F41040326C3CA03D58EB9B05ED5452E2EFD34F4A121BF57BA42A96840
F12A9CDE7A8B3BD0992B153E1CDCFF61AB94F73588D5CF056C83FF93CF3EE52CA6
AA15BFBC9B200167A0A02AF739D5E803BE8F11B4DC0442BC949ACD0EE27E604FCD
```

```
C8A59D5DFA41043149E778491245D26E933242F1CF8E3D566A2D8AEB522B2902D1
9EDD33DD51213EF04615C462E31BD89DE628A266388169CF86358C4BB9A3579C28
2A413416B64104256BF7368F63DFC182CED8C7249B84CF3A93233CDA2D9395DB8A
EE6C1C2D42DB3F6800354A39CADF252ED18913E177C59A880BA757C594A4291714
19C66253E220F68BD04192E05BC9F2E73E207B40E68E9E7900206987D930840CBD
44D9B7412E 9000 (698)
<--
04434152444104120FD2D28BDB700F6A05492F1D68C2DE05AAC601A30FE6255F27
F7FD132D0DA24E537D75DA25CE1D9E041B3AB190D191011F1BA933B43AC8468615
A534D871C741040F632A00F9E4953CAA01957EF33828670F6AEC23A99629254B97
7AB2C10499557EA1480724340525C46E1BC808686DD541B4533A17DE6EC485ED17
CFF38BC08F41045647C54A98F24428AAE92DF267887E3F68E332F4719F07CA95B5
6751B4A5891513DF32C2E37AEABDB6A471B4F4862F6F03340C3E9073D5B3FB6823
9D543767774104120FD2D28BDB700F6A05492F1D68C2DE05AAC601A30FE6255F27
F7FD132D0DA24E537D75DA25CE1D9E041B3AB190D191011F1BA933B43AC8468615
A534D871C7410432B9184B3A2231ED4B33B26721DA6A3E6CB3B9616853019273EE
71E46525684408436F1108BF69269543B1D90B6EB415C5F1A5A89E92F512D0A862
D164F815D020064AEF20845F88DCF30F5B76C4379EB11F50133602EE10FD0AD4CD
AC816E2CC441040F632A00F9E4953CAA01957EF33828670F6AEC23A99629254B97
7AB2C10499557EA1480724340525C46E1BC808686DD541B4533A17DE6EC485ED17
CFF38BC08F41040326C3CA03D58EB9B05ED5452E2EFD34F4A121BF57BA42A96840
F12A9CDE7A8B3BD0992B153E1CDCFF61AB94F73588D5CF056C83FF93CF3EE52CA6
AA15BFBC9B200167A0A02AF739D5E803BE8F11B4DC0442BC949ACD0EE27E604FCD
C8A59D5DFA41043149E778491245D26E933242F1CF8E3D566A2D8AEB522B2902D1
9EDD33DD51213EF04615C462E31BD89DE628A266388169CF86358C4BB9A3579C28
2A413416B64104256BF7368F63DFC182CED8C7249B84CF3A93233CDA2D9395DB8A
EE6C1C2D42DB3F6800354A39CADF252ED18913E177C59A880BA757C594A4291714
19C66253E220F68BD04192E05BC9F2E73E207B40E68E9E7900206987D930840CBD
44D9B7412E 9000 (698) [98 ms]
-->
AAEE0000F008434F4D505554455241043C6C420B2BC5084DD2DA1649CAF6543AEC
F1233D7259775688A8F3B2F86822F67DF2D1D932F3F71E92F3150ED69E2DC45038
CBE3528AADF46B82A14BFE3F6ADE41044E9A5621A0237E7141039BF5499C3A524F
1E309352069137B7F052D24B46EB2955209F4707B36B4D845D086CFAA87FC336BC
6A2AACB137694D38F36395C97DBC41043879C9CD3BE586E44F9BEE6404CCD86B07
C6157A4620EEABCCF2CB5D8DC99FE2400788D3423271ECEDB441D88BFA8DFFA5F6
34DFF8E0AEBD494460B25FDD9532200AD8D772F956FE17FBA44FCEE42D672B79DC
8898E89EAC2A6B6662511EB69C25
-->
[AAEE0000F008434F4D505554455241043C6C420B2BC5084DD2DA1649CAF6543AE
CF1233D7259775688A8F3B2F86822F67DF2D1D932F3F71E92F3150ED69E2DC4503
8CBE3528AADF46B82A14BFE3F6ADE41044E9A5621A0237E7141039BF5499C3A524
F1E309352069137B7F052D24B46EB2955209F4707B36B4D845D086CFAA87FC336B
C6A2AACB137694D38F36395C97DBC41043879C9CD3BE586E44F9BEE6404CCD86B0
7C6157A4620EEABCCF2CB5D8DC99FE2400788D3423271ECEDB441D88BFA8DFFA5F
634DFF8E0AEBD494460B25FDD9532200AD8D772F956FE17FBA44FCEE42D672B79D
C8898E89EAC2A6B6662511EB69C25] 245
<-- 9000
<-- 9000 [36 ms]
PC   : New shared session key established.
PC <-> SC: Key agreement finished successfully.
```

```
################################################
Begin: demo password storage

PC --> SC: Set password `All your` to slot 0.
-->
EE000000806C5544797A91115DC3330EBD003851D207253D39C257A973E99ECF8F
A9A7E49A4F421B560D00E07B4315C096CE2FC2DE0B27C0B900A7C7423D90C4564E
C07C872C17D954DC3F6264EF09C2C9B025A7A2A3CC3B10A5133079C651D00A7ED6
A081CCC579FA6105A87E01B75BD447621227F60FF1BEFB6E80015B0CCA7498D14E
CF
-->
[EE000000806C5544797A91115DC3330EBD003851D207253D39C257A973E99ECF8
FA9A7E49A4F421B560D00E07B4315C096CE2FC2DE0B27C0B900A7C7423D90C4564
EC07C872C17D954DC3F6264EF09C2C9B025A7A2A3CC3B10A5133079C651D00A7ED
6A081CCC579FA6105A87E01B75BD447621227F60FF1BEFB6E80015B0CCA7498D14
ECF] 133
<-- 9000
<-- 9000 [13 ms]
Error: Error in communication with the card.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ "IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!" @
@     PIN IS INCORRECT OR THIS SMARTCARD IS MALICIOUS     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Detailed info about this error:
applet.EmeraldProtocolException
      at
applet.SecureChannelManagerBase.decrypt(SecureChannelManagerBase.j
ava:172)
      at
emapplication.EmeraldApplicationCli.setPassword(EmeraldApplication
Cli.java:197)
      at
emapplication.EmeraldApplicationCli.demoPasswordStorage(EmeraldApp
licationCli.java:296)
      at
emapplication.EmeraldApplicationCli.run(EmeraldApplicationCli.java
:85)
      at
emapplication.EmeraldApplicationCli.main(EmeraldApplicationCli.jav
a:52)
exit
```