

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 19, 2016

A. Kato
NTT Software Corporation
T. Hardjono
MIT
T. Kobayashi
T. Saito
K. Suzuki
NTT
March 18, 2016

FSU Key Exchange
draft-kato-fsu-key-exchange-01

Abstract

This draft proposes an identity-based authenticated key exchange protocol following the extended Canetti-Krawczyk (id-eCK) model. The protocol is currently the most efficient among the id-eCK protocols.

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 September 19, 2016.

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

Internet-Draft

FSU Key Exchange

March 2016

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	3
1.1.	Our Motivation	3
2.	Requirements Terminology	4
3.	Notation	4
4.	Data Type and Its Conversions	6
4.1.	BitString-to-OctetString Conversion (BS2OSP)	7
4.2.	OctetString-to-BitString Conversion (OS2BSP)	7
4.3.	FieldElement-to-Integer Conversion (FE2IP)	7
4.4.	Integer-to-FieldElement Conversion (I2FEP)	7
4.5.	FieldElement-to-OctetString Conversion (FE2OSP)	7
4.6.	OctetString-to-FieldElement Conversion (OS2FEP)	8
4.7.	EllipticCurvePoint-to-OctetString Conversion (ECP2OSP)	8
4.8.	OctetString-to-EllipticCurvePoint Conversion (OS2ECP)	8
5.	Building Block of FSU Key Exchange	8
5.1.	Key Derivation Function	8
5.2.	Hashing to Point	9
5.2.1.	IHF1	10
5.2.2.	OS2FQE	11
5.3.	Group Membership Test Function	12
6.	FSU Key Exchange	13
6.1.	System Parameter Setup	13
6.2.	Key Distribution by KGC	14
6.3.	FSU Key Exchange Protocol	14
7.	Security Considerations	16
8.	Acknowledgements	16
9.	Algorithm Identifiers	16
10.	Change log	17
11.	Test Vectors	17
12.	References	17
12.1.	Normative References	17
12.2.	Informative References	17
Appendix A.	Construction of Data Conversion	19
A.1.	Construction of BS2OSP	19
A.2.	Construction of OS2BSP	19
A.3.	Construction of FE2IP	20
A.4.	Construction of I2FEP	20
A.5.	Construction of FE2OSP	21

A.6.	Construction of OS2FEP	22
A.7.	Construction of ECP20SP	22
A.8.	Construction of OS2ECPP	24
Authors' Addresses	25

[1.](#) Introduction

Authenticated key exchange (AKE) is a core security function within many deployed systems today. It is a foundational function that allows end-users and systems alike to be authenticated prior to access to resource and services. Over the past two decades key exchange schemes have been proposed, based on symmetric and asymmetric key cryptography.

A more recent approach to AKE protocol has been the introduction of identity binding to the exchange [[7](#)] [[8](#)], obviating the need to rely on a public key infrastructure in which digital certificates need to be exchanged by users or end-points that wish to communicate signed and/or encrypted messages.

Identity-based AKE (ID-AKE) schemes rely on the use of the trusted intermediary referred to as the Key Generation Center (KGC). The role of the KGC, among others, is to generate a pair of master public and secret keys based on the user's identity and to extract a user's secret key corresponding to his or her identity.

In a 2-pass ID-AKE scheme, an "initiator" entity wishing to share a key with a second entity (referred to as the "responder") sends ephemeral public information to the responder. In its turn, the responder sends another ephemeral public information to the initiator entity. Following this, each entity would then generate a session from a number of parameters, notably their respective secret keys (given by the KGC), their own secret values of the ephemeral information, the identity of the peer they're communicating with, and the ephemeral information they received from that peer.

We propose a provably secure ID-AKE scheme called "FSU" [[4](#)] [[5](#)] [[6](#)] based on the previous model of [[9](#)] and which builds on the previous efforts in [[10](#)] [[11](#)]. The model underlying the FSU was chosen due to the merit of provable security based on an adversarial model in which the adversary has the freedom to choose keys reveal.

[1.1.](#) Our Motivation

In order to establish secure communications, the encryption is used, and a key exchange protocol is necessary to use the encryption. If the key exchange protocol has vulnerability, an attacker can intercept all messages, so encrypted session becomes meaningless. In practice, man in the middle attack and a forward security of key exchange protocol are serious issues.

In recent years, IoT technology gathers many attentions. It is expected that 26-30 billion devices will be wirelessly connected by

2020. And to set up a huge number of devices with certificates or passwords for key exchange and to maintain the certificates or passwords require many costs. Furthermore, the leakage of a secret key for key exchange and a session key for encryption likely to occur because of resource restriction of device and installation environment of device.

To resolve above problems, we propose an ID-based authenticated key exchange protocol FSU. In usual PKI based cryptography, a device must set up password or generate own secret key. On the other hand, in the FSU protocol the trusted third party generate the secret key for a huge number of IoT devices, so the manufacture and users of the devices can maintain secret key for the devices unitarily. The FSU Protocol use existing ID, which can be any string, e.g., e-mail address and serial number, as public key instead of certificate or password. Thus, the authentication server is not required to manage the certificates and the passwords of device any more. Finally, the FSU protocol provides the highest security against leakages of secret keys. Thus, security of a session key is preserved even if some secret keys are leaked because of resource restriction and installation environment of devices.

