www.czasopisma.pan.pl PAN www.journals.pan.pl

jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 3, PP. 431-441
Manuscript received February 18, 2020; revised July, 2020. DOI: 10.24425/ijet.2020.131896

SIDH Hybrid Schemes with Classical Component Based on the Discrete Logarithm Problem over Finite Field Extension

Michał Wroński, Elżbieta Burek, and Łukasz Dzierzkowski

***Abstract*—The concept of a hybrid scheme with connection of SIDH and ECDH is nowadays very popular. In hardware implementations it is convenient to use a classical key exchange algorithm, which is based on the same finite field as SIDH. Most frequently used hybrid scheme is SIDH-ECDH. On the other hand, using the same field as in SIDH, one can construct schemes over $\mathbb{F}_{p^n}$, like Diffie-Hellman or XTR scheme, whose security is based on the discrete logarithm problem.**

In this paper, idea of such schemes will be presented. The security of schemes, which are based on the discrete logarithm problem over fields $\mathbb{F}_p, \mathbb{F}_{p^2}, \mathbb{F}_{p^4}, \mathbb{F}_{p^6}$ and $\mathbb{F}_{p^8}$, for primes p used in SIDH, will be analyzed. At the end, the propositions of practical applications of these schemes will be presented.

***Keywords*—SIDH, Diffie-Hellman algorithm, Hybrid schemes**

I. Introduction

HYBRID scheme, using post-quantum and classical primitives, is a popular conception in the cryptographic society nowadays. In hybrid schemes a primitive $\mathcal{P}$, which is a longstanding classically-secure, is partnered alongside with a post-quantum primitive $\mathcal{Q}$. It is worth to note, that primitives, that are believed nowadays to be quantumly secure, have not been sufficiently studied yet, so it is possible, that some quantum or even classical attacks on the component $\mathcal{Q}$ will be found. Some protection against such possibility is the use of hybrid schemes, because if there will be found any attack for the component $\mathcal{Q}$ using classical computers, the hybrid scheme $\mathcal{P}+\mathcal{Q}$ is likely to still remain secure, until proper quantum computer will be invented. In such case, it will be possible to change broken primitive $\mathcal{Q}$ to the new one, which will be believed to be secure. On the other hand, if proper quantum computer will be built, then classically-secure primitive $\mathcal{P}$ may be broken by some quantum attack, but the hybrid scheme, if any new attack on the post-quantum component $\mathcal{Q}$ will not be found, will be still secure.

M. Wroński is with Institute of Mathematics and Cryptology, Faculty of Cybernetics, Military University of Technology, Warsaw, Poland (e-mail: michal.wronski@wat.edu.pl).

E. Burek is with Institute of Mathematics and Cryptology, Faculty of Cybernetics, Military University of Technology, Warsaw, Poland (e-mail: elzbieta.burek@wat.edu.pl).

Ł. Dzierzkowski is with Institute of Mathematics and Cryptology, Faculty of Cybernetics, Military University of Technology, Warsaw, Poland (e-mail: lukasz.dzierzkowski@student.wat.edu.pl).

Very convenient for hardware implementations of such hybrid schemes is the Supersingular Isogeny Diffie-Hellman (SIDH) algorithm, which is based on a supersingular elliptic curve arithmetic. It is also possible to use the same prime p as in SIDH, to construct an ordinary elliptic curve over $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$ and use the Elliptic Curve Diffie-Hellman (ECDH) cryptosystem as a classical component of such hybrid scheme.

In this article, hardware applications of hybrid cryptosystems are considered. Moreover, it is assumed that parallel $\mathbb{F}_{p^2}$ arithmetic, which is required for SIDH algorithm, is implemented.

In such context, the possibility of use of ECDH over $\mathbb{F}_p$, together with SIDH was firstly posted by Costello, Longa and Naehrig in [1]. Usage of ECDH over $\mathbb{F}_{p^2}$, where the GLS method may be used, was proposed by Wroński, Kijko and Dryło in [2]. However, these solutions should be used for a high or very high level of security.

In applications where low level of security is sufficient, the alternative solutions should be considered, because ECDH is not the most efficient cryptosystem. From this point of view, using in a hybrid scheme protocols based on the discrete logarithm problem and defined over extension fields, for characteristic p the same as used in SIDH algorithm, may be a reasonable choice, assuming a reasonable level of classical security.

II. Classical key-exchange protocols most suitable for the hybrid SIDH scheme

In this section, key-exchange protocols will be shortly described, that are considered for the hybrid SIDH scheme.

A. Diffie-Hellman protocol

Diffie-Hellman (DH) algorithm was the first publicly presented key agreement protocol. The purpose of this protocol is to establish a symmetric key for two parties over an insecure channel. The security of the Diffie-Hellman protocol is based on the difficulty of computation of discrete logarithm over finite fields.

For this protocol, there are two public key parameters:

- p - a large prime number,
- g - a primitive root of 1, whose order is equal to r (usually called a generator).

The Diffie Hellman key-exchange scheme can be described in the following steps:

- **Step 1:** Alice generates a random secret number $1 < a < p-1$, calculates $A = g^a$ and sends A to Bob.
- **Step 2:** Bob also generates a random secret number $1 < b < p-1$, calculates $B = g^b$ and sends B to Alice.
- **Step 3:** Alice, after receiving Bob's public key B, verifies that the order of B is r, by calculating B^r. If the result is equal to 1, then Bob's public key B is correct.
- **Step 4:** Alice calculates $S = B^a = g^{ab}$, where S is a shared secret key.
- **Step 5:** Bob, after receiving Alice's public key A, verifies that the order of A is r, by calculating A^r. If the result is equal to 1, then Alice's public key A is correct.
- **Step 6:** Bob calculates $S = A^b = g^{ab}$, where S is a shared secret key.

A step which ensures the security of the protocol is a verification of received public key. As described above, in the Diffie-Hellman protocol receiver verifies the order of received public key, by raising it to a power equal to suitable order. This verification requires additional computations, that increases the computational cost of this protocol.

B. ECDH protocol

Elliptic Curve Diffie-Hellman protocol allows two communicating parties over an insecure channel to agree a symetric key. ECDH is based on the discrete logarithm problem over the group of points of an elliptic curve. Let $E/\mathbb{F}_q$ be an elliptic curve over a finite field $\mathbb{F}_q$. The public key parameters, that are assumed to be publicly known, are following:

- elliptic curve parameters, that define the elliptic curve E,
- $G = (x_G, y_G) \in E(\mathbb{F}_q)$, which is a generator of subgroup of large prime order r.

The Elliptic Curve Diffie-Hellman key-exchange scheme consists of the following steps:

- **Step 1:** Alice generates a random secret number $1 < a < r-1$, calculates point $A = [a]\,G$ and sends A to Bob.
- **Step 2:** Bob also generates a random secret number $1 < b < r-1$, calculates point $B = [b]\,G$ and sends B to Alice.
- **Step 3:** Alice, after receiving Bob's public key B, verifies that B is correct.
- **Step 4:** Alice calculates $S = [a]\,B = [ab]\,G$, where S is a shared secret key.
- **Step 5:** Bob, after receiving Alice's public key A, verifies that A is correct.
- **Step 6:** Bob calculates $S = [b]\,A = [ab]\,G$, where S is a shared secret key.

The verification of a received public key depends on the order $\#E(\mathbb{F}_q)$ of the elliptic curve and is a protection against small subgroup attacks.

- If the order of elliptic curve r is a prime number, then it is enough to verify that the received point $P \in E(\mathbb{F}_q)$.
- If the order of elliptic curve is equal to rh, where h is small integer, then it is enough to multiply the received point by h and next, multiply such result by the secret key.
- If the order of elliptic curve is equal to rh, where h is large integer, then the received point is multiplied by r, where r is a required order of point. If the result is the point at infinity, then the received point is proper.

C. SIDH protocol

Supersingular Isogeny Diffie-Hellman protocol is based on an elliptic curve cryptography. This protocol uses supersingular elliptic curves, which are rarely used in the traditional elliptic curve cryptography. The security of SIDH protocol is based on the difficulty of finding the isogeny of high order between two supersingular elliptic curves.
The public parameters of SIDH protocol are:

- l_A, l_B and $p = l_A^{e_A} l_B^{e_B} \mp 1$ - prime numbers,
- $E(\mathbb{F}_{p^2})$ - a supersingular elliptic curve over a finite field $\mathbb{F}_{p^2}$, with order equal to $(p \pm 1)^2$,
- (P_A, Q_A) - a base points in $E\,[l_A^{e_A}]$, where $E\,[l_A^{e_A}]$ denotes the subgroup of $l_A^{e_A}$ torsion points,
- (P_B, Q_B) - a base points in $E\,[l_B^{e_B}]$, where $E\,[l_B^{e_B}]$ denotes the subgroup of $l_B^{e_B}$ torsion points.

