THE GPST ADAPTIVE ATTACK

YAN BO TI

1. Introduction

Central to the active attack on SIDH are the auxiliary points that are sent as part of the message from one party to another. The auxiliary points are needed so that Alice and Bob can then compute their secret isogenies on a different domain. Recall that when taking part in SIDH, Alice randomly selects secret scalars (a_1, a_2) and computes the point $[a_1]P_A + [a_2]Q_A$ that generates the subgroup $G_A = \langle [a_1]P_A + [a_2]Q_A \rangle$ which forms the kernel of the isogeny ϕ_A. She then sends the tuple $(E_A, \phi_A(P_B), \phi_A(Q_B))$ to Bob, and will receive $(E_B, \phi_B(P_A), \phi_B(Q_A))$. In this tutorial, we will see how the presence of the points $\phi_B(P_A)$ and $\phi_B(Q_A)$ (called the auxiliary points) allows for an adaptive attack. In particular, we will illustrate how an attacker, playing the role of Bob, is able to recover Alice's secret key by sending her malformed tuples.

The main reference for this tutorial is [GPST16]. We aim to go in-depth and explain the intuition from that paper.

SIDH. First, a quick overview of the SIDH scheme. Setup of the scheme involves the choice of a prime of the form $p = \ell_A^n \cdot \ell_B^m \cdot f \pm 1$, where ℓ_A, ℓ_B are small primes, $\ell_A^n \approx \ell_B^m$, and f is a small cofactor. Typically, $\ell_A = 2$ and $\ell_B = 3$, so we will assume this case in the following discussion. A supersingular elliptic curve E_0 is then constructed over the field $\mathbb{F}_{p^2}$, and made public along with chosen bases $\langle P_A, Q_A \rangle = E_0[2^n]$, $\langle P_B, Q_B \rangle = E_0[3^m]$ (these torsion subgroups have order 2^{2n} and 3^{2m} respectively).

To perform the key exchange, Alice will select $0 \leq a_1, a_2 < 2^n$ not both divisible by 2, and similarly Bob will select $0 \leq b_1, b_2 < 3^m$ not both divisible by 3. These integers allow each party to compute a secret subgroup each

$$G_A = \langle [a_1]P_A + [a_2]Q_A \rangle, \qquad G_B = \langle [b_1]P_B + [b_2]Q_B \rangle.$$

Next, they construct isogenies $\phi_i : E_0 \to E_i = E_0/G_i$, $i \in \{A, B\}$. Alice will send to Bob the tuple $\big(E_A, \phi_A(P_B), \phi_A(Q_B)\big)$ and Bob will reciprocate appropriately. The exchanged points will allow Alice (and in the same manner Bob) to compute

$$\langle [a_1]\phi_B(P_A) + [a_2]\phi_B(Q_A) \rangle = \langle \phi_B([a_1]P_A + [a_2]Q_A) \rangle = \phi_B(G_A).$$

These new subgroups can then be used to compute isogenies $E_A \to E_A/\phi_A(G_B) \simeq E_B/\phi_B(G_A) \simeq E_0/\langle G_A, G_B \rangle$. Because the j-invariant $j\left(E_0/\langle G_A, G_B \rangle\right)$ of these curves is isomorphism-invariant, it can be used as the shared key. This is summarised in Figure 1.

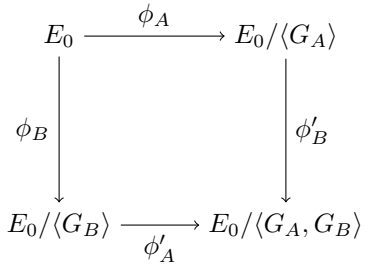

Figure 1. The SIDH key exchange.

Date: July 2021.

As is now standard any choice of secret integers (a_1, a_2) is equivalent to either $(1, \alpha)$ or $(\alpha', 1)$. It thus suffices to consider only these cases. For the rest of this work we use keys of the form $(1, \alpha)$.