[2.](#) Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this memo are to be interpreted as described in [\[1\]](#).

[3.](#) Notation

This section shows notation used in this memo.

Let F_q be a finite field with $q = p^n$ elements for a prime p and an integer n and let $E(F_q)$ be an elliptic curve with an order r and an embedding degree k defined over F_q . An embedding degree k is defined as a minimum integer k such that r is a divisor of $q^k - 1$.

Let G_1 (resp. G_2) be an additive group with an order r generated by $E(F_q)$ (resp. $E'(F_q)$). Let G_T be multiplicative groups with the same order r . Let P_1, P_2 be generators of G_1, G_2 respectively. We say that (G_1, G_2, G_T) are bilinear map groups if there exists a pairing $e: (G_1, G_2) \rightarrow G_T$ satisfying the following properties:

1. Bilinearity: for any Q_1 in G_1 , for any Q_2 in G_2 , for any a, b in \mathbb{Z}_r , we have the relation $e(aQ_1, bQ_2) = e(Q_1, Q_2)^{ab}$.

Kato, et al.

Expires September 19, 2016

[Page 4]

Internet-Draft

FSU Key Exchange

March 2016

2. Non-degeneracy: for any Q_1 in G_1 , $e(Q_1, Q_2) = 1$ only if $Q_2 = O_{G_2}$ and for any Q_2 in G_2 , $e(Q_1, Q_2) = 1$ only if $Q_1 = O_{G_1}$.
3. Computability: for any Q_1 in G_1 , for any Q_2 in G_2 , the bilinear map is efficiently computable.

This pairing is described in specification of optimal ate pairing specification[3]. It is defined by Pairing-Param-ID following way.

```
Pairing-Param-ID = {  
    G1-Curve-ID,  
    G2-Curve-ID  
    GT-Field-ID  
}
```

G1-Curve-ID and G2-Curve-ID is an identifiers of elliptic curve. And GT-Field-ID is an identifier of the G_T range of finite field. G1-Curve-ID, G2-Curve-ID and GT-Field-ID are described in [2] the following way.

```
G1-Curve-ID = {
  p_b      : A prime specifying base field F_p.
  A, B     : The coefficients of the equation  $y^2 = x^3 A * x + B$ 
              defining E.
  G = (x, y) : The base point, i.e., a point with x and y
              being its x- and y-coordinates in E, respectively.
  r        : The prime order of the group generated by G.
  h        : The cofactor of G in E.
}
G2-Curve-ID = {
  p_b      : A prime specifying base field F_p.
  e2       : The constant of an irreducible polynomial specifying
              extension field  $F_{p^2} = F_p[u] / (u^2 - e2)$ .
  A', B'   : The coefficients of the equation  $y^2 = x^3 A' * x +$ 
              B' defining E'.
```

```

    G' = (x', y') : The base point, i.e., a point with x' and y'
                    being its x- and y-coordinates in E', respectively.
    r'           : The prime order of the group generated by G'.
    h'           : The cofactor of G' in E'.
}

GT-Filed-ID = {
    p_b      : A prime specifying base field.
    r        : The prime order of the subgroup of  $F_{\{p^{12}\}}$ .
    e2       : The constant of the irreducible polynomial of  $F_{\{p^2\}} =$ 
                $F_p[u] / (u^2 - e2)$ .
    e6       : The constant of the irreducible polynomial of  $F_{\{p^6\}} =$ 
                $F_{\{p^2\}}[v] / (v^3 - e6)$ .
    e12      : The constant of the irreducible polynomial of  $F_{\{p^{12}\}}$ 
                $= F_{\{p^6\}}[w] / (w^2 - e12)$ .
    h''      : The cofactor of  $G_T$ 
}

```

In addition, this memo uses the following functions.

`floor(x)` : The function returning an integer such that $\max\{x' \text{ in } Z \mid x' \leq x\}$.

`ceil(x)` : The function returning an integer such that $\min\{x' \text{ in } Z \mid x' \geq x\}$.

`O_E` : The point at infinity over elliptic curve E.

[4.](#) Data Type and Its Conversions

This section describes data type and its conversion used in this memo.

[4.1.](#) BitString-to-OctetString Conversion (BS2OSP)

This memo uses conversion from bit strings to octet strings. Informally, the idea is to pad the bit string with 0's on the left to make its length a multiple of 8, then chop the result up into octets. Formally, the conversion routine, BS2OSP(B), is specified in [Appendix A.1](#)

[4.2.](#) OctetString-to-BitString Conversion (OS2BSP)

This memo uses conversion from octet strings to bit strings. Informally, the idea is simply to view the octet string as a bit string. Formally, the conversion routine, OS2BSP(M), is specified in [Appendix A.2](#)

[4.3.](#) FieldElement-to-Integer Conversion (FE2IP)