The Supersingular Isogeny Diffie-Hellman key-exchange scheme can be realized in the following way [3]:

- **Step 1:** Alice generates a random secret elements $0 < a_1, a_2 < l_A^{e_A}$ and computes the point $A = [a_1]\,P_A + [a_2]\,Q_A$, secret isogeny $\alpha : E \to E_A = E/\langle A \rangle$ and sends E_A, and points $\alpha(P_B)$, $\alpha(Q_B)$ to Bob.
- **Step 2:** Bob generates a random secret elements $0 < b_1, b_2 < l_B^{e_B}$ and computes the point $B = [b_1]\,P_B + [b_2]\,Q_B$, secret isogeny $\beta : E \to E_B = E/\langle B \rangle$ and sends E_B, and points $\beta(P_A)$, $\beta(Q_A)$ to Alice.
- **Step 3:** Alice receives Bob's public key and verifies that:
  - received curve is a supersingular elliptic curve,
  - received curve has a proper order,
  - the points $\beta(P_A)$, $\beta(Q_A)$ are from different torsion subgroups.
- **Step 4:** Alice calculates $E/\langle A, B \rangle = E_B/\langle \beta(A) \rangle$. The *j - invariant* of $E/\langle A, B \rangle$ is the shared secret key.
- **Step 5:** Bob receives Alice's public key and verifies that:
  - received curve is a supersingular elliptic curve,
  - received curve has proper order,
  - the points $\alpha(P_B)$, $\alpha(Q_B)$ are from different torsion subgroups.
- **Step 6:** Bob calculates $E/\langle A, B \rangle = E_A/\langle \alpha(B) \rangle$. The *j - invariant* of $E/\langle A, B \rangle$ is the shared secret key.

III. Security

In this section there will be shortly described security of hybrid schemes, SIDH, DH and ECDH cryptosystems.

A. Security of components of hybrid scheme

In this article the same methodology is used as was chosen in NIST PQC:

- it is assumed that algorithm $\mathcal{R}$ ensures a classical security CS at the n-bit level ($CS(\mathcal{R}) = n$) if the fastest known classical attack on this algorithm is not faster than the fastest known classical attack on AES-n,
- in the same way, it is assumed, that an algorithm ensures quantum security QS at the n-qubit level, if the fastest known quantum attack on this algorithm is not faster than the fastest known quantum attack on AES-n.

In the case of hybrid algorithms, one will say that hybrid algorithm has security HS at the level n if $HS(\mathcal{Q},\mathcal{P}) = n = \min\{\min\{CS(\mathcal{P}), CS(\mathcal{Q})\}, QS(\mathcal{Q})\}$. To fully use the power of post-quantum algorithm, also should hold $CS(\mathcal{P}) \geq CS(\mathcal{Q})$ and $CS(\mathcal{P}) \geq QS(\mathcal{Q})$.
Unfortunately, using the last condition ($CS(\mathcal{P}) \geq QS(\mathcal{Q})$), it is easy to see, that usage of classical components based on discrete logarithm problem over $\mathbb{F}_{p^n}$ in the hybrid $SIDH$ scheme will make sense iff a degree n of the field extension will be high or the security level of $HS(\mathcal{Q},\mathcal{P})$ will be low.

B. Security of SIDH algorithm

SIDH is based on the problem of searching an isogeny of given degree between two supersingular elliptic curves over $\mathbb{F}_{p^2}$. Firstly, it seemed that the best known algorithms let to break SIDH in $O\left(\sqrt[4]{p}\right)$ operations on a classical computer. It was believed that Tani's claw finding algorithm requires $O\left(\sqrt[6]{p}\right)$ operations on a quantum computer. However, the analysis by Jaques and Schanck [4] showed, that Tani's algorithm has a real complexity equal to $O\left(\sqrt[3]{p}\right)$. Moreover, Jaques and Schanck made a non-asymptotical analysis, by which it is possible to estimate the cost of quantum and classical attacks on SIDH as Tani's quantum claw finding, direct application of Grover's algorithm to claw finding attack and (classical) attack of van Oorschot and Wiener. Using these ideas and others, presented in chapter 5 of SIKE submission supporting documentation [5], the security levels of SIDH algorithm for primes from 120 to 550 bits were estimated by us, both for classical and quantum attacks.

C. Attacks on the Discrete Logarithm Problem over a finite field

The integer factorization problem and the discrete logarithm problem are dominant problems in public-key cryptosystems. Let $\mathbb{F}_q$ be a finite field, $g \in \mathbb{F}_q$ be a primitive element of $\mathbb{F}_q$ and $w \in \mathbb{F}_q$ be a non-zero element of the finite field $\mathbb{F}_q$. The discrete logarithm of w in the base g, defined as $\log_g(w)$, is the least non-negative integer n such that $w = g^n$.

1) The Pohlig - Hellman algorithm: Pohlig and Hellman noted that to solve the discrete logarithm problem for a finite group G it is enough to solve it for subgroups in G, whose order is the power of the prime number. The Pohlig - Hellman algorithm sequence is described below [6]. Let G be a group, and $g \in G$ be an element of this group G. Let's assume, that the order of g is equal to r, where $r = r_1 r_2$, and $gcd(r_1, r_2) = 1$. Then the cyclic groups $\langle g^{r_1}\rangle$ and $\langle g^{r_2}\rangle$ of order r_1 and r_2, respectively, form a cyclic group $\langle g\rangle$. Using the Chinese Remainder Theorem, it is possible to compute the $\log_g(w) \equiv b\cdot x\cdot r_1 + a\cdot y\cdot r_2 \ (mod\ r)$, where $b = \log_{g^{r_1}}(w^{r_1})$ and $a = \log_{g^{r_2}}(w^{r_2})$. The Euclidean GCD algorithm is used to compute x and y.
Suppose, that the order of g is a prime power, for example $r = p^k$. Then the determination of $log_g(w)$ is reduced to k calculations of a discrete logarithm, in a cyclic group of p elements.
The consequence of developing this algorithm is the requirement, that the order of the group G must contain a large prime factor.

2) The Pollard's-rho algorithm: Pollard developed two randomized algorithms to calculate a discrete logarithm in any group: the rho method and the lambda (kangaroo) method.
The rho method is based on a birthday paradox, that assumes that if one uses a random walk for a graph of r vertices, with a cycle, it is very likely that the same vertex will be visited again after about $\sqrt{r}$ steps. The algorithm sequence is presented below [7].
Let $\langle g\rangle$ be a cyclic group of order r, which is divided into three sets, G_1, G_2, G_3, of similar size. To compute $\log_g(h)$, it is necessary to define a sequence w_0, w_1, ..., where $w_0 = g$ and for $i > 0$ Pollard's iteration function $f_P : G \to G$ is defined as follows:

$$f_P(w_i) \equiv \begin{cases} g \cdot w_i & (mod\ r) \quad if\ w_i \in G_1, \\ w_i^2 & (mod\ r) \quad if\ w_i \in G_2, \\ h \cdot w_i & (mod\ r) \quad if\ w_i \in G_3, \end{cases}$$

where $f_P(w_i) = w_{i+1}$.
Each $w_i = g^{a_i} h^{b_i}$, for some integers a_i and b_i. If the transition from w_i to w_{i+1} behaves like a random walk, then after $O(\sqrt{r})$ steps one will find i, for which $w_i = w_{2i}$ and then: $a_i + b_i \log_g(h) \equiv a_{2i} + b_{2i} \log_g(h)\ mod\ r$.

3) The Pollard's-lambda algorithm: The Pollard's-lambda algorithm [8] is a random algorithm, that was developed to solve a DLP. Like the Pollard's-rho algorithm, the Pollard's-lambda algorithm uses a random walk, but jumps are smaller. In the primary version of this method, there are two kangaroos. The first kangaroo (a tame kangaroo) starts at point $g^{N/2}$, where N is some interval and N is less than the order of g. This kangaroo jumps towards the right using the random walk. The second kangaroo (a wild kangaroo) starts at the point w and jumps in the same way as the first kangaroo. For the first kangaroo one stores such h that $z = g^h$, where z is the current group element and for the second kangaroo one stores such h that $z = wg^h$. The distinguished elements of the group, together with the flag indicating which kangaroo it concerns, are stored in a binary tree, hash table or list. When the same element of group is visited twice, by the both kangaroos, the DLP is solved.
The Pollard's-lambda algorithm requires $O(\sqrt{r})$ group opera-

tions. However, Pollard's-lambda was developed for searching a discrete logarithm in the given interval, nowadays the fastest algorithm for this purpose is Gaudry-Schost algorithm with improvements of Ruprai [9].

