S. V. Grebnev, Limonnitsa: making Limonnik-3 post-quantum, *Mat. Vopr. Kriptogr.*, 2020, Volume 11, Issue 2, 25–42

DOI: 10.4213/mvk319

МАТЕМАТИЧЕСКИЕ ВОПРОСЫ КРИПТОГРАФИИ
2020 Т. 11 № 2 С. 25–42

УДК 519.719.2 DOI https://doi.org/10.4213/mvk319

Limonnitsa: making Limonnik-3 post-quantum

S. V. Grebnev

Technical Committee for Standardization, «Cryptography and security mechanisms» (TC 26), Moscow

Получено 05.XI.2019

Abstract. We propose *Limonnitsa*, a secure authenticated key exchange (AKE) scheme which brings together the standardized in 2017 by Rosstandart *Limonnik*-3 AKE scheme (a part of Standardization Recommendations R 1323565.1.004-2017 "Key agreement schemes based upon public-key methods" and the supersingular elliptic curves isogeny cryptographic framework alongside with standardized cryptographic primitives, which makes the protocol secure against even the efficient quantum computers. The protocol does not require a digital signature as a "standalone" primitive, allows the parties to use different sets of parameters. We describe the protocol, discuss *Limonnitsa*'s basic cryptographic properties and preliminary choice of its basic parameters that conforms with another standardized cryptographic primitives. We show that the protocol is secure against known classes of attacks, including the problem of determining the parties' secret keys. We give security arguments in a modified Canetti–Krawczyk model based upon the assumption of the hardness of supersingular isogeny analogue of the Diffie–Hellman problem. Thus, we show that *Limonnitsa* is a versatile, secure cryptographic protocol that conforms the requirements expected from modern authenticated key exchange protocols.

Keywords: authenticated key exchange, isogenies, Limonnik-3, post-quantum cryptography, supersingular elliptic curves

Лимонница: постквантовая модификация Лимонника-3

С. В. Гребнев

Технический комитет по стандартизации «Криптографическая защита информации» (ТК 26), Москва

Аннотация. Предлагается протокол выработки общего ключа с аутентификацией на основе открытого ключа "Лимонница". Протокол основан на структуре протокола "Лимонник-3", вошедшего в состав Рекомендаций по стандартизации Р 1323565.1.004-2017 "Схемы выработки общего ключа с аутентификацией на основе открытого ключа",

Citation: Matematicheskie Voprosy Kriptografii, 2020, v. 11, № 2, pp. 25–42 (Russian)

утвержденных в 2017 году Росстандартом, но в дополнение к стандартизированным криптографическим решениям предполагает использование механизма изогений суперсингулярных эллиптических кривых, что делает возможным его использование даже при появлении эффективных квантовых вычислителей, способных решать задачи криптографического анализа. Протокол не требует использования цифровой подписи как отдельного примитива, позволяет двум участникам использовать различные наборы параметров для выработки и сертификации открытого ключа. В статье описывается протокол "Лимонница", исследуются заложенные в его основу синтезные решения и криптографические требования, предъявляемые к протоколу при его разработке, изучаются вопросы криптографической стойкости. При условии применения предлагаемых в статье параметров и алгоритмов показана стойкость протокола относительно известных классов атак, в том числе задачи определения секретного ключа. Получено формальное доказательство стойкости протокола в модифицированной модели Канетти–Кравчика в предположении о вычислительной сложности аналога задачи Диффи–Хеллмана для случая изогений суперсингулярных эллиптических кривых. Таким образом, показано, что протокол "Лимонница" является гибким, стойким криптографическим решением, удовлетворяющим всем предъявляемым к современным протоколам выработки общего ключа требованиям.

Ключевые слова: выработка общего ключа, изогении, Лимонник-3, постквантовая криптография, суперсингулярные эллиптические кривые

1. Introduction

An emerging threat of quantum computers leads cryptographers to review many of existing public key cryptographic systems. For example, cryptanalysis of the schemes based upon the factorization problem, such as RSA and Rabin, as well as discrete logarithm based schemes, including Diffie–Hellman–Merkle and ElGamal, is reduced to a polynomial-time quantum algorithm.

Thus, although the prospectives of the construction of a powerful enough (from a cryptanalyst's point of view) quantum computer are unclear, many researchers are concerned about creating "post-quantum" schemes which are to withstood both "classical" (that is, based upon the Turing-style computations) and "quantum" cryptanalysis. We mention the NIST proposal for the post-quantum family of cryptosystems which has brought anomalous amount of research into the post-quantum field.

In 2017, Russia officially accepted a family of AKE protocols designed by the author as recommendations for standardization (that is, a candidate to become a national standard). This family includes *Echinacea*-2, *Echinacea*-3 and *Limonnik*-3 protocols[1].

Both the protocols are based upon the elliptic curve Diffie–Hellman scheme and are thus quantum-insecure. The *Echinacea*-3 protocol is constructed on the base of the ISO-STS-MAC [10], using KEA+C [18] ideas. The *Limonnik*-3 protocol is constructed on the base of the MTI/A0 protocol [19] with influences by [6] and [20].

Unlike *Echinacea*, *Limonnik*-3 does not require digital signatures, it may be viewed as the outputs of two elliptic curve Diffie–Hellman processes, each one mixing a static and an ephemeral key, hashed together to compute a shared secret key. Thus, we choose this scheme for post-quantum conversion, replacing the Diffie–Hellman protocol by its post-quantum analogue.

Amongst the multiple post-quantum proposals, we have chosen SIDH, the supersingular elliptic curves isogeny-based Diffie–Hellman key exchange protocol [8] for the following reasons:

– unlike most NIST competitors, the protocol allows for static keys (see, however, [13, 17] for discussion of several attacks against static keys), which are mandatory for an AKE scheme,

– the protocol, for a given security parameter, provides keys of moderate size,

– the protocol may be implemented quite efficiently with a well-studied mechanisms.

We proceed with a general description of the *Limonnik*-3 and basic ideas of supersingular elliptic curves cryptography.

2. Limonnik-3

We choose protocol parameters h_2, h_3 as two fixed distinct non-empty strings. The function $\pi\colon E(GF(p)) \to \{0,1\}^*$ represents the x-coordinate of a point as a binary string, $\texttt{KDF}(\dots)$ is a key derivation function. We suppose that KDF outputs a pair of keys (K, M). Further, $\texttt{MAC}_K(\dots)$ is a message authentication code defined in [3].