This memo uses conversion from field elements to integers. An finite field element should be represented as a polynomial with subfield coefficients, which can be represented as a sequence of the coefficients. Informally, the idea is simply to view the sequence of the coefficients as the radix- p^m representation of the base field elements, where p^m is the number of the subfield elements. Formally, the conversion routine, FE2IP(a), is specified in [Appendix A.3](#)

[4.4.](#) Integer-to-FieldElement Conversion (I2FEP)

This memo uses conversion from integers to field elements. A field element should be represented as a polynomial with subfield coefficients, and it can be represented as a sequence of the coefficients. Informally, the idea is to represent the integer with radix- p^m positional number system where p^m is the number of the subfield element, and then convert the each digit to the each coefficient of the polynomial. Formally, the conversion routine, I2FEP(x), is specified in [Appendix A.4](#):

[4.5.](#) FieldElement-to-OctetString Conversion (FE2OSP)

This memo uses conversion from field elements to octet strings. This conversion is constructed by using FE2IP and I2SOP conversions. Formally, the conversion routine, FE2OSP(a), is specified in [Appendix A.5](#).

[4.6.](#) OctetString-to-FieldElement Conversion (OS2FEP)

This memo uses conversion from octet strings to field elements. This conversion is constructed by using OS2IP and I2FEP conversions. Formally, the conversion routine, OS2FEP(M), is specified in [Appendix A.6](#).

[4.7.](#) EllipticCurvePoint-to-OctetString Conversion (ECP2OSP)

This memo uses conversion from elliptic curve points to octet strings. Informally the idea is that, if point compression is being used, the compressed y-coordinate is placed in the leftmost octet of the octet string along with an indication that point compression is on, and the x-coordinate is placed in the remainder of the octet string; otherwise if point compression is off, the leftmost octet indicates that point compression is off, and remainder of the octet string contains the x-coordinate followed by the y-coordinate. Formally, the conversion routine, ECP2OSP(P,R), is specified in [Appendix A.7](#).

[4.8.](#) OctetString-to-EllipticCurvePoint Conversion (OS2ECP)

This memo uses conversion from octet strings to elliptic curve points. Informally, the idea is that, if the octet string represents a compressed point, the compressed y-coordinate is recovered from the leftmost octet, the x-coordinate is recovered from the remainder of the octet string, and then the point compression process is reversed; otherwise the leftmost octet of the octet string is removed, the x-coordinate is recovered from the left half of the remaining octet string, and the y-coordinate is recovered from the right half of the remaining octet string. Formally, the conversion routine, OS2ECP(M), is specified in [Appendix A.8](#).

[5.](#) Building Block of FSU Key Exchange

This section describes building block for constructing FSU Key Exchange.

[5.1.](#) Key Derivation Function

MGF1 is a mask generation function, parameterized by a hash function. MGF1(M,n) is defined as follows:

System parameters:

- o Hash : a hash function
- o hashLen : the length in octets of the hash function output

Input:

- o `M` : a seed from which a mask is generated, an octet string
- o `n` : the octet length of the output, a positive integer

Output:

- o `mask` : a mask, an octet string of length `n`

Method:

1. Let `n_0` be the octet length of `M`. If `n_0 + 4` is greater than the input limitation for the hash function, output INVALID and stop.
2. Set `cThreshold = ceil(n / hashLen)`
3. If `cThreshold > 2^32`, output INVALID and stop
4. Let `M'` be the empty octet string
5. Set `counter = 0`
6. `B = B_{0}, ..., B_{31}` such that `counter = B_{31} + B_{30}*2 + ... + B_{0}*2^{31}`
7. Compute `C = BS2OSP(B)`
8. Compute `H = Hash(M || C)`
9. Set `M' = M' || H`
10. Set `counter = counter + 1`
11. If `counter < cThreshold`, go back to step 6.
12. Set `mask = M'_0M'_1...M'_{n-1}` where `M' = M'_0M'_1M'_2...`
13. Output `mask`

[5.2.](#) Hashing to Point

Hashed value should be converted to elliptic curve point as described in this section. Formally, the conversion routine, `HASHINGTOPOINT(Curve-ID, Hash, M)`, is specified as follows:

Input:

- o Curve-ID : an elliptic curve parameter
- o Hash : a hash function
- o M : an octet string

Output:

- o P : an elliptic curve point

Method:

1. Set $i = 0$
2. $B = B_{\{0\}}, \dots, B_{\{15\}}$ such that $\text{counter} = B_{\{15\}} + B_{\{14\}} \times 2 + \dots + B_{\{0\}} \times 2^{\{15\}}$
3. Compute $C = \text{BS2OSP}(B)$
4. $x_0 = \text{OS2FQE}(C || M, \text{Hash}, F_{\{p^m\}})$ in $F_{\{p^m\}}$
5. $t = x_0^3 + A * x_0 + B$
6. If $t=0$, set $P = (x_0, 0)$ and output $h' * P$
7. If t is not square in $F_{\{p^m\}}$, set $i = i + 1$ and go back to step 2
8. Set α be one of square roots of t . Then, $-\alpha$ is another square root of t .
9. Set $y_1 = \text{FE2IP}(\alpha)$
10. Set $y_2 = \text{FE2IP}(-\alpha)$
11. If $y_1 > y_2$, set $y_0 = -\alpha$
12. Else (i.e. $y_1 \leq y_2$), set $y_0 = \alpha$