D. Security of the ECDH algorithm

ECDH is based on the discrete logarithm problem over the group of points on an elliptic curve (ECDLP). Security of ECDH protocol depends on the choice of public key parameters.

Let $E/\mathbb{F}_q$ be an elliptic curve defined over finite field $\mathbb{F}_q$. For given points $P, Q \in E(\mathbb{F}_q)$, the ECDLP problem is to find an integer k, for which $Q = [k]\,P$.

The best known generic attack to break the ECDLP is a Pollard's-rho algorithm. Additionally, such attacks as Pohlig-Hellman reduction, MOV attack, anomalous attack, or fault attack can be used to solve the ECDLP problem for special elliptic curves.

1) MOV attack: The MOV attack uses the Weil pairing to reduce the discrete logarithm problem on an elliptic curve $E/\mathbb{F}_q$ to the discrete logarithm problem in a multiplicative group $\mathbb{F}^*_{q^m}$. Then such problem may be solved by the sieve attack, if m is small.

Let $P, Q \in E(\mathbb{F}_q)$ and $ord(P) = N$. If $gcd(N, q) = 1$, then we have $E\,[N] \cong \mathbb{Z}_N \oplus \mathbb{Z}_N$. One can select the point R such that $\{P, R\}$ is a basis of $E\,[N]$, and then $Q = [a]\,P + [b]\,R$. The Weil pairing is a map $e_N : E\,[N] \times E\,[N] \to \mu_N$, defined as $e_N(P, R) = \zeta$, where ζ is a primitive Nth root of unity.

In MOV attack one selects such m that $E\,[N] \subset E(\mathbb{F}_{q^m})$ and then for all primitive roots of unity ζ, one has $\mu_N \subset E(\mathbb{F}_{q^m})$. Then it is necessary to choose a random point S_i and compute its order $M_i = ord(S_i)$. There also should be defined $d_i = gcd(M_i, N)$ and the point $T_i = (M_i/d_i)S_i$. For the received point T_i, one computes: $\zeta_{1i} = e_N(P, T_i)$ and $\zeta_{2i} = e_N(Q, T_i)$. Basing on the ζ_{1i} and ζ_{2i}, one can solve the discrete logarithm problem $\zeta_{1i}^{k_i} = \zeta_{2i}$ in $\mathbb{F}^*_{q^m}$. The result is $k_i\ (mod\ d_i)$. The above calculations are repeated for subsequent iterations i, until $lcm(d_1, d_2, ..., d_k) = N$.

It is necessary to use the $k_i\ (mod\ d_i)$ values to find a $k\ (mod\ N)$.

If the order of the elliptic curve is equal to $\#E(\mathbb{F}_q) = h \cdot r$, where r is the order of generator, then the MOV attack is executed for the smallest number of m, for which $q^m \equiv 1\ (mod\ r)$. For the currently used key sizes, it is sufficient to verify whether the above equation is not satisfied for $m \leq 30$ [10].

2) Anomalous attack: The set of elliptic cures for which the discrete logarithm problem is very simple are anomalous elliptic curves. Anomalous curve is the elliptic curve, whose order is equal to the size of the finite field $\#E(\mathbb{F}_q) = q$ (the trace of Frobenius is equal to one). The anomalous attack sequence is following [11].

Let $E/\mathbb{F}_q$ be an anomalous elliptic curve and let $\overline{P}, \overline{Q} \in E(\mathbb{F}_q)$. The algorithm to find m such that $\overline{Q} = [m]\,\overline{P}$ will be described. Firstly, one calculates the lifts $P, Q \in E(\mathbb{Q}_q)$ of the points $\overline{P}, \overline{Q}$. The field $\mathbb{Q}_q$ is a set of q-adric numbers, where nonzero q-adric number n is defined as $n = q^{\alpha}(\sum_{i=0}^{\infty} n_i q^i)$, where $\alpha \in \mathbb{Z}$, $n_i \in \{0, ..., q-1\}$ and $n_0 \neq 0$. In addition, let's define $ord_q(n) = \alpha$ and $|n|_q = q^{-\alpha}$. Let $E/\mathbb{Q}_q$ be an elliptic curve over field of q-adric numbers. One defines:

- $E_1(\mathbb{Q}_q)$ is a set of points in $E(\mathbb{Q}_q)$ that reduce modulo q to the point at infinity: $E_1(\mathbb{Q}_q) = \{P \in E(\mathbb{Q}_q) | \overline{P} = \mathcal{O}\}$.
- $E_0(\mathbb{Q}_q)$ is a set of points in $E(\mathbb{Q}_q)$ that are reduced modulo q to the points in $E(\mathbb{F}_q)$.
- $E_2(\mathbb{Q}_q)$ is a set of points in $E(\mathbb{Q}_q)$ of the form: $E_2(\mathbb{Q}_q) = \{P \in E(\mathbb{Q}_q) : ord_q(P_x) \leq -4\} \cup \{\mathcal{O}\}$, where P_x denotes the x-coordinate of P.

Then one has $Q - [m]\,P = R \in E_1(\mathbb{Q}_q)$. Let's note that $E_0(\mathbb{Q}_q)/E_1(\mathbb{Q}_q) \cong E(\mathbb{F}_q)$ and also that $E_1(\mathbb{Q}_q)/E_2(\mathbb{Q}_q) \cong \mathbb{F}_q$. From this fact and since $\#E(\mathbb{F}_q) = \#\mathbb{F}_q = q$ it is clear that one has $[q]\,Q - [m]\,([q]\,P) = [p]\,R \in E_2(\mathbb{Q}_q)$. By computing the q-adric logarithm for both sides, one obtains an equation $\vartheta_q([q]\,Q) - m\vartheta_q([q]\,P) = \vartheta_q([p]\,R) \equiv 0(mod\ q^2)$. The value m can be determined from the congruence $m \equiv \dfrac{\vartheta_q([q]\,Q)}{\vartheta_q([q]\,P)}\ (mod\ q)$.

E. General security of algorithms based on discrete logarithm problem over finite fields

Diffie-Hellman scheme is based on the discrete logarithm problem. There are many different attacks on discrete logarithm in finite fields, but the most powerful are sieve methods, which are insensitive for the size of subgroup generated by generator g and depends only on the size of the field. The second important and powerful attack is the Pollard's-rho method, which is sensitive to size of a subgroup generated by the generator g.

If $Ord(g) = r$, where r is largest prime, for which $r|Ord(g)$, then the complexity of Pollard's-rho method is equal to $O(\sqrt{r})$. The complexity of sieves methods for discrete logarithm defined over field $\mathbb{F}_q$, where $q = p^n$, depends on the form of p and n, but all such attacks have subexponential complexity equal to $L_{p^n}\,[1/3, c]$.

In [12] it was presented that:

1) for $\mathbb{F}_p$ fields, for primes p of the general form, the fastest known attack has complexity equal to $L_p\left[1/3, \sqrt[3]{64/9}\right]$;
2) for $\mathbb{F}_{p^n}$ fields, for primes p of the general form and extensions $n \in \{2, 3\}$, the fastest known attack has complexity equal to $L_p\left[1/3, \sqrt[3]{64/9}\right]$;
3) for $\mathbb{F}_{p^n}$ fields, for primes p of the general form and medium characteristic, the fastest known attack has complexity equal to $L_{p^n}\left[1/3, \sqrt[3]{48/9}\right]$;
4) for $\mathbb{F}_{p^n}$ fields, for general n and primes p of special form (vulnerable to variants of the Special Number Field Sieve attack), the fastest known attack has a complexity equal to $L_{p^n}\left[1/3, \sqrt[3]{32/9}\right]$ (it is the case of primes used in the SIDH algorithm).

The primes used in SIDH has a form $p = fa^m b^n \pm 1$, where $a^m \approx b^n$ and the most frequently $a = 2$, $b = 3$, $f = 1$.

Such primes are vulnerable to variants of SNFS attack, because it is easy to present them (or their small multiplicities) as irreducible polynomial of small degree $d \in \{3,4,5,6\}$, with small coefficients.