An optionally used information connected to the session (timestamps, IP addressess, previously shared secret strings etc.) which may be used

[1] *Эхинацея* (*Echinacea purpurea*) and *Лимонник* (*Schizandra chinensis*) are medicinal herbs extensively used in Russian complementary medicine.

Limonnik-3

$A:$	$k_A \in_R [1, q_B - 1]$
$A \to B$	$\mathtt{ID}_A, \mathtt{Cert}_A, k_A P_B$
$B:$	$k_B \in_R [1, q_A - 1], Q = c_A k_B S_A, R = c_B s_B k_A P_B$
	$(K, M) = \mathtt{KDF}(\pi(Q), \pi(R), \mathtt{ID}_A \parallel \mathtt{ID}_B[\parallel \mathtt{OI}])$
	$tag_B = \mathtt{MAC}_M(h_2, k_B P_A, k_A P_B, \mathtt{ID}_B, \mathtt{ID}_A)$
$B \to A$	$\mathtt{ID}_B, \mathtt{Cert}_B, k_B P_A, tag_B$
$A:$	$Q = c_A s_A k_B P_A, R = c_B k_A S_B$
	$(K, M) = \mathtt{KDF}(\pi(Q), \pi(R), \mathtt{ID}_A \parallel \mathtt{ID}_B[\parallel \mathtt{OI}])$
	If $tag_B \neq \mathtt{MAC}_M(h_2, k_B P_A, k_A P_B, \mathtt{ID}_B, \mathtt{ID}_A)$, terminates the session with an error
	$tag_A = \mathtt{MAC}_M(h_3, k_A P_B, k_B P_A, \mathtt{ID}_A, \mathtt{ID}_B)$
$A \to B$	tag_A
$B:$	If $tag_A \neq \mathtt{MAC}_M(h_3, k_A P_B, k_B P_A, \mathtt{ID}_A, \mathtt{ID}_B)$, terminates the session with an error

during key generation is denoted $\mathtt{OI}$. Concatenation of strings a, b is denoted by $a \parallel b$.

A party A's identity is denoted by ID_A. We suppose that the communicating parties A and B are using two (possibly different) elliptic curves $E_A(GF(p_A))$ and $E_B(GF(p_B))$ defined over corresponding prime fields $GF(p_A), GF(p_B)$.

A party's curve has the following parameters important for the description:

– $m_A = |E_A|$,

– P_A is a point of large prime order q_A, $q_A | m_A$,

– $c_A = m_A / q_A$ is the cofactor.

Static key pairs (s_A, S_A) and (s_B, S_B) are defined as $S_A = s_A P_A$, $S_B = s_B P_B$, where $0 < s_A < q_A$, $0 < s_B < q_B$, and certified by $\mathtt{Cert}_A, \mathtt{Cert}_B$.

We also assume that any party verifies validity of certificate received and correctness of elliptic curve points, terminating the session with an error if an invalid certificate or a "bad" point (i.e. not belonging to the given elliptic curve or having a small order) is provided by another party.

If the scheme succcessfully completes, the parties A and B are mutually authenticated and provided with an implicitly verified shared secret key K.

Note that the key M is used only for the purposes of key confirmation and must be destroyed after the session is established, see [14].

2.1. Isogenies in brief

Consider an elliptic curve $E_{a,b}(\mathcal{F})$ defined over a field $\mathcal{F}$, $\operatorname{char}\mathcal{F} \neq 2, 3$, $E_{a,b}(\mathcal{F}) = \{(x, y) : x, y \in \mathcal{F}, y^2 = x^3 + ax + b\}$.

Definition 1. Let $E_{a,b}$, E_{a_1,b_1} be elliptic curves over K, denote by $\overline{\mathcal{F}}(E_{a,b})$ the field of rational functions over $E_{a,b}$. A *rational map* $E_{a,b}$ to E_{a_1,b_1} is a map

$$\psi = \psi(x, y) = (f_1(x, y), f_2(x, y)),$$

where $f_1(x, y), f_2(x, y) \in \overline{\mathcal{F}}(E_{a,b})$, such that $(f_1(x_0, y_0), f_2(x_0, y_0)) \in E_{a_1,b_1}(\mathcal{F})$ for any point $(x_0, y_0) \in E_{a,b}$ where the functions are defined.

Definition 2. A rational map defined at every point of $E_{a,b}(\mathcal{F})$, is a *morphism.*

Definition 3. If ψ is a morphism and $\psi(\mathcal{O}) = \mathcal{O}_1$, then ψ is an *isogeny.* If such a map exists, the curves are called *isogenous.*

Definition 4. For any isogeny $\psi\colon E \to E'$ there exists an unique *dual* isogeny $\hat{\psi}\colon E' \to E$ such that $\hat{\psi} \circ \psi = [m]_E$ and $\psi \circ \hat{\psi} = [m]_{E'}$, where m is the *degree* of the isogeny ψ.

Definition 5. Considering three elliptic curves $E(\mathcal{F}), E'(\mathcal{F}), E''(\mathcal{F})$ and isogenies $\phi, \psi\colon$ $\phi\colon E \mapsto E'$, $\psi\colon E' \mapsto E''$, we define the *composition* of isogenies $\psi\phi\colon E \mapsto E''$.

We have that $\widehat{\psi\phi} = \hat{\phi}\hat{\psi}$ and $\deg\psi\phi = \deg\psi \deg\phi$.
Let now $\operatorname{char}\mathcal{F} = p$. Denote by $E[p^e]$ the torsion group of E.

Definition 6. If $E[p^e] = \{\mathcal{O}\}$ for any $e = 1, 2, \ldots$, the curve E is called *supersingular.*

There are about $\lfloor p/12 \rfloor$ distinct supersingular curves defined over $GF(p^2)$, see [8]; it is sufficient for cryptographic applications.

2.2. Computation of isogenies

One can use Vélu's formulae [23] to compute isogenies φ with a given kernel (i.e. a subgroup $G \subset E$), $\varphi\colon E \mapsto E' = E/G$. Given curve coefficients a, b

for E, and all of the x-coordinates x_i of the subgroup $G \subset E$, Vélu's formulae output a', b' for E', and the map

$$\varphi\colon E \to E' = E/G, \quad (x,y) \mapsto \left(\frac{f_1(x,y)}{g_1(x,y)}, \frac{f_2(x,y)}{g_2(x,y)}\right),$$
$$\frac{f_1(x,y)}{g_1(x,y)}, \frac{f_2(x,y)}{g_2(x,y)} \in \overline{\mathcal{F}}(E_{a,b}).$$

The complexity of computation of isogeny of degree l is $O(l)$ field operations. For isogenies of smooth degrees, however, the complexity may be lowered by decomposing it into a composition of isogenies of small degrees.

