Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 18, 2017

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

December 15, 2016

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

draft-smyshlyaev-sespake-13

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 June 18, 2017.

Table of Contents

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 is not 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 an 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:

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

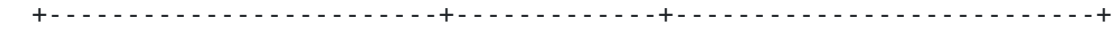
4.2. Initial values of the protocol counters

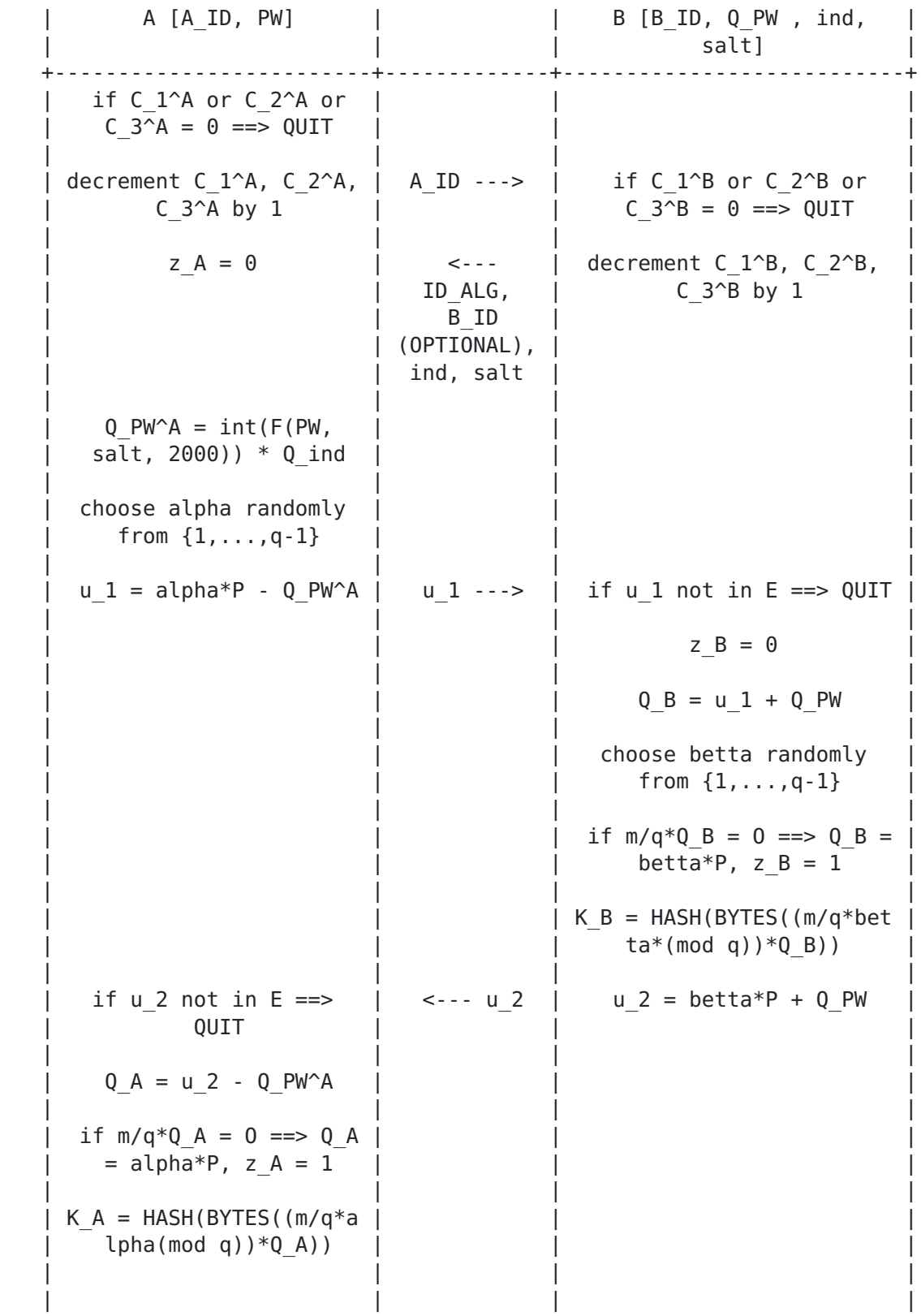
After the setup of a new password value PW the values of the counters MUST be assigned as follows:

- 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, C_3^A by 1
z_A = 0
Q_PW^A = int(F(PW, salt, 2000)) * Q_ind
choose alpha randomly from {1,...,q-1}
u_1 = alpha*P - Q_PW^A
A_ID --->
<--- ID_ALG, B_ID (OPTIONAL), ind, salt
u_1 --->
<--- u_2
if u_2 not in E ==> 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*alpha(mod q))*Q_A))
if C_1^B or C_2^B or C_3^B = 0 ==> QUIT
decrement C_1^B, C_2^B, C_3^B by 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*betta*(mod q))*Q_B))
u_2 = betta*P + Q_PW

U_1 = BYTES(u_1), U_2 = BYTES(u_2)		
MAC_A = HMAC(K_A, 0x01 || ID_A || ind || salt || U_1 || U_2 || ID_ALG (OPTIONAL) || DATA_A)	DATA_A, MAC_A --->	U_1 = BYTES(u_1), U_2 = BYTES(u_2)
		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, 0x02 || ID_B || ind || salt || U_1 || U_2 || ID_ALG (OPTIONAL) || DATA_A || DATA_B) ==> QUIT	<--- DATA_B, MAC_B	MAC_B = HMAC(K_B, 0x02 || ID_B || ind || salt || U_1 || U_2 || ID_ALG (OPTIONAL) || DATA_A || DATA_B)
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=0x0014f139e61f1c165019916ddd4fba581f24396ab2ff03a26a10007389879688
Y=0xc4b1c98e5319dcdfe1a348043fc369df9c4cce0b20727bda642d001f109ce195
SEED:
00 00 00 02

A.1.2. Curve id-GostR3410-2001-CryptoPro-B-ParamSet
Point Q_1
X=0x59694b16c95587a194acc31139cdecbc01f036cfa3589feb911defd7cf37cef6
Y=0x78393a073d54191db188e534c2309fdecf95b56fef54e11155e2c16b2f99f25f
SEED:
00 00 00 00

A.1.3. Curve id-GostR3410-2001-CryptoPro-C-ParamSet
Point Q_1
X=0x5ece29ddf3c2c2fdab15c7153e1217ff6066a25da0eaf0fbf1845a005b4fc6a8
Y=0x98b98c809ba902349145cb964ad092c7e072fc841b293927cf42d148d0749c83
SEED:
00 00 00 00

A.1.4. Curve id-tc26-gost-3410-2012-512-paramSetA
Point Q_1
X=0x12b157bde7474f99630eef43fdd5446f5b1aeaecab036a0dd716fb101c4c3410
f198be4a66d9db3c32f848e249b4a127d7f48d3dde2820ef35ae329c4462b469
Y=0x0bd049e5186a28897bc5779240cb9c3b0a135ff1c29805f41be7bbb344059348
dafaefe95e4ec9ef014f2cf5fa93c4976fdfa13c0c47c87a11c088e4c26f7ca6
SEED:
00 00 00 01

A.1.5. Curve id-tc26-gost-3410-2012-512-paramSetB
Point Q_1
X=0x5282e1bd4b4129c6bb7598651678ec0cf7bd70a7f6d03bebcf637be0db902b0a
9964fb4014e02df816ca91646e74dfdd85b8f2ee24bafa9ef3a765a1d74f9ca3
Y=0x22081bfacf0af5d47e0cfd1c0d38a405efd00acbe8bca1e35b39f712f239ffd9
e024c5fd43be538bea8c500dca83b80693a0436148b30201d42b675176522f85
SEED:
00 00 00 02