Exercise 1. *Prove that any choice of secret integers (a_1, a_2) is equivalent to either $(1, \alpha)$ or $(\alpha', 1)$.*

2. Adaptive Attacks

Adaptive attacks are a standard type of attack on a cryptosystem that is using a static private key. The cryptosystem can be treated as an oracle that produces an output based on an input. The output will be related to the input and the secret using a known computation. The attacker will adapt its interaction with the oracle in such a way as to learn the secret from the input and output. Taking SIDH as an example, inputting the tuple $\big(E_B, \phi_B(P_A), \phi_B(Q_A)\big)$ may lead to an output[1] of $j(E_B/\phi_B(G_A))$.

Consider the setting of cryptosystems based on discrete logarithm problems, where a user can be treated as an oracle that on input of a group element $g \in G$, outputs the group element $g^a \in G$, where a is some static secret. The "small subgroup attack" of Lim and Lee [LL97] explains how an attacker can recover the secret with carefully chosen group elements g if the group G has many small subgroups.

Exercise 2. *In this exercise, we will play the role of an attacker that is learning the secret from an oracle f by sending carefully chosen queries.*

Let N be 2021, and let $G = (\mathbb{Z}/N\mathbb{Z})^*$.

(a) Compute the order of G, find the group structure of G.

(b) Verify that 988, 1033, 565, 236, 130 are generators of G, i.e.

$$\langle 988, 1033, 565, 236, 130\rangle = G.$$

(c) Given a function $f : G \to G$ such that $f(g) = g^k$ for some $g \in G$, and also, given the following congruences:

$$\begin{aligned} f(988) &= 988,\\ f(1033) &= 1033,\\ f(565) &= 1,\\ f(236) &= 1411,\\ f(130) &= 388 \end{aligned}$$

Recover k using the Chinese Remainder Theorem.

Remark 1. *In this exercise, an attacker is given access to an oracle with a secret k. The task of the attacker is to recover k by only querying the oracle and observing the output. The final part of the exercise shows that the secret k can be recovered by querying the oracle on carefully chosen inputs.*

2.1. Oracles.

The adaptive attack on SIDH is also based on querying an oracle using carefully chosen inputs to work out the secret in the oracle. Consider the SIDH protocol, where Alice is using a static long-term secret, and Bob is a malicious party trying to learn her secret.

Recall that Bob will send Alice the tuple $(E_B, \phi_B(P_A), \phi_B(Q_B))$, which she will use to compute

$$\phi_B(P_A) + [\alpha]\phi_B(Q_A)$$

which is the generator of the kernel that yields the supersingular elliptic curve E_{AB}.

The idea behind the SIDH adaptive attack is to send Alice (or the oracle) points which are different from $\phi_B(P_A)$, $\phi_B(Q_A)$. Let's call these points $R = \phi_B(P_A)$ and $S = \phi_B(Q_A)$, and call $E = E_B$. So Bob's tuple is (E, R, S). Under normal circumstances, what Alice computes cannot be seen by anyone except for Alice. However, further interactions with Alice can reveal information about her final computation.

For instance, consider the following oracle model:

$$O(E, R, S) = j(E/\langle R + [\alpha]S\rangle).$$

If the scheme under attack is the key exchange scheme, this corresponds to Alice taking Bob's protocol message, completing her side of the protocol, and outputting the shared key. In the encryption protocol, this

[1] We will address the specific outputs later in this tutorial.

would correspond to an encryption $c = m \oplus j(E_{AB})$ *without the hash function* and Alice decrypting Bob's ciphertext and returning the plaintext m.

We will see another instance of an oracle that only returns a single bit of information. That instance of an oracle is evidently weaker than the one presented above, which implies that an attack using a weaker oracle must be a stronger attack.

2.2. **First version of the adaptive attack with a stronger oracle.** This section of the tutorial will guide you through an adaptive attack that is able to recover the secret from the oracle model from the previous section.

Exercise 3. *Suppose that you are an attacker playing the role of Bob, and you are trying to recover the static secret key of Alice. This means that she has a corresponding static public key given by* $(E_A, \phi_A(P_B), \phi_A(Q_B))$. *The first step is to honestly generate* $(E_B, \phi_B(P_A), \phi_B(Q_A))$ *as per the SIDH protocol.*

Henceforth, we will replace Alice with the oracle described above, i.e. for an input (E, R, S), *the oracle will return* $j(E/\langle R + [\alpha]S\rangle)$. *Also, for ease of notation, we will write* $E = E_B$, $R = \phi_B(P_A)$, $S = \phi_B(Q_A)$, *and suppose that* $\langle P_A, Q_A\rangle = E_0[2^n]$.