We recall that isomorphic curves have the same j-invariant. Since construction of an isomorphism is a simple operation, the isogeny problem is actually the problem of finding isogenies between classes of isomorphic curves, every one of which is represented by its j-invariant.

3. Supersingular Isogeny Diffie–Hellman

We proceed with the description of the Supesingular Isogeny Diffie–Hellman scheme (SIDH), following [8].

We fix the public parameters: $p = l_A^{e_A} l_B^{e_B} \cdot f \pm 1$, where l_A, l_B are distinct small prime numbers (e.g., $l_A = 2$ and $l_B = 3$), $(l_A, f) = (l_B, f) = 1$, a supersingular elliptic curve $E_0(GF(p^2))$ and bases $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$, which generate correspondingly $E_0[l_A^{e_A}]$ and $E_0[l_B^{e_B}]$, i.e. $\langle P_A, Q_A\rangle = E_0[l_A^{e_A}]$ and $\langle P_B, Q_B\rangle = E_0[l_B^{e_B}]$.

The party A chooses two random elements $m_A, n_A \in_R \mathbb{Z}/l_A^{e_A}\mathbb{Z}$, not both divisible by l_A, and constructs an isogeny $\varphi_A : E_0 \to E_A$ with the kernel $\mathcal{K}_A := \langle [m_A]P_A + [n_A]Q_A\rangle$. The party A also computes the image $\{\varphi_A(P_B), \varphi_A(Q_B)\}$ and sends these points to the party B together with E_A.

Simultaneously, the party B chooses two random elements $m_B, n_B \in_R \mathbb{Z}/l_B^{e_B}\mathbb{Z}$, not both divisible by l_B, and constructs an isogeny $\varphi_B : E_0 \to E_B$ with the kernel $\mathcal{K}_B := \langle [m_B]P_B + [n_B]Q_B\rangle$. The party B also computes the image $\{\varphi_B(P_B), \varphi_B(Q_B)\}$ and sends these points to the party A.

Having received the party B's set $E_B, \varphi_B(P_B), \varphi_B(Q_B)$, the party A constructs an isogeny $\varphi'_A : E_B \to E_{AB}$ with the kernel $\langle [m_A]\varphi_B(P_A) + [n_A]\varphi_B(Q_A)\rangle$; the party B operates in a similar way.

The shared key may be computed as the j-invariant of the curve

$$E_{AB} = \varphi'_B(\varphi_A(E_0)) = \varphi'_A(\varphi_B(E_0))$$
$$= E_0/\left\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B\right\rangle.$$

We have a following commutative diagram:

$$\begin{array}{ccc} E & \xrightarrow{\varphi} & E/\langle P\rangle \\ {\scriptstyle\psi}\downarrow & & \downarrow \\ E/\langle Q\rangle & \longrightarrow & E/\langle P,Q\rangle \end{array} \tag{1}$$

where φ, ψ are walks in the graphs of isogenies of degrees equal to powers of l_A, l_B.

The protocol implements an analogue of the Diffie–Hellman scheme over this commutative diagram, where the party A chooses φ, and B chooses ψ.

4. Putting things together: Limonnitsa

In this section we describe an AKE scheme that is derived from *Limonnik*-3 by merging into it the ideas of supersingular elliptic curves isogenies cryptography. The new protocol is named *Limonnitsa*[2].

So, we fix two (possibly distinct) sets of the public parameters for the parties:

- $p_A = 2^{e_{a2}}3^{e_{a3}} - 1$,
- $E_{A0}(GF(p_A^2))$ — a supersingular elliptic curve, for example, $y^2 = x^3 + x$,
- linearly independent points $P_{A2}, Q_{A2} \in E_{A0}[2^{e_{a2}}]$ (that is, $|\langle P_{A2}, Q_{A2}\rangle| = 2^{2e_{a2}}$) and linearly independent points $P_{A3}, Q_{A3} \in E_{A0}[3^{e_{a3}}]$ (that is, $|\langle P_{A3}, Q_{A3}\rangle| = 3^{2e_{a3}}$).

For the party B, we have:

- $p_B = 2^{e_{b2}}3^{e_{b3}} - 1$,
- $E_{B0}(GF(p_B^2))$ — a supersingular elliptic curve, for example, $y^2 = x^3 + x$,
- linearly independent points $P_{B2}, Q_{B2} \in E_{B0}[2^{e_{b2}}]$ (that is, $|\langle P_{B2}, Q_{B2}\rangle| = 2^{2e_{b2}}$), and linearly independent points $P_{B3}, Q_{B3} \in E_{B0}[3^{e_{b3}}]$ (that is, $|\langle P_{B3}, Q_{B3}\rangle| = 3^{2e_{b3}}$).

Now, the party A selects its secret static key as an integer s_A such that $0 < s_A < 2^{e_{a2}}$, calculates $E_A = E_{A0}/\langle P_{A2} + [s_A]Q_{A2}\rangle$, constructs the isogeny $\varphi_A\colon E_A \to E_A/\langle P_{A2} + [s_A]Q_{A2}\rangle$, the points $P_A = \varphi_A(P_{A3})$,

[2] *Лимонница* (*limonnitsa*) stands for a brimstone butterfly (*Gonepteryx rhamni*) in Russian.

$Q_A = \varphi_A(Q_{A3})$, sets its static public key to $\{E_A, P_A, Q_A\}$, and acquires a certificate $\mathtt{Cert}_A$.

B selects its static key as an integer (s_B such that $0 < s_B < 2^{e_{b2}}$, calculates $E_B = E_{B0}/\langle P_{B2} + [s_B]Q_{B2}\rangle$, constructs the isogeny $\varphi_B\colon E_B \to E_B/\langle P_{B2} + [s_B]Q_{B2}\rangle$, the points $P_B = \varphi_B(P_{B3})$, $Q_B = \varphi_B(Q_{B3})$, sets is static public key as $\{E_B, P_B, Q_B\}$, and acquires a certificate $\mathtt{Cert}_B$ as well.

We use SIKE-style [15] modification to the original scheme and generate kernels of the isogenies for public key calculation in the form $\langle P + [k]Q\rangle$, that is, we generate a single random value. As shown by Galbraith [12], the computational problems of finding the secret isogeny are equivalent.

Our protocol reminds Galbraith's variant of the NAXOS protocol from [11]; however, in our setting (provided that the parties' parameters may differ) the ephemeral-to-ephemeral shared key which is employed in NAXOS key generation cannot be produced.