For example, number $p = 2^{128} \cdot 3^{81} - 1$ (or its small multiplicity), may be presented as:

- $p = 2^3 \cdot 3 \cdot (2^{25}3^{16})^5 - 1 = 24x^5 - 1$, where $x = 2^{25}3^{16}$;
- $p = 3 \cdot (2^{32}3^{20})^4 - 1 = 3x^4 - 1$, where $x = 2^{32}3^{20}$;
- $3^3 \cdot p = 2^2 \cdot (2^{21}3^{14})^6 - 27 = 4x^6 - 27$, where $x = 2^{21}3^{14}$;
- $2 \cdot p = (2^{43}3^{27})^3 - 2 = x^3 - 2$, where $x = 2^{43}3^{27}$.

In every case the result is an irreducible polynomial of a small degree and with a small coefficients.

IV. Security of discrete logarithm problem over prime field $\mathbb{F}_p$ and its small extensions for SIDH primes

A. Special attacks on discrete logarithms over SIDH primes

Let's note that the small subgroups attack is possible for primes of the form $p = fa^m b^n \pm 1$ over fields $\mathbb{F}_{p^n}$. The example for prime $p = fa^m b^n - 1$ and $\mathbb{F}_{p^2}$ field, generated by the irreducible polynomial $f(t)$ will be shown. Let's assume that $g \in \mathbb{F}_{p^2} \backslash \mathbb{F}_p$ and $Ord(g) = p^2 - 1$. Then, if $g^k \equiv h (mod\ f(t))$ and k is the secret key, then $g'^k \equiv \left(g^{p-1}\right)^k \equiv h^{p-1} \equiv h'(mod\ f(t))$, so $Ord(g') = p+1, Ord(h')|p+1$ and, because $p + 1 = fa^m b^n$, Pohlig-Hellman reduction is possible.

To prevent this attack, every element g should be firstly checked if it has a large prime order r, where $r|p-1$, by exponentiation g^r. If $g^r \equiv 1 (mod\ f(t))$, then $Ord(g) = r$ and element g is proper.

B. Security of DH scheme over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$

In general in SIDH, $p = fa^m b^n \pm 1$. Let's note, that for DH over $\mathbb{F}_p$ primes of the form $p = fa^m b^n + 1$ are not allowed, because in this case $p - 1 = fa^m b^n$ and $p - 1$ does not have large prime factors. However, for DH over $\mathbb{F}_{p^2}$, such primes may be used in some situations, because the order of the multiplicative group $\mathbb{F}^*_{p^2}$ is equal to $p^2 - 1$ and then there exist elements, whose order is a factor of $p + 1$, which may be large prime.

Primes of the form of $p = fa^m b^n - 1$ may be used (if $p - 1$ has a large prime factor) for DH scheme over $\mathbb{F}_p$, but should not be used for the DH scheme over $\mathbb{F}_{p^2}$. However, elements of an order r, where $r|p-1$ may be found both over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$, but every element $g \in \mathbb{F}_{p^2}$ has order h, for which $GCD(h, p+1) > 1$ and in this case it is easy to transform the discrete logarithm problem over $\mathbb{F}_{p^2}$ to the discrete logarithm problem over $\mathbb{F}_p$. Moreover, because $p+1 = fa^m b^n$, a Pohlig-Hellman reduction and a small subgroup attack are possible. The security of DH scheme over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$ for SIDH primes is presented in the Figure 1 and the Figure 2.

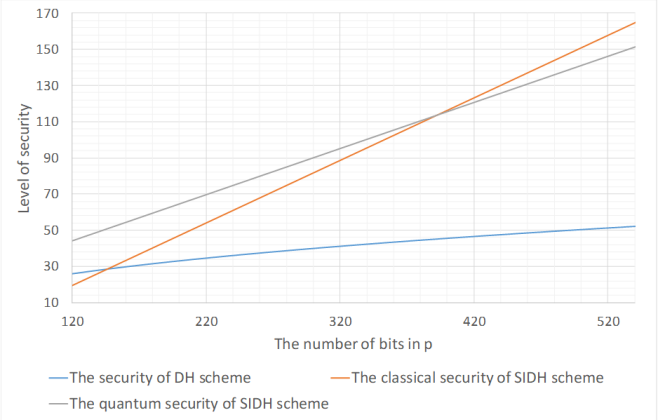

Fig. 1. Security of DH scheme over $\mathbb{F}_p$ for SIDH primes

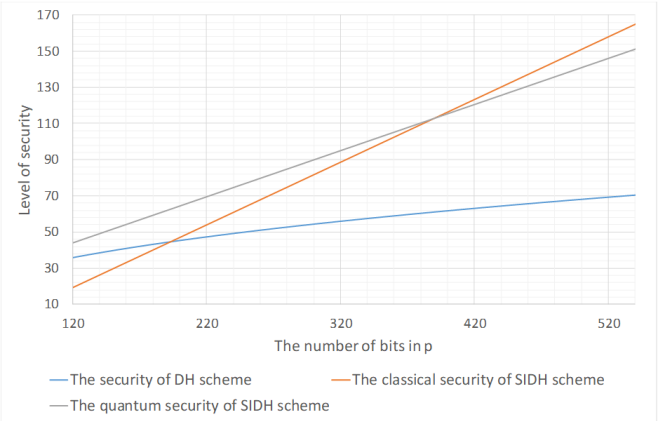

Fig. 2. Security of DH scheme over $\mathbb{F}_{p^2}$ for SIDH primes

C. Problem of factorisation of integers of the form $p^2 - 1$, where $p = f\ell_1^m \ell_2^n + 1$

To check if given field $\mathbb{F}_{p^2}$ has elements of proper order, the factorisation of $p \pm 1$ must be performed.

Factorisation of integers is computationally hard and no classical polynomial time algorithm is known (however, there is of course quantum Shor's algorithm [13] which has a polynomial time complexity). The fastest algorithm for factorisation of large integers has subexponential complexity.

In fact, there are several methods which allow to speed-up the factorisation of the number of a form $p \pm 1$ for $p = f\ell_1^m \ell_2^n \pm 1$.

- If $p = 2^m 3^n \pm 1$, then $p \pm 1 = 2^m 3^n \pm 2 = 2\left(2^{m-1}3^n \pm 1\right)$. If $GCD(m-1, n) = d \neq 1$, one can perform factorisation of $p \pm 1 = 2\left(\left(2^{\frac{m-1}{d}}3^{\frac{n}{d}}\right)^d \pm 1\right)$. If the substitution $w = 2^{\frac{m-1}{d}}3^{\frac{n}{d}}$ is made, then the factorisation of the number $w^d \pm 1$, with addition of factor 2, will give the full factorisation of $p \pm 1$.
- If $p = 2a^m b^n \pm 1$, then $p \pm 1 = 2a^m b^n \pm 2 = 2\left(a^m b^n \pm 1\right)$. If $GCD(m, n) = d \neq 1$, one can perform factorisation of $p \pm 1 = 2\left(\left(a^{\frac{m}{d}} b^{\frac{n}{d}}\right)^d \pm 1\right)$. If the

substitution $w = a^{\frac{m}{d}} b^{\frac{n}{d}}$ is made, then the factorisation of the number $w^d \pm 1$, with addition of factor 2, will give the full factorisation of $p \pm 1$.

D. Factorisation of numbers of the form $w^n \pm 1$

In this subsection it will be shortly described how to perform factorisation of numbers of the form $w^d \pm 1$, which may be applied to the numbers $p \pm 1$ for SIDH primes p.

1) Algebraic factors method: The algebraic and Aurifeuillean factorisation is well described in [14].

Let's consider the polynomial $x^d - 1$. Let's denote by $\Phi_e(x)$ the e-th cyclotomic polynomial, then for $d \geq 1$ one has

$$x^d - 1 = \prod_{e|d} \Phi_e(x).$$

Although the cyclotomic polynomials are irreducible over the integers, $\prod_{e|d} \Phi_e(x)$ with $x = w$, in general does not give the complete factorisation of $w^d - 1$ since any factor $\Phi_d(b)$ might be composite.

One may factor $w^d + 1$ in a similar fashion using the identity

$$x^d + 1 = \left(x^{2^d} - 1\right) / \left(x^d - 1\right) =$$

$$= \prod_{e|2d} \Phi_e(x) / \prod_{e|d} \Phi_e(x) = \prod_{e|m} \Phi_{2^t e}(x),$$

where $2d = 2^t m$ with m odd.

