**Mutual Key Generation Between SIM Card and Service Provider**

**Date:** 23 December 2016

**Author:** Kerem Ok

Contents

[1. SIMSec Protocol 2](#_Toc475536014)

[1.1 Details of the SIMSec Protocol 2](#_Toc475536015)

[1.2 Protocol Values 3](#_Toc475536016)

[1.3. Implementation of the SIMSec Protocol on un-keyed SIM Cards 5](#_Toc475536017)

[2. SIMSec Sample Value Set 6](#_Toc475536018)

[3. Conclusion 9](#_Toc475536019)

# 1. SIMSec Protocol

The objective of the proposed protocol is to design a security infrastructure to provide end-to-end encryption between the SP (Service Provider) and the SIM card. After this infrastructure is set up, SP and SIM card will be able to generate symmetric encryption keys at both sides.

## 1.1 Details of the SIMSec Protocol

The SIMSec protocol is given in Figure 1. We describe firstly the protocol here and then the values and hash functions used in the protocol.

As can be seen from the protocol, the SP initially generates a random 10-character long value, V, in step (1); and sends this value to the SIM card in step (2).

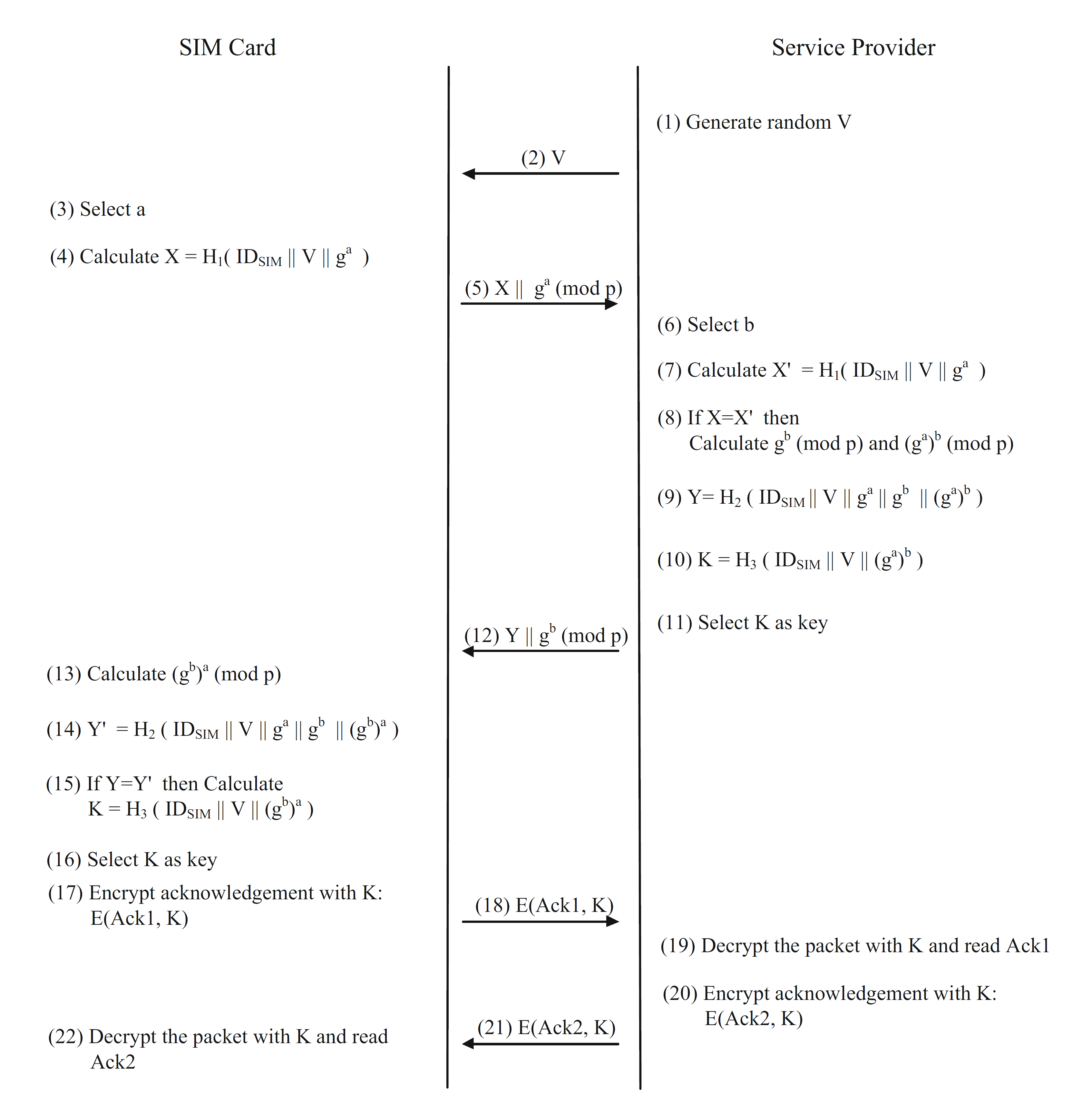
In step (3), the SIM card randomly generates a private Diffie–Hellman secret value, a. Afterwards; it calculates ga and computes the hash of IDSIM || V || ga values in step (4). Then in step (5), it sends this hash value with ga to the SP.