13. Set $P = (x_0, y_0)$

14. Output $h * P$

[5.2.1.](#) IHF1

Bit string should be converted to hashed non-negative integer less than an assigned integer as described in this section. Formally, the conversion routine, $IHF1(s,n,Hash)$ is defined as follows:

Input:

- o s : an octet string
- o n : an integer
- o $Hash$: a hash function

Output:

- o v in Z_n

Method:

1. Set $hashLen$ be the length of the output of the hash function $Hash$
2. Set h_0 be the zero string of length $hashLen$
3. $h_1 = Hash(h_0 || s)$
4. $B = B_0, \dots, B_{l-1} = OS2BSP(h_1)$
5. $a_1 = \sum_{i=0}^{l-1} 2^{l-1-i} * B_{\{i\}}$
6. $h_2 = Hash(h_1 || s)$
7. $B = B_0, \dots, B_{l-1} = OS2BSP(h_2)$
8. $a_2 = \sum_{i=0}^{l-1} 2^{l-1-i} * B_{\{i\}}$
9. $v = 2^{hashLen} * a_1 + a_2 \bmod n$

10. Output v

[5.2.2.](#) OS2FQE

Octet string should be converted to hashed finite field element as described in this section. Formally, the conversion routine, $\text{OS2FQE}(s, \text{Hash}, F_{\{p^m\}})$ is defined as follows:

Input:

- o s : an octet string
- o Hash : a hash function

- o $F_{\{p^m\}}$: a finite field with p^m elements where p is a prime, and $m > 0$ is an integer

Output:

- o a : an element in $F_{\{p^m\}}$

Method:

1. Set $i = 0$
2. $B = B_{\{0\}}, \dots, B_{\{31\}}$ such that $\text{counter} = B_{\{31\}} + B_{\{30\}} * 2 + \dots + B_{\{0\}} * 2^{\{31\}}$
3. Compute $C = \text{BS2OSP}(B)$
4. Compute $t_i = \text{IHF1}(C || s, p, \text{Hash})$
5. If $i < m$, set $i = i + 1$ and go back to step2
6. Compute $a = \sum_{i=0}^{m-1} t_i * \text{beta}^i$ where beta is the variable of the polynomial
7. Output a

5.3. Group Membership Test Function

GROUPMEMBERSHIPTEST(Curve-ID, P) is a test function that an elliptic curve point is on the correct curve and group. GROUPMEMBERSHIPTEST is defined as follows:

Input:

- o Curve-ID : an elliptic curve identifier
- o $P = (x, y)$: an elliptic curve point

Output:

- o boolean : an integer in $\{0, 1\}$

Method:

1. If $P = O_E$, then output 1
2. If $y^2 \neq x^3 + A * x + B$, then output 0
3. If $h \neq 1 \ \&\& \ r * P \neq O_E$, then output 0

4. Output 1

6. FSU Key Exchange

This section provides the specification of ID-based authenticated key exchange protocol FSU [4] that is an extension of FSU (Fujioka-Suzuki-Ustaoglu) protocol standardized in ISO/IEC11770-3 [5] [6].

6.1. System Parameter Setup

Key Generation Center (KGC) defines the following system parameters in FSU:

- o Pairing-Param-ID : An identifier for showing asymmetric pairing. i.e., G1-Curve-ID, G2-Curve-ID and GT-Filed-ID.
- o G1-Curve-ID is an identifier for showing an elliptic curve which defines cyclic groups G_1 with prime p_{b_1} , coefficients A_1 and

B_1 , generator P_1 , order r , and cofactor h_1 .

- o G2-Curve-ID is an identifier for showing an elliptic curve which defines cyclic groups G_2 with prime p_{b_2} , irreducible polynomial $e_{2,2}$, coefficients A_2 and B_2 , generator P_2 , order r , and cofactor h_2 .
- o GT-Field-ID is an identifier for showing a pairing co-domain group which is subgroup of order r in $G_{\{\phi_{12}(p)\}}$. $G_{\{\phi_{12}(p)\}}$ is the 12-th cyclotomic subgroup of order $p^4 - p^2 + 1$ in $F_{\{p^{12}\}}^*$.
- o HASH-ID : An identifier for showing a hash function i.e., Hash : $\{0,1\}^* \rightarrow \{0,1\}^{\text{hashLen}}$.
- o hashLen : Length of output by Hash.
- o KDF-ID : An identifier for showing key derivation function, i.e., MGF1: $\{0,1\}^* \rightarrow \{0,1\}^n$.
- o n : Length of output by key derivation function.
- o R : A point compression type of conversion between elliptic curve point and octet string specifically "Compressed", "Uncompressed", or "Hybrid".

KGC generates the master secret key MSK and master public key MPK from system parameters as following.