2) Aurifeuillean factorisation: Numbers of the form $w^d - 1$ or $\Phi_d(w)$, where $w = s^2 \cdot t$ with square-free t, have Aurifeuillean factorisation if and only if one of the following conditions holds:

1) $t \equiv 1 (mod\ 4)$ and $d \equiv t (mod\ 2t)$
2) $t \equiv 2, 3 (mod\ 4)$ and $d \equiv 2t (mod\ 4t)$

Thus, when $w = s^2 \cdot t$ with square-free t, and d is congruent to $t \mod 2t$, then if t is congruent to $1 \mod 4$, $w^d - 1$ has Aurifeuillean factorization, otherwise, $w^d + 1$ has Aurifeuillean factorization. When the number is of a particular form (the exact expression varies with the base), Aurifeuillean factorization may be used, which gives a product of two or three numbers.

E. Security of DH scheme over $\mathbb{F}_{p^4}$ for SIDH primes

Another field, which may be used for DH scheme, is $\mathbb{F}_{p^4}$. Because order of multiplicative group of $\mathbb{F}_{p^4}$ field is equal to $p^4 - 1 = (p-1)(p+1)(p^2+1)$, there are elements which belong to $\mathbb{F}_{p^4} \backslash \mathbb{F}_{p^2}$, whose order divides $p^2 + 1$. If $p^2 + 1$ has a large prime divisor r, then one can find an element $g \in \mathbb{F}_{p^4} \backslash \mathbb{F}_{p^2}$, for which $Ord(g) = r$. If r is large, then one can use the DH scheme over $\mathbb{F}_{p^4}$. Moreover, the security of such algorithm (Figure 3) is equal to $O\left(e^{\left(\sqrt[3]{\frac{32}{9}} + o(1)\right)(4 \ln p)^{\frac{1}{3}} \cdot (\ln (4 \ln p))^{\frac{2}{3}}}\right)$ and is much higher than for the DH scheme over $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$.

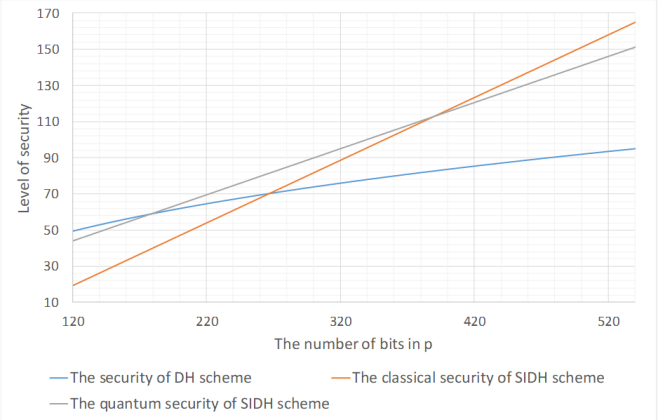

Fig. 3. Security of the DH scheme over $\mathbb{F}_{p^4}$ for SIDH primes

The $\mathbb{F}_{p^4}$ arithmetic may be built using sequence of $\mathbb{F}_{p^2}$ arithmetic, which requires, for properly chosen irreducible polynomial, 3 multiplications in $\mathbb{F}_{p^2}$ and few additions in $\mathbb{F}_{p^2}$.

F. Security of DH scheme over $\mathbb{F}_{p^6}$ for SIDH primes

The next finite field, which may be applied for DH scheme, is $\mathbb{F}_{p^6}$. The order of a multiplicative group of $\mathbb{F}_{p^6}$ field is equal to $p^6 - 1 = (p^2 - p + 1)(p^2 + p + 1)(p + 1)(p - 1)$. It is possible to find the element $g \in \mathbb{F}_{p^6} \backslash (\mathbb{F}_{p^3} \cup \mathbb{F}_{p^2})$, for which $Ord(g) = r$, where r is large prime (in most cases) and r divides $p^2 - p + 1$. The DH scheme over $\mathbb{F}_{p^6}$ may be used if r is large. The security of DH scheme over $\mathbb{F}_{p^6}$ for SIDH primes is equal to $O\left(e^{\left(\sqrt[3]{\frac{32}{9}} + o(1)\right)(6 \ln p)^{\frac{1}{3}} \cdot (\ln (6 \ln p))^{\frac{2}{3}}}\right)$. The comparison of security of the DH scheme over $\mathbb{F}_{p^6}$ with a classical and quantum level of security of SIDH over $\mathbb{F}_{p^2}$ is presented in the Figure 4.

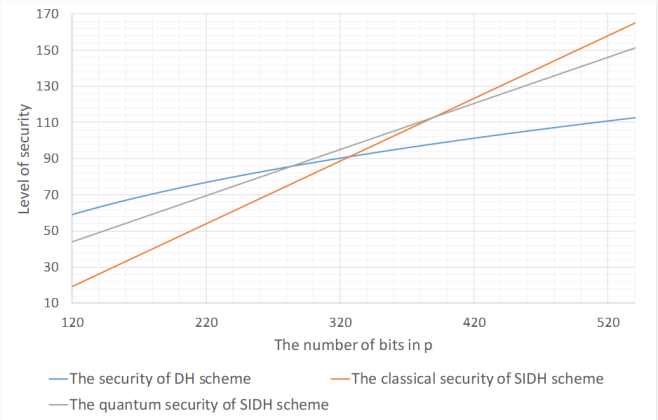

Fig. 4. Security of DH scheme over $\mathbb{F}_{p^6}$ for SIDH primes

The $\mathbb{F}_{p^6}$ arithmetic may be built using the $\mathbb{F}_{p^2}$ arithmetic, for properly chosen irreducible polynomial of degree 3 over $\mathbb{F}_{p^2}$. In such situation one multiplication in $\mathbb{F}_{p^6}$ requires 6 multiplications in $\mathbb{F}_{p^2}$.

G. Security of DH scheme over $\mathbb{F}_{p^8}$ for SIDH primes

Another finite field, which may be applied is $\mathbb{F}_{p^8}$. The order of multiplicative group of $\mathbb{F}_{p^8}$ field is equal to $p^8 - 1 = (p^4+1)(p^2+1)(p-1)(p+1)$. There exist elements which belong to $\mathbb{F}_{p^8} \backslash \mathbb{F}_{p^4}$, whose order divides p^4+1. If p^4+1 has a large prime divisor r, then one can find an element $g \in \mathbb{F}_{p^8} \backslash \mathbb{F}_{p^4}$, for which $Ord(g) = r$, where r is large. Then one can use the DH scheme over $\mathbb{F}_{p^8}$. The security of such algorithm is equal to $O\left(e^{\left(\sqrt[3]{\frac{32}{9}}+o(1)\right)(8\ln p)^{\frac{1}{3}}\cdot(\ln(8\ln p))^{\frac{2}{3}}}\right)$. The comparison of security of the DH scheme over $\mathbb{F}_{p^8}$ with classical and quantum level of security of SIDH over $\mathbb{F}_{p^2}$ is presented in the Figure 5.

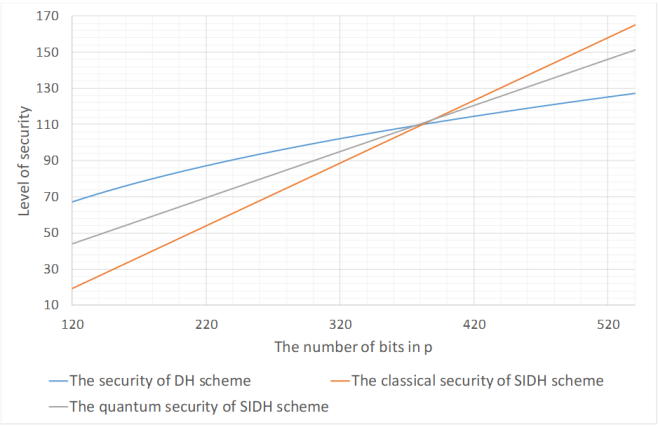

Fig. 5. Security of DH scheme over $\mathbb{F}_{p^8}$ for SIDH primes

In order to build the $\mathbb{F}_{p^8}$ arithmetic using the $\mathbb{F}_{p^2}$ arithmetic, it is necessary to choose irreducible polynomial of degree 2, whose coefficients belong to $\mathbb{F}_{p^2}$, and then it is necessary to choose irreducible polynomial of degree 2, whose coefficients belong to $\mathbb{F}_{p^4}$. Then one multiplication in $\mathbb{F}_{p^8}$ requires 9 multiplications in $\mathbb{F}_{p^2}$.

H. Security of DH scheme over finite fields with high extension degrees

For the DH scheme, finite fields with high extension degrees $n > 8$ may be used. However, if the arithmetic of $\mathbb{F}_{p^n}$ will use the $\mathbb{F}_{p^2}$ arithmetic, then the degree of an extension field n should be divisible by two. In order to verify if there are elements, whose order is a large prime, it is necessary to find the factorisation of the number $p^n - 1$ into prime factors.

