Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2017

S. Smyshlyaev, Ed.
E. Alekseev
I. Oshkin
V. Popov
CRYPTO-PRO

October 27, 2016

The Security Evaluated Standardized Password Authenticated Key Exchange (SESPAKE) Protocol

draft-smyshlyaev-sespake-11

Abstract

This document specifies the Security Evaluated Standardized Password Authenticated Key Exchange (SESPAKE) protocol. The SESPAKE protocol provides password authenticated key exchange for usage in the systems for protection of sensitive information. The security proofs of the protocol were made for the case of an active adversary in the channel, including MitM attacks and attacks based on the impersonation of one of the subjects.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Conventions used in this document 2
3. Notations 3
4. Protocol description 4
4.1. Protocol parameters 5
4.2. Initial values of the protocol counters 6
4.3. Protocol steps 6
5. Construction of points Q_1,...,Q_N 11
6. Acknowledgments 12
7. Security Considerations 12
8. References 13
8.1. Normative References 13
8.2. Informative References 14
Appendix A. Test examples for GOST-based protocol implementation 14
A.1. Examples of points 14
A.2. Test examples 20
Authors' Addresses 31

1. Introduction

The current document contains the description of the password authenticated key exchange protocol SESPAKE (security evaluated standardized password authenticated key exchange) for usage in the systems for protection of sensitive information. The protocol is intended to use for establishment of keys that are then used for organization of secure channel for protection of sensitive information. The security proofs of the protocol were made for the case of an active adversary in the channel, including MitM attacks and attacks based on the impersonation of one of the subjects.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Notations

This document uses the following parameters of elliptic curves in accordance with [RFC6090]:

E an elliptic curve defined over a finite prime field GF(p), where p > 3;

p the characteristic of the underlying prime field;

a, b the coefficients of the equation of the elliptic curve in the canonical form;

m the elliptic curve group order;

q the elliptic curve subgroup order;

P a generator of the subgroup of order q;

X, Y the coordinates of the elliptic curve point in the canonical form;

O zero point (point of infinity) of the elliptic curve.

This memo uses the following functions:

HASH the underlying hash function;

HMAC the function for calculating a message authentication code, based on a HASH function in accordance with [RFC2104];

F(PW, salt, n) the value of the function PBKDF2(PW,salt,n,len), where PBKDF2(PW,salt,n,len) is calculated according to [RFC2898] The parameter len is considered equal to minimal integer that is a multiple of 8 and satisfies the following condition:
len >= floor(log_2(q)).

This document uses the following terms and definitions for the sets and operations on the elements of these sets

B_n the set of byte strings of size n, n >= 0, for n = 0 the B_n set consists of a single empty string of size 0; if b is an element of B_n, then b = (b_1,...,b_n), where b_1,...,b_n are elements of {0,...,255};

|| concatenation of byte strings A and C, i.e., if A in B_n1, C in B_n2, A = (a_1,a_2,...,a_n1) and C = (c_1,c_2,...,c_n2),

then A||C = (a_1,a_2,...,a_n1,c_1,c_2,...,c_n2) is an element of B_(n1+n2);

int(A) for the byte string A= (a_1,...,a_n) in B_n an integer int(A) = 256^(n-1)a_n +...+ 256^(0)a_1;

bytes_n(X) the byte string A in B_n such that int(A) = X, where X is integer, 0 <= X < 256^n;

BYTES(Q) for Q in E, the byte string bytes_n(X) || bytes_n(Y), where X, Y are standard Weierstrass coordinates of point Q and n = ceil(log_{256}(p)).

4. Protocol description

The main point of the SESPAKE protocol is that parties sharing a weak key (a password) generate a strong common key. The active adversary who has an access to a channel isn't able to obtain any information that can be used to find a key in offline mode, i.e. without interaction with legitimate participants.

The protocol is used by the subjects A (client) and B (server) that share some secret parameter that was established in an out-of-band mechanism: a client is a participant who stores a password as a secret parameter and a server is a participant who stores a password-based computed point of the elliptic curve.

The SESPAKE protocol consists of two steps: the key agreement step and the key confirmation step. During the first step (the key agreement step) the parties exchange keys using Diffie-Hellman with public components masked by element that depends on the password - one of the predefined elliptic curve points multiplied by the password-based coefficient. This approach provides an implicit key authentication, which means that after this step one party is assured that no other party aside from a specifically identified second party may gain access to the generated secret key. During the second step (the key confirmation step) the parties exchange strings that strongly depend on the generated key. After this step the parties are assured that a legitimate party and no one else actually has possession of the secret key.

To protect against online guessing attacks the failed connections counters were introduced in the SESPAKE protocol. There is also a special way of a small order point processing and a mechanism that provides a reflection attack protection by using different operations for different sides.

4.1. Protocol parameters

Various elliptic curves can be used in the protocol. For each elliptic curve supported by clients the following values MUST be defined:

- o the protocol parameters identifier ID_ALG (which can also define a HASH function, PRF used in PBKDF2 function, etc.), that is a byte string of an arbitrary length;

- o the point P, that is a generator point of the subgroup of order q of the curve;

- o the set of distinct curve points {Q_1,Q_2,...,Q_N} of order q, where the total number of points N is defined for protocol instance.

The method of generation of the points {P,Q_1,Q_2,...,Q_N} is described in Section 5.

The protocol parameters that are used by subject A are the following:

1. The secret password value PW, which is a byte string that is uniformly randomly chosen from a subset of cardinality 10^10 or greater of the set B_k, where k >= 6 is password length.

2. The list of curve identifiers supported by A.

3. Sets of points {Q_1,Q_2,...,Q_N}, corresponding to curves supported by A.

4. The C_1^A counter, that tracks the total number of unsuccessful authentication trials in a row, and a value of CLim_1 that stores the maximum possible number of such events.

5. The C_2^A counter, that tracks the total number of unsuccessful authentication events during the period of usage of the specific PW, and a value of CLim_2 that stores the maximum possible number of such events.

6. The C_3^A counter, that tracks the total number of authentication events (successful and unsuccessful) during the period of usage of the specific PW, and a value of CLim_3 that stores the maximum possible number of such events.

7. The unique identifier ID_A of the subject A (OPTIONAL), which is a byte string of an arbitrary length.

The protocol parameters that are used by subject B are the following:

1. The values ind and salt, where ind is in {1,...,N}, salt is in {1,...,2^128-1}.

2. The point Q_PW, satisfying the following equation:

   Q_PW = int (F (PW, salt, 2000))*Q_ind.

   It is possible that the point Q_PW is not stored and is calculated using PW in the beginning of the protocol. In that case B has to store PW and points Q_1,Q_2,...,Q_N.

3. The ID_ALG identifier.

4. The C_1^B counter, that tracks the total number of unsuccessful authentication trials in a row, and a value of CLim_1 that stores the maximum possible number of such events.

5. The C_2^B counter, that tracks the total number of unsuccessful authentication events during the period of usage of the specific PW, and a value of CLim_2 that stores the maximum possible number of such events.

6. The C_3^B counter, that tracks the total number of authentication events (successful and unsuccessful) during the period of usage of the specific PW, and a value of CLim_3 that stores the maximum possible number of such events.

7. The unique identifier ID_B of the subject B (OPTIONAL), which is a byte string of an arbitrary length.

4.2. Initial values of the protocol counters

After the setup of a new password value PW the values of the counters MUST be assigned as follows:

- C_1^A = C_1^B = CLim_1, where CLim_1 is in {3,...,5};

- C_2^A = C_2^B = CLim_2, where CLim_2 is in {7,...,20};

- C_3^A = C_3^B = CLim_3, where CLim_3 is in {10^3,10^3+1,...,10^5}.

4.3. Protocol steps

The basic SESPAKE steps are shown in the scheme below:

```
+-------------------------+-------------+---------------------------+
```

```
   |       A [A_ID, PW]      |             |    B [B_ID, Q_PW , ind,   |
   |                         |             |           salt]           |
   +-------------------------+-------------+---------------------------+
   |   if C_1^A or C_2^A or  |             |                           |
   |    C_3^A = 0 ==> QUIT   |             |                           |
   |                         |             |                           |
   | decrement C_1^A, C_2^A, |  A_ID --->  |    if C_1^B or C_2^B or   |
   |        C_3^A by 1       |             |     C_3^B = 0 ==> QUIT    |
   |                         |             |                           |
   |         z_A = 0         |    <---     |  decrement C_1^B, C_2^B,  |
   |                         |   ID_ALG,   |         C_3^B by 1        |
   |                         |    B_ID     |                           |
   |                         | (OPTIONAL), |                           |
   |                         |  ind, salt  |                           |
   |                         |             |                           |
   |    Q_PW^A = int(F(PW,   |             |                           |
   |   salt, 2000)) * Q_ind  |             |                           |
   |                         |             |                           |
   |  choose alpha randomly  |             |                           |
   |     from {1,...,q-1}    |             |                           |
   |                         |             |                           |
   |  u_1 = alpha*P - Q_PW^A |   u_1 --->  |  if u_1 not in E ==> QUIT |
   |                         |             |                           |
   |                         |             |          z_B = 0          |
   |                         |             |                           |
   |                         |             |      Q_B = u_1 + Q_PW     |
   |                         |             |                           |
   |                         |             |   choose betta randomly   |
   |                         |             |      from {1,...,q-1}     |
   |                         |             |                           |
   |                         |             |  if m/q*Q_B = O ==> Q_B = |
   |                         |             |      betta*P, z_B = 1     |
   |                         |             |                           |
   |                         |             | K_B = HASH(BYTES((m/q*bet |
   |                         |             |     ta*(mod q))*Q_B))     |
   |                         |             |                           |
   |   if u_2 not in E ==>   |   <--- u_2  |    u_2 = betta*P + Q_PW   |
   |           QUIT          |             |                           |
   |                         |             |                           |
   |    Q_A = u_2 - Q_PW^A   |             |                           |
   |                         |             |                           |
   |  if m/q*Q_A = O ==> Q_A |             |                           |
   |    = alpha*P, z_A = 1   |             |                           |
   |                         |             |                           |
   | K_A = HASH(BYTES((m/q*a |             |                           |
   |    lpha(mod q))*Q_A))   |             |                           |
   |                         |             |                           |
   |                         |             |                           |
```

```
|                          |             |                          |
| U_1 = BYTES(u_1), U_2 = |             |                          |
|        BYTES(u_2)        |             |                          |
|                          |             |                          |
|  MAC_A = HMAC(K_A, 0x01 |   DATA_A,   |  U_1 = BYTES(u_1), U_2 =  |
|  || ID_A || ind || salt |  MAC_A ---> |         BYTES(u_2)        |
| || U_1 || U_2 || ID_ALG |             |                          |
|  (OPTIONAL) || DATA_A)  |             |                          |
|                          |             |                          |
|                          |             |   if MAC_A != HMAC(K_B,   |
|                          |             |   0x01 || ID_A || ind ||  |
|                          |             |   salt || U_1 || U_2 ||   |
|                          |             |    ID_ALG (OPTIONAL) ||   |
|                          |             |      DATA_A) ==> QUIT     |
|                          |             |                          |
|                          |             |    if z_B = 1 ==> QUIT    |
|                          |             |                          |
|                          |             | C_1^B = CLim_1, increment |
|                          |             |         C_2^B by 1        |
|                          |             |                          |
|  if MAC_B != HMAC(K_A,   |     <---    | MAC_B = HMAC(K_B, 0x02 || |
|  0x02 || ID_B || ind || |   DATA_B,   |   ID_B || ind || salt ||  |
|  salt || U_1 || U_2 ||  |    MAC_B    |    U_1 || U_2 || ID_ALG   |
|   ID_ALG (OPTIONAL) ||  |             |  (OPTIONAL) || DATA_A ||  |
|  DATA_A || DATA_B) ==>  |             |          DATA_B)          |
|           QUIT           |             |                          |
|                          |             |                          |
|   if z_A = 1 ==> QUIT    |             |                          |
|                          |             |                          |
|     C_1^A = CLim_1,      |             |                          |
|   increment C_2^A by 1   |             |                          |
+--------------------------+-------------+--------------------------+
```