If the SP receives the packet from the SIM card in the pre-set time period after the exchange of V, it randomly generates a private Diffie–Hellman secret, b, in step (6). The SP calculates hash value X′ in step (7) which is expected to be same as X. If the two hash values are not equal, then the SP terminates the protocol. If the two hash values are equal, the SP authenticates the SIM card and proceeds to step (8) by calculating gb and (ga)b values. In step (9), the SP calculates the Y value as the hash of IDSIM || V || ga || gb || (ga)b. Then in step (10), the SP calculates K by computing the hash of IDSIM || V || (ga)b values. After calculating the K value in step (11), SP sends the Y and gb values to the SIM card in step (12).

After the SIM card receives the packet, it calculates (gb)a in step (13) which should give the same result as (ga)b calculated by SP in step (8). In step (14), the SIM card calculates the Y′ value which should be equal to the Y value that is calculated by the SP. The only difference between Y′ and Y is that when calculating Y′, (gb)a is inputted to the hash function instead of (ga)b. Then in step (15), the SIM card calculates the K value as in step (11) and selects the K value as a key in step (16).

In order to notify the SP of the key agreement, the SIM card creates an acknowledgment and encrypts it with K in step (17), and sends it to the SP in step (18). The SP decrypts the packet and registers mutual identification of the key. Then, the SP creates an acknowledgement and encrypts it using K in step (20) and sends this encrypted packet to the SIM card in step (21). The SIM card decrypts the packet and reads the acknowledgement. If the SIM card does not receive the final acknowledgement in step (22) (e.g. in case of a packet loss), it resends the packet in step (18) to the SP if the lifetime of the V value (described below) has not expired. Otherwise, the protocol re-starts. In case of any packet loss in steps (2), (5), or (12), unreceived packets will not be resent, thus V’s lifetime will expire, and the protocol will be restarted from the beginning.

Note: All ga, gb, (ga)b, (gb)a calculations are in mod p. || stands for concatenation.

  
Figure 1. SIMSec Protocol

## 1.2 Protocol Values

This section describes the values used throughout the protocol.

**p Value**

p is a public variable which is used as the modulus in all computations. The length is 1024 bits.

FF FF FF FF FF FF FF FF C9 0F DA A2 21 68 C2 34

C4 C6 62 8B 80 DC 1C D1 29 02 4E 08 8A 67 CC 74

02 0B BE A6 3B 13 9B 22 51 4A 08 79 8E 34 04 DD

EF 95 19 B3 CD 3A 43 1B 30 2B 0A 6D F2 5F 14 37

4F E1 35 6D 6D 51 C2 45 E4 85 B5 76 62 5E 7E C6

F4 4C 42 E9 A6 37 ED 6B 0B FF 5C B6 F4 06 B7 ED

EE 38 6B FB 5A 89 9F A5 AE 9F 24 11 7C 4B 1F E6

49 28 66 51 EC E6 53 81 FF FF FF FF FF FF FF FF

**g Value**

g is a public variable that is used as a base variable in exponentiation operations. The length is 1024 bits.

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D

**a Value**

a is generated randomly by the SIM card, and it remains private. The SIM card uses this value as a power in exponentiation. The length of a value is 384 bits.

**b Value**

b is generated randomly by the SP, and it remains private. SP uses this value as a power in exponentiation. The length of b value is 384 bits.

**V Value**

In SIMSec protocol, SMS channel is used for the communication except the exchange of V value. V is 10 characters long in which each character contains 64 possibilities (alphanumeric values and additional signs), which is generated randomly by SP for one time use only and has a limited lifetime (e.g. 5 minutes).

V value is sent to the SIM card owner using a communication channel other than SMS. We propose a secure Internet communication protocol such as https. For a secure exchange, the channel is expected to perform mutual authentication between the user and the SP. This is typically performed via the SP's secure web page such as an Internet branch. The user can enter this website by digitally authenticating herself and can get the V value. As the V value is received by the user, she inputs it to the SP's SIM toolkit application that is installed to the SIM card.