(a) Consider the tuple $(E, R, [2^{n-1}]S)$. *Compute the kernel that the oracle computes upon receipt of this tuple in terms of* α.

(b) Let α_0 *denote the first bit of* α. *Compute the kernel in the case where* α_0 *is* 0 *and when it is* 1. *Prove that the kernels are different.*
Is is possible to recover α_0*?*

(c) Assuming that you have recovered α_0, *consider the tuple* $(E, R, [2^{n-2}]S)$. *Recover the second bit of* α.

(d) Change the roles of Alice and Bob, and recover Bob's secret key using the ideas from above.

The exercise above is a guide to implement an adaptive attack that recovers the first two bits of Alice's secret. It should be clear how one can extend this attack to recover the entirety of Alice's secret.

However, it is also clear that the malformed points sent to Alice are not of the correct order. In general, there are two checks that Alice can perform that can check for malformed points:

(1) Point order check
(2) Weil pairing check

Refer to Algorithm 1 for a pseudocode of this attack.

3. The GPST attack

Consider the oracle

$$O(E, R, S, E') = \begin{cases} 1 & \text{if } j(E') = j(E/\langle R + [\alpha]S\rangle) \\ 0 & \text{otherwise.} \end{cases}$$

In the key exchange setting, this corresponds to Alice taking Bob's protocol message, completing her side of the protocol, and then performing some operations using the shared key that return an error message if the shared key is not the same as the j-invariant provided (e.g., the protocol involves verifying a MAC corresponding to a key derived from the session key).

In the encryption scenario, this would correspond to Bob having access to a decryption oracle for Alice. By choosing a random ciphertext c Bob could ask for a decryption of (E_B, R, S, c) and get m such that $c = m \oplus H_k(j(E_{AB}))$. Bob can then check whether or not $c \oplus m = H_k(j(E'))$. Hence a decryption oracle for the encryption scheme gives an oracle O of this type.

There are different oracles that suit different cryptographic applications, and it is up to the attacker to decide which oracle model is required for the use-case at hand.

3.1. **First step.** We will make the assumption that Alice's secret is $(1, \alpha)$.

The first step of the attack reveals the first bit of α. This is done by honestly generating the ephemeral key $(E_B, R = \phi_B(P_A), S = \phi_B(Q_A))$, and querying the oracle on $(E_B, R, S + [2^{n-1}]R, E_{AB})$.

Notice that the oracle computes

$$\langle R + [\alpha](S + [2^{n-1}]R)\rangle = \langle R + [\alpha]S + [\alpha][2^{n-1}]R\rangle\,.$$

If the subgroups $\langle R + [\alpha]S\rangle$ and $\langle R + [\alpha]S + [2^{n-1}]R\rangle$ are different, then the oracle's response will disclose the first bit of α. A result of 1 from the oracle informs the attacker that α is even, otherwise the attacker learns that α is odd.

Exercise 4. *(a) Prove that $\langle R + [\alpha]S\rangle$ and $\langle R + [\alpha]S + [2^{n-1}]R\rangle$ are different subgroups.*
(b) Suppose that the attacker is unsure if Alice has secret scalars of the form $(1, \alpha)$ or $(\alpha, 1)$. How would the attacker recover this information?

3.2. **Iterative step.** We now describe the remainder of the attack. Suppose we have learnt the first i bits of α. We can write α in the form

$$\alpha = K_i + \alpha_i 2^i + \alpha' 2^{i+1},$$

where K_i is the i-th partial key that we now know, $\alpha_i \in \{0, 1\}$ is the next bit of α we hope to learn, and α' is unknown. The attacker will honestly generate $(E_B, R = \phi_B(P_A), S = \phi_B(Q_A))$ and E_{AB} as before, according to the protocol. He will then query the oracle with