A.1.6. Curve id-tc26-gost-3410-2012-256-paramSetA
Point Q_1
X=0x16cffad2a33d8b6637454a5bda0fb9df8fbd59ae540c1efc569eb3ae371bf5f0
Y=0x74b3eb70bc445e5fdbd874c2dd4041b9b9d7d0a780efa498d34ee94d194de56f
U=0xa8eb62fc576364640cc2e8611296b8cdfb58a20eeb6d2ab8090d7221874a61e3
V=0xd3f3d67ac863d4eca8d84bbe99e6380e5736325527685093f4f51ae5cd835554
SEED:
00 00 00 00

A.1.7. Curve id-tc26-gost-3410-2012-512-paramSetC
Point Q_1
X=0x8337e003dbef8caa5a58ff2aa0bc9470da4c669570623a1d3d5ce092a7f28a80
143fbadf4c7056ceb5ae42e8ad0dccf5a248ff716ca4f6704b578e6c0b75d6ad
Y=0x5713159ce967dafd885b8b8caf3574341e04b5987bf5e41ec1971735a3ca350a
c65105407d32c1f9a42976d92b52c547a20014bdac0b2ee0171cff120da682d1
U=0x884e49002dd3ecd2bfb66adae0afde4992c652a7bd2f1bbbe5617d4ffe6919c1
369342e80612520e868efe3e9f2c4237cad144cda890339bb7ed29d6e745b2f4
V=0x0756ecf5083432acd5cab218d34f1ac5165c570b6aa5172b4bc784157f0c6c9a
b1ab08cf7734c1d5723e9bd0f07c6344c45f287e490f69a634d075cc26a40a92
SEED:
00 00 00 00

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

The input protocol parameters in this example take the following values:
N= 1
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 = 0x0014F139E61F1C165019916DDD4FBA581F24396AB2FF03A26A10007389879688
Y = 0xC4B1C98E5319DCDFE1A348043FC369DF9C4CCE0B20727BDA642D001F109CE195
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 = 0xA830AABA7E71AA33225475D38B7426799A5AC73FB9D8D56F804F59B3D382D003
Y = 0xB483264EA9CEA6C47BE5B6F3B542214F7598546CB4F84117390703F1A75E3B16
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=0x1F2538097D5A031FA68BBB43C84D12B3DE47B7061C0D5E24993E0C873CDBA6B3
alphaP:
  X = 0xBBC77CF42DC1E62D06227935379B4AA4D14FEA4F565DDF4CB4FA4D31579F9676
  Y = 0x8E16604A4AFDF28246684D4996274781F6CB80ABBBA1414C1513EC988509DABF
u_1:
  X = 0xF1B5F1C865832DA6696CCD5C91C47DA7C1D479D72C73E923A9C6574C67A31136
  Y = 0x87C1275A36073C86F31DE7FD5D791D63771E3A53A747BAD2443F2430E2753964
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=0xDC497D9EF6324912FD367840EE509A2032AEDB1C0A890D133B45F596FCCBD45D
src:
  2E 01 A3 D8 4F DB 7E 94 7B B8 92 9B E9 36 3D F5
  F7 25 D6 40 1A A5 59 D4 1A 67 24 F8 D5 F1 8E 2C
  A0 DB A9 31 05 CD DA F4 BF AE A3 90 6F DD 71 9D
  BE B2 97 B6 A1 7F 4F BD 96 DC C7 23 EA 34 72 A9
K_B:
  1A 62 65 54 92 1D C2 E9 2B 4D D8 D6 7D BE 5A 56
  62 E5 62 99 37 3F 06 79 95 35 AD 26 09 4E CA A3
betta*P:
  X = 0x6097341C1BE388E83E7CA2DF47FAB86E2271FD942E5B7B2EB2409E49F742BC29
  Y = 0xC81AA48BDB4CA6FA0EF18B9788AE25FE30857AA681B3942217F9FED151BAB7D0
u_2:
  X = 0xE97E2423F7862B4863252346AE5367FF0567854A4741E05B5FBD4351E96B0497
  Y = 0x8E1E30A8F0A411BE2D231D29C17C3806EEE3B5397B6FC6FB08E64F5379C4EDCA
During processing a message u_2 and calculation the key on the subject A the K_A key takes the following value:
K_A:
  1A 62 65 54 92 1D C2 E9 2B 4D D8 D6 7D BE 5A 56
  62 E5 62 99 37 3F 06 79 95 35 AD 26 09 4E CA A3
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:
  35 FF E4 CB 21 6E 5D 23 9D 26 68 E0 AB 86 C5 30
  94 E2 BD 1A 3E 30 67 81 39 C1 C0 F5 15 9E 83 A9
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:
  B0 4B 36 E2 09 FC 23 7B 22 7C 25 F2 47 B4 6B 69
  9A E3 11 89 10 E1 51 E0 1B A6 14 B6 52 1C 0D 6F

