draft-urien-tls-psk-emv-00

EMV support for TLS-PSK

P.Urien, Telecom ParisTech

L.Cogneau and P. Martin, Xiring

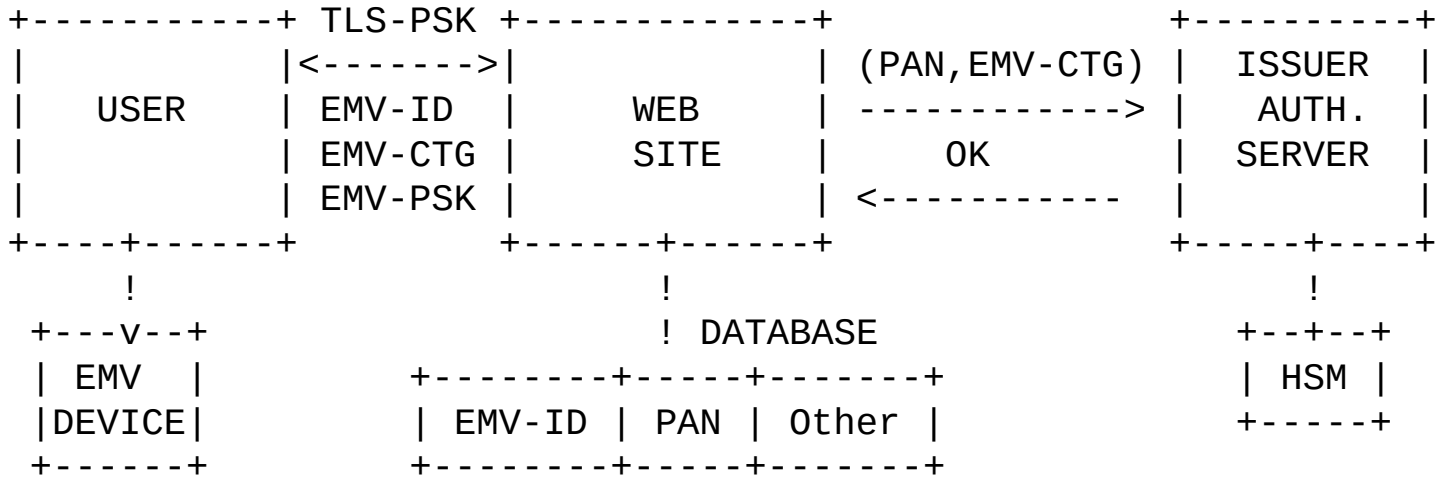
Pascal.Urien@telecom-paristech.fr

Goal

- **This draft describes an authentication protocol based on TLS pre shared key (TLS-PSK), RFC 4279.**
  - Identity and psk attributes, defined in TLS-PSK are extracted from EMV chips, which are widely deployed for payments transactions.
  - The goal of this protocol is to provide a strong mutual authentication transparent for the end users and guarantying the confidentiality and the integrity of exchanged data over Internet network.
- **This is a new step avoiding the use of static passwords for on-line services, such as electronic banking or electronic payment**

Global architecture

- The user
  - Holds an EMV device, whose card number is called PAN
  - Works with TLS-PSK
- The EMV device is registered to the WEB site
  - A database establishes a relation between the EMV-ID, used in TLS-PSK and the PAN (card number)
- The cryptogram (EMV-CTG) produced by the EMV-Device is checked by the EMV device issuer



RFC 4279, PSK

```
struct {
select (KeyExchangeAlgorithm)
 {   case psk:
     opaque psk_identity_hint<0..2^16-1>;
};
} ServerKeyExchange;

struct {
select (KeyExchangeAlgorithm)
{   case psk:
     opaque psk_identity<0..2^16-1>;
 } exchange_keys;
 } ClientKeyExchange;
```

- **PreMasterSecret**
  - 02 bytes (PSK length)
  - N bytes NULL (0)
  - 02 bytes (PSK Length)
  - N bytes (PSK value)

RFC 4279, DHE-PSK

```
struct {
select (KeyExchangeAlgorithm)
 {   case diffie_hellman_psk:
     opaque psk_identity_hint<0..2^16-1>;
};
} ServerKeyExchange;

struct {
select (KeyExchangeAlgorithm)
{   case diffie_hellman_psk:
     opaque psk_identity<0..2^16-1>;
 } exchange_keys;
 } ClientKeyExchange;
```

- **PreMasterSecret**
  - 02 bytes (DH length)
  - DH value
  - 02 bytes (PSK Length)
  - PSK value

RFC 4279, RSA-PSK

```
struct {
select (KeyExchangeAlgorithm)
 {   case rsa_psk:
     opaque psk_identity_hint<0..2^16-1>;
};
} ServerKeyExchange;

Struct {
select (KeyExchangeAlgorithm)
{   case rsa_psk:
     opaque psk_identity<0..2^16-1>;
     EncryptedPreMasterSecret;
 } exchange_keys;
 } ClientKeyExchange;
```

- **PreMasterSecret**
  - 02 bytes (0048)
  - 02 bytes, version
  - 46 bytes, random
  - 02 bytes PSK Length
  - N bytes (PSK value)