Table 1: SESPAKE protocol steps

The full description of the protocol consists of the following steps:

1. If any of the counters C_1^A, C_2^A, C_3^A is equal to 0, A finishes the protocol with an error that informs of exceeding the number of trials that is controlled by the corresponding counter.

2. A decrements each of the counters C_1^A, C_2^A, C_3^A by 1, requests open authentication information from B and sends the ID_A identifier.

3. If any of the counters C_1^B, C_2^B, C_3^B is equal to 0, B finishes the protocol with an error that informs of exceeding

the number of trials that is controlled by the corresponding counter.

4. B decrements each of the counters C_1^B, C_2^B, C_3^B by 1.

5. B sends the values of ind, salt and the ID_ALG identifier to A. B also can OPTIONALLY send the ID_B identifier to A. All following calculations are done by B in the elliptic curve group defined by the ID_ALG identifier.

6. A sets the curve defined by the received ID_ALG identifier as the used elliptic curve. All following calculations are done by A in this elliptic curve group.

7. A calculates the point Q_PW^A = int (F (PW, salt, 2000))*Q_ind.

8. A chooses randomly (according to the uniform distribution) the value alpha, alpha is in {1,...,q-1}, and assigns z_A = 0.

9. A sends the value u_1 = alpha*P - Q_PW^A to B.

10. After receiving u_1, B checks that u_1 is in E. If it is not, B finishes with an error, considering the authentication process unsuccessful.

11. B calculates Q_B = u_1 + Q_PW, assigns z_B = 0 and chooses randomly (according to the uniform distribution) the value betta, betta is in {1,...,q-1}.

12. If m/q*Q_B = O, B assigns Q_B = betta*P and z_B = 1.

13. B calculates K_B = HASH (BYTES((m/q*betta*(mod q))*Q_B)).

14. B sends the value u_2 = betta*P + Q_PW to A.

15. After receiving u_2, A checks that u_2 is in E. If it is not, A finishes with an error, considering the authentication process unsuccessful.

16. A calculates Q_A = u_2 - Q_PW^A.

17. If m/q*Q_A = O, then A assigns Q_A = alpha*P and z_A = 1.

18. A calculates K_A = HASH (BYTES((m/q*alpha(mod q))*Q_A)).

19. A calculates U_1 = BYTES(u_1), U_2 = BYTES(u_2).

20. A calculates MAC_A = HMAC (K_A, 0x01 || ID_A || ind || salt || U_1 || U_2 || ID_ALG (OPTIONAL) || DATA_A), where DATA_A is an OPTIONAL string that is authenticated with MAC_A (if it is not used, then DATA_A is considered to be of zero length).

21. A sends DATA_A, MAC_A to B.

22. B calculates U_1 = BYTES(u_1), U_2 = BYTES(u_2).

23. B checks that the values MAC_A and HMAC (K_B, 0x01 || ID_A || ind || salt || U_1 || U_2 || ID_ALG (OPTIONAL) || DATA_A) are equal. If they are not, it finishes with an error, considering the authentication process unsuccessful.

24. If z_B = 1, B finishes, considering the authentication process unsuccessful.

25. B sets the value of C_1^B to CLim_1 and increments C_2^B by 1.

26. B calculates MAC_B = HMAC(K_B, 0x02 || ID_B || ind || salt || U_1 || U_2 || ID_ALG (OPTIONAL) || DATA_A || DATA_B), where DATA_B is an OPTIONAL string that is authenticated with MAC_B (if it is not used, then DATA_B is considered to be of zero length).

27. B sends DATA_B, MAC_B to A.

28. A checks that the values MAC_B and HMAC (K_A, 0x02 || ID_B || ind || salt || U_1 || U_2 || ID_ALG (OPTIONAL) || DATA_A || DATA_B) are equal. If they are not, it finishes with an error, considering the authentication process unsuccessful.

29. If z_A = 1, A finishes, considering the authentication process unsuccessful.

30. A sets the value of C_1^A to CLim_1 and increments C_2^A by 1.

After the successful finish of the procedure the subjects A and B are mutually authenticated and each subject has an explicitly authenticated value of K = K_A = K_B.

N o t e s :

1. In the case when the interaction process can be initiated by any subject (client or server) the ID_A and ID_B options MUST be used and the receiver MUST check that the identifier he had received is not equal to his own, otherwise, it finishes the protocol. If an OPTIONAL parameter ID_A (or ID_B) is not used in the protocol,

it SHOULD be considered equal to a fixed byte string (zero-length string is allowed) defined by a specific implementation.

2. The ind, ID_A, ID_B and salt parameters can be agreed in advance. If some parameter is agreed in advance, it is possible not to send it during a corresponding step. Nevertheless, all parameters MUST be used as corresponding inputs to HMAC function during stages 20, 23, 26 and 28.

3. The ID_ALG parameter can be fixed or agreed in advance.

4. The ID_ALG parameter is RECOMMENDED to be used in HMAC during stages 20, 23, 26 and 28.

5. Continuation of protocol interaction in case of any of the counters C_1^A, C_1^B being equal to zero MAY be done without changing password. In this case these counters can be used for protection against denial-of-service attacks. For example, continuation of interaction can be allowed after a certain delay.

6. Continuation of protocol interaction in case of any of the counters C_2^A, C_3^A, C_2^B, C_3^B being equal to zero MUST be done only after changing password.

7. It is RECOMMENDED that during the stages 9 and 14 the points u_1 and u_2 are sent in a non-compressed format (BYTES(u_1) and BYTES(u_2)). However, the point compression MAY be used.

8. The use of several Q points can reinforce the independence of the data streams in case of working with several applications, when, for example, two high-level protocols can use two different points. However, the use of more than one point is OPTIONAL.

5. Construction of points Q_1,...,Q_N

This section provides an example of possible algorithm for generation of each point Q_i in the set {Q_1,...,Q_N} that corresponds to the given elliptic curve E.

The algorithm is based on choosing points with coordinates with a known preimages of a cryptographic hash function H, which is the GOST R 34.11-2012 hash function (see [RFC6986]) with 256-bit output, if 2^254 < q < 2^256, and the GOST R 34.11-2012 hash function (see [RFC6986]) with 512-bit output , if 2^508 < q < 2^512.

The algorithm consists of the following steps:

1. Choose an arbitrary SEED value with length of 32 bytes or more.

2. Calculate X = INT (H (SEED)) mod p, where INT is the function that maps the byte string A = (a_1,...,a_n), A is in B_n, into the integer a = 256^(n-1)a_1 +...+ 256^(0)a_n;

3. Check that the value of X^3 + aX + b is a quadratic residue in the field F_p. If it is not, return to Step 1.

4. Choose the value of Y arbitrarily from the set {+sqrt(R),-sqrt(R)}, where R = X^3 + aX + b. Here sqrt(R) is an element of F_p, for which (sqrt(R))^2 = R mod p.

5. Check that for point Q = (X,Y) the following relations hold: Q != O and q*Q = O. If they do not, return to Step 1.

With the defined algorithm for any elliptic curve E point sets {Q_1,...,Q_N} are constructed. Constructed points in one set MUST have distinct X-coordinates.

N o t e :The knowledge of a hash function preimage prevents knowledge of the multiplicity of any point related to generator point P. It is of primary importance, because such a knowledge could be used to implement an attack against protocol with exhaustive search of password.

N o t e : In case when N = 1 it is RECOMMENDED to generate Q_1 with X-coordinate equal to INT(HASH(BYTES(P)||seed)) mod p for the appropriate seed value.

6. Acknowledgments

We thank Lolita Sonina, Georgiy Borodin, Sergey Agafin and Ekaterina Smyshlyaeva for their careful readings and useful comments.

7. Security Considerations

Any cryptographic algorithms, particularly HASH function and HMAC function, that are used in the SESPAKE protocol MUST be carefully designed and MUST be able to withstand all known types of cryptanalytic attack.

It is RECOMMENDED that the HASH function satisfies the following condition:
hashlen <= log_2(q) + 4, where hashlen is the lengths of the HASH function output.

The output length of hash functions that are used in the SESPAKE protocol is RECOMMENDED to be greater or equal to 256 bits.

The points Q_1, Q_2,... Q_N and P MUST be chosen in such a way that they are provable pseudorandom. As a practical matter, this means that the algorithm for generation of each point Q_i in the set {Q_1,...,Q_N} (see Section 5) ensures that multiplicity of any point under any other point is unknown.

For a certain ID_ALG using N = 1 is RECOMMENDED.

N o t e: The exact adversary models, which have been considered during the security evaluation, can be found in the paper [SESPAKE-SECURITY], containing the security proofs.

8. References

8.1. Normative References

[GOST3410-2012]
Federal Agency on Technical Regulating and Metrology (In Russian), "Information technology. Cryptographic data security. Signature and verification processes of [electronic] digital signature", GOST R 34.10-2012, 2012.