$$\left(E_B, R - [2^{n-i-1}K_i]S, [1 + 2^{n-i-1}]S, E_{AB}\right)$$

If the response of the oracle is 1, then $\alpha_i = 0$, otherwise $\alpha_i = 1$, because

$$\begin{aligned}
&\langle R - [2^{n-i-1}K_i]S + [\alpha][1 + 2^{n-i-1}]S\rangle \\
&= \langle R + [\alpha]S + [-2^{n-i-1}K_i + 2^{n-i-1}(K_i + 2^i\alpha_i + 2^{i+1}\alpha')]S\rangle \\
&= \langle R + [\alpha]S + [\alpha_i 2^{n-1}]S\rangle \\
&= \begin{cases} \langle R + [\alpha]S\rangle & \text{if } \alpha_i = 0\,, \\ \langle R + [\alpha]S + [2^{n-1}]S\rangle & \text{if } \alpha_i = 1\,. \end{cases}
\end{aligned}$$

Exercise 5. *This exercise will show that the malformed points chosen above are able to recover the secret α, and evade standard countermeasures (order checking and Weil pairing).*

Consider the query $\left(E_B, R' = R - [2^{n-i-1}K_i]S, S' = [1 + 2^{n-i-1}]S, E_{AB}\right)$ to the oracle. Now, consider the following conditions:

(1) If $\alpha_i = 0$, then the oracle returns 1.
(2) If $\alpha_i = 1$, then oracle returns 0.
(3) The malformed points both have order 2^n.
(4) The Weil pairing must be equal to

$$e_{2^n}(\phi_B(P_A), \phi_B(Q_A)) = e_{2^n}(P_A, Q_A)^{\deg \phi_B} = e_{2^n}(P_A, Q_A)^{3^m}$$

The first two conditions help us distinguish the bit α_i and the latter two prevent the attack from being detected via order checking and Weil pairing validation checks respectively.

(a) Check if the query (E_B, R', S', E_{AB}) satisfies the first three conditions.
(b) Compute the Weil pairing on the pair R', S'.
(c) Compute the Weil pairing on the pair $\theta R', \theta S'$, and solve for θ such that $e_{2^n}(\theta R', \theta S') = e_{2^n}(P_A, Q_A)^{3^m}$.
(d) Check that $\langle [\theta]R' + [\alpha][\theta]S'\rangle = \langle [\theta](R' + [\alpha]S')\rangle = \langle R' + [\alpha]S'\rangle$ if θ is coprime to the order 2^n. Does this masking hold for all steps?

Exercise 6. *(a) Change the malformed points such that the attacker can recover Alice's secret which is of the form $(\alpha, 1)$.*
(b) Suppose the attacker is playing the role of Alice, and is trying to recover Bob's private key. Show that the first bit (ternary now) can be recovered.
(c) Implement the GPST attack for yourself.

With luck, this section would have sketched the GPST adaptive attack to you. Refer to Algorithm 2 for a pseudocode.

4. Cryptographic consequences

This attack does not solve the mathematical problem underpinning the protocol because one is not able to use this attack to solve the computational supersingular isogeny problem. The consequence of having an adaptive attack on SIDH is purely cryptographic, in that only certain protocols will be affected. However, this is still a significant attack on the cryptographic protocol. It is clear that using static keys for non-interactive key exchanges are now insecure without proper protections. Furthermore, we have shown that simple countermeasures such as point order checking and Weil pairing checking can be circumvented by this adaptive attack.

The only option to realise a non-interactive key exchange (NIKE) protocol would be to perform a transform of the CPA-secure SIDH to a CCA-secure key exchange scheme. One such transformation is known as the Fujisaki–Okamoto transform. The FO transform takes a CPA-secure protocol and turns it into a CCA-secure one. This is the technique used in many of the third round NIST PQC candidates, including SIKE, the isogeny-based submission.

The method allows for one party to validate the other, but for the ease of exposition, let us suppose that Alice is using a static secret and Bob needs to prove to her that he is performing the protocol correctly.

Bob would prove to Alice that he performed the protocol correctly by executing the key-exchange, encrypting the random seed used to generate his private key and sending this ciphertext to Alice for her to verify that the random seed leads to the correct keys.