EMV-Device structure

- **An EMV smart card contains one or several EMV applications.**
- **An EMV application manages a set of information that can be freely read.**
  - These data are encoding according to the ASN.1 syntax.
- **An EMV application produces cryptograms that authenticate payment transactions.**
- **A Data Object List (DOL), is a list of tuples TAG value (one or two bytes) and object length (one byte)**

EMV details1/2

- **Application Primary Account Number (PAN)**
  - The card number
  - Tag 5A, Length 08, Value: 49 73 01 97 61 90 02 78
- **Application PAN Sequence Number (PSN)**
  - An additional identifier for the card
  - Tag 34, Length 01, Value: 00
- **Signed Static Application Data (SSAD)**
  - A signature for a set of information stored in the card.
  - Tag 93

EMV Details 2/2

- **Card Risk Management Data 1 (CDOL1)**
  - CDOL1 is the list of objects required for the generation of a transaction cryptogram
  - Tag 8C, Length 1B, Value: 9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 9F 45 02 9F 4C 08
- **Card Risk Management Data 2 (CDOL2)**
  - CDOL2 is the list of objects required for the completion of a transaction.
  - It is the concatenation of the Authorization Response Code (TAG 8A, length 02) and the CDOL1 value.
  - Tag 8D, Length 1A, Value: 8A 02 9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 9F 4C 08

ARQC & AAC

- **ARQC**
  - The Authorization Request Cryptogram (ARQC) starts an EMV transaction.
  - A set of values, whose elements are listed by the CDOL1 object, and without TAG or length information, are pushed towards the card.
  - The content of CDOL1 is noted xCDOL1 and the response to ARQC is noted yCDOL1.
    - xCDOL1 comprises an Unpredictable Number (TAG 9F37, length 04), i.e. a random value of 32 bits.
    - The response yCDOL1, includes an 8 bytes cryptogram and additional information.
- **AAC**
  - The Application Authentication Cryptogram ends an EMV transaction.
  - A set of values, whose elements are listed by the CDOL2 object, and without TAG or length information, are pushed towards the card.
  - The content of CDOL2 is noted xCDOL2 and the response to AAC is noted xCDOL2.

ARQC & AAC example

- **ARQC**
  - xCDOL1
    - 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 01 01 01 00
    - **00 00 00 00 = r32** = 32 bits random number
    - 00 00 00 00 00 00 00 00 00 00
  - yCDOL1
    - 77 21 9F 27 01 80 9F 36 02 01 2E 9F 26 08
    - **3F 79 8C 12 3E F2 9A 51 = AC1**
    - 9F 10 0A 06 16 0A 03 A4 80 00 02 00 00
- **AAC**
  - xCDOL2
    - 5A 33 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 01 01 01 00
    - 00 00 00 00 = 32 bits random number
    - 00 00 00 00 00 00 00 00
  - yCDOL2
    - 77 21 9F 27 01 00 9F 36 02 01 2E 9F 26 08
    - 82 8E 0A 3E 70 D4 4A D4 = AC2
    - 9F 10 0A 06 16 0A 03 25 80 00 02 00 00

TLS-PSK-EMV

- **The basic idea of this protocol is to used the PAN, i.e. the card number as a static PSK.**
  - However the PAN entropy is small, about 36 bits, so brute force attacks are possible. In order to avoid these issues, the PAN value is replaced by others parameters stored in the card and providing sufficient entropy, e.g. greater than 80 bits.
  - The psk-identity is a list of information proving that the client holds the card associated with the PAN.
    - This proof is based on an ARQC associated with a 32 bits random number, which is noted r32.

TLS-PSK-EMV definitions

- **EMV-ID**
  - EMV-ID = h(h(SSAD)), where
    - SSAD is the Signed Static Application Data,
    - and h is a digest function
- **EMV-CPG**
  - The EMV cryptogram (emv-cpg) is the response (yCDOL1) to an ARQC associated with the r32 random number. The ARQC request is followed by an AAC operation that cancels the EMV transaction.
  - Values used for ARQC and AAC (xCDOL1 and xCDOL2) are fix, apart from the unpredictable number set to r32.
  - By convention, the R32 number is a concatenation of multiple r32i values (up to four), and EMV-CPG is the concatenation of associated emv-cpgi, with the index i ranging between 1 and R32-length/4.
- **EMV-PSK**
  - EMV-PSK = h(SSAD), where
    - SSAD is the Signed Static Application Data,
    - and h is a digest function

TLS-PSK & TLS-PSK-DH

- **psk = EMV-PSK.**
- **psk-identity**
  - RH = h(ClientRandom | ServerRandom), where h is a digest function.
    - R32 is the 32 less significant bits of RH.
  - The psk-identity is the concatenation of the following values:
    - uint16(length) | R32
    - uint16(length) | EMV-ID
    - unit16(length) | PSN
    - uint16(length) | CDOL1
    - uint16(length) | EMV-CPG

TLS-PSK-RSA

- **psk = EMV-PSK**
- **psk-identity**
  - RH = h(ClientRandom | ServerRandom | ServerPublicKey), where h is a digest function.
  - R32 is the 32 (or more) less significant bits of RH.
  - The psk-identity is the concatenation of the following values:
    - uint16(length) | R32
    - uint16(length) | EMV-ID
    - unit16(length) | PSN
    - uint16(length) | CDOL1
    - uint16(length) | EMV-CPG

Questions ?