A.2.2 Curve id-GostR3410-2001-CryptoPro-B-ParamSet

The input protocol parameters in this example take the following values:
N= 1
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 = 0x59694B16C95587A194ACC31139CDECBC01F036CFA3589FEB911DEFD7CF37CEF6
  Y = 0x78393A073D54191DB188E534C2309FDECF95B56FEF54E11155E2C16B2F99F25F
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 = 0x6352105EFCE276140F6BF63681C2EC113DE366B1BF3C8701EE16B196DBA2A7C2
  Y = 0x034AED344DE92EBBD8CF40AF707078B003DF729C38351E41199E09E73390D755
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 = 0x5E70CB0ED63DFB011C7F7CB599C21F1576796F4F93A70F819E4313997FB8550F
  Y = 0x4B94B5945FD4FB97FD49CB606CFC13463AD4B4647EC0E7E926B153062562DCDD
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 = 0x0ABD49A25A734D569DEC9F3081BF127415D0BB85EBFD24187AF3BD28404AA302
  Y = 0x4CBE6942F8DA7D39545B2BFE3B7C598739D7F7B673B347AF3489A1942D1BD275
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:
  27 87 85 0F 20 FC 2A 71 99 04 C8 1F 63 96 13 F5
  27 5E 90 A1 64 D2 0A 15 9E BD 7B 3E A3 71 30 22
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:
  85 C1 E0 79 83 6D 7B 81 50 28 EB 31 7F 11 C2 DC
  AE C9 43 0D C5 B6 13 50 5D 6B 19 7D EB 51 D1 EA

A.2.3 Curve id-GostR3410-2001-CryptoPro-C-ParamSet

The input protocol parameters in this example take the following values:
N= 1
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 = 0x5ECE29DDF3C2C2FDAB15C7153E1217FF6066A25DA0EAF0FBF1845A005B4FC6A8
  Y = 0x98B98C809BA902349145CB964AD092C7E072FC841B293927CF42D148D0749C83
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 = 0x8B38C3A25DE0CF319CCDE95B772CF1972FDE525B5B36D6BE2304661B7EAEBFF0
  Y = 0x7930C3A4D4BB5F5CE1172BD47E9515E574FE344D3D48FBFF2DB3803D3416CF27
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 = 0x0A3FDC5475FF9098CF6DA936C856261AFF23882BC05E63E894AC839D1F65ACA9
  Y = 0x8B57F088956A2FB7C856888F474784322037DF612924D66F19459A751DCCEF50
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 = 0x8FF321793B2FC1A3F778C95AC9D7ECAC259D715AFF36F989D6441706B226A7DE
Y = 0x29121DB477E1EE44CFB771DFCD72004A6C56784610D50073B81A61BE1772AAB9
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:
70 E0 F3 91 50 7F DC 8B 2F 9C 05 98 87 D4 85 DB
A2 97 FE 2C 80 24 4D 7E 50 E0 5F 9E 23 76 12 0F
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:
D8 D6 57 FE E9 1D 3D 11 95 B5 25 86 1D C5 1A 48
C6 C2 E8 92 D8 47 45 2B 79 CE 62 E4 BB 7A 58 3E