The protocol is a combination of static-to-ephemeral sessions, and thus may be subject to an attack against static keys by [17]. In order to thwart the attack, a party must ensure that the public keys it receives are valid, i.e. elliptic curves are built as prescribed by the protocols, the generators are chosen at random and are of prescribed order and linearly independent. Several validation techniques are described in [13]. The paper [22] states that the key validation problem may be equivalent to the CSSI problem (see below); thus, we would rather use the following variant of Kirkwood's trick from [16].

Instead of choosing random ephemeral secret key k_A, the party A chooses a single random seed $r_A \in \{0,1\}^*$ and uses a pseudo-random function $\mathtt{prf}$ to output $k_A = \mathtt{prf}(r_A)$. Then, tag_A is calculated as

$$tag_A = \mathtt{ENC}_M(h_3, r_A, \mathcal{K}_A, \mathcal{K}_B, \mathtt{ID}_A, \mathtt{ID}_B),$$

where $\mathtt{ENC}_K(\dots)$ is the «Kuznyechik» encryption [2] using the key M. The party B, having calculated the session key, recovers the seed r_A and repeats A's computations in order to verify that the keys were constructed as prescribed, otherwise, terminates the session. The parties B and A proceed vice versa. See Appendix A for the complete description of the protocol.

5. Analysis

The security of the protocol relies on the hardness of the following problem.

Problem 1. *Computational Supersingular Isogeny — CSSI: let* $\phi_1\colon E_0 \to E_1$ *be an isogeny with the kernel* $[m_1]R_1 + [n_1]S_1$, *where*

Limonnitsa

$A:$	$k_A \in_R [1, 3^{e_{b3}}], S_{AB} = P_{B3} + [k_A]Q_{B3},$
	$\varphi_{AB} : E_B \to E'_A = E_B/\langle S_{AB}\rangle$ – an isogeny with the kernel $\langle S_{AB}\rangle$
	$E_{AB} = E_{B0}/\langle S_{AB}\rangle$ (that is, $E_{AB} = \varphi_{AB}(E_{B0})$)
	$\mathcal{K}_A = \{E'_A, \varphi_{AB}(P_{B2}), \varphi_{AB}(Q_{B2})\}$ – A's ephemeral public key
$A \to B$	$\mathtt{ID}_A, \mathtt{Cert}_A, \mathcal{K}_A$
$B:$	$k_B \in_R [1, 3^{e_{a3}}], S_{BA} = P_{A3} + [k_B]Q_{A3},$
	$\varphi_{BA} : E_A \to E'_B = E_A/\langle S_{BA}\rangle$ – an isogeny with the kernel $\langle S_{BA}\rangle$
	$E_{BA} = E_{A0}/\langle S_{BA}\rangle$ (that is, $E_{BA} = \varphi_{BA}(E_{A0})$)
	$\mathcal{K}_B = \{E'_B, \varphi_{BA}(P_{A2}), \varphi_{BA}(Q_{A2})\}$ – B's session public key
	$T_{AB} = P_A + [k_B]Q_A$
	$T'_{AB} = \varphi_{AB}(P_{B2}) + [s_B]\varphi_{AB}(Q_{B2})$
	$\psi_{AB} : E'_A \to E'_A/\langle T_{AB}\rangle$ – an isogeny with the kernel $\langle T_{AB}\rangle$
	$\psi'_{AB} : E_B \to E_B/\langle T'_{AB}\rangle$ – an isogeny with the kernel $\langle T'_{AB}\rangle$
	$E_{AB} = \psi_{AB}(E'_A); E'_{AB} = \psi'_{AB}(E_B)$
	$(K, M) = \mathtt{KDF}(j(E_{AB}) \parallel j(E'_{AB}) \parallel \mathtt{ID}_A \parallel \mathtt{ID}_B[\parallel \mathtt{OI}])$
	$tag_B = \mathtt{MAC}_M(h_2, \mathcal{K}_B, \mathcal{K}_A, \mathtt{ID}_B, \mathtt{ID}_A)$
$B \to A$	$\mathtt{ID}_B, \mathtt{Cert}_B, \mathcal{K}_B, tag_B$
$A:$	$T_{BA} = \varphi_{BA}(P_{A2}) + [s_A]\varphi_{BA}(Q_{A2})$
	$T'_{BA} = P_B + [k_A]Q_B$
	$\psi'_{BA} : E'_B \to E'_B/\langle T_{BA}\rangle$ – an isogeny with the kernel $\langle T_{BA}\rangle$
	$\psi_{BA} : E_A \to E_A/\langle T'_{BA}\rangle$ – an isogeny with the kernel $\langle T'_{BA}\rangle$
	$E'_{BA} = \psi'_{BA}(E'_B); E_{BA} = \psi'_{AB}(E_A)$
	$(K, M) = \mathtt{KDF}(j(E'_{BA}) \parallel j(E_{BA}) \parallel \mathtt{ID}_A \parallel \mathtt{ID}_B[\parallel \mathtt{OI}])$
	If $tag_B \neq \mathtt{MAC}_M(h_2, \mathcal{K}_B, \mathcal{K}_A, \mathtt{ID}_B, \mathtt{ID}_A)$,
	terminates the session with an error
	$tag_A = \mathtt{MAC}_M(h_3, \mathcal{K}_A, \mathcal{K}_B, \mathtt{ID}_A, \mathtt{ID}_B)$
$A \to B$	tag_A
$B:$	If $tag_A \neq \mathtt{MAC}_M(h_3, \mathcal{K}_A, \mathcal{K}_B, \mathtt{ID}_A, \mathtt{ID}_B)$,
	terminates the session with an error

m_1, n_1 are chosen uniformly at random from the interval $[1, l_1^{e_1}]$, and are not both divisible by l. Given E_1 and images $\phi_1(R_2), \phi_1(S_2)$ of the points, find the generator of $\langle [m_1]R_1 + [n_1]S_1\rangle$.

Note that we choose the j-invariants of the elliptic curves resulting from two SIDH processes with different parameters and hash them together with a KDF function to obtain a shared secret value. Then, a naive deduction implies that an adversary has to solve two distinct instances of CSSI, which

should be twice as hard.