I. Security recomendations

Basing on performed calculations and the above analyses, the maximum number of bits of the finite field characteristic p, for which the DH scheme may be used, has been determined. These results are presented in the Table I.

TABLE I
MAXIMAL SECURITY LEVELS OF $SIDH - DH$ HYBRID SCHEMES FOR DIFFERENT DEGREES OF EXTENSION n.

	DH($\mathbb{F}_{p^4}$)	$DH(\mathbb{F}_{p^6})$	DH($\mathbb{F}_{p^8}$)
Degree n of field extension	4	6	8
Maximal bit-length of p	176b	281b	374b
Maximal level of $HS(SIDH(\mathbb{F}_{p^2}), DH(\mathbb{F}_{p^n}))$	58	85	108

V. ARITHMETIC IN EXTENSION FIELDS

A. $\mathbb{F}_{p^4}$ arithmetic using $\mathbb{F}_{p^2}$ arithmetic

A field $\mathbb{F}_{p^4}$ may be constructed using an irreducible polynomial of degree 2 with coefficients from an $\mathbb{F}_{p^2}$ field. An example of such construction will be presented for prime p_{271}. For this prime, a polynomial $f(t) = t^2+1$ is irreducible over $\mathbb{F}_p$ and therefore its roots t generate the $\mathbb{F}_{p^2}$ field. Next, the polynomial $g(s) = s^2 - (t+1)s - t - 1$ with coefficients from $\mathbb{F}_{p^2}$ is irreducible over this field and therefore its root s generates the $\mathbb{F}_{p^4}$ field.

The addition/subtraction of two elements $A = a_1 s \pm a_0$ and $B = b_1 s \pm b_0$, where $A, B \in \mathbb{F}_{p^4}$, may be computed using two additions/subtractions in $\mathbb{F}_{p^2}$, because $C = A \pm B = (a_1 \pm b_1)s + (a_0 \pm b_0)$.

The multiplication of two elements $A = a_1 s \pm a_0$ and $B = b_1 s \pm b_0$ may be computed using algorithm 1.

Algorithm 1 Multiplication in $\mathbb{F}_{p^4}$ generated by irreducible polynomials $f(t) = t^2+1$ and $g(s) = s^2 - (t+1)s - t - 1$

Input: $A = a_1 s + a_0$, $B = b_1 s + b_0$, $A, B \in \mathbb{F}_{p^4}$, $a_0, a_1, b_0, b_1 \in \mathbb{F}_{p^2}$

Output: $C = c_1 s + c_0 = A \cdot B$

1: $D := a_1$;
2: $E := a_0$;
3: $F := b_1$;
4: $G := b_0$;
5: $H = D + E$;
6: $E = E \cdot G$;
7: $G = F + G$;
8: $D = D \cdot F$;
9: $F = G \cdot H$;
10: $F = F - E$;
11: $E = D + E$;
12: $D = D \cdot t$;
13: $c_1 = D + F$;
14: $c_0 = D + E$;

return $c_1 s + c_0$;

Completing an optimization algorithm causes a reduction of necessary memory registers. A multiplication of 2 elements from $\mathbb{F}_{p^4}$ field may be carried out using only 5 variables. Assuming that 1 register corresponds to 1 element from the $\mathbb{F}_{p^2}$ field, where a bit-length of p is equal to 271, 1 register requires 542 bits and the whole multiplication needs 2710 bits of memory.

B. $\mathbb{F}_{p^6}$ arithmetic using $\mathbb{F}_{p^2}$ arithmetic

The $\mathbb{F}_{p^6}$ field may be constructed using an irreducible polynomial of degree 3 with coefficients from $\mathbb{F}_{p^2}$ field. An examplary solution assumes that this polynomial is $g(s) = s^3 + s - 1$, so its roots s generate the $\mathbb{F}_{p^6}$ field.
The addition/subtraction of two elements $A = a_2s^2 \pm a_1s \pm a_0$ and $B = b_2s^2 \pm b_1s \pm b_0$, where $A, B \in \mathbb{F}_{p^6}$, may be computed using three additions/subtractions in $\mathbb{F}_{p^2}$, because $C = A \pm B = (a_2 \pm b_2)s^2 + (a_1 \pm b_1)s + (a_0 \pm b_0)$.
The multiplication of two elements $A = a_2s^2 \pm a_1s \pm a_0$ and $B = b_2s^2 \pm b_1s \pm b_0$ may be computed using algorithm 2.

Algorithm 2 Multiplication in $\mathbb{F}_{p^6}$ generated by irreducible polynomials $f(t) = t^2 + 1$ and $g(s) = s^3 + s - 1$

Input: $A = a_2s^2+a_1s+a_0$, $B = b_2s^2+b_1s+b_0$, $A, B \in \mathbb{F}_{p^6}$, $a_0, a_1, a_2, b_0, b_1, b_2 \in \mathbb{F}_{p^2}$
Output: $C = c_2s^2 + c_1s + c_0 = A \cdot B$

1: $D = a_2$;
2: $E = a_1$;
3: $F = a_0$;
4: $G = b_2$;
5: $H = b_1$;
6: $I = b_0$;
7: $J = D + E$;
8: $K = D + F$;
9: $L = G + I$;
10: $K = K \cdot L$;
11: $L = E + F$;
12: $D = D \cdot G$;
13: $E = E \cdot H$;
14: $F = F \cdot I$;
15: $G = G + H$;
16: $H = H + I$;
17: $G = J \cdot G$;
18: $I = G - D$;
19: $E = F - E$;
20: $c_0 = I + E$;
21: $D = 2 \cdot D$;
22: $K = K - D$;
23: $c_2 = K - E$;
24: $L = L \cdot H$;
25: $G = D - G$;
26: $G = G + L$;
27: $c_1 = G - F$;

return $c_2s^2 + c_1s + c_0$;

As well as in the multiplication in the $\mathbb{F}_{p^4}$ field, an optimization allows to reduce a memory cost of algorithm 2. To multiply 2 elements from $\mathbb{F}_{p^6}$ field, 9 registers are needed. As assumed earlier, 1 register requires 542 bits, so the optimized multiplication in $\mathbb{F}_{p^6}$ field needs 4878 bits of memory.

C. $\mathbb{F}_{p^8}$ arithmetic using $\mathbb{F}_{p^2}$ arithmetic

To construct the $\mathbb{F}_{p^8}$ field, an irreducible polynomial of degree 2 with coefficients from $\mathbb{F}_{p^4}$ field is necessary. The choice fell on $h(u) = u^2 - su + 1$. Because a degree of chosen polynomial is 2, as well as in $\mathbb{F}_{p^4}$, the arithmetic in both fields will be similar.

The addition/subtraction of two elements $A = a_1u \pm a_0$ and $B = b_1u \pm b_0$ may be computed using two additions/subtractions in $\mathbb{F}_{p^4}$, because $C = A \pm B = (a_1 \pm b_1)u + (a_0 \pm b_0)$.

The multiplication of two elements $A = a_1u \pm a_0$ and $B = b_1u \pm b_0$ may be computed using an algorithm 3.

Algorithm 3 Multiplication in $\mathbb{F}_{p^8}$ generated by polynomials $f(t) = t^2 + 1$, $g(s) = s^2 - (t + 1)s - t - 1$ and $h(u) = u^2 - su + 1$.

Input: $A = a_1u + a_0$, $B = b_1u + b_0$, $A, B \in \mathbb{F}_{p^8}$, $a_0, a_1, b_0, b_1 \in \mathbb{F}_{p^4}$
Output: $C = c_1u + c_0 = A \cdot B$

1: $D = a_1$;
2: $E = a_0$;
3: $F = b_1$;
4: $G = b_0$;
5: $H = D + E$;
6: $D = D \cdot F$;
7: $E = E \cdot G$;
8: $F = F + G$;
9: $H = H \cdot F$;
10: $c_0 = E - D$;
11: $D = s \cdot D$;
12: $D = H + D$;
13: $D = D + c_0$;
14: $D = D - E$;
15: $c_1 = D - E$;

return $c_1u + c_0$;

To multiply 2 elements from the $\mathbb{F}_{p^8}$ field with the optimized algorithm, 5 variables are needed. However, 1 register corresponds to 1 element from the $\mathbb{F}_{p^2}$ field and in the algorithm 3 coefficients are from the $\mathbb{F}_{p^4}$ field, so all of them require 2 registers. Summing up, the algorithm of multiplication in $\mathbb{F}_{p^8}$ needs 10 registers (5420 bits) to be completed.
The results of above analyses are presented in Table II.