A.2.4 Curve id-tc26-gost-3410-2012-512-paramSetA

The input protocol parameters in this example take the following values:
N= 1
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 = 0x12B157BDE7474F99630EEF43FDD5446F5B1AEAECAB036A0DD716FB101C4C3410
F198BE4A66D9DB3C32F848E249B4A127D7F48D3DDE2820EF35AE329C4462B469
Y = 0x0BD049E5186A28897BC5779240CB9C3B0A135FF1C29805F41BE7BBB344059348
DAFAEFE95E4EC9EF014F2CF5FA93C4976FDFA13C0C47C87A11C088E4C26F7CA6
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 = 0xA76B5059BF7E26E6F8F27A1821E1D1571DE5E30F5967E405EE58BD65E45BD849
        46867E5D95D9042AB834523B7CC5A84BE0D2487DB2903911002C6299E8E63F21
  Y = 0x810E12C68174D32FAF169FBD7243D3F41D26863F9FFAD2E7341F9BF9ED3ACDEA
        0FB6971A2FAA1DE6BC2A5DF9EFE4DB8626B15909B7ACC05E9F6BB049868E08DF
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 = 0xA0676AA3D2F628A53466AF138010C93579212EF836D7AB415E5664B68A944AD9
        D1B2F4FCCE845D77A3CDFA2B382A5D7EC1E67710BABA3FA234E0F00704BB6E8D
  Y = 0x7857F9D3AB63ADAE1189FBEE749AB9A597CC6039FC0C4F14452BE9BEFE78CB02
        DB7FFE3A6F44056748801B7D7797ECE9E2FC64461F5E647F1C1A2F71A4820BBE
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 = 0x86341289178788169B32CA9C9B991CADD827CAC3E898543708730C6147EA9052
        B848B5C2A58DF2708B7DC90672613DB2BE72C5329BCFE32893ACB329209D53D4
  Y = 0x60F833538B3583820AD0C2FE2022D0EB4BA242296FCF711AE6127BB2C9651CAE
```

3E7E9DD4BC46C0E7950B7AB7994FE4FAF14BD23749F102F7E069E767BBE192F3
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:
  39 16 61 47 3D 35 82 82 7C 57 D1 98 F0 8B 07 A6
  AA D0 50 99 74 16 4E 7B E5 70 28 D8 82 EE 20 57
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:
  5A 25 BE EE C9 55 5C A7 AD 90 84 30 5D 14 D9 2A
  01 DB B2 71 27 1E D7 87 DD 7B 05 88 F8 5F 09 A4

A.2.5 Curve id-tc26-gost-3410-2012-512-paramSetB

The input protocol parameters in this example take the following values:
N= 1
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 = 0x5282E1BD4B4129C6BB7598651678EC0CF7BD70A7F6D03BEBCF637BE0DB902B0A
        9964FB4014E02DF816CA91646E74DFDD85B8F2EE24BAFA9EF3A765A1D74F9CA3
  Y = 0x22081BFACF0AF5D47E0CFD1C0D38A405EFD00ACBE8BCA1E35B39F712F239FFD9
        E024C5FD43BE538BEA8C500DCA83B80693A0436148B30201D42B675176522F85
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 = 0x7593F9CF924158C6EB8D82C17EA553ADC33F44247AEBDC8AB2631C79677FD446
        3C6F4DAF99860914CB48A4E633598C0900A1E336CC8C0E102D98CFC644F55595
  Y = 0x72E3F2B7A87F0005FF1FB71971C481276ECCCC3140BC866A3CAD2D496D341653
        A50D6B6E61F7D097BBF4F3F68FFA2DDE97E37314F611CE761C3B68C3FC966DDE
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 = 0x262E01CEC119A98FB0109B8E753DEDB79D8AD63DFDB3DB9EDD16DC9EB2992BB9