**IDSIM Value**

IDSIM value is the identification data of the SIM card. Unique digits of IMSI and ICCID are used to form IDSIM value and the length of IDSIM is 96 bits. Since both MNO and SIM card are capable of calculating this value, SIM card calculates this value itself. When required, SP receives IDSIM value from MNO via their existing secure communication channel before the protocol starts. This secure communication channel generally exist between MNOs and SPs; otherwise they need to set it up.

**H1 Function**

H1 implements a hash function (SHA-1) and uses 128 Least Significant Bits (LSB) of the output. SP and SIM card should agree on the hashing function that they will use when developing the SIMSec application. Following data is inputted to the each hash function in the provided order:

1. 32 bit function type (1 for H1)
2. 32 bit value for the main input's bit length
3. Main input which is the concatenation of SIM card id, V and ga (mod p):

(IDSIM || V || ga (mod p))

**H2 Function**

H2 implements a hash function (SHA-1) and uses 128 LSB bit of the output. Following data is inputted to the hash function in order:

1. 32 bit function type (2 for H2)
2. 32 bit value for the main input's bit length
3. Main input which is the concatenation SIM card id, V, ga (mod p), gb (mod p), and (gb)a (mod p)

IDSIM || V || ga (mod p) || gb (mod p) || (gb)a (mod p)

**H3 Function**

H3 implements a hash function (SHA-1) and uses 160 LSB bit of the output. Following data is inputted to the hash function in order:

1. 32 bit function type (3 for H3)
2. 32 bit value for the main input's bit length
3. Main input which is the concatenation SIM card id, V, (gb)a (mod p)

IDSIM || V || (gb)a (mod p)

**Final Key**

Final key is the concatenation of H3 function’s output and V value. The length is 168 bits.

## 1.3. Implementation of the SIMSec Protocol on un-keyed SIM Cards

Calculating ga (mod p) and (gb)a (mod p) values are possible with RSA encryption. In order to calculate the Diffie-Hellman values, following mappings are used:

* For the message content (m value) in RSA encryption, we used g value of Diffie-Hellman,
* For the first public key of user A (e value) in RSA encryption, we used g value of Diffie-Hellman,
* For the second public key of user A (n value) in RSA encryption, we used p value of Diffie-Hellman.

In the protocol, RSA encryption (No padding) needs to be used to calculate Diffie-Hellman values using the crypto primitives provided in Java card. The details of the matching Diffie-Hellman values in RSA encryption are given in Table 1.

Table 1. Matching Diffie-Hellman values in RSA Encryption

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Base Value | Exponent | Mod | Result |
| RSA Encryption | M | e | n | me (mod n) |
| 1st Diffie-Hellman | G | a | p | ga (mod p) |
| 2nd Diffie-Hellman | gb (mod p) | a | p | (gb)a (mod p) |