1. KGC selects a random integer z in Z_r .

2. KGC computes $Z_v = z * P_v$ for v is in $\{1, 2\}$.
3. KGC sets $MSK = z$ and $MPK = (Z_1, Z_2)$.

Hash function H_v are defined as $H_v(M) = \text{HASHINGTOPOINT}(Gv\text{-Curve-ID}, \text{Hash}, \text{"FSU"} || \text{ECP2OSP}(Z_1, R) || \text{ECP2OSP}(Z_2, R) || M)$ for v in $\{1, 2\}$.
Hash function H is defined as $H(M) = \text{MGF1}(\text{"FSU"} || \text{ECP2OSP}(Z_1, R) || \text{ECP2OSP}(Z_2, R) || M, n)$.

[6.2](#). Key Distribution by KGC

This subsection explains operations of key distribution by KGC. There are two types of static secret key in FSU Key Exchange, respectively static secret key based on cyclic groups in G_1 and in G_2 . FSU Key Exchange requires that an initiator and a responder use static secret key with different types, respectively. Hence, KGC needs to define a rule for key distribution for users. For example, clients use static secret keys in G_1 and servers use them in G_2 .

KGC generates static secret key $D_{\{i, v\}}$ for an identifier ID_i for i in $\{A, B\}$ of user in G_v as following.

1. Let MPK be (Z_1, Z_2) and MSK be z .
2. KGC Compute $D_{\{i, v\}} = z * H_v(ID_i)$.
3. Distribute $D_{\{i, v\}}$ to a user with ID_i .

6.3. FSU Key Exchange Protocol

This subsection describes FSU Key Exchange Protocol in an initiator U_A with an identifier ID_A and static secret key $D_{\{A,1\}}$ and a responder U_B with an identifier ID_B and static secret key $D_{\{B,2\}}$.

Computation of ephemeral public key by U_A

1. U_A selects a random integer x_A in Z_r .
2. U_A computes the ephemeral public key $X_{\{A,v\}} = x_A * P_v$ for v in $\{1,2\}$.
3. U_A computes $XOS_{\{A,v\}} = ECP2OSP(X_{\{A,v\}}, R)$ for v in $\{1,2\}$.
4. U_A sends $(ID_A, ID_B, XOS_{\{A,1\}}, XOS_{\{A,2\}})$ to U_B .

Computation of ephemeral public key by U_B

1. U_B receives $(ID_A, ID_B, XOS_{\{A,1\}}, XOS_{\{A,2\}})$.

2. U_B computes $X_{\{A,v\}} = OS2ECP(XOS_{\{A,v\}})$ for v in $\{1,2\}$.
3. If $(GROUPMEMBERSHIPTEST(G1-Curve-ID, X_{\{A,1\}}) = 0 \mid \mid$
 $GROUPMEMBERSHIPTEST(G2-Curve-ID, X_{\{A,2\}}) = 0 \mid \mid e(X_{\{A,1\}}, P_2)$

$\neq e(P_1, X_{\{A,2\}}))$, then abort.

4. U_B selects a random ephemeral secret key x_B in Z_r .
5. U_B computes the ephemeral public key $X_{\{B,v\}} = x_B * P_v$ for v in $\{1,2\}$.
6. U_B computes $XOS_{\{B,v\}} = ECP2OSP(X_{\{B,v\}}, R)$ for v in $\{1,2\}$.
7. U_B sends $(ID_B, ID_A, XOS_{\{B,1\}}, XOS_{\{B,2\}})$ to U_A .

Computation of session key by U_B

1. U_B computes $\sigma_1 = e(H_1(ID_A), D_{\{B,2\}})$.
2. U_B computes $\sigma_2 = e(H_1(ID_A) + X_{\{A,1\}}, D_{\{B,2\}} + x_B * Z_2)$.
3. U_B computes $\sigma_3 = x_B * X_{\{A,1\}}$.
4. U_B computes $\sigma_4 = x_B * X_{\{A,2\}}$.
5. U_B computes $\sigma_{OS_j} = FE2OSP(\sigma_j)$ for j in $\{1,2\}$.
6. U_B computes $\sigma_{OS_{j'}} = ECP2OSP(\sigma_{j'}, R)$ for j' in $\{3,4\}$.
7. Set $sid = (ID_A || ID_B || XOS_{\{A,1\}} || XOS_{\{A,2\}} || XOS_{\{B,1\}} || XOS_{\{B,2\}})$.
8. U_B computes session key $K = H(\sigma_{OS_1} || \sigma_{OS_2} || \sigma_{OS_3} || \sigma_{OS_4} || sid)$.

Computation of session key by U_A

1. U_A computes $X_{\{B,v\}} = OS2ECP(XOS_{\{B,v\}})$ for v in $\{1,2\}$.
2. If $(GROUPMEMBERSHIPTEST(G1-Curve-ID, X_{\{B,1\}}) = 0 || GROUPMEMBERSHIPTEST(G2-Curve-ID, X_{\{B,2\}}) = 0 || e(X_{\{B,1\}}, P_2) \neq e(P_1, X_{\{B,2\}}))$, then abort.
3. U_A computes $\sigma_1 = e(D_{\{A,1\}}, H_2(ID_B))$.
4. U_A computes $\sigma_2 = e(D_{\{A,1\}} + x_A * Z_1, H_2(ID_B) + X_{\{B,2\}})$.