E2BE54D106F11AC93D04A4BE04060A071A27BE663E71E877E9454D73C24B9D5B
Y = 0x10CDD37FF8931A39C893276CBF3867BD5DF31DF77664B51C924778F521A3D710
5DC07F3664AA45AE312DFA06B2BFAEE1B1362A2447D20C4F1B8B6C751BAE1BAB
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 = 0x0EA6EEDDFC36219035A83F4E9DEB13213A86904BFDD320D0135FB57F577FDB4D
9BFD073BA4FB5CEA25697C3D369173839B8E681F4C9D48531257906E52FF5049
Y = 0x73A6DEDD417A8D8BA2379403733762DCFC2501569155C410959F1EEA7692098D
255E506707B8AAD293D46D20D97F8D685213E461CC91CE602D483E69E64D7F05
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:
F8 7D A6 A5 F8 5F 12 F0 ED F3 4F F3 99 EF 0B 78
F6 CF 9E 44 B3 FA AF 47 E2 F4 8E F6 DA 9A A8 34

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:
  60 FA EB AD 09 FF 1F 28 7B 73 6B 58 EF C4 B8 E9
  D3 6A C4 95 C3 D1 33 FF E6 C3 43 C6 A3 DD 83 6B

A.2.6 Curve id-tc26-gost-3410-2012-256-paramSetA

The input protocol parameters in this example take the following values:
N= 1
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 = 0x16CFFAD2A33D8B6637454A5BDA0FB9DF8FBD59AE540C1EFC569EB3AE371BF5F0
  Y = 0x74B3EB70BC445E5FDBD874C2DD4041B9B9D7D0A780EFA498D34EE94D194DE56F
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 = 0xE77AE34BB9072E2A7A1C70D547F8E291183D71EE833603B9A837E76F37100883
  Y = 0x553FFB01DE6EB0913BF47097B905F8ADAAED8D148750AB15BDFC324A056C4471
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 = 0x1EFE487831CBB17062DA7475DC798431A513FB52C53BE598C61F3AC531A8222C
  Y = 0xDEA9398960CAF3F465FFC0A8FDBC3D9A20A8281F3C814870F99E817E4F75B24C
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 = 0x7F51C2475207551516E41F4DCB6CE30C0710AB309E9840A9671B58414EE9A01B
Y = 0x4A5A4BE45892214E911FD25E98627F954620CDCCB21A4648287C2DE9E8F25874
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:
E1 DA D1 73 EE E6 79 DD E0 BE A1 74 5E A0 B7 A4
81 EE BE 1E AD 0B 0C B6 B8 3A B5 9C 12 82 76 99
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:
AD F3 AA D1 A8 5A 71 30 FB 7D 78 6C D1 FE 30 62
48 C0 CB 7E DF AE D9 E3 3A DA 03 DD 24 6B E1 F2

A.2.7 Curve id-tc26-gost-3410-2012-512-paramSetC

The input protocol parameters in this example take the following values:
N= 1
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 = 0x8337E003DBEF8CAA5A58FF2AA0BC9470DA4C669570623A1D3D5CE092A7F28A80
143FBADF4C7056CEB5AE42E8AD0DCCF5A248FF716CA4F6704B578E6C0B75D6AD
Y = 0x5713159CE967DAFD885B8B8CAF3574341E04B5987BF5E41EC1971735A3CA350A
C65105407D32C1F9A42976D92B52C547A20014BDAC0B2EE0171CFF120DA682D1
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 = 0x613019CF4D3A2E3EEDD032C5AFC627C11B2689C600678D63EC3745B8BBF14C38