[GOST3411-2012]
Federal Agency on Technical Regulating and Metrology (In Russian), "Information technology. Cryptographic Data Security. Hashing function", GOST R 34.11-2012, 2012.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, DOI 10.17487/RFC2898, September 2000, <http://www.rfc-editor.org/info/rfc2898>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <http://www.rfc-editor.org/info/rfc6090>.

[RFC6986] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.11-2012: Hash Function", RFC 6986, DOI 10.17487/RFC6986, August 2013, <http://www.rfc-editor.org/info/rfc6986>.

[RFC7091] Dolmatov, V., Ed. and A. Degtyarev, "GOST R 34.10-2012: Digital Signature Algorithm", RFC 7091, DOI 10.17487/RFC7091, December 2013, <http://www.rfc-editor.org/info/rfc7091>.

[RFC7836] Smyshlyaev, S., Ed., Alekseev, E., Oshkin, I., Popov, V., Leontiev, S., Podobaev, V., and D. Belyavsky, "Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012", RFC 7836, DOI 10.17487/RFC7836, March 2016, <http://www.rfc-editor.org/info/rfc7836>.

8.2. Informative References

[SESPAKE-SECURITY] Smyshlyaev, S., Oshkin, I., Alekseev, E., and L. Ahmetzyanova, "On the Security of One Password Authenticated Key Exchange Protocol", 2015, <http://eprint.iacr.org/2015/1237.pdf>.

Appendix A. Test examples for GOST-based protocol implementation

The following test examples are made for the protocol implementation that is based on the Russian national standards GOST R 34.10-2012 [GOST3410-2012] and GOST R 34.11-2012 [GOST3411-2012]. The English versions of these standards can be found in [RFC7091] and [RFC6986].

A.1. Examples of points

There are three points (Q_1, Q_2, Q_3) for each of the elliptic curves below. This points were constructed using the method described in Section 5, where the GOST R 34.11-2012 hash function (see [RFC6986]) with 256-bit output is used if 2^254 < q < 2^256, the GOST R 34.11-2012 hash function (see [RFC6986]) with 512-bit output is used if 2^508 < q < 2^512.

The same method should be used for constructing, if necessary, additional points. Each of the points complies with the GOST R 34.10-2012 [GOST3410-2012] standard and is represented by a pair of (X, Y) coordinates in the canonical form and by a pair of (U, V) coordinates in the twisted Edwards form in accordance with the document [RFC7836] for the curves that have the equivalent representation in this form. There is a SEED value for each point, by which it was generated.

A.1.1 Curve id-GostR3410-2001-CryptoPro-A-ParamSet

```
Point Q_1
  X= 0xa33ce065b0c23e1d3d026a206f8a1f8747ed1cd92a665bf85198cdb10ac90a5c
  Y= 0xb00d0dc0733883f05de9f55fd711f55998f5508cc40bead80c913b4d5b533667
  SEED:
     f8 18 95 b4 13 69 d9 08 9e 3d 3c 56 e8 70 ba 5e
     9d 55 5e 20 eb d9 c7 22 66 10 6d 79 c2 83 48 b8
     ce 63 70 52 9a 82 9f 18 a4 d3 e3 fd 7b f2 dd 73
     b1 1b bc d4 10 9d 27 c9 a3 d4 bd 3a 42 cc 26 ae
     43 5b 52 f5 89 a4 c3 b7 61 c0 1b a2 88 b7 e0 8d
     f9 4e 22 40 29 f3 aa 96 11 5c 43 f5 eb 87 99 70
Point Q_2
  X= 0x4ce9c2bcf17212b9efcab65c3c815c0ff96d7461c957634dbfd1fe7c9a324d27
  Y= 0xf7500d7adea2c2b4a16d838a8faa02b46639eb881f124d0f2506efca0e24289d
  SEED:
     fd 99 6d bc c7 2e 49 a4 37 e7 49 a8 85 ad de 28
     4b 58 64 bd 3b 7e 60 fc b5 2f c8 36 0e 0a bf 98
     fd 35 7a 3f 98 c5 f6 20 c8 68 3d b2 ca b9 27 b6
     13 f2 91 a1 52 45 c0 65 71 dc 62 b0 4f 2e e5 76
     56 a9 fa 51 12 23 5d 0b 80 67 59 af e2 33 b1 09
     6a 94 84 91 45 f2 18 50 65 b7 9b 86 ab 68 8a 39
Point Q_3
  X= 0x31fb8e5070b1e0f52f047f40477c38c6020fd8da9f685791f9237cc47bd89324
  Y= 0x8ba1184a4e296dc5c5873639747339ecc71b7fa44d31cc8e35b6615a4f797dd7
  SEED:
     29 0c 12 66 47 91 2e de 11 cc 43 78 0c f8 87 d4
     7d a1 63 bc fb 91 1d 92 86 2a ae 4f 53 a6 80 70
     08 c1 ec 0c 8f e7 2e 0a a0 81 df a3 32 7c 86 ad
     5f 24 da 28 a7 1d 07 f5 fd b7 61 31 a1 fb 04 d5
     b2 31 c7 7f ca 26 d3 a6 42 99 9e 3b 10 74 b5 a7
     b3 54 2f 03 b0 39 63 2a 6b 44 56 36 fb 52 8d 58
```

A.1.2 Curve id-GostR3410-2001-CryptoPro-B-ParamSet

```
Point Q_1
  X= 0x0ad754474a915d9d706c6b8dc879858a1cb85cc8f6c148fc3120825393ecd394
  Y= 0x68c33b6d0343cf72cb19666ffd487fa94294dc677b28c8e27ec36068ff85ed83
  SEED:
     78 1d 7d 85 ff 04 12 b9 92 a7 6e 65 37 dd 83 81
     9b 81 2f fa bf e8 92 3c d0 12 fe dc 00 c4 96 69
     f6 52 44 4d 38 9a f2 b2 6f 57 b5 3e 2d 0f 81 2e
     db 2c f3 d6 a7 18 42 52 32 da 47 18 75 1e ec ef
     8f 2b c5 03 17 fb 49 6f 02 05 d7 99 bc 3c 34 87
     12 f5 1b d3 ef aa 7f f5 ba c2 52 07 80 46 34 77
Point Q_2
  X= 0x1cd96e72fdf1ce6b544dec12d0d7bcb9f6ba65bba3d9f7af732bcb133c1b6437
  Y= 0x34ab5b63c286a2b885ca443ac875a8f9ec0c2f148f1622bc64c83b80e6e3d31f
  SEED:
     62 93 40 15 63 0c 9a 09 ce 76 32 6c fd 1c 04 36
```

```
     ee 08 bc 92 b9 c0 3a d9 63 c6 db 00 18 12 12 fa
     e0 1a 46 38 8b b6 81 df ae 4f 64 3e cc 0c 93 8c
     e4 10 36 2f d9 6d 5c fd 99 f3 9c 13 fd 30 52 a1
     3e 8b 35 8b ed 1c 31 b0 39 9c 03 dc 5a 94 2b 41
     f8 ff 9d 62 41 bd eb 9d bf cf 54 b6 c8 cc d1 06
Point Q_3
  X= 0x18dda7154e5abef001dc9943554439cb44b9e26256def176849da5f09b5f690d
  Y= 0x3ef584be59673d1751b2fd6e3fdc619e3d756c0d355595b3a62196de048ece44
  SEED:
     33 17 39 c1 38 82 98 88 14 68 83 c6 97 14 86 8c
     d0 d2 1a 28 41 51 99 a9 33 40 15 0b 30 88 35 01
     4a 41 42 f8 d8 9a a6 bd e1 a6 81 23 94 19 e8 a0
     ef 3d 36 02 ef ff 38 e6 10 4b 11 2f 7b b5 50 42
     5a 7b 39 a6 00 53 1a 92 fc cc 2b 0d 95 dd ea 95
     42 d4 27 6f a8 0f ae 45 b2 d6 f4 38 c1 52 17 5d
```

A.1.3 Curve id-GostR3410-2001-CryptoPro-C-ParamSet

```
Point Q_1
  X= 0x339f791f62938871f241c1c89643619aa8b2c7d7706ce69be01fddff3f840003
  Y= 0x31d6d9264cc6f8fe09bf7aa48910b4ad5ddfd74a2ef4699b76de09ffed295f11
  SEED:
     0e 29 35 9d 45 dd a3 b4 57 9b 17 e8 87 d9 9e 63
     b9 d6 04 e3 ac 74 83 11 91 2a 5b d4 86 7b 5d 9c
     5d 07 70 64 cd f1 2d 93 f7 f0 2e f0 0a e1 7b 8b
     c1 87 50 b3 8f 39 bb 95 68 21 5c 42 e2 4e 8c fe
     59 e9 0f a6 05 0b 76 68 a2 94 da 5f 2c 9a 27 28
     1f 3a 7e 4e 14 54 10 21 01 6f 2c a2 97 77 94 12
Point Q_2
  X= 0x80f4d03b00b1b9b53f6bb4ffa52be65a6d316de846e27f44ccd795bc62d89e23
  Y= 0x38dd712518ddec19b46afccccba97338d89d1292427dc12985d4e848066cd1ab
  SEED:
     f5 61 4e 92 8f e5 5c 77 26 37 ab ac 1b 1e 3c dd
     2a 37 77 be 25 23 cc 58 9a 79 5a 60 28 db 9e 64
     f8 62 73 01 98 3e dd 23 0b eb 07 3e 81 9b cb d9
     94 bc bf 7f 9e 5f e1 8f a5 8a ce 9e f2 99 0e 9d
     fb ee 1c 64 38 22 33 c3 1b e7 05 9e c2 e9 bb 46
     b9 dc 15 19 9d e0 9f cb 65 d5 6d 46 2f 01 21 65
Point Q_3
  X= 0x0c8b64c3f0ec7ece81b6232db2e8054666d051ee28254d4b9a4bcb1460ca546b
  Y= 0x88c98b48b22b90d0d3a018da55ca0d05cedd82b6c838bd62aba2b823ce82b28f
  SEED:
     8a 1b 29 62 38 f5 c2 e2 9b 4a c0 5b 6d 57 99 88
     86 69 a4 1b b9 f6 60 f3 a3 15 26 e5 f4 33 1e ae
     80 9a 38 52 f5 44 86 91 71 76 1c ab 77 0a b6 2e
     c3 6f d6 4d 3c 31 a3 67 2a 82 25 bf d6 ae c9 95
     66 95 b8 87 39 6a 3e bf ef 28 65 16 b9 51 29 1d
     65 df 12 7a eb 4c ec f1 6f 08 f5 98 36 0a b9 a0
```