TABLE II
COMPARISON OF $ECDH$ OVER $\mathbb{F}_p$ AND DH OVER $\mathbb{F}_{p^n}$ FOR DIFFERENT VALUES OF n; $bl(x)$ DENOTES THE BIT-LENGTH OF NUMBER x, $HW(x)$ DENOTES THE HAMMING WEIGHT OF NUMBER x AND $S(x) = bl(x) + HW(x)$.

	$ECDH(\mathbb{F}_p)$	$DH(\mathbb{F}_{p^4})$	DH($\mathbb{F}_{p^6}$)	DH($\mathbb{F}_{p^8}$)
Degree n of field extension	1	4	6	8
Length of private key	$bl(p)$	$bl(r)$	$bl(r)$	$bl(r)$
Length of public key	$2 \cdot bl(p)$	$n \cdot bl(p)$	$n \cdot bl(p)$	$n \cdot bl(p)$
Single step cost	11M	3M	6M	9M
Validation cost	$(S(r) + 3)\,M$	$S(r)$M	$S(r)$M	$S(r)$M

VI. EXAMPLES OF DH SCHEME FOR SIDH HYBRID SCHEMES AND ITS EFFICIENCY ANALYSIS

In this section some examples of hybrid SIDH-DH schemes, with a complexity and security analysis, will be presented.

A. Assumptions

In every example and every protocol we assumed that its efficiency is based on the number of multiplications and inversions in the $\mathbb{F}_{p^2}$ field, because additions/subtractions have a small impact on the efficiency of the algorithm. Moreover, for the ECDH algorithm, we assumed that the fastest is application of ECDH based on the Montgomery curve $E : By^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$, with cofactor equal to 4 and parameter B equal to 1.

We also assumed, that in every protocol, when computations using the secret key are performed, the ladder technique is used. A Montgomery ladder for the Montgomery curve may be performed in 11 multiplications.

However, using the Montgomery curve with point representation in XZ coordinates requires checking if $\frac{x^3+Ax^2+x}{B}$ is quadratic residue in $\mathbb{F}_p$. If the Legendre symbol $\left(\frac{B}{p}\right) = B^{\frac{p-1}{2}}$ is previously computed, then verification requires only the computation of Legendre symbol $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}}$, where $a = x^3 + Ax^2 + x$.

In every protocol we also assumed, that every element sent by the second person is checked if is proper.

B. Example I

Let's consider the number $p = 2^{137}3^{84} - 1$ (it is 271 bit-length prime number). For such prime, the SIDH algorithm over $\mathbb{F}_{p^2}$ has 83 qubits level of quantum security and 72 bits level of classical security. DH over $\mathbb{F}_{p^6}$ has 85 bits level of classical security.

To check if there exist elements having a proper order, it is necessary to factorize the number p^2-p+1 (it is a cyclotomic polynomial which divides $p^6 - 1$). In this case $p^2 - p + 1$ is equal to $3 \cdot 7 \cdot 19 \cdot 67 \cdot 829 \cdot r_{33} \cdot r_{35} \cdot r_{38} \cdot r_{44} \cdot r_{68} \cdot r_{302}$, where r_i is i-bit prime.

If one chooses an element g of the order r_{302}, then Pollard's-rho attack can break the DH over $\mathbb{F}_{p^6}$ by $c_1 \cdot 2^{151}$ operations, for some constant c_1. Because sieve methods can break DH over $\mathbb{F}_{p^6}$ by $c_2 \cdot 2^{85}$ operations, where c_2 is some constant, finally, DH scheme over $\mathbb{F}_{p^6}$ in this case ensures 85 bit level of security.

So if a generator g has order r_{302} and the private key is 302 bit-length number, then g^a requires 604 multiplications in $\mathbb{F}_{p^6}$, if Montgomery ladder is used.

In the second phase of DH algorithm (we describe the operations which have to be performed by Alice), at first it is necessary to check if received element B is proper. It may be checked by computing $B^{r_{302}}$. This operation does not have to be computed securely and requires 451 multiplications in $\mathbb{F}_{p^6}$, assuming that standard left-to-right method is used. At the end, it is necessary to compute B^a, which requires 604 multiplications in $\mathbb{F}_{p^6}$.

Finally, the protocol requires in average case 9954 multiplications in $\mathbb{F}_{p^2}$.

DH scheme is in this case for $26,88\%$ slower than ECDH scheme based on Montgomery curve, and the public key is about six times longer (DH public key bit-length is equal to 1626 long and ECDH public key bit-length is equal to 271).

This example shows, that using DH over $\mathbb{F}_{p^6}$ is not comparable with ECDH over $\mathbb{F}_p$. It is clear, that if one wants to use DH algorithm, at least at 80 bits level of security, the different approach should be considered. Such approach is presented in the next section.

VII. Application of algorithms based on discrete logarithm over finite field using finite field elements compression

There are several algorithms equivalent to Diffie-Hellman and using compression of finite fields elements, like LUC [15], Gong-Harn algorithm [16] and XTR [17]. However, LUC allows one to compress element from field $\mathbb{F}_{p^{2n}}$ to the element from the $\mathbb{F}_{p^n}$ and Gong-Harn allows one to compress element from field $\mathbb{F}_{p^{3n}}$ to the element from the $\mathbb{F}_{p^n}$. XTR allows one to compress element from field $\mathbb{F}_{p^{6n}}$ to the element from the $\mathbb{F}_{p^{2n}}$ and, in the context of this article, seems to be the most useful. In the next part of this section applications of XTR algorithm will be shown.

A. XTR algorithm

The name of XTR algorithm comes from Efficient and Compact Subgroup Trace Representation. The main idea of XTR was to compress a public key representation, by using elements from $\mathbb{F}_{p^2}$, without losing a security of $\mathbb{F}_{p^6}$ field. It may be achievable with an element g of order $q > 6$ which divides $p^2 - p + 1$. The element g generates a subgroup of a multiplicative group from $\mathbb{F}_{p^6}$, because polynomial p^2-p+1 divides p^6-1, which is an order of q of mentioned multpilicative subgroup. As q is not a divisor of $p^k - 1$ for $k = 1, 2, 3$, the subgroup generated by g is not a part of any proper subfield of $\mathbb{F}_{p^6}$. The thing is that powers of g may be represented using an element of $\mathbb{F}_{p^2}$ and that they may be computed using arithmetic in $\mathbb{F}_{p^2}$, avoiding using an $\mathbb{F}_{p^6}$ arithmetic, which needs almost 2 times more memory and has bigger computational complexity. Using the above fact, trace of g may be acquired as $Tr(g) = g+g^{p^2}+g^{p^4}$, where $g \in \mathbb{F}_{p^6}$ and $Tr(g) \in \mathbb{F}_{p^2}$.

Algorithm 4 Computing $S_n(c)$

Input: c

Output: $S_n(c)$

case n is:

1) when $0 \rightarrow S_0(c) = (c^p, 3, c)$;
2) when $1 \rightarrow S_1(c) = (3, c, c^2 - 2c^p)$;
3) when $2 \rightarrow S_2(c) = (c, c^2 - 2c^p, c_3)$;
4) when $>2 \rightarrow$
   a) $m = n$;
   b) if m is even $\rightarrow m = m - 1$;
   c) $\bar{S}_k(c) = S_3(c)$;
   d) let $\frac{m-1}{2} = \sum_{j=0}^{r} m_j 2^j$, where $m_j \in \{0, 1\}$ and $m_r = 1$;
   e) for $j = r - 1, r - 2, ..., 0$ do:
      i) if $m_j = 0 \rightarrow \bar{S}_{2k}(c) = (c_{4k}, c_{4k+1}, c_{4k+2})$ and $k = 2k$;
      ii) if $m_j = 1 \rightarrow \bar{S}_{2k+1}(c) = (c_{4k+2}, c_{4k+3}, c_{4k+4})$ and $k = 2k + 1$;
   f) if n is even $\rightarrow S_{m+1}(c) = (c_m, c_{m+1}, c_{m+2})$ and $m = m + 1$;
5) $S_n(c) = S_m(c)$;

return $S_n(c)$;

In the XTR algorithm, computing a three-element set $S_n(c) = (c_{n-1}, c_n, c_{n+1}) \in (\mathbb{F}_{p^2})^3$ is necessary. It is possible

by using algorithm 4 and using below formulas with given c, c_{n-1}, c_n, c_{n+1}:

- $c_{2n} = c_n^2 - 2c_n^p$;
- $c_{n+2} = c \cdot c_{n+1} - c^p \cdot c_n + c_{n-1}$;
- $c_{2n-1} = c_{n-1} \cdot c_n - c^p \cdot c_n^p + c_{n+1}^p$;
- $c_{2n+1} = c_{n+1} \cdot c_n - c \cdot c_n^p + c_{n-1}^p$;
- $\bar{S}_t(c) = S_{2t+1}(c)$.