# 2. SIMSec Sample Value Set

Table 2 gives a sample test set for SIMSec protocol

**Table 2:** SIMSec uygulamasının verileri.

| **Value Name** | **Value Length** | **Value** |
| --- | --- | --- |
| g | 128 bytes | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0C |
| p | 128 bytes | FF FF FF FF FF FF FF FF C9 0F DA A2 21 68 C2 34 C4 C6 62 8B 80 DC 1C D1 29 02 4E 08 8A 67 CC 74 02 0B BE A6 3B 13 9B 22 51 4A 08 79 8E 34 04 DD EF 95 19 B3 CD 3A 43 1B 30 2B 0A 6D F2 5F 14 37 4F E1 35 6D 6D 51 C2 45 E4 85 B5 76 62 5E 7E C6 F4 4C 42 E9 A6 37 ED 6B 0B FF 5C B6 F4 06 B7 ED EE 38 6B FB 5A 89 9F A5 AE 9F 24 11 7C 4B 1F E6 49 28 66 51 EC E6 53 81 FF FF FF FF FF FF FF FF |
| a | 48 bytes | 0E A9 BB 7A BD 7D 65 40 2B 08 C6 DF C9 4B 09 6A 29 3B C2 42 88 23 44 AF 08 84 21 FE 0B A4 CA F9 7D BC FC 82 4C FF 42 A4 B8 D2 DA CC EE C5 34 ED |
| b | 48 bytes | 88 23 44 AF 08 84 21 FE 0B A4 CA F9 7D BC FC 0E A9 BB 7A BD 7D 65 40 2B 08 C6 DF C9 4B 09 6A 29 3B C2 42 82 4C FF 42 A4 B8 D2 DA CC EE C5 34 ED |
| ga (mod p) | 128 bytes | 4a a9 76 37 f5 2e aa 30 39 6e bf 64 89 04 f9 2b e1 93 44 82 06 80 eb 17 3e dd b8 a8 b9 56 cb 7d a8 72 ad a4 38 67 36 bd 79 e1 a4 64 c3 fd 04 ba 03 be 7a d1 41 a0 95 bd ba 60 5c 2f 25 62 85 20 46 75 11 62 ca fa 38 62 e9 c7 68 e3 09 74 f7 11 66 f0 30 34 7c 18 9d ea 66 3f 2a 90 79 72 51 73 5e d0 ae 79 cf 3c 8f 3d 16 7c 29 8e a0 66 6b 7a c5 f8 9b b0 bd cc 2f 73 6c b5 02 0f b6 98 73 b7 |
| gb (mod p) | 128 bytes | 36 cd 9c 43 eb 45 46 8f 5d d2 b9 6f a1 16 39 de 35 5c 39 4f 03 d5 52 44 59 72 1c eb 48 66 da 80 0f a3 13 ac 08 43 aa 19 1d 73 ea 96 84 18 9e e1 34 5c 97 e2 c8 24 b7 07 d1 f8 60 26 a2 d1 c2 6d 41 37 d7 f3 45 31 2f e0 fe b3 5f ad ca c1 c3 9b d2 13 a0 71 fe 90 bf 7c a4 d9 6b 74 0a 75 16 bb 31 de f6 82 98 c9 03 78 ad 60 af e4 c6 1e 77 92 44 86 1a 7a be 7d 45 29 71 91 a1 fc 41 82 d9 12 |
| gab (mod p) | 128 bytes | 5f 36 6f de 5f 35 cf 74 c5 d4 64 61 24 61 45 8a 61 1d ab 34 02 36 a6 5a f7 25 12 4e 7e 26 5c 22 3f 51 45 3b d9 5c 89 f9 a9 2d 4a 5d 54 80 94 6b 3c 27 e2 53 9a 5b e4 b6 11 0c 5d 0b 6f b2 ad c3 8a 2e d8 60 8b 40 c2 db e0 cb 7f 0a 6a ab f5 5a 76 a1 0e ba cc cc d0 bb 32 96 4e a9 f2 11 e6 57 00 0d 60 6e 2e 86 5f ec 72 07 4c 5b da 17 8a b1 f3 9a 23 0a e2 aa b1 13 00 56 53 f2 7e 96 c4 30 |
| gba (mod p) | 128 bytes | 5f 36 6f de 5f 35 cf 74 c5 d4 64 61 24 61 45 8a 61 1d ab 34 02 36 a6 5a f7 25 12 4e 7e 26 5c 22 3f 51 45 3b d9 5c 89 f9 a9 2d 4a 5d 54 80 94 6b 3c 27 e2 53 9a 5b e4 b6 11 0c 5d 0b 6f b2 ad c3 8a 2e d8 60 8b 40 c2 db e0 cb 7f 0a 6a ab f5 5a 76 a1 0e ba cc cc d0 bb 32 96 4e a9 f2 11 e6 57 00 0d 60 6e 2e 86 5f ec 72 07 4c 5b da 17 8a b1 f3 9a 23 0a e2 aa b1 13 00 56 53 f2 7e 96 c4 30 |
| V | 10 bytes | O c – Y J b 0 V w 0  4F 63 2D 59 4A 62 30 56 77 30 (corresponding ASCI bytes) |
| IDSIM | 12 bytes | 31 50 92 00 00 70 F8 15 32 62 23 71 |