A.1.4 Curve id-tc26-gost-3410-2012-512-paramSetA

```
Point Q_1
  X= 0x301aac1a3b3e9c8a65bc095b541ce1d23728b93818e8b61f963e5d5b13eec0fe
        e6b06f8cd481a07bb647b649232e5179b019eef7296a3d9cfa2b66ee8bf0cbf2
  Y= 0x191177dd41ce19cc849c3938abf3adaab366e5eb2d22a972b2dcc69283523e89
        c9907f1d89ab9d96f473f96815da6e0a47297fcdd8b3adac37d4886f7ad055e0
  SEED:
     64 1c 90 19 c5 d7 68 91 de d1 9a 31 28 4e 7c d3
     c6 8b 74 e5 e6 a7 20 b5 2c fb 45 17 9f 91 b3 f6
     3a 0c b2 5e 3f 91 e3 eb 80 3d 80 4f 79 98 a3 57
     f2 e5 dc 5d 84 ab d6 7d 33 a3 2b 89 66 db c6 94
     96 8f 96 2d 37 9e 33 c0 fd 14 32 dd 02 70 fb 61
     1a 88 4c 6d ae 1b 58 20 24 6e 80 80 5d cd a8 66
Point Q_2
  X= 0x7edc38f17f88e3105bafb67c419d58fe6a9094dd4dc1a83bcaccc61f020ac447
        92eba888457c658ee2d82557b7c6ab6efd61ba0c3327741d09a561a8b860a085
  Y= 0x3af1400a7a469058d9ba75e65ea5d3f4d0bdb357fa57eb73fa4900e2dca4da78
        b8e5ff35ca70e522610bb1fc76b102c81cc4729f94b12822584f6b6229a57ea1
  SEED:
     3d ad a1 b4 fb 87 3e 13 1e 51 62 60 1f ee f1 54
     b0 77 e0 71 1b cf da 74 a2 20 7e a3 20 01 c3 f5
     79 00 5f 10 9f c1 83 83 4e 29 46 b3 29 8a 4c 10
     0c 69 f4 c6 40 92 3f ed af b2 68 08 0b 6b 1c 07
     48 a1 18 29 6e 64 9b f6 1d eb 26 27 b4 77 9e e8
     e0 ff c1 db 48 5d 8b c1 10 8c 58 b1 af 07 5f 7b
Point Q_3
  X= 0x387acfba7bbc5815407474a7c1132a1bded12497243d73ef8133d9810eb21716
        95dde2ff15597e159464a1db207b4d1ff98fbb989f80c2db13bc8ff5fea16d59
  Y= 0x4c816d1ca3e145ac448478fb79a77e1ad2dfc69576685e2f6867ec93fbad8aa4
        4111acd104036317095bce467e98f295436199c8ead57f243860d1bde8d88b68
  SEED:
     7c 8d a6 91 96 0d 9d 06 65 92 23 08 df cf 51 71
     bd 7c 4b f8 50 1b 3f fd 3c b1 58 3a 30 e1 a7 17
     4e 09 e2 5f 1d 19 35 6e b0 51 66 1c d0 2a c1 9e
     48 22 38 49 76 0d 43 4e 20 ea d1 80 73 84 1c e8
     36 a6 8e f3 24 bb 2d 57 45 32 a5 d4 e6 08 73 fa
     d3 8c 32 e8 af a1 c5 25 8c ff 3d 52 ca ac 98 d1
```

A.1.5 Curve id-tc26-gost-3410-2012-512-paramSetB

```
Point Q_1
  X= 0x488cf12b403e539fde9ee32fc36b6ed52aad9ec34ff478c259159a85e99d3dda
        dfd5d73606ecee351e0f780a14c3e9f14e985d9d7ddec93b064fc89b0c843650
  Y= 0x7bc73c032edc5f2c74dd7d9da12e1856a061ce344a77253f620592752b1f3a3d
        cbbc87eb27ec4ed5e236dfeb03f3972404747e277671e53a9e412e82aaf6c3f7
  SEED:
     40 57 8b 1a c0 bd 53 8d 75 97 2d 49 9b 1c c6 73
     94 c8 f7 d4 76 cd fe 15 59 02 fa 0f 28 b8 06 e1
     81 4c fb d0 4d 62 86 0c 4a ce c1 0e 88 13 da 2a
```

```
     d4 fd 7a 13 4d ba 75 0d 2c 80 f2 68 ba c1 b4 34
     98 ea fe 10 51 86 60 b7 70 30 f8 64 6f 21 d9 40
     aa da 62 3e ad 44 3f 93 73 a5 6b c3 15 55 3c bd
Point Q_2
  X= 0x175166b97248bda12ec035df2e312a2771d0b16977c9cbc79461ff05e01f719c
        92ae8b53f3b7e3edcacffcc5063b5e9c8de18d0cb87da358350992132173df69
  Y= 0x10e2943dc1a18a841ab76ac756fa974948d5a18d071d458a4769c2494fe2a6c5
        966e3c8931e624d87259156aea9317157502698e4a4a489c327b89277cf59b4c
  SEED:
     26 01 07 1b 3d 3e 6d e7 0e d0 22 ae be 81 be 47
     51 77 49 b6 5d 29 d1 07 5c df cb f4 56 a8 77 54
     2b e9 91 50 34 06 b3 aa 71 c5 ce 16 b6 5f e9 93
     e7 48 99 58 b1 26 81 10 9f 9b e4 30 38 73 77 13
     f0 6a 4f 30 05 b2 66 76 9f b8 1b 5f 39 55 52 97
     ab 46 6b 5d 2e 19 2d 12 f3 2a b3 18 72 71 52 62

Point Q_3
  X= 0x01f4583db894cdebd7c591af848783ee011a20567751ca1561f398a6118ace08
        a4efe1501bda67f39d060270ba660526dc53063c6b40fa5548c9a9e7688f2239
  Y= 0x7bc640641d70c8296bd9257c9eebb5b1bd3196a169bac04f7579bf27b5847d4e
        7b4f63748ad81b5469070ed35ad93e5a5258652306f84094eae04a91954536ee
  SEED:
     bb 9a 63 a5 67 7d 40 7c f3 4d 06 df 96 7d d9 e9
     ca 4d 42 eb d6 7d a5 69 a4 9b d8 b1 04 64 2e 20
     fb a9 9d 84 2f cb 54 76 61 dc 7a a4 de 72 6f 67
     4a 09 85 46 20 04 7c c1 75 2c ab 67 99 8b 5c 8e
     6a 88 6d 0a 06 e6 a3 fa e8 19 34 21 1a ec 81 8d
     89 03 9e 45 dc a1 85 03 7e c3 49 37 33 ee 3c 2e
```

A.1.6 Curve id-tc26-gost-3410-2012-256-paramSetA

```
Point Q_1
  X= 0x5161b08a973d521bdde0cbd45b68aa0470e1058dd936e5bd618fd3373770eed9
  Y= 0xc1633db551677c62b9c2b69d47e503c0f8ca83b6b3109dece0a5f985d77a83a7
  U= 0x9c5ad63ddc3314ac009d879780d6219720bf4573f4fe6b4bf7a0a88860677f9d
  V= 0x8ee071a767f3d6f0435eb6100d1a936f984e43d9af0bc91c864a9e65cee025fb
  SEED:
    c4 7e 5e 42 31 4e dd 8e e9 ac 39 fb c8 da ea c8
    e6 5b fd 26 58 27 4e 1f 99 e9 33 e1 1e 5d f2 62
    4a e3 97 f1 7e db f9 83 60 f3 ec 2e 8f 6f 2e ff
    d4 aa 80 5c 71 d6 ed 5b a1 5b c9 d0 ad d6 38 23
    84 c1 55 20 a5 b8 bb bd 7b 23 1f c8 fb 8c 77 71
    57 b9 77 25 91 55 3f 17 46 8c 4b b3 64 6f 9f 53
Point Q_2
  X= 0xd47abd59dccad35849dec9dc721ffa1e44419ca8686406a9f441e61294b210ed
  Y= 0xa78b64220bf3375d08de0ea5e2920cfd8f204da6757bf1878ac870fb7e5ca0e8
  U= 0xf0195efb6b249eb8018c19376907c787511bf30516a5c27d045fc7ba2af58ed0
  V= 0xacae88466127df000b663863bc7bd394eafa6996fcedad11d7834f502a6a2686
  SEED:
```

```
    30 5f d0 bb ce c7 16 49 ac e4 1b 4d ca 07 6c a6
    96 a8 8c c6 fd 06 91 a8 79 13 5d e1 90 96 e3 c8
    03 c5 b4 ad 41 68 36 9b e7 b9 ed 81 d6 e2 bd 0c
    a2 8e b0 e9 6f 74 2d 50 e2 df b6 a1 86 ae 15 60
    96 a3 5a 97 a2 20 fa 6d 0f cf 88 db 6d 86 cd db
    19 7c 3a 21 5f 10 cc ea 42 95 ef aa b2 63 95 d5
Point Q_3
  X= 0xe0d610ff42ce21eb308980964ca368963fbe5cb08c277187d22d0c94f4bf0762
  Y= 0x82619b88da25b666e07b617ff487be8afd5af8b092568b493ecef44ee0c04b5f
  U= 0x723df0719311d095b814ee05ca086e18c410c375a48789dc03c3fe844ed3b7c9
  V= 0x160e1b5338337ee0620745206dbe5556a7ff5d19735418a3cc03bf7f2735ce25
  SEED:
    22 16 91 c7 21 8d 93 d4 a8 ad 15 e4 72 33 84 90
    ca 5c 5e 3b 84 84 57 4e bc df 83 26 68 84 5e f4
    09 53 71 79 7a f8 e2 a6 e3 99 93 de 2a 7c 65 f0
    37 26 2e cc fa 95 58 9a c6 e8 b1 2d e6 09 af be
    f9 2f 12 d0 a3 08 56 9a b3 c0 fa d8 ec 5d 7b 9c
    f4 27 1f aa 54 bc bb da 31 61 b7 cd f5 40 d6 b8
```