It is believed that the best classical algorithm [7] to attack the problem has the complexity $O(\sqrt[2]{l^{e_l}})$, where $l^{e_l} = \min(l_1^{e_1}, l_2^{e_2})$, while the claw-finding quantum algorithm [21] has the complexity $O(\sqrt[3]{l^{e_l}})$. Recent research [4,16] show that the actual quantum complexity of breaking the isogeny problem is estimated by $O(\sqrt[4]{p})$ operations, but we choose to be a little on a safe side. We discuss the choice of parameters in the following sections.

Note that the protocol inherits implicit key confirmation, KCI- and UKS-immunity from *Limonnik*-3 [14]. The protocol provides forward security against A, B (but not A AND B, since if long-term keys of the both parties are compromised, all the sessions involving them both are compromised, too; this property arises from the basic structure of the MTI/A0 protocol [19].

6. Security arguments

Consider the following computational problem [8, 22].

Problem 2. *Computational isogeny Diffie–Hellman, SSCDH: let* $\varphi_A\colon E_0 \to E_A$ *be an isogeny with kernel* $\langle P_A + [n_A]Q_A\rangle$, *and* $\varphi_B\colon E_0 \to E_B$ *be an isogeny with kernel* $\langle P_B + [n_B]Q_B\rangle$, *where* n_A *is chosen uniformly random from* $\mathbb{Z}/l_A^{e_A}\mathbb{Z}$ *and* n_B *is chosen uniformly random from* $\mathbb{Z}/l_B^{e_B}\mathbb{Z}$. *Given* E_A, E_B *and the images* $\varphi_A(P_B), \varphi_A(Q_B), \varphi_B(P_A), \varphi_B(Q_A)$, *find the* j*-invariant of the curve* $E_0/\langle P_A + [n_A]Q_A, P_B + [n_B]Q_B\rangle$.

The decisional version of the problem may be stated as follows.

Problem 3. *Decisional isogeny Diffie–Hellman, SSDDH: Given a tuple sampled with probability* 1/2 *from each of the following two distributions*

– $(E_A, E_B, \varphi_A(P_B), \varphi_A(Q_B), \varphi_B(P_A), \varphi_B(Q_A), E_{AB})$, *where* $(E_A, E_B, \varphi_A(P_B), \varphi_A(Q_B), \varphi_B(P_A), \varphi_B(Q_A)$ *are as before,*

$$E_{AB} \cong E_0/\langle P_A + Q_A, [m]P_B + [n]Q_B\rangle,$$

– $(E_A, E_B, \varphi_A(P_B), \varphi_A(Q_B), \varphi_B(P_A), \varphi_B(Q_A), E_C)$, *where* $(E_A, E_B, \varphi_A(P_B), \varphi_A(Q_B), \varphi_B(P_A), \varphi_B(Q_A)$ *are as before, and*

$$E_C \cong E_0/\langle P_A + [n']Q_A, P_B + [n']Q_B\rangle,$$

where m', n' *are chosen at random from* $\mathbb{Z}/l_B^{e_B}\mathbb{Z}$),

we have to determine from which distribution the tuple is sampled.

We cannot state an analogue of the Diffie–Hellman problem (GDHP) for the supersingular isogeny case, since decisional problems here are equivalent to computational. Thus, security arguments for *Limonnik*-3, proven secure under the GDHP hardness assumption, cannot be directly transformed for *Limonnitsa*. However, as pointed out by Galbraith, we may consider a weaker adversary model.

We state now a weaker version of the security definition adapted from [6]. We allow an adversary M to perform any of the following queries.

- *Initiate* a session between any chosen parties.
- *Send* messages from a party to another, which is followed by a correct (prescribed by the protocol) response.
- *Execute* a correct session between any chosen parties.
- *Corrupt* a party (that is, to learn any secret keys, as well as all generated shared keys and any local state information).

Note that M cannot perform any Reveal queries.

Define as $\Lambda(n)$ the set of all *Limonnitsa* public parameters for a chosen security parameter n: that is, all primes of an appropriate form with bit-lentgh n, all possible supersingular elliptic curves defined over the corresponding primes.

Definition 7. A key agreement protocol is said to be *weak-AKE-secure* if the following conditions hold:

1) If two honest parties complete matching sessions then, except with negligible probability, they both compute the same session key.
2) No polynomially bounded adversary M defined above can distinguish the session key of a fresh session from a randomly chosen session key with probability larger than $1/2$ plus a negligible fraction.

Then the following theorem holds.

Theorem. *Let the SSDDH problem for* Λ *be computationally hard. Let* `KDF` *be modelled by a pseudorandom function, let* `MAC` *be secure against forgery attack. Then Limonnitsa is secure in the sense of Definition* 7.

Proof. We define a protocol L-2 by removing from *Limonnitsa* computation and verification of tag_A, tag_B and replacing `KDF` by a hash-function H.

Our proof follows the method of [18], since L-2 is similar to the protocol KEA+.

L-2

$A:$	$k_A \in_R [1, 3^{e_{b3}}], S_{AB} = P_{B3} + [k_A]Q_{B3},$
	$\varphi_{AB} : E_B \to E'_A = E_B/\langle S_{AB}\rangle$ – an isogeny with the kernel $\langle S_{AB}\rangle$
	$E_{AB} = E_{B0}/\langle S_{AB}\rangle$ (that is, $E_{AB} = \varphi_{AB}(E_{B0})$)
	$\mathcal{K}_A = \{E'_A, \varphi_{AB}(P_{B2}), \varphi_{AB}(Q_{B2})\}$
$A \to B$	$\mathtt{ID}_A, \mathtt{Cert}_A, \mathcal{K}_A$
$B:$	$k_B \in_R [1, 3^{e_{a3}}], S_{BA} = P_{A3} + [k_B]Q_{A3},$
	$\varphi_{BA} : E_A \to E'_B = E_A/\langle S_{BA}\rangle$ – an isogeny with the kernel $\langle S_{BA}\rangle$
	$E_{BA} = E_{A0}/\langle S_{BA}\rangle$ (that is, $E_{BA} = \varphi_{BA}(E_{A0})$)
	$\mathcal{K}_B = \{E'_B, \varphi_{BA}(P_{A2}), \varphi_{BA}(Q_{A2})\}$
	$T_{AB} = P_A + [k_B]Q_A$
	$T'_{AB} = \varphi_{AB}(P_{B2}) + [s_B]\varphi_{AB}(Q_{B2})$
	$\psi_{AB} : E'_A \to E'_A/\langle T_{AB}\rangle$ – an isogeny with the kernel $\langle T_{AB}\rangle$
	$\psi'_{AB} : E_B \to E_B/\langle T'_{AB}\rangle$ – an isogeny with the kernel $\langle T'_{AB}\rangle$
	$E_{AB} = \psi_{AB}(E'_A); E'_{AB} = \psi'_{AB}(E_B)$
	$K = H(j(E_{AB}) \parallel j(E'_{AB}) \parallel \mathtt{ID}_A \parallel \mathtt{ID}_B)$
$B \to A$	$\mathtt{ID}_B, \mathtt{Cert}_B, \mathcal{K}_B$
$A:$	$T_{BA} = \varphi_{BA}(P_{A2}) + [s_A]\varphi_{BA}(Q_{A2})$
	$T'_{BA} = P_B + [k_A]Q_B$
	$\psi'_{BA} : E'_B \to E'_B/\langle T_{BA}\rangle$ – an isogeny with the kernel $\langle T_{BA}\rangle$
	$\psi_{BA} : E_A \to E_A/\langle T'_{BA}\rangle$ – an isogeny with the kernel $\langle T'_{BA}\rangle$
	$E'_{BA} = \psi'_{BA}(E'_B); E_{BA} = \psi'_{AB}(E_A)$
	$K = H(j(E'_{BA}) \parallel j(E_{BA}) \parallel \mathtt{ID}_A \parallel \mathtt{ID}_B)$