```
        3922A1DDB5E65284511D5FF6B7B5675F5A209DD73F4EB9F672EC9B9960F58641
  Y = 0x2FF035731C2516233268FA44CBE069B3838A27ED7F53DE3897F7D6ECB74B44E9
        DC83BF5D840CFD254339396671CF7C619D91C495FB6ABD03609EA43B5C839AB7
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=0x332F930421D14CFE260042159F18E49FD5A54167E94108AD80B1DE60B13DE799
        9A34D611E63F3F870E5110247DF8EC7466E648ACF385E52CCB889ABF491EDFF0
alphaP:
  X = 0x561655966D52952E805574F4281F1ED3A2D498932B00CBA9DECB42837F09835B
        FFBFE2D84D6B6B242FE7B57F92E1A6F2413E12DDD6383E4437E13D72693469AD
  Y = 0xF6B18328B2715BD7F4178615273A36135BC0BF62F7D8BB9F080164AD36470AD0
        3660F51806C64C6691BADEF30F793720F8E3FEAED631D6A54A4C372DCBF80E82
u_1:
  X = 0x8337D772EB93B6B437E35B9520FB8B9C0F2166F7A105E2755F7371BAEC3A2EBF
        F3B1F7FC7B805773AC26DF7B4BC13AF3E0DC0455283C8CD66212E86272EA4667
  Y = 0x345FC5809CB38607583F5EE5C915C7B711BADCBF05737E2DB38D0D51CE27CED7
        E99C389A98D40DFFD8DBFDF5E5B6D5636C73EE383FF47108DD4F07F0CEE150A1
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=0x38481771E7D054F96212686B613881880BD8A6C89DDBC656178F014D2C093432
        A033EE10415F13A160D44C2AD61E6E2E05A7F7EC286BCEA3EA4D4D53F8634FA2
src:
  4F 4D 64 B5 D0 70 08 E9 E6 85 87 4F 88 2C 3E 1E
  60 A6 67 5E ED 42 1F C2 34 16 3F DE B4 4C 69 18
  B7 BC CE AB 88 A0 F3 FB 78 8D A8 DB 10 18 51 FF
  1A 41 68 22 BA 37 C3 53 CE C4 C5 A5 23 95 B7 72
  AC 93 C0 54 E3 F4 05 5C ED 6F F0 BE E4 A6 A2 4E
  D6 8B 86 FE FA 70 DE 4A 2B 16 08 51 42 A4 DF F0
  5D 32 EC 7D DF E3 04 F5 C7 04 FD FA 06 0F 64 E9
  E8 32 14 00 25 F3 92 E5 03 50 77 0E 3F B6 2C AC
K_B:
  A0 83 84 A6 2F 4B E1 AE 48 98 FC A3 6D AA 3F AA
  45 1B 3E C5 B5 9C E3 75 F8 9E 92 9F 4B 13 25 8C
betta*P:
  X = 0xB7C5818687083433BC1AFF61CB5CA79E38232025E0C1F123B8651E62173CE687
        3F3E6FFE7281C2E45F4F524F66B0C263616ED08FD210AC4355CA3292B51D71C3
  Y = 0x497F14205DBDC89BDDAF50520ED3B1429AD30777310186BE5E68070F016A44E0
        C766DB08E8AC23FBDFDE6D675AA4DF591EB18BA0D348DF7AA40973A2F1DCFA55
u_2:
  X = 0x223ED4B60B27B1D145A19545FA2EEAACCCAFC4CE05EED1882A89DB8FC1673021
        C0696744455F2F18266EC82D270B74E3D3889719FD89860F34EB977B2E14C8DE
  Y = 0xFC9F377DCABFFCD9AF96BFFFAC496E346ABC8D4A4DF5D0E8146AD63DBB7FFB90
        D6E9A785E4329C1CF4F329E66081B9F30F60EAB2288BE28AD377AFE02329F047
During processing a message u_2 and calculation the key on the subject A
the K_A key takes the following value:
K_A:
  A0 83 84 A6 2F 4B E1 AE 48 98 FC A3 6D AA 3F AA
```

```
  45 1B 3E C5 B5 9C E3 75 F8 9E 92 9F 4B 13 25 8C
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:
  F3 56 03 36 0D 28 7D 7C D8 27 CC D2 52 E5 DE E9
  67 08 70 AE 74 3E 12 50 EB 63 A4 0C F3 88 78 8D
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:
  4D 02 E7 26 0A 47 53 49 CF C1 40 F1 DC A0 FE 7A
  90 E1 37 C2 7F A1 0A 49 F8 BF AF 32 FF 2E 0D D6
```

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