A.1.7 Curve id-tc26-gost-3410-2012-512-paramSetC

```
Point Q_1
  X= 0x5b065ead2e94de0ee2e462de204c93c6b2bf3498ad920393cb60259e1a8ffc7c
        7e7d4defa20ff4282abf70207e4611d532f40db6800e29d2b53f6ac0713e5b38
  Y= 0xa39a28c59ff7f796b85223b8834384907c626086415487288ed1182ca4487dc1
        ae5f37af90fd267b7c0dc8542ea52cd984af54731bc84271d6186d973c91359b
  U= 0x3c80e89805380f52cfe86ff990501801d70e5b4636e8478674d2d5706a56a666
        63eb03abdc332584f7ea8c3255b1be3ca75e4685a060e0ea88e569612d9e7227
  V= 0xd8f2cf17c484f4bb6a0208b3796a2609971c55d56bffadf155c0bfb76f7afe99
        7d6b6e8fde9e2cefd0ab3e31a1862953425a70334e4e2404c9cd9079856c7259
  SEED:
    03 8a 01 b5 ea a2 28 3b bb 29 7b 81 ad 94 92 01
    e4 32 11 df 76 a3 70 ec c0 09 ec 49 1f 9f 8d 33
    f2 ee 24 08 c7 88 27 cd 0c 51 17 a7 e3 8d 58 5b
    3d 15 50 30 a3 29 1b ad 6a 21 ab 48 38 1d 66 bc
    1b d6 b9 ba 6e d8 6a 21 65 c5 99 84 dc 5d 51 81
    f3 f1 97 fe 4a 86 81 c2 e5 0a 22 a0 61 2c 55 7e
Point Q_2
  X= 0xb3e6c475f173af4494dd02ad7c9df3bd6a5ca82c3d65ad86fbb330dfb1c40e34
        c4cd04d93f609cff2daea5907d0e08192a29be3ff27522223b868e8bcc6a7b74
  Y= 0x53ffcf818281bcf383d9b6542b3b1fcee5bd20cd1c805ed1dacb83ba161167a5
        eb96df52c1d290496043ea514c465ecb37970fcd7ffbb6ca35a767cd0227fe8c
  U= 0x8dd3f6f455ffaea85c3935750792b65fa1ba990c7ac8bc449a77bb86aeb87eb6
        ecb6bf387924885b0ea1e30fc4d742919504cd7baf4926b777ed40b898be41f8
  V= 0x2d7edc1ca6078878d2d8ecafabc2abc83fcb269c049baa11951ff2523b69b1d1
        4ee6c0fa7cbd5a566cb32246d14568eb9fa04e3b53ee6175bb32887796870ba9
  SEED:
    63 ce db 44 3d f8 df 68 8d 5d fd d0 ea 54 41 62
    f5 6d 78 92 73 0c 86 88 53 e2 24 4d dd 87 1e 4a
```

```
    0e 3f b7 32 40 c8 7a bc e8 fd b3 16 dd 0e 9b 23
    ec 1f c7 40 86 29 8c fc f2 a1 d9 18 31 af a3 cf
    1e 98 b8 0d 42 0c f6 73 8d 57 44 77 8e 1a d3 e4
    42 d7 26 39 6c 91 b7 f5 e8 84 09 8d be 02 aa 80
Point Q_3
  X= 0xbe963ad90f84ff9ff6ff7ddd39d91cea649e849bf20b8cc1e72040cf689a974f
        40f24e10c737bfa558b514c605b7c156e24251b859202b12ef311b0f363171eb
  Y= 0x007cfa56f5ae239694e74f7996e1f44fcd4f62205a555fdb627e4212576b4591
        7f88667bcd924a3271f40dc4bbd2f2e216b4fcf59c25fdd8154241d40f42e2ad
  U= 0xff6697883ce5c6cc165fa78ff158c03b31add23dc01b24902b18c5487f1835ff
        eaf4af5ede44f8b254748704e504810597d0e4418daf50e0253f33915de97b7b
  V= 0x54e1ba656234479c5845c06af70bbefa741a863d5186ca720ee2f43bdd4d5b74
        71594871d532ce263928aa30ae3c25efc6a2f82d4163c45869339426888be3bc
  SEED:
    06 ce 08 fa 5d 11 50 45 e2 d2 a8 03 01 d2 9e 2a
    39 c3 ea e7 f1 61 37 f9 3e 51 d1 46 f3 21 b1 89
    fb 5c 17 70 26 5b 30 8b 6d f2 87 7c d9 d3 6f 8a
    3c b4 e7 1d f0 99 a4 73 69 1d d5 46 8d 43 50 f9
    87 df e5 e5 de ff 3c 7b b8 f4 62 ed 19 9b f3 33
    7a 6f f9 0e c5 f0 bc bb 1a 59 1e cd c2 9b 52 64
```

A.2. Test examples

This protocol implementation uses the GOST R 34.11-2012 hash function (see [RFC6986]) with 256-bit output as the H function and the HMAC_GOSTR3411_2012_512 function defined in [RFC7836] as a PRF function for the F function. The parameter len is considered equal to 256, if 2^254 < q < 2^256, and equal to 512, if 2^508 < q < 2^512.

The test examples for one of the three points of each curve in Appendix A.1 of this document are given below.

A.2.1 Curveid-GostR3410-2001-CryptoPro-A-ParamSet

The input protocol parameters in this example take the following values:

```
N= 3
ind= 1
ID_A:
  00 00 00 00
ID_B:
  00 00 00 00
PW:
  31 32 33 34 35 36 ('123456')
salt:
  29 23 be 84 e1 6c d6 ae 52 90 49 f1 f1 bb e9 eb
Q_ind:
  X= 0xa33ce065b0c23e1d3d026a206f8a1f8747ed1cd92a665bf85198cdb10ac90a5c
  Y= 0xb00d0dc0733883f05de9f55fd711f55998f5508cc40bead80c913b4d5b533667
```

The function F (PW, salt, 2000) takes the following values:

```
F(PW,salt,2000):
  bd 04 67 3f 71 49 b1 8e 98 15 5b d1 e2 72 4e 71
  d0 09 9a a2 51 74 f7 92 d3 32 6c 6f 18 12 70 67
```

The coordinates of the point Q_PW are:

```
X= 0x9d339b3396ae4a816388a14c79ab3a8dd495fa4c53f0d4076579022ef2aaeb68
Y= 0xdad91482e208590fd316bf959480f5ec2c17463ec8fc8f63030649b452cddda8
```

During the calculation of the message u_1 on the subject A the parameter alpha, the point alpha*P and the message u_1 take the following values:

```
alpha=0xfccbd45d1f2538097d5a031fa68bbb43c84d12b3de47b7061c0d5e24993e0c87
alphaP:
  X= 0x24538e096781b9d53316c342ae5bbd49ccfb2db627c3659175bc4fa9d95b4618
  Y= 0xf6e39a7490ae0ac449f5abe7e2135697c582daf3a038c40a05e6e8be3e466a2b
u_1:
  X= 0xcf73b30dd577369fb98e2a93d6d98d7450f9ceef2bada1e3dcb8bb1016dff1e1
  Y= 0x1cf05014caedbdb1635120b30e0a445060b8f1cca52965cf83c4838d554ca4e2
```

During processing a message u_1, calculation the K_B key and the message u_2 on the subject B the parameters betta, src, K_B = HASH(src), betta*P and u_2 take the following values:

```
betta=0xf2144faddc497d9ef6324912fd367840ee509a2032aedb1c0a890d133b45f596
src:
  39 c0 e8 83 59 91 cd 6d 56 88 fc ad 55 29 1f 79
  e5 0f 87 9d 94 b5 0a b2 db d6 bd f7 e8 39 b7 1a
  10 b5 a7 8d c0 36 b8 73 f7 e4 b1 6b 12 48 6f eb
  69 d7 39 d4 01 4d ae e2 cc 5c 2f c7 4a 2c c8 06
K_B:
  e0 4e e0 14 7f 9f 19 8d e2 5a af 33 a2 84 99 e0
  ce 7d 31 6e 47 39 76 2f d5 19 f8 e9 91 d7 fc 00
betta*P:
  X= 0xb11f1a8fb043bc6d4068667b897e4ff637b8410f5eb19e11b0a7028f34d6936a
  Y= 0x266d952955e2ab3f3ba75d14a919795d6b8ac04dbcff1cfaac6ba32291c099fd
u_2:
  X= 0x6e1bfb24b6131a3ad0b60e477a38715c6f96f21bb0b2f9ebd67680e804a77199
  Y= 0x873ee3c546c41e8f707298f11b955fe64f7577d52d7dadc1beccb9925178ca80
```

During processing a message u_2 and calculation the key on the subject A the K_A key takes the following value:

```
K_A:
  e0 4e e0 14 7f 9f 19 8d e2 5a af 33 a2 84 99 e0
  ce 7d 31 6e 47 39 76 2f d5 19 f8 e9 91 d7 fc 00
```

The message MAC_A=HMAC (K_A, 0x01 || ID_A || ind || salt || u_1 || u_2) from the subject A takes the following value:

```
MAC_A:
  bd 35 c5 0e 90 60 e6 4f 04 2e 7b e6 cc 02 99 84
  0c 8e 27 82 b8 e5 9c 3d d4 47 50 11 16 73 c5 ea
```

The message MAC_B=HMAC (K_B, 0x02 || ID_B || ind || salt || u_1 || u_2) from the subject B takes the following value:

```
MAC_B:
  c8 c8 2c 0f ed 8e 4d 1e 41 42 d7 a9 f0 55 b4 5f
```

```
  f6 71 2d 2f 41 bf 26 ef 2f bc 37 c5 56 4b 86 d3
```

A.2.2 Curveid-GostR3410-2001-CryptoPro-B-ParamSet

The input protocol parameters in this example take the following values:

```
N= 3
ind= 1
ID_A:
  00 00 00 00
ID_B:
  00 00 00 00
PW:
  31 32 33 34 35 36 ('123456')
salt:
  29 23 be 84 e1 6c d6 ae 52 90 49 f1 f1 bb e9 eb
Q_ind:
  X= 0x0ad754474a915d9d706c6b8dc879858a1cb85cc8f6c148fc3120825393ecd394
  Y= 0x68c33b6d0343cf72cb19666ffd487fa94294dc677b28c8e27ec36068ff85ed83
```

The function F (PW, salt, 2000) takes the following values:

```
F(PW,salt,2000):
  bd 04 67 3f 71 49 b1 8e 98 15 5b d1 e2 72 4e 71
  d0 09 9a a2 51 74 f7 92 d3 32 6c 6f 18 12 70 67
```

The coordinates of the point Q_PW are:

```
X= 0x7a7211a430fd4e31b815e6d2454eea9574f034c5c442dce1723d69555d3ee4c9
Y= 0x2995e857187808e80d3e40a00fb87128e203f2d91c1f15d8193a5aad95964734
```