This is fully elaborated in [KLM$^+$15]. See also [JAC$^+$17].

Exercise 7. *Verify that this countermeasure will thwart the GPST adaptive attack.*

5. Further directions and readings

In response to the GPST attack, countermeasures that do not involve an FO transform have been proposed. A proposal for such a countermeasure is k-SIDH [AJL17], which implements k^2 instances of SIDH and uses a hash function to combine the outputs of the k^2 shared secrets. The hope is that k-SIDH would be able to withstand the adaptive attack for a small k. However, the GPST adaptive attack has been extended to break this countermeasure (with added computational complexity) [DGL$^+$20, BKM$^+$20].

Extending this work to genus 2 has been completed recently in [KTW21].

References

[AJL17] Reza Azarderakhsh, David Jao, and Christopher Leonardi, *Post-quantum static-static key agreement using multiple protocol instances*, Selected Areas in Cryptography – SAC 2017 (Carlisle Adams and Jan Camenisch, eds.), vol. 10719, Springer International Publishing, 2017, pp. 45–63.

[BKM$^+$20] Andrea Basso, Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Charlotte Weitkämper, *On adaptive attacks against jao-urbanik's isogeny-based protocol*, Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings (Abderrahmane Nitaj and Amr M. Youssef, eds.), Lecture Notes in Computer Science, vol. 12174, Springer, 2020, pp. 195–213.

[DGL$^+$20] Samuel Dobson, Steven D. Galbraith, Jason T. LeGrow, Yan Bo Ti, and Lukas Zobernig, *An adaptive attack on 2-sidh*, Int. J. Comput. Math. Comput. Syst. Theory **5** (2020), no. 4, 282–299.

[GPST16] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti, *On the security of supersingular isogeny cryptosystems*, Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I (Jung Hee Cheon and Tsuyoshi Takagi, eds.), Lecture Notes in Computer Science, vol. 10031, 2016, pp. 63–91.

[JAC$^+$17] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Aaron Hutchinson, Amir Jalali, Koray Karabina, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Geovandro Pereira, Joost Renes, Vladimir Soukharev, and David Urbanik, *Supersingular Isogeny Key Encapsulation*, `https://sike.org`, 2017.

[KLM$^+$15] Daniel Kirkwood, Bradley C. Lackey, John McVey, Mark Motley, Jerome A. Solinas, and David Tuller, *Failure is not an option: Standardization issues for post-quantum key agreement*, 2015, Workshop on Cybersecurity in a Post-Quantum World.

[KTW21] Sabrina Kunzweiler, Yan Bo Ti, and Charlotte Weitkämper, *An adaptive attack on genus-2 sidh*, Cryptology ePrint Archive, Report 2021/990, 2021, `https://ia.cr/2021/990`.

[LL97] Chae Hoon Lim and Pil Joong Lee, *A key recovery attack on discrete log-based schemes using a prime order subgroupp*, Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings, 1997, pp. 249–263.

Appendix A. Pseudocodes

Algorithm 1: Low order adaptive attack using oracle $O(E, R, S)$.

Data: n, E, P_A, Q_A, P_B, Q_B, E_A, $\phi_A(P_B)$, $\phi_A(Q_B)$
Result: α

Set $K_0 \leftarrow 0$;
for $i \leftarrow 0$ **to** $n-1$ **do**
  Choose random (b_1, b_2);
  Set $G_B \leftarrow \langle [b_1]P_B + [b_2]Q_B \rangle$;
  Set $E_B \leftarrow E/G_B$ and let $\phi_B : E \to E_B$ be the isogeny with kernel G_B;
  Set $(R, S) \leftarrow (\phi_B(P_A), \phi_B(Q_A))$;
  Set $j_i \leftarrow$ Query the oracle on $\left(E_B, R, [\ell^{n-i-1}]S\right)$;
  for $x \leftarrow 0$ **to** $\ell - 1$ **do**
    Set $j_{\text{att}} \leftarrow j(E_B/\langle R + [K_i]S + [\ell^{n-1}x]S \rangle)$;
    if $j_{att} = j_i$ **then** $\alpha_i \leftarrow x$;
  end
  Set $K_{i+1} \leftarrow K_