5. U_A computes $\text{sigma}_3 = x_A * X_{\{B,1\}}$.
6. U_A computes $\text{sigma}_4 = x_A * X_{\{B,2\}}$.
7. U_A computes $\text{sigmaOS}_j = \text{FE2OSP}(\text{sigma}_j)$ for j in $\{1,2\}$.
8. U_A computes $\text{sigmaOS}_{j'} = \text{ECP2OSP}(\text{sigma}_{j'}, R)$ for j' in $\{3,4\}$.
9. Set $\text{sid} =$
 $(\text{ID}_A || \text{ID}_B || \text{XOS}_{\{A,1\}} || \text{XOS}_{\{A,2\}} || \text{XOS}_{\{B,1\}} || \text{XOS}_{\{B,2\}})$.
10. U_A compute session key $K =$
 $H(\text{sigmaOS}_1 || \text{sigmaOS}_2 || \text{sigmaOS}_3 || \text{sigmaOS}_4 || \text{sid})$.

7. Security Considerations

This memo specifies identity-based authenticated key exchange protocol FSU [4] [6] [5] which is secure in the id-eCK(id-based extended Canetti-Krawczyk) security model under the GBDH(gap bilinear DH) assumption [4].

id-eCK security model is the most strong security model in the meaning of that it ensures the safety of session key if any non-trivial combinations of master key, static key, and ephemeral key are leaked.

And id-eCK security model guarantees following 4 security notions:

MitM(resistance to man in the middle attacks),

wPFS(weak perfect forward security),

KCI(resistance to key compromise impersonation attacks),

RLE(resilience to leakage of ephemeral private keys).

8. Acknowledgements

TBD

9. Algorithm Identifiers

TBD

Internet-Draft

FSU Key Exchange

March 2016

[10.](#) Change log

NOTE TO RFC EDITOR: Please remove this section in before final RFC publication.

[11.](#) Test Vectors

TBD

[12.](#) References

[12.1.](#) Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997.
- [2] Kasamatsu, K., Kanno, S., Kato, A., Scott, M., Kobayashi, T., and Y. Kawahara, "Barreto-Naehrig Curves", [draft-kasamatsu-bncurves-02](#) (work in progress), 2015.
- [3] Kato, A., Scott, M., Kobayashi, T., and Y. Kawahara, "Barreto-Naehrig Curves", [draft-kato-optimal-ate-pairings-01](#) (work in progress), 2015.

[12.2.](#) Informative References

- [4] Fujioka, A., Hoshino, F., Kobayashi, T., Suzuki, K., Ustaglu, B., and K. Yoneyama, "id-eCK Secure ID-Based Authenticated Key Exchange on Symmetric and Asymmetric Pairing", Proceedings IEICE Transactions 96-A(6): 1139-1155, 2013.
- [5] Fujioka, A., Suzuki, K., and B. Ustaglu, "Ephemeral Key Leakage Resilient and Efficient ID-AKEs That Can Share Identities, Private and Master Keys", Proceedings Pairing 2010 Lecture Notes in Computer Science Volume 6487, pp 187-205, 2010.
- [6] "Information technology -- Security techniques -- Key management -- Part 3: Mechanisms using asymmetric

techniques.", ISO/IEC 11770-3: 2015, 2015.

- [7] Shamir, A., "Identity-based Cryptosystems and Signature Schemes", Proceedings CRYPTO '84, LNCS 196, pages 47-53, Springer-Verlag, 1984.

Kato, et al.

Expires September 19, 2016

[Page 17]

Internet-Draft

FSU Key Exchange

March 2016

- [8] Boneh, D. and M. Franklin, "Identity-Based Encryption from the Weil Pairing", Proceedings CRYPTO 2001, LNCS 2139, pages 213-229, Springer-Verlag, 2001.
- [9] Huang, H. and Z. Cao, "An ID-based Authenticated Key Exchange Protocol Based on Bilinear Diffie-Hellman Problem", Proceedings the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS '09) pp. 333-342, ACM, 2009.
- [10] Canetti, R. and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", Proceedings Eurocrypt 2001 (LNCS2015), pp. 453-474, Springer-Verlag, 2001.
- [11] LaMacchia, B., Lauter, K., and A. Mityagin, "Stronger Security of Authenticated Key Exchange", Proceedings in Provable Security (LNCS 4784), pp. 1-16, Springer, 2007.

[Appendix A](#). Construction of Data Conversion

[A.1](#). Construction of BS2OSP

Concrete construction of BS2OSP(B) is specified as follows:

Input:

- o $B = B_0 B_1 \dots B_{l-1}$: a bit string of length l

Output:

- o $M = M_0 M_1 \dots M_{n-1}$: an octet string of length $n = \text{ceil}(l/8)$.

Method:

1. If $l = 0$, then output empty octet string and stop.
2. For j in $\{0, \dots, 8n-1\}$, if $j \geq 8n - l$, set $B'_j = B_{j-(8n-l)}$, otherwise set $B'_j = 0$.
3. For i in $\{0, \dots, n-1\}$, set $M_i = B'_{8i} B'_{8i+1} \dots B'_{8i+7}$.
4. Output $M = M_0 M_1 \dots M_{n-1}$.