During the calculation of the message u_1 on the subject A the parameter alpha, the point alpha*P and the message u_1 take the following values:

```
alpha=0x499d72b90299cab0da1f8be19d9122f622a13b32b730c46bd0664044f2144fad
alphaP:
  X= 0x61d6f916db717222d74877f179f7ebef7cd4d24d8c1f523c048e34a1df30f8dd
  Y= 0x3ec48863049cfcfe662904082e78503f4973a4e105e2f1b18c69a5e7fb209000
u_1:
  X= 0x35e78fcbc24998eb3039445a9de7032aadf291e7768196ef618e45bed80edf88
  Y= 0x1970a4697295f6d361d2c3edd3885794c1254bac3f4adb4a3346ad01a911d13c
```

During processing a message u_1, calculation the K_B key and the message u_2 on the subject B the parameters betta, src, K_B = HASH(src), betta*P and u_2 take the following values:

```
betta=0x0f69ff614957ef83668edc2d7ed614be76f7b253db23c5cc9c52bf7df8f4669d
src:
  50 14 0a 5d ed 33 43 ef c8 25 7b 79 e6 46 d9 f0
  df 43 82 8c 04 91 9b d4 60 c9 7a d1 4b a3 a8 6b
  00 c4 06 b5 74 4d 8e b1 49 dc 8e 7f c8 40 64 d8
  53 20 25 3e 57 a9 b6 b1 3d 0d 38 fe a8 ee 5e 0a
K_B:
  a6 26 de 01 b1 68 0f f7 51 30 09 12 2b ce e1 89
  68 83 39 4f 96 03 01 72 45 5c 9a e0 60 cc e4 4a
betta*P:
  X= 0x33bc6f7e9c0ba10cfb2b72546c327171295508ea97f8c8ba9f890f2478ab4d6c
  Y= 0x75d57b396c396f492f057e9222ccc686437a2aad464e452ef426fc8eeed1a4a6
```

```
u_2:
  X= 0x20d7a92b238143e3f137be904d52fa35c45a29f02a7226a7ac83a1172c2a55cd
  Y= 0x5fc4cd6ffb0e76ea8603ce9e6dab5164285617969ab3bfab09fbeb8595d1f47b
```

During processing a message u_2 and calculation the key on the subject A the K_A key takes the following value:

```
K_A:
  a6 26 de 01 b1 68 0f f7 51 30 09 12 2b ce e1 89
  68 83 39 4f 96 03 01 72 45 5c 9a e0 60 cc e4 4a
```

The message MAC_A=HMAC (K_A, 0x01 || ID_A || ind || salt || u_1 || u_2) from the subject A takes the following value:

```
MAC_A:
  55 7a 59 61 42 60 39 a1 52 c8 23 a7 65 04 59 b0
  62 be 3d 47 56 53 03 09 95 57 1c e7 53 40 26 47
```

The message MAC_B=HMAC (K_B, 0x02 || ID_B || ind || salt || u_1 || u_2) from the subject B takes the following value:

```
MAC_B:
  3b c5 5e 27 07 84 19 94 c4 b9 ca ba 43 e6 ce 6a
  09 2d e9 08 83 76 5f b6 c3 44 c6 1d 76 02 96 e9
```

A.2.3 Curveid-GostR3410-2001-CryptoPro-C-ParamSet

The input protocol parameters in this example take the following values:

```
N= 3
ind= 1
ID_A:
  00 00 00 00
ID_B:
  00 00 00 00
PW:
  31 32 33 34 35 36 ('123456')
salt:
  29 23 be 84 e1 6c d6 ae 52 90 49 f1 f1 bb e9 eb
Q_ind:
  X= 0x339f791f62938871f241c1c89643619aa8b2c7d7706ce69be01fddff3f840003
  Y= 0x31d6d9264cc6f8fe09bf7aa48910b4ad5ddfd74a2ef4699b76de09ffed295f11
```

The function F (PW, salt, 2000) takes the following values:

```
F(PW,salt,2000):
  bd 04 67 3f 71 49 b1 8e 98 15 5b d1 e2 72 4e 71
  d0 09 9a a2 51 74 f7 92 d3 32 6c 6f 18 12 70 67
```

The coordinates of the point Q_PW are:

```
X= 0x8b666917d42c455331358c50c3c12c85b898a2e454b50dd773541da02e1c3068
Y= 0x8a9b6c4703934b7f0dc903f52c16275e1d38b568117c7cff3bd322a99a311fe9
```

During the calculation of the message u_1 on the subject A the parameter alpha, the point alpha*P and the message u_1 take the following values:

```
alpha=0x3a54ac3f19ad9d0b1eac8acdcea70e581f1dac33d13feafd81e762378639c1a8
alphaP:
  X= 0x96b7f09c94d297c257a7da48364c0076e59e48d221cba604ae111ca3933b446a
  Y= 0x54e4953d86b77ecceb578500931e822300f7e091f79592ca202a020d762c34a6
u_1:
```

```
  X= 0x2124a22e00b1be2114f5ca42d58d55a0a9f2b08f8cb10275eddf8243402abb7a
  Y= 0x62497815861d15877b7ad2e86768a2deb0f755a8b1a8897fc5235da783914a59
```

During processing a message u_1, calculation the K_B key and the message u_2 on the subject B the parameters betta, src, K_B = HASH(src), betta*P and u_2 take the following values:

```
betta=0x448781782bf7c0e52a1dd9e6758fd3482d90d3cfccf42232cf357e59a4d49fd4
src:
  16 a1 2d 88 54 7e 1c 90 06 ba a0 08 e8 cb ec c9
  d1 68 91 ed c8 36 cf b7 5f 8e b9 56 fa 76 11 94
  d2 8e 25 da d3 81 8d 16 3c 49 4b 05 9a 8c 70 a5
  a1 b8 8a 7f 80 a2 ee 35 49 30 18 46 54 2c 47 0b
K_B:
  be 7e 7e 47 b4 11 16 f2 c7 7e 3b 8f ce 40 30 72
  ca 82 45 0d 65 de fc 71 a9 56 49 e4 de ea ec ee
betta*P:
  X= 0x4b9c0ab55a938121f282f48a2cc4396eb16e7e0068b495b0c1dd4667786a3eb7
  Y= 0x223460aa8e09383e9df9844c5a0f2766484738e5b30128a171b69a77d9509b96
u_2:
  X= 0x47ad0110d1620fe38832e90b58971d2e0b9183dd52de23422b6fc47bec64541a
  Y= 0x8296af496b3c52640e738a195d63ab7bfb457aba7c71b5649cc3e300829cbf0a
```

During processing a message u_2 and calculation the key on the subject A the K_A key takes the following value:

```
K_A:
  be 7e 7e 47 b4 11 16 f2 c7 7e 3b 8f ce 40 30 72
  ca 82 45 0d 65 de fc 71 a9 56 49 e4 de ea ec ee
```

The message MAC_A=HMAC (K_A, 0x01 || ID_A || ind || salt || u_1 || u_2) from the subject A takes the following value:

```
MAC_A:
  47 58 fa 64 9f 2e 31 3b f2 70 8b 76 a7 f7 a7 5a
  37 ce 9e 7f 55 c3 fc 5a 55 77 e8 77 a7 a2 c1 ea
```

The message MAC_B=HMAC (K_B, 0x02 || ID_B || ind || salt || u_1 || u_2) from the subject B takes the following value:

```
MAC_B:
  2f 33 b9 bf f0 7d cd e3 44 67 bd b0 7f 62 fc a8
  b3 52 3a 64 39 ef f1 c9 93 ba 0b 4c e6 c2 ed e4
```

A.2.4 Curveid-tc26-gost-3410-2012-512-paramSetA

The input protocol parameters in this example take the following values:

```
N= 3
ind= 1
ID_A:
  00 00 00 00
ID_B:
  00 00 00 00
PW:
  31 32 33 34 35 36 ('123456')
salt:
  29 23 be 84 e1 6c d6 ae 52 90 49 f1 f1 bb e9 eb
```

```
Q_ind:
  X= 0x301aac1a3b3e9c8a65bc095b541ce1d23728b93818e8b61f963e5d5b13eec0fe
       e6b06f8cd481a07bb647b649232e5179b019eef7296a3d9cfa2b66ee8bf0cbf2
  Y= 0x191177dd41ce19cc849c3938abf3adaab366e5eb2d22a972b2dcc69283523e89
       c9907f1d89ab9d96f473f96815da6e0a47297fcdd8b3adac37d4886f7ad055e0
The function F (PW, salt, 2000) takes the following values:
F(PW,salt,2000):
  bd 04 67 3f 71 49 b1 8e 98 15 5b d1 e2 72 4e 71
  d0 09 9a a2 51 74 f7 92 d3 32 6c 6f 18 12 70 67
  1c 62 13 e3 93 0e fd da 26 45 17 92 c6 20 81 22
  ee 60 d2 00 52 0d 69 5d fd 9f 5f 0f d5 ab a7 02
The coordinates of the point Q_PW are:
X= 0xa8b54a6339b296f5c5227670fb1482010b4b07e3642974b40c58a5f1da33370e
     fed546eb17c6a707f3fc69671deba10a6de03a55f859473e9074a89b4a7b5488
Y= 0xfebf437ecf21536328b32f4c8e0430d5c0c096001c08a378ac30b8634412f44c
     5ba9b7096642f51cc3a018cd1599c849cd62917a370eca3bbc6bed5eedabdd77
During the calculation of the message u_1 on the subject A the parameter
alpha, the point alpha*P and the message u_1 take the following values:
alpha=0x3ce54325db52fe798824aead11bb16fa766857d04a4af7d468672f16d90e7396
     046a46f815693e85b1ce5464da9270181f82333b0715057bbe8d61d400505f0e
alphaP:
  X= 0xb93093eb0fcc463239b7df276e09e592fcfc9b635504ea4531655d76a0a3078e
       2b4e51cfe2fa400cc5de9fbe369db204b3e8ed7edd85ee5cca654c1aed70e396
  Y= 0x809770b8d910ea30bd2fa89736e91dc31815d2d9b31128077eedc371e9f69466
       f497dc64dd5b1fadc587f860ee256109138c4a9cd96b628e65a8f590520fc882
u_1:
  X= 0xe8732d5471901b3eb9a31aaebeac7a6155c2c8fc1c960cb475e14074987dd2c8
       4eccafac0835735a5c2df3d1c8dacf4a1d2e38e1e4419f5df4e25b7f8dd90b50
  Y= 0xd680a41eaec979d49f4752008e9e92eb0efc1950d74b85e852be47f3958d5500
       0442d859e5b459de5dc7acaa0c36383cd1f98f271333c6083dcecaf07ac825b8
During processing a message u_1, calculation the K_B key and the message
u_2 on the subject B the parameters betta, src, K_B = HASH(src), betta*P
and u_2 take the following values:
betta=0xb5c286a79aa8e97ec0e19bc1959a1d15f12f8c97870ba9d68cc12811a56a3bb1
     1440610825796a49d468cdc9c2d02d76598a27973d5960c5f50bce28d8d345f4
src:
  84 59 c2 0c b5 c5 32 41 6d b9 28 eb 50 c0 52 0f
  b2 1b 9c d3 9a 4e 76 06 b2 21 be 15 ca 1d 02 da
  08 15 de c4 49 79 c0 8c 7d 23 07 af 24 7d da 1f
  89 ec 81 20 69 f5 d9 cd e3 06 af f0 bc 3f d2 6e
  d2 01 b9 53 52 a2 56 06 b6 43 e8 88 30 2e fc 8d
  3e 95 1e 3e b4 68 4a db 5c 05 7b 8f 8c 89 b6 cc
  0d ee d1 00 06 5b 51 8a 1c 71 7f 76 82 ff 61 2b
  bc 79 8e c7 b2 49 0f b7 00 3f 94 33 87 37 1c 1d
K_B:
  53 24 de f8 48 b6 63 cc 26 42 2f 5e 45 ee c3 4c
  51 d2 43 61 b1 65 60 ca 58 a3 d3 28 45 86 cb 7a
betta*P:
```

```
  X= 0x238b38644e440452a99fa6b93d9fd7da0cb83c32d3c1e3cfe5df5c3eb0f9db91
       e588daedc849ea2fb867ae855a21b4077353c0794716a6480995113d8c20c7af
  Y= 0xb2273d5734c1897f8d15a7008b862938c8c74ca7e877423d95243eb7ebd02fd2
       c456cf9fc956f078a59aa86f19dd1075e5167e4ed35208718ea93161c530ed14
u_2:
  X= 0x1830804bf1fb07ebd43f27d03ff71ad9c7c31becaf1d3585dfb9e356c36638dc
       d82aba559dec06d46c862566653dfe0b116eb1a68439b0283f4d79ce48408eee
  Y= 0x23b33ae97fba92e06095c41525aedf7b5d96fe9ca8e0244ed6c8a565d542d05e
       d3044cafb1a8ac9a570c5133ba846d61da77f54da2daf13b0def7d90a0796f06
```