| H1 input | 158 bytes | 31 50 92 00 00 70 f8 15 32 62 23 71 4f 63 2d 59 4a 62 30 56 77 30 4a a9 76 37 f5 2e aa 30 39 6e bf 64 89 04 f9 2b e1 93 44 82 06 80 eb 17 3e dd b8 a8 b9 56 cb 7d a8 72 ad a4 38 67 36 bd 79 e1 a4 64 c3 fd 04 ba 03 be 7a d1 41 a0 95 bd ba 60 5c 2f 25 62 85 20 46 75 11 62 ca fa 38 62 e9 c7 68 e3 09 74 f7 11 66 f0 30 34 7c 18 9d ea 66 3f 2a 90 79 72 51 73 5e d0 ae 79 cf 3c 8f 3d 16 7c 29 8e a0 66 6b 7a c5 f8 9b b0 bd cc 2f 73 6c b5 02 0f b6 98 73 b7 |
| X  (H1 output) | LSB 16 bytes | 94 36 0d e5 01 25 b2 22 90 5c b0 06 cb c1 1b 1c |
| H2 input | 414 bytes | 31 50 92 00 00 70 f8 15 32 62 23 71 4f 63 2d 59 4a 62 30 56 77 30 4a a9 76 37 f5 2e aa 30 39 6e bf 64 89 04 f9 2b e1 93 44 82 06 80 eb 17 3e dd b8 a8 b9 56 cb 7d a8 72 ad a4 38 67 36 bd 79 e1 a4 64 c3 fd 04 ba 03 be 7a d1 41 a0 95 bd ba 60 5c 2f 25 62 85 20 46 75 11 62 ca fa 38 62 e9 c7 68 e3 09 74 f7 11 66 f0 30 34 7c 18 9d ea 66 3f 2a 90 79 72 51 73 5e d0 ae 79 cf 3c 8f 3d 16 7c 29 8e a0 66 6b 7a c5 f8 9b b0 bd cc 2f 73 6c b5 02 0f b6 98 73 b7 36 cd 9c 43 eb 45 46 8f 5d d2 b9 6f a1 16 39 de 35 5c 39 4f 03 d5 52 44 59 72 1c eb 48 66 da 80 0f a3 13 ac 08 43 aa 19 1d 73 ea 96 84 18 9e e1 34 5c 97 e2 c8 24 b7 07 d1 f8 60 26 a2 d1 c2 6d 41 37 d7 f3 45 31 2f e0 fe b3 5f ad ca c1 c3 9b d2 13 a0 71 fe 90 bf 7c a4 d9 6b 74 0a 75 16 bb 31 de f6 82 98 c9 03 78 ad 60 af e4 c6 1e 77 92 44 86 1a 7a be 7d 45 29 71 91 a1 fc 41 82 d9 12 5f 36 6f de 5f 35 cf 74 c5 d4 64 61 24 61 45 8a 61 1d ab 34 02 36 a6 5a f7 25 12 4e 7e 26 5c 22 3f 51 45 3b d9 5c 89 f9 a9 2d 4a 5d 54 80 94 6b 3c 27 e2 53 9a 5b e4 b6 11 0c 5d 0b 6f b2 ad c3 8a 2e d8 60 8b 40 c2 db e0 cb 7f 0a 6a ab f5 5a 76 a1 0e ba cc cc d0 bb 32 96 4e a9 f2 11 e6 57 00 0d 60 6e 2e 86 5f ec 72 07 4c 5b da 17 8a b1 f3 9a 23 0a e2 aa b1 13 00 56 53 f2 7e 96 c4 30 |
| Y  (H2 output) | 16 bytes | 91 53 14 32 82 3d 10 d3 04 08 1e 7f 17 5c f4 c4 |
| H3 Input | 158 bytes | 31 50 92 00 00 70 f8 15 32 62 23 71 4f 63 2d 59 4a 62 30 56 77 30 5f 36 6f de 5f 35 cf 74 c5 d4 64 61 24 61 45 8a 61 1d ab 34 02 36 a6 5a f7 25 12 4e 7e 26 5c 22 3f 51 45 3b d9 5c 89 f9 a9 2d 4a 5d 54 80 94 6b 3c 27 e2 53 9a 5b e4 b6 11 0c 5d 0b 6f b2 ad c3 8a 2e d8 60 8b 40 c2 db e0 cb 7f 0a 6a ab f5 5a 76 a1 0e ba cc cc d0 bb 32 96 4e a9 f2 11 e6 57 00 0d 60 6e 2e 86 5f ec 72 07 4c 5b da 17 8a b1 f3 9a 23 0a e2 aa b1 13 00 56 53 f2 7e 96 c4 30 |
| H3 Output | 16 bytes | 99 23 9b a5 e8 48 d3 a5 00 fa bd b3 41 11 44 9f ab d8 a9 0c |
| Key | 21 bytes | 99 23 9b a5 e8 48 d3 63 a5 00 fa bd b3 41 11 2d 44 9f ab d8 a9 0c 4f 59 |

# 3. Conclusion

In this study, we propose a key exchange protocol between un-keyed SIM cards and SPs. When SIM card and SP execute the developed SIMSec protocol, the parties collaboratively create a symmetric key that can be further used for end-to-end encryption and consequently enables the SIM card and SP to establish and facilitate emerging applications via SIM cards such as NFC payment.