Define *signature* of a session

$$\Omega = (j(E_{AB}), j(E'_{AB}), \mathtt{ID}_A, \mathtt{ID}_B).$$

The secret key generated here is

$$K = H(j(E_{AB}), j(E'_{AB}), \mathtt{ID}_A, \mathtt{ID}_B).$$

Since H is modelled by a random function with a negligible collision probability, an adversary has the following abilities to determine the secret key:

1) run $H(SSCDH(S_A, \mathcal{K}_B), SSCDH(S_B, \mathcal{K}_A), \mathtt{ID}_A, \mathtt{ID}_B)$, that is, solve the SSCDH,

2) initiate a session with the same signature and reveal its secret key.

We dismiss the second possibility, since the sessions with identical signatures include the parties' identities, and it is obvious that the sessions keys are identical, and the sessions are identical, except for a negligible fraction defined by the probability of collision of H.

Now it suffices to show that if a polynomially bounded adversary $\mathfrak{M}$ can distinguish between a generated key and a random string with a non-negligible probability, then we can construct a polynomial algorithm S solving SSCDH with a non-negligible success probability.

Let S take input $E_A, E_B, \varphi_A(P_B), \varphi_A(Q_B), \varphi_B(P_A), \varphi_B(Q_A)$.

We suppose that S has access to an oracle for $SSDDH$. Suppose that S runs an experiment with n honest parties, each participating in at most k sessions, and with an adversary $\mathfrak{M}$. In this experiment S selects a random party A, sets its parameters $G_A := \{E_A, P_A, Q_A\}$ and static key $S_A := \{E_A, \varphi_A(P_A), \varphi_A(Q_A)\}$. Other parties set their parameters according to the protocol, choosing them at random from Λ. Then S chooses a random $u \in_R [1, \ldots, nk]$ and sets a counter $i := 1$.

During the experiment, S processes the queries of $\mathfrak{M}$ in the following manner.

1) A query to the oracle $H(v)$ is processed by a function $H_1(v)$ which we define below.

2) $Initiate(B, C, role)$, where B, C are two parties distinct from A, is processed according to the protocol L-2, except for the calculation of H, where H_1 is used.

3) $Initiate(A, C, role)$ cannot be processed according to the protocol L-2, since S cannot determine s_A.

   Thus, depending on the role of A, $role$, the algorithm S proceeds as follows.

   If A is the initiator, then S chooses $n \in_R \mathbb{Z}$, sends to C the set $\kappa = \{E'_A, \varphi_{AC}(P_{C2}), \varphi_{AC}(Q_{C2})\}$ and, having received the response ω, calculates the key $H_2(1, \omega, \kappa, \mathtt{ID}_A, \mathtt{ID}_C)$, where $H_2(i, \omega, \kappa, \mathtt{ID}_A, \mathtt{ID}_C)$ is defined below. If A is the responder, S waits for a request α, chooses $n_A \in_R \mathbb{Z}$ and calculates the key $H_2(2, \alpha, \kappa, \mathtt{ID}_C, \mathtt{ID}_A)$.

4) When processing $Initiate(B, A, role)$, S checks the counter i. If $i = u$, the session is marked as "special", the party B sends Y – the second parameter of our instance of $SSCDH$ – as its public key, and does not compute the secret key. If $i \neq u$, S increases the counter $i := i + 1$ and continues similarly to the step 2.

For the special session S stops the experiment.

5) To process $Corrupt(C)$, where C is a party distinct from A and B, S returns to $\mathfrak{M}$ the static private key s_C, as well as all keys generated by C during the experiment. $Corrupt(A)$ or $Corrupt(B)$ returns an error.

When $\mathfrak{M}$ terminates, S views all performed by $\mathfrak{M}$ requestrs to the oracle $H()$ and checks, where there is $SSCDH(X, Y)$ amongst them. Having detected such a value (by calling the $SSDDH$ oracle), S returns it as the solution to $SSCDH$. If this value is not detected, S terminates with an error.

Define now H_1 and H_2.

$H_1(Z_1, Z_2, B, C)$ simulates H over the correct signatures of L-2:

– if H_1 for given input is already computed, return it,

– otherwise, review the transcription of previous calls to $H_2(\cdot)$, and for each call $H_2(i, \omega, Z, B', C') = v$ check that

$$B = B', C = C', Z = Z_{3-i} \text{ and } SSDDH(X, \omega, Z_i) = 1.$$

If all conditions are satisfied, return v.

– If such v is not found, select a random w, $w \in \{0,1\}^\lambda$, remember that $H_1(Z_1, Z_2, B, C) := w$, and return w.

The function $H_2(i, \dots)$ implements the oracle over signatures, unknown to S: by the conditions of experiment, it takes on input (Z_1, Z_2, B, C) such that $Z_i = SSCDH(X, \omega)$ and $Z_{3-i} = Z$.

– If H_2 for given input is already computed, return it,

– otherwise, review the transcription of previous calls to $H_1(\cdot)$, and for each call $H_1(Z_1, Z_2, B', C') = v$ check that

$$B = B', C = C', Z = Z_{3-i} \text{ and } SSDDH(X, \omega, Z_i) = 1.$$

If all conditions are satisfied, return v.

– If such v is not found, select a random $w \in \{0,1\}^\lambda$, remember that $H_2(i, \omega, Z, B, C) := w$ and return w.