During processing a message u_2 and calculation the key on the subject A the K_A key takes the following value:

```
K_A:
  53 24 de f8 48 b6 63 cc 26 42 2f 5e 45 ee c3 4c
  51 d2 43 61 b1 65 60 ca 58 a3 d3 28 45 86 cb 7a
```

The message MAC_A=HMAC (K_A, 0x01 || ID_A || ind || salt || u_1 || u_2) from the subject A takes the following value:

```
MAC_A:
  37 e6 1a 43 2d 85 75 9b 30 13 a2 9d d6 82 f1 4d
  33 ca 86 89 37 db 4b f2 02 91 ed cf 6b e2 4b 4e
```

The message MAC_B=HMAC (K_B, 0x02 || ID_B || ind || salt || u_1 || u_2) from the subject B takes the following value:

```
MAC_B:
  72 dc de 19 5f 26 4b b8 a8 1d 2a fe 2f d9 da 2d
  60 12 81 9c 15 f7 11 db 2b c4 c5 74 85 9e 05 3e
```

A.2.5 Curveid-tc26-gost-3410-2012-512-paramSetB

The input protocol parameters in this example take the following values:

```
N= 3
ind= 1
ID_A:
  00 00 00 00
ID_B:
  00 00 00 00
PW:
  31 32 33 34 35 36 ('123456')
salt:
  29 23 be 84 e1 6c d6 ae 52 90 49 f1 f1 bb e9 eb
Q_ind:
  X= 0x488cf12b403e539fde9ee32fc36b6ed52aad9ec34ff478c259159a85e99d3dda
       dfd5d73606ecee351e0f780a14c3e9f14e985d9d7ddec93b064fc89b0c843650
  Y= 0x7bc73c032edc5f2c74dd7d9da12e1856a061ce344a77253f620592752b1f3a3d
       cbbc87eb27ec4ed5e236dfeb03f3972404747e277671e53a9e412e82aaf6c3f7
```

The function F (PW, salt, 2000) takes the following values:

```
F(PW,salt,2000):
  bd 04 67 3f 71 49 b1 8e 98 15 5b d1 e2 72 4e 71
  d0 09 9a a2 51 74 f7 92 d3 32 6c 6f 18 12 70 67
  1c 62 13 e3 93 0e fd da 26 45 17 92 c6 20 81 22
  ee 60 d2 00 52 0d 69 5d fd 9f 5f 0f d5 ab a7 02
```

The coordinates of the point Q_PW are:

```
X= 0x2383039092052ed0e8ca3f751c11ebb891b8f32f7c66a437dec86345c63efc4b
     a1ecd04dfc11826dd581cbc1d744754e284c00b04eef9cd6eff22c12432c46fd
Y= 0x374202580afbaf2f68da8a5c03ab82e71eb4c1f1fdd881aa2911d0206d470039
     275d298d5477901565ab826ec4492f67eebcf3194442f272fd2cad9a5f04234f
```

During the calculation of the message u_1 on the subject A the parameter alpha, the point alpha*P and the message u_1 take the following values:

```
alpha=0x715e893fa639bf341296e0623e6d29dadf26b163c278767a7982a989462a3863
     fe12aef8bd403d59c4dc4720570d4163db0805c7c10c4e818f9cb785b04b9997
alphaP:
  X= 0x10c479ea1c04d3c2c02b0576a9c42d96226ff033c1191436777f66916030d87d
       02fb93738ed7669d07619ffce7c1f3c4db5e5df49e2186d6fa1e2eb5767602b9
  Y= 0x039f6044191404e707f26d59d979136a831cce43e1c5f0600d1ddf8f39d0ca3d
       52fbd943bf04ddced1aa2ce8f5ebd7487acdef239c07d015084d796784f35436
u_1:
  X= 0x0ab9e56fc0d48e4982ee0a0b09507a63dc530181611d9f00d0464724415757b9
       de1c647178783a0fb4648dfd8e3da1efeb4db29de4711c8599191054ca7de6c4
  Y= 0x4decae941f8d19c44daae9eb132019e116478124e76430b8bee16ce6910a06c8
       a2fed68f4907e4ba17c4f4e3356dc3b3b8647165b9c1aae54b1c13239bfa8213
```

During processing a message u_1, calculation the K_B key and the message u_2 on the subject B the parameters betta, src, K_B = HASH(src), betta*P and u_2 take the following values:

```
betta=0x30fa8c2b4146c2dbbe82bed04d7378877e8c06753bd0a0ff71ebf2befe8da8f3
     dc0836468e2ce7c5c961281b6505140f8407413f03c2cb1d201ea1286ce30e6d
src:
  3f 04 02 e4 0a 9d 59 63 20 5b cd f4 fd 89 77 91
  9b ba f4 80 f8 e4 fb d1 25 5a ec e6 ed 57 26 4b
  d0 a2 87 98 4f 59 d1 02 04 b5 f4 5e 4d 77 f3 cf
  8a 63 b3 1b eb 2d f5 9f 8a f7 3c 20 9c ca 8b 50
  b4 18 d8 01 e4 90 ae 13 3f 04 f4 f3 f4 d8 fe 8e
  19 64 6a 1b af 44 d2 36 fc c2 1b 7f 4d 8f c6 a1
  e2 9d 6b 69 ac ce ed 4e 62 ab b2 0d ad 78 ac f4
  fe b0 ed 83 8e d9 1e 92 12 ab a3 89 71 4e 56 0c
K_B:
  d5 90 e0 5e f5 ae ce 8b 7c fb fc 71 be 45 5f 29
  a5 cc 66 6f 85 cd b1 7e 7c c7 16 c5 9f f1 70 e9
betta*P:
  X= 0x34c0149e7bb91ae377b02573fcc48af7bfb7b16deb8f9ce870f384688e3241a3
       a868588cc0ef4364cca67d17e3260cd82485c202adc76f895d5df673b1788e67
  Y= 0x608e944929bd643569ed5189db871453f13333a1eaf82b2fe1be8100e775f13d
       d9925bd317b63bfaf05024d4a738852332b64501195c1b2ef789e34f23ddafc5
u_2:
  X= 0x66defd2a42f0efe38ed3d4a4dfbed6b86d40f4adf156c86fee1605dbf6b057b1
       2fe82a0be4823f7f215b5110673e02e3bf44f0ae26630005fcfd9f01473127eb
  Y= 0x36168c6d20c9514556ab442bf63ded0115346916ef45af7e5517f59205d1cc52
       ae2e72c3036f13cab7de12932e4a3acd0789f5e2474ff722b81334676c8a3371
```

During processing a message u_2 and calculation the key on the subject A the K_A key takes the following value:

K_A:
  d5 90 e0 5e f5 ae ce 8b 7c fb fc 71 be 45 5f 29
  a5 cc 66 6f 85 cd b1 7e 7c c7 16 c5 9f f1 70 e9
The message MAC_A=HMAC (K_A, 0x01 || ID_A || ind || salt || u_1 || u_2) from the subject A takes the following value:
MAC_A:
  9e c1 a8 74 93 b2 87 c9 ca c3 da c2 a2 d7 1b 82
  8d c5 97 7c b0 03 93 42 c1 5a cd fb 66 c8 cf 89
The message MAC_B=HMAC (K_B, 0x02 || ID_B || ind || salt || u_1 || u_2) from the subject B takes the following value:
MAC_B:
  a9 b2 f1 9b d9 c1 fd 0f 0c ab fd 09 52 94 c6 e6
  3c d5 9f 12 cf 8e fd 01 12 46 0d b7 aa 20 bb 6e