[A.2](#). Construction of OS2BSP

Concrete construction of OS2BSP(M) is specified as follows:

Input:

- o $M = M_0M_1\dots M_{n-1}$: an octet string of length n .

Output:

- o $B = B_0B_1\dots B_{l-1}$: a bit string of length $l = 8*n$

Method:

1. If $l = 0$, then output empty octet string and stop.
2. For i in $\{0, \dots, n-1\}$, j in $\{0, \dots, 7\}$, set B_{8i+j} in $\{0,1\}$ as $M_i = B_{8i} B_{8i+1} \dots B_{8i+7}$.
3. Output $B = B_0 B_1 \dots B_{l-1}$.

[A.3.](#) Construction of FE2IP

Concrete construction of FE2IP(a) is specified as follows:

System parameters:

- o $F_{\{p^{m_2}\}}/F_{\{p^{m_1}\}}$: a field extension with an irreducible polynomial $\text{Irr}(F_{\{p^{m_2}\}} / F_{\{p^{m_1}\}}; \beta)$

Input:

- o a : a field element in $F_{\{p^{m_2}\}}$

Output:

- o x : an integer in $\{0, \dots, p^{m_2} - 1\}$

Method:

1. If $m_2 = 1$ (i.e. $F_{\{p^{m_2}\}}$ is prime field)

A field element of $F_{\{p^{m_2}\}}$ must be represented as an integer in $\{0, \dots, p-1\}$

(A) Set $x = a$

(B) Output x

2. Else (i.e. $m_2 > 1$)

(A) Let the coefficients a_i in $F_{\{p^{m_1}\}}$ for i in $\{0, \dots, m_2 / m_1 - 1\}$ such that $a = \sum_{i=0}^{m_2 / m_1 - 1} a_i \cdot \beta^i$

(B) Compute $x = \sum_{i=0}^{m_2 / m_1 - 1} \text{FE2IP}(a_i) \cdot (p^{m_1})^i$

(C) Output x

[A.4.](#) Construction of I2FEP

Concrete construction of I2FEP(x) is specified as follows:

System parameters:

- o $F_{\{p^{m_2}\}}/F_{\{p^{m_1}\}}$: a field extension with an irreducible polynomial $\text{Irr}(F_{\{p^{m_2}\}} / F_{\{p^{m_1}\}}; \beta)$

Input:

- o x : an integer in $\{0, \dots, p^{m_2} - 1\}$

Output:

- o a : a field element in $F_{\{p^{m_2}\}}$

Method:

1. If $m_2 = 1$ (i.e. $F_{\{p^{m_2}\}}$ is prime field)

A field element of $F_{\{p^{m_2}\}}$ must be represented as an integer in $\{0, \dots, p-1\}$

(A) Set $a = x$

(B) Output a

2. Else (i.e. $m_2 > 1$)

(A) Let x_i be an element in $\{0, \dots, p^{m_1}-1\}$ for i in $\{0, \dots, m_2 / m_1 - 1\}$ such that $x = \sum_{i=0}^{m_2 / m_1 - 1} x_i * p^{m_1 i}$

(B) Compute $a = \sum_{i=0}^{m_2 / m_1 - 1} \text{I2FEP}(x_i) * \text{beta}^i$

(C) Output a

[A.5.](#) Construction of FE2OSP

System parameter:

- o $F_{\{p^m\}}$: a finite field with p^m elements where p is a prime, and $m > 0$ is an integer
- o n : an integer equivalent to $\text{ceil}(m * \log_2 p / 8)$

Input:

- o a : a field element in $F_{\{p^m\}}$

Output:

- o M : an octet string

Method:

1. Compute $I = \text{FE2IP}(a)$

2. Compute $X = x_{\{0\}}, \dots, x_{\{n-1\}}$ such that $I = x_{\{n-1\}} + x_{\{n-2\}} * 2 + \dots + x_{\{1\}} * 2^{\{n-2\}} + x_{\{0\}} * 2^{\{n-1\}}$

3. Compute $M = \text{BS2OSP}(X)$
4. Output M

[A.6.](#) Construction of OS2FEP

System parameter:

- o $F_{\{p^m\}}$: a finite field with p^m elements where p is a prime, and $m > 0$ is an integer
- o n : an integer equivalent to $\text{ceil}(m * \log_2 p / 8)$

Input:

- o M : an octet string

Output:

- o a : a field element in $F_{\{p^m\}}$

Method:

1. Compute $X = \text{OS2BSP}(M)$
2. Let X be $x_0, \dots, x_{\{l-1\}}$
3. Compute $I = \sum_{i=0}^{l-1} 2^{l-1-i} * x_{\{i\}}$
4. Compute $a = \text{I2FEP}(I)$
5. Output a

[A.7.](#) Construction of ECP2OSP

Concrete construction of $\text{ECP2OSP}(P, R)$, is specified as follows:

System parameters:

- o Curve-ID : an elliptic curve parameter

Input:

- o P : a point on an elliptic curve over $F_{\{p^m\}}$
- o R : compression type specifically "Compressed", "Uncompressed", or "Hybrid"

Output:

- o M : an octet string of length n

Method:

1. If $P = O_E$
 - (A) Compute $M = \text{BS2OSP}(00000000)$
 - (B) Output M
2. If $P = (x, y) \neq O_E$ && $R = \text{Compressed}$
 - (A) Set $X = \text{FE2OSP}(x)$
 - (B) If p is odd && $y = 0$, set $y' = 0$
 - (C) Else if p is odd && $y \neq 0$, set $y' = y_i \bmod 2$ such that $y = y_{\{m-1\}} * \beta^{m-1} + \dots + y_1 * \beta + y_0$ and i is the smallest integer such that $y_i \neq 0$
 - (D) If $y' = 0$, compute $L = \text{BS2OSP}(00000100)$
 - (E) If $y' = 1$, compute $L = \text{BS2OSP}(00000101)$
 - (F) Output $M = L || X$
3. If $P = (x, y) \neq O_E$ && $R = \text{Uncompressed}$
 - (A) Set $X = \text{FE2OSP}(x)$
 - (B) Set $Y = \text{FE2OSP}(y)$
 - (C) Compute $L = \text{BS2OSP}(00000100)$
 - (D) Output $M = L || X || Y$
4. If $P = (x, y) \neq O_E$ && $R = \text{Hybrid}$
 - (A) Set $X = \text{FE2OSP}(x)$
 - (B) Set $Y = \text{FE2OSP}(y)$

Internet-Draft

FSU Key Exchange

March 2016

(C) If $y = 0$, set $y' = 0$

(D) Else (i.e. $y \neq 0$) $y' = y_i \bmod 2$ such that $y = y_{m-1} * \beta^{m-1} + \dots + y_1 * \beta + y_0$ and i is the smallest integer such that $y_i \neq 0$

(E) If $y' = 0$, compute $L = \text{BS2OSP}(00000110)$

(F) If $y' = 1$, compute $L = \text{BS2OSP}(00000111)$

(G) Output $M = L || X || Y$

[A.8.](#) Construction of OS2ECPP

Concrete construction of OS2ECPP(M), is specified as follows:

System parameters

- o Curve-ID : an elliptic curve parameter

Input:

- o M : an octet string

Output:

- o P : an elliptic curve point

Method:

1. If $M = \text{BS2OSP}(00000000)$, output $P = O_E$

2. If M has length $\text{ceil}(m * \log_2 p / 8) + 1$

(A) Let M be $L || X$ where L is a single octet

(B) Compute $x = \text{OS2FEP}(X)$

(C) If $L = \text{BS2OSP}(00000010)$, then set $y' = 0$

(D) Else if $L = \text{BS2OSP}(00000011)$, then set $y' = 1$

(E) Else output INVALID and stop

(F) Compute $w = x^3 + A * x + B$

(G) Compute $\gamma = \text{square}(w)$

(H) If there is no γ in $F_{\{p^m\}}$, then output INVALID and stop

(I) Else if $\gamma = 0$, then set $y = 0$

(J) Else if $\gamma_i = y' \bmod 2$ where $\gamma = \gamma_{\{m-1\}} * \beta^{\{m-1\}} + \dots + \gamma_{\{1\}} * \beta + \gamma_{\{0\}}$ and i is the smallest integer such that $\gamma_i \neq 0$

(K) Else if $\gamma_i \neq y' \bmod 2$, set $y = -\gamma$ where $\gamma = \gamma_{\{m-1\}} * \beta^{\{m-1\}} + \dots + \gamma_{\{1\}} * \beta + \gamma_{\{0\}}$ and i is the smallest integer such that $\gamma_i \neq 0$

(L) Output $P = (x, y)$

3. If M has length $2 * \text{floor}(m * \log_2 p / 8) + 1$

(A) Let M be $L || X || Y$ where L is a single octet, X is $\text{floor}(m * \log_2 p / 8)$ octets, and Y is $\text{floor}(m * \log_2 p / 8)$ octets

(B) Unless L is BS2OSP(00000100), BS2OSP(00000110) or BS2OSP(00000111), output INVALID and stop.

(a) Compute $x = \text{OS2FEP}(X)$

(b) Compute $y = \text{OS2FEP}(Y)$

(c) If (x, y) does not satisfy the equation of elliptic curve, then output INVALID and stop

(d) Output $P = (x, y)$

Akihiro Kato
NTT Software Corporation

EMail: kato.akihiro-at-po.ntts.co.jp

Thomas Hardjono
MIT

EMail: hardjono-at-mit.edu

Kato, et al.

Expires September 19, 2016

[Page 25]

Internet-Draft

FSU Key Exchange

March 2016

Tetsutaro Kobayashi
NTT

EMail: kobayashi.tetsutaro-at-lab.ntt.co.jp

Tsunekazu Saito
NTT

EMail: saito.tsunekazu-at-lab.ntt.co.jp

Koutarou Suzuki
NTT

EMail: suzuki.koutarou-at-lab.ntt.co.jp