We show that the transcription of simulated experiment is distributed similarly to the transcription of experiment performed by $\mathfrak{M}$, except for a negligible fraction.

Indeed, the transcription of $\mathfrak{M}$ consists of all static public keys, static secret keys of corrupted parties, session keys and generated secret keys of corrupted parties, and the oracle's responses.

All the secret and corresponding public keys, except for s_A unknown to $\mathfrak{M}$, are distributed identically to the session of $\mathfrak{M}$. The probability that $\mathfrak{M}$ chooses the session performed by A as a matching session is at least $1/n^2t$.

In this case $\mathfrak{M}$ determines the key by providing to S the signature of a test session, containing $SSCDH(X, Y)$.

The time of S is defined by executing $\mathfrak{M}$ (in time t which is polynomial in λ) and processing $O(t^2)$ requests to the oracles, thus $\mathfrak{M}$ is also polynomial.

Thus, we have construct an algorithm which solves $SSCDH$ with the probability

$$\frac{1}{n^2k} Pr[Success(\mathfrak{M})],$$

where $Pr[Success(\mathfrak{M})]$ is the probability that $\mathfrak{M}$ breaks the weak AKE-security of L-2.

It follows now that *Limonnitsa* is weak-AKE-secure in the Universally Composable model [5]. That is, if $SSCDH$ for the family Λ is computationally hard, `MAC` is secure against forgery, and the hash function is modelled by a random oracle, the *Limonnitsa* is weak-AKE-secure. □

7. Preliminary choice of parameters

Consider primes of the form $p_A = 2^{e_2}3^{e_3} - 1$. In order to keep up the classic and quantum (see [9]) complexity with the standardized block cipher *Kuznyechik* [2], which has 128-bit block size and 256-bit keys, we choose the parameter p as the smallest prime of the form $p_A = 2^{e_2}3^{e_3} - 1$ such that $\log_2 p/6 \geqslant 128$ and the factors $2, 3$ are balanced: $e_2 \approx e_3/\log_2 3$. Thus we obtain the *Limonnitsa-prime* $p_\lambda = 2^{451}3^{284} - 1$, $\log_2 p_\lambda \approx 902$.

Note that SIKE NIST proposal [15], following NIST requirements and estimations of the quantum security of AES, provides a 751-bit prime for the same classical security level.

Limonnitsa allows for distinct parameters of the parties; this means that, for example, a party with a SIKE public parameters may run a *Limonnitsa* session with a Limonnitsa-prime-based party. The only practical problem here may be mutual public parameter verification.

Elliptic curve operations may be implemented by various techniques; for example, Montgomery or Edwards forms of an elliptic curve may be used.

The protocol execution takes (almost) exactly twice the complexity of executing a SIDH protocol with analogous choice of parameters. Its feasibility for embedded systems may be a subject of discussion as well as that of SIDH/SIKE.

8. Conclusion

We have proposed a post-quantum variant of the officially adopted key exchange protocol. We have studied its basic cryptographic properties. We have shown that the protocol is both classical and quantum secure and conforms to the cryptographic requirements.

However, implementation and efficiency issues of *Limonnitsa*, including parameters optimization for specific processors, as well as side-channel attack protection not yet investigated.

Acknowledgement: The author would like to thank anonymous reviewers for valuable comments which greatly helped to improve the protocol.

References

[1] *R* 1323565.1.004-2017. *Standardization recommendations. Key agreement schemes based upon public-key methods*, Standartinform, Moscow, 2017 (in Russian).

[2] *GOST R* 34.12-2015. *National standard of Russian Federation. Block ciphers*, Standartinform, Moscow, 2015 (in Russian).

[3] *GOST R* 34.13-2015. *National standard of Russian Federation. Block cipher modes*, Standartinform, Moscow, 2015 (in Russian).

[4] Biasse J.-F., Jao D., Sankar A., "A quantum algorithm for computing isogenies between supersingular elliptic curves", INDOCRYPT 2014, Lect. Notes Comput. Sci., **8885**, 2014, 428–442.

[5] Canetti R., Krawczyk H., "Analysis of key-exchange protocols and their use for building secure channels", EUROCRYPT 2001, Lect. Notes Comput. Sci., **2045**, 200, 453–474.

[6] Chatterjee S., Menezes A., Ustaoglu B., "A generic variant of NIST's KAS2 key agreement scheme", Proc. ACISP, Lect. Notes Comput. Sci., **6812**, 2011, 353–370.

[7] Costello C., Longa P., Naehrig M., Renes J., Virdia F., *Improved classical cryptanalysis of SIKE in practice*, Cryptology ePrint Archive, Report 2019/298, 2019.

[8] De Feo L., Jao D., Plût J., "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies", *J. Math. Cryptology*, **8(3)** (2014), 209-–247.

[9] Denisenko D., Marshalko G., Nikitenkova M., Rudskoy V., Shishkin V., "Estimation of Grover's algorithm implementation for searching GOST R 34.10-2015 block cipher keys", *J. Exp. Theor. Phys.*, **155**:4 (2019), 645–653 (in Russian).

[10] Diffie W., van Oorschot P., Wiener M., "Authentication and authenticated key exchanges", *Des., Codes Cryptogr.*, **2**, 107–125.

[11] Galbraith S., *Authenticated key exchange for SIDH*, Cryptology ePrint Archive, Report 2018/266, 2018.

[12] Galbraith S., Petit P., Silva J., *Schemes Based On Supersingular Isogeny Problems*, Cryptology ePrint Archive, Report 2016/1154, 2016.

[13] Galbraith S., Petit P., Shani B., Yan Bo Ti, *On the Security of Supersingular Isogeny Cryptosystem*, Cryptology ePrint Archive, Report 2016/859, 2016.

[14] Grebnev S., "Security properties of Limonnik-3", *Bezopasnost' Informacionnykh Tekhnologii*, **26**:2 (2019), 6–20 (in Russian).

[15] Jao D., Azarderakhsh R., Campagna M., Costello C., De Feo L., Hess B., Jalali A., Koziel B., LaMacchia B., Longa P., Naehrig M., Renes J., Soukharev V., Urbanik D., "Supersingular isogeny key encapsulation", 2017, https://sike.org/#nist-submission.

[16] Jaques S., Schanck J.M., *Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE*, Cryptology ePrint Archive, Report 2019/103, 2019.

[17] Kirkwood D., Lackey B.C., McVey J., Motley M., Solinas J.A., Tuller D., "Failure is not an option: Standardization issues for post-quantum key agreement", NIST Workshop on Cybersecurity in a Post-Quantum World, **2** (2015).