A.2.6 Curveid-tc26-gost-3410-2012-256-paramSetA

The input protocol parameters in this example take the following values:
N= 3
ind= 1
ID_A:
  00 00 00 00
ID_B:
  00 00 00 00
PW:
  31 32 33 34 35 36 ('123456')
salt:
  29 23 be 84 e1 6c d6 ae 52 90 49 f1 f1 bb e9 eb
Q_ind:
  X= 0x5161b08a973d521bdde0cbd45b68aa0470e1058dd936e5bd618fd3373770eed9
  Y= 0xc1633db551677c62b9c2b69d47e503c0f8ca83b6b3109dece0a5f985d77a83a7
The function F (PW, salt, 2000) takes the following values:
F(PW,salt,2000):
  bd 04 67 3f 71 49 b1 8e 98 15 5b d1 e2 72 4e 71
  d0 09 9a a2 51 74 f7 92 d3 32 6c 6f 18 12 70 67
The coordinates of the point Q_PW are:
X= 0xa0fd0bcfaa07f640c802aa95f42e80b28bb758fbcb7ee2aca2cc0a615b567207
Y= 0x52cf0c960f362894bd097d198999e965bd940c7828e0d2ad38a0097f68135047
During the calculation of the message u_1 on the subject A the parameter alpha, the point alpha*P and the message u_1 take the following values:
alpha=0x147b72f6684fb8fd1b418a899f7dbecaf5fce60b13685baa95328654a7f0707f
alphaP:
  X= 0x33fbac14eae538275a769417829c431bd9fa622b6f02427ef55bd60ee6bc2888
  Y= 0x22f2ebcf960a82e6cdb4042d3ddda511b2fba925383c2273d952ea2d406eae46
u_1:
  X= 0x8e8929226c7f679ea8c2dfb833d1f8062d62a9672493df02ad7462014c0edbc6
  Y= 0x20f2382c2425aaa638f61e8b70fcf70dae6bcb2f9f341b33ae577c62395aa816
During processing a message u_1, calculation the K_B key and the message u_2 on the subject B the parameters betta, src, K_B = HASH(src), betta*P and u_2 take the following values:

```
betta=0x30d5cfadaa0e31b405e6734c03ec4c5df0f02f4ba25c9a3b320ee6453567b4cb
src:
  a3 39 a0 b8 9c ef 1a 6f fd 4c a1 28 04 9e 06 84
  df 4a 97 75 b6 89 a3 37 84 1b f7 d7 91 20 7f 35
  11 86 28 f7 28 8e aa 0f 7e c8 1d a2 0a 24 ff 1e
  69 93 c6 3d 9d d2 6a 90 b7 4d d1 a2 66 28 06 63
K_B:
  7d f7 1a c3 27 ed 51 7d 0d e4 03 e8 17 c6 20 4b
  c1 91 65 b9 d1 00 2b 9f 10 88 a6 cd a6 ea cf 27
betta*P:
  X= 0x2b2d89fab735433970564f2f28cfa1b57d640cb902bc6334a538f44155022cb2
  Y= 0x10ef6a82eef1e70f942aa81d6b4ce5dec0ddb9447512962874870e6f2849a96f
u_2:
  X= 0x47182ed8f018fa93a5d837e52724af6051c168ef15e4a40fe926473bc3f1032a
  Y= 0x97f3e1e674da53b0ec3ebb1a62a25c7424f4334950daec4d33045f78d9faeeb4
During processing a message u_2 and calculation the key on the subject A
the K_A key takes the following value:
K_A:
  7d f7 1a c3 27 ed 51 7d 0d e4 03 e8 17 c6 20 4b
  c1 91 65 b9 d1 00 2b 9f 10 88 a6 cd a6 ea cf 27
The message MAC_A=HMAC (K_A, 0x01 || ID_A || ind || salt || u_1 || u_2)
from the subject A takes the following value:
MAC_A:
  f5 69 f6 e7 68 9e f0 ba 08 46 98 cc 0e bc ac 59
  67 8c 93 26 af 21 f5 4d 3e 90 05 29 32 6b 41 ee
The message MAC_B=HMAC (K_B, 0x02 || ID_B || ind || salt || u_1 || u_2)
from the subject B takes the following value:
MAC_B:
  80 d5 f0 3b 48 22 37 76 43 b4 ff 92 05 dd ed b1
  9f 22 80 1f b4 de 0b fb e0 74 55 c2 54 32 45 1e
```

A.2.7 Curveid-tc26-gost-3410-2012-512-paramSetC

```
The input protocol parameters in this example take the following values:
N= 3
ind= 1
ID_A:
  00 00 00 00
ID_B:
  00 00 00 00
PW:
  31 32 33 34 35 36 ('123456')
salt:
  29 23 be 84 e1 6c d6 ae 52 90 49 f1 f1 bb e9 eb
Q_ind:
  X= 0x5b065ead2e94de0ee2e462de204c93c6b2bf3498ad920393cb60259e1a8ffc7c
       7e7d4defa20ff4282abf70207e4611d532f40db6800e29d2b53f6ac0713e5b38
  Y= 0xa39a28c59ff7f796b85223b8834384907c626086415487288ed1182ca4487dc1
       ae5f37af90fd267b7c0dc8542ea52cd984af54731bc84271d6186d973c91359b
```

The function F (PW, salt, 2000) takes the following values:

```
F(PW,salt,2000):
  bd 04 67 3f 71 49 b1 8e 98 15 5b d1 e2 72 4e 71
  d0 09 9a a2 51 74 f7 92 d3 32 6c 6f 18 12 70 67
  1c 62 13 e3 93 0e fd da 26 45 17 92 c6 20 81 22
  ee 60 d2 00 52 0d 69 5d fd 9f 5f 0f d5 ab a7 02
```

The coordinates of the point Q_PW are:

```
X= 0x463e9d38239ddac18e7cc7f6caa7244ae5c49d58dcdfd6a56510d7779496744d
     75e3e0d5795d4e603f7baea8d24ada989d4179e1db33d1912602fc59470192df
Y= 0x088874b12c160930aa840f046ee75fa86206f19ca5f431d81e2381d6d947b7b0
     30577e40f09b1c16f8e6ef84daddba028f8b6e397a27ece0e13197662659af4d
```

During the calculation of the message u_1 on the subject A the parameter alpha, the point alpha*P and the message u_1 take the following values:

```
alpha=0x0b3fe942126aefc0287f82c6290505aeb117aa8dcb033cee56222dd1b9f9e1e5
     377583ba300211ec2c399546b4f54578ee925c238d52530c159c7034ccfa0ddd
alphaP:
  X= 0x61427a12468974b5829de1263d91fdfd8e26ea337c6c223595e05b4da4f8fe93
       2b532f33c0f4631729422c04f7018a7bf619c026ef0edc4ba2a96b79397eba92
  Y= 0x6833806e26791ef1dd01e60c10cc247173b97d7d8d7fea53de4a8a6a444bacc7
       042bf35394aef4cfde0f236788f2e9fca9e10f7d7fee54fff951ae17996808c1
u_1:
  X= 0x03664ef83e51beaec1f11711f8742b180001c7734a715e4a693758acd9851b38
       c6d7e0a316d809b75694ae1b356951a93c91a9b85aa3e3a561742211fd238852
  Y= 0x2b92fa93fab060fa86c3039eb2904bc18cbe45032dc3c93ce1c6ba1542a29e0d
       790a5f7b63928ed9e50d1fefd6bd00ade4eb021bc62a560567a3419e74dfc08a
```

During processing a message u_1, calculation the K_B key and the message u_2 on the subject B the parameters betta, src, K_B = HASH(src), betta*P and u_2 take the following values:

```
betta=0x0d494d54fb777781d1324ed6088bb0d9d86b8b0a252aa6a3ee70af8ef44b87a6
4cea3a432b61a699bad2d9760d700c2891b6285be0b0bb90f16a40a9b2e0e36a
src:
  c2 a3 1a 15 08 52 8a fb 70 be d5 7a 3a 97 9a 3a
  8c ed 00 2f 1b 05 8f 99 cb ee 64 56 5c cb ae 42
  c5 80 b1 39 18 04 b3 e3 34 d0 2f 70 55 18 ef 16
  b7 cf 0e 79 91 76 6f 7e 22 81 f2 87 b1 df cd 34
  5c 56 04 ef 1d 9a 8e b8 27 3f 2e 7a 3b fb a4 13
  ad 7f 19 59 99 41 f8 f6 73 63 2b e1 43 b1 65 7f
  d3 3a 3a de 7d 9f 71 6c e4 0c b5 9e 9d dd a5 0d
  db 87 66 57 e7 37 8f f1 55 94 fc 7a 9e 4b 03 48
K_B:
  84 14 e1 12 6c 56 a1 1e 1f 5e a0 b7 c3 bd ab e9
  8b 26 8b 59 d4 08 f9 7c d0 ea d7 c2 7e e4 9c 15
betta*P:
  X= 0xe247677c90ac3c74952c70da0d43f25ece4ac22eda732f7ddd772de7c3e69b22
       f7679cc01cae009e442c630c7aa9403a9f11e0fb62cf7af84e77b95210a17edd
  Y= 0x63be6dc920e57cb1c5b63fd8b623db6c934b87e0b14468de32c9387515cf3d35
       618e945a986424708ef0515ccaa30061ac6870ab56c29c43340736a6c6179c2e
u_2:
```

```
  X= 0x32260df3ddeabaa9c5c1f55248e8e9a3552cefb81a19f0ac1e10f3b7280a844c
       5362b527da1c6ec7eeace2a77aa1167f5e18a4bb6bc6445b4f479ca239245002
  Y= 0x04e0612a0c8cd4323535899d0698dd09bb9fc4302016f1b236c86692358ffd98
       1cd082c0129763bd4749ee5bb014255d1de0fd7775deccb564213ebc7100001d
```

During processing a message u_2 and calculation the key on the subject A the K_A key takes the following value:

```
K_A:
  84 14 e1 12 6c 56 a1 1e 1f 5e a0 b7 c3 bd ab e9
  8b 26 8b 59 d4 08 f9 7c d0 ea d7 c2 7e e4 9c 15
```

The message MAC_A=HMAC (K_A, 0x01 || ID_A || ind || salt || u_1 || u_2) from the subject A takes the following value:

```
MAC_A:
  53 0b 77 63 c5 9e 7c 98 52 59 ad eb af a4 16 41
  c6 f4 35 47 85 01 bd c9 7e a9 cf 88 a6 9a 12 8c
```

The message MAC_B=HMAC (K_B, 0x02 || ID_B || ind || salt || u_1 || u_2) from the subject B takes the following value:

```
MAC_B:
  3f 48 65 b8 8c 81 e5 ac 56 1e 31 c1 b3 d1 d9 0c
  57 e1 e7 4b ac 77 b1 63 ac 60 74 82 4e 99 d3 cc
```

Authors' Addresses

Stanislav Smyshlyaev (editor)
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: svs@cryptopro.ru

Evgeny Alekseev
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: alekseev@cryptopro.ru

Igor Oshkin
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: oshkin@cryptopro.ru

Vladimir Popov
CRYPTO-PRO
18, Suschevsky val
Moscow 127018
Russian Federation

Phone: +7 (495) 995-48-20
Email: vpopov@cryptopro.ru