Because computing $S_a(Tr(g))$ is obligatory to obtain $Tr(g^a)$, algorithm 5 presents how to do this:

Algorithm 5 Computation of $Tr(g)$

Output: $Tr(g)$, where g is some element of order q

1) Choose random $c \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$;
2) Compute c_{p+1};
3) If $c_{p+1} \in \mathbb{F}_p \rightarrow$ return to Step 1;
4) Compute $c_{(p^2-p+1)/q}$;
5) If $c_{(p^2-p+1)/q} = 3 \rightarrow$ return to Step 1;
6) $Tr(g) = c_{(p^2-p+1)/q}$;

return $Tr(g)$;

XTR version of the Diffie-Hellman protocol is shown in algorithm 6:

Algorithm 6 XTR-DH

Input: public key data - $p, q, Tr(g)$
Output: shared secret key - K

1) Alice chooses random $a \in \mathbb{Z}$, where $1 < a < q - 2$;
2) Alice computes $S_a(Tr(g)) = (Tr(g^{a-1}), Tr(g^a), Tr(g^{a+1}))$ and sends $Tr(g^a)$ to Bob;
3) Bob chooses random $b \in \mathbb{Z}$, where $1 < b < q - 2$;
4) Bob computes $S_b(Tr(g)) = (Tr(g^{b-1}), Tr(g^b), Tr(g^{b+1}))$ and sends $Tr(g^b)$ to Alice;
5) Alice computes $S_a(Tr(g^b)) = (Tr(g^{(a-1)b}), Tr(g^{ab}), Tr(g^{(a+1)b}))$ and knows the $K = Tr(g^{ab})$;
6) Bob computes $S_b(Tr(g^a)) = (Tr(g^{a(b-1)}), Tr(g^{ab}), Tr(g^{a(b+1)}))$ and knows the $K = Tr(g^{ab})$;

B. Example of XTR application

For the previously presented 271 bit-length prime number p, XTR algorithm is more than twice faster than ECDH algorithm and its security level is equal to 83 bits (it is the same security level as for the DH over $\mathbb{F}_{p^6}$).

The public key in XTR algorithm is about twice longer (542 bits), comparing to the bit-length of public key (269 bits) in ECDH algorithm based on Montgmery XZ arithmetic.

However, in the XTR algorithm it is suggested to use $\mathbb{F}_{p^2}$ field which is generated by the root of irreducible polynomial $f(t) = t^2 + t + 1$, because in such case t is the generator of an optimal normal basis and computation of the Frobenius endomorphism $\pi(g)$ is just the rotation of the elements. It is worth to note, that for the irreducible polynomial $f(t) = t^2 + 1$, for which the multiplication in a polynomial basis is the most efficient, computation of $\pi(g)$ for the element $g = g_1 t + g_0, g \in \mathbb{F}_{p^2}$ requires only one subtraction: $\pi(g) = -g_1 t + g_0$. Computation of Frobenius endomorphism is in such case very efficient and it is not necessary to make any conversion between a polynomial and normal bases.

The whole XTR algorithm requires in this case 3624 multiplications in the $\mathbb{F}_{p^2}$ field and is therefore 53% faster than the previously presented ECDH algorithm using Montgomery curve over $\mathbb{F}_p$.

The comparison of efficiency of $ECDH$ over $\mathbb{F}_p$ and XTR over $\mathbb{F}_{p^6}$ is presented in the Table III.

TABLE III
COMPARISON OF EFFICIENCY OF $ECDH$ OVER $\mathbb{F}_p$ USING MONTGOMERY CURVE AND XTR OVER $\mathbb{F}_{p^6}$, WHERE $p = 2^{137}3^{84} - 1$ IS 271-BITS PRIME.

	$ECDH(\mathbb{F}_p)$	$XTR(\mathbb{F}_{p^6})$
Length of private key	269b	302b
Length of public key	269b	542b
Cost for 1 bit of private key	11M	4M
1st (2nd) round	2959M	1208M
Validation	477M	1208M
Additional computations (inversion)	949M	0M
Whole algorithm	7845M	3624M
$QS(\mathcal{Q})$	83q	83q
$CS(\mathcal{Q})$	63b	63b
$CS(\mathcal{P})$	135b	85b
$HS(\mathcal{Q}, \mathcal{P})$	83	83

VIII. CONCLUSION AND FURTHER WORKS

In this paper was analyzed possibility of an application of Diffie-Hellman scheme over extension fields and protocols based on the discrete logarithm problem over finite fields, which uses elements compression. The levels of security and examples of applications of such protocols were presented. It seems that XTR algorithm using traces of $\mathbb{F}_{p^6}$ elements over $\mathbb{F}_{p^2}$ field may be used in the SIDH-XTR hybrid scheme for low levels of security, up to 85 bits. In the example presented in a previous subsection, XTR algorithm is about 53% faster than ECDH based on Montgomery curve over $\mathbb{F}_p$, having public key twice longer.

The further works should focus on a hardware implementation of presented solution and investigation of other XTR-like algorithms, whose efficiency in a hybrid scheme would be comparable to an application of ECDH over $\mathbb{F}_p$ field, but a security level of such solution would be at least 128 bits.

REFERENCES

[1] C. Costello, P. Longa, and M. Naehrig, "Efficient algorithms for supersingular isogeny diffie-hellman," in *Advances in Cryptology – CRYPTO 2016*, M. Robshaw and J. Katz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 572–601.

[2] R. D. M. Wroński, T. Kijko, "Methods of generation of elliptic curves for hybrid sidh scheme over large fields," *To appear in: Proceedings of the Romanian Academy, Series A*, vol. January-March, 2020.
[3] D. B. Roy and D. Mukhopadhyay, "Post quantum ecc on fpga platform." [Online]. Available: https://eprint.iacr.org/2019/568
[4] S. Jaques and J. M. Schanck, "Quantum cryptanalysis in the ram model: Claw-finding attacks on sike," in *CRYPTO*, 2019.
[5] D. J. et. al., "Supersingular isogeny key encapsulation (version from april 17, 2019," NIST PQC, 2019, https://sike.org/files/SIDH-spec.pdf.
[6] G. L. Mullen and D. Panario, *Handbook of finite fields*. Chapman & Hall/CRC, 2013.
[7] P. Wang and F. Zhang, "Improved pollard rho method for computing discrete logarithms over finite extension fields," *Journal of Computational and Applied Mathematics*, vol. 236, no. 17, p. 4336–4343, 2012.
[8] S. D. Galbraith, J. M. Pollard, and R. S. Ruprai, "Computing discrete logarithms in an interval," *Mathematics of Computation*, vol. 82, no. 282, p. 1181–1195, 2012.
[9] R. S. Ruprai, "Improvements to the gaudry-schost algorithm for multidimensional discrete logarithm problems and applications," *Department of Mathematics, Royal Holloway University of London*, 2010.
[10] "1363-2000 - ieee standard specifications for public-key cryptography." [Online]. Available: https://standards.ieee.org/standard/1363-2000.html
[11] I. F. Blake, G. Seroussi, N. P. Smart, and K. Witold, *Krzywe eliptyczne w kryptografii*. Wydawnictwa Naukowo-Techniczne, 2004.
[12] A. Guillevic, "Discrete logarithm computation in finite fields $\mathfrak{O}_{p^n}$ with nfs variants and consequences in pairing-based cryptography," personal site, 2019, https://members.loria.fr/AGuillevic/files/talks/19_Rennes_STNFS.pdf.
[13] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov 1994, pp. 124–134.
[14] S. Wagstaf, "The cunningham project," https://pdfs.semanticscholar.org/66af/f30505c7cdf318756785a937744bca3b1e5b.pdf.
[15] P. J. Smith and M. J. Lennon, "Luc: A new public key system," in *SEC*, 1993, pp. 103–117.
[16] G. Gong and L. Harn, "Public-key cryptosystems based on cubic finite field extensions," *IEEE Transactions on Information Theory*, vol. 45, no. 7, pp. 2601–2605, 1999.
[17] A. K. Lenstra and E. R. Verheul, "The xtr public key system," in *Annual International Cryptology Conference*. Springer, 2000, pp. 1–19.