[18] Lauter K., Mityagin A., "Security analysis of KEA authenticated key exchange protocol", PKC 2006, Lect. Notes Comput. Sci., **3958**, 2006, 378–394.

[19] Matsumoto T., Takashima Y., Imai H., "On seeking smart public-key distribution systems", *Trans. IECE of Japan*, **E69(2)** (1986), 99–106.

[20] Matyukhin D., "On some properties of PKI-based key agreement schemes in the context of developing standardized solutions", *Obozr. Prikl. Promyshl. Mathem.*, **18** (2011), 793–794 (in Russian).

[21] Seiichiro T., *Claw finding algorithms using quantum walk*, http://arxiv.org/abs/0708.2584, 2008.

[22] Urbanik D., Jao D., *SoK: The problem landscape of SIDH*, Cryptology ePrint Archive, Reporrt 2018/336, 2018.

[23] Vélu J., "Isogenies entre courbes elliptiques", *C.R. Acad. Sci. Paris*, *Ser. A*, **273** (1971), 238–241.

A. Applying Kirkwood's trick

In this section we provide the complete description of the variant of the protocol obtained by applying the Kirkwood's trick to *Limonnitsa*, denoted *Limonnitsa+*.

Note that in this setting a party's ephemeral secret key is uncovered to another party, thus, it becomes the party's responsibility to generate unique value each time. Now many practical issues arise (for example, storing and searching through a database of any previously generated values in a secure manner may be too expensive). We propose to use a secure pseudorandom numbers generator instead.

Limonnitsa+

$A:$	$c_A \in_R \{0,1\}^*, k_A = H(s_A), S_{AB} = P_{B3} + [k_A]Q_{B3},$
	$\varphi_{AB} : E_B \to E_B/\langle S_{AB}\rangle$ – an isogeny with the kernel $\langle S_{AB}\rangle$
	$E_{AB} = E_{B0}/\langle S_{AB}\rangle$ (that is, $E_{AB} = \varphi_{AB}(E_{B0})$)
	$\mathcal{K}_A = \{E'_A, \varphi_{AB}(P_{B2}), \varphi_{AB}(Q_{B2})\}$ – A's ephemeral public key
$A \to B$	$\mathtt{ID}_A, \mathtt{Cert}_A, \mathcal{K}_A$
$B:$	$c_B \in_R \{0,1\}^*, k_B = H(c_B), S_{BA} = P_{A3} + [k_B]Q_{A3},$
	$\varphi_{BA} : E_A \to E'_B/\langle S_{BA}\rangle$ – an isogeny with the kernel $\langle S_{BA}\rangle$
	$E_{BA} = E_{A0}/\langle S_{BA}\rangle$ (that is, $E_{BA} = \varphi_{BA}(E_{A0})$)
	$\mathcal{K}_B = \{E'_B, \varphi_{BA}(P_{A2}), \varphi_{BA}(Q_{A2})\}$ – B's session public key
	$T_{AB} = P_A + [k_B]Q_A$
	$T'_{AB} = \varphi_{AB}(P_{B2}) + [s_B]\varphi_{AB}(Q_{B2})$
	$\psi_{AB} : E'_A \to E'_A/\langle T_{AB}\rangle$ – an isogeny with the kernel $\langle T_{AB}\rangle$
	$\psi'_{AB} : E_B \to E_B/\langle T'_{AB}\rangle$ – an isogeny with the kernel $\langle T'_{AB}\rangle$
	$E_{AB} = \psi_{AB}(E'_A); E'_{AB} = \psi'_{AB}(E_B)$
	$(K, M) = \mathtt{KDF}(j(E_{AB}) \parallel j(E'_{AB}) \parallel \mathtt{ID}_A \parallel \mathtt{ID}_B[\parallel \mathtt{OI}])$
	$tag_B = \mathtt{ENC}_M(h_2, c_B, \mathcal{K}_B, \mathcal{K}_A, \mathtt{ID}_B, \mathtt{ID}_A)$
$B \to A$	$\mathtt{ID}_B, \mathtt{Cert}_B, \mathcal{K}_B, tag_B$
$A:$	$T_{BA} = \varphi_{BA}(P_{A2}) + [s_A]\varphi_{BA}(Q_{A2})$
	$T'_{BA} = P_B + [k_A]Q_B$
	$\psi'_{BA} : E'_B \to E'_B/\langle T_{BA}\rangle$ – an isogeny with the kernel $\langle T_{BA}\rangle$
	$\psi_{BA} : E_A \to E_A/\langle T'_{BA}\rangle$ – an isogeny with the kernel $\langle T'_{BA}\rangle$
	$E'_{BA} = \psi'_{BA}(E'_B); E_{BA} = \psi'_{AB}(E_A)$
	$(K, M) = \mathtt{KDF}(j(E'_{BA}) \parallel j(E_{BA}) \parallel \mathtt{ID}_A \parallel \mathtt{ID}_B[\parallel \mathtt{OI}])$
	recovers A's initial value as $c'_B, k'_B = H(c'_B)$, computes φ'_{BA}
	– an isogeny with the kernel $P_{A3} + [k'_B]Q_{A3}$,
	if $\varphi'_{BA}(P_A 2) \neq \varphi_{BA}(P_A 2)$ or $\varphi'_{BA}(Q_A 2) \neq \varphi_{BA}(Q_A 2)$,
	sets $(K, M) = \mathtt{KDF}(j(E'_{BA}) \parallel n \parallel \mathtt{ID}_A \parallel \mathtt{ID}_B[\parallel \mathtt{OI}]), n \in_R [1, p_B^2]$
	$tag_A = \mathtt{ENC}_M(h_3, c_A, \mathcal{K}_A, \mathcal{K}_B, \mathtt{ID}_A, \mathtt{ID}_B)$
$A \to B$	tag_A
$B:$	decrypts $c_A, k_A = H(c_A)$, computes $\varphi' AB$
	– an isogeny with the kernel $P_{B3} + [k'_B]Q_{B3}$,
	if $\varphi'_{BA}(P_A 2) \neq \varphi_{BA}(P_A 2)$ or $\varphi'_{BA}(Q_A 2) \neq \varphi_{BA}(Q_A 2)$,
	sets $(K, M) = \mathtt{KDF}(n \parallel j(E'_{AB}) \parallel \mathtt{ID}_A \parallel \mathtt{ID}_B[\parallel \mathtt{OI}]), n \in_R [1, p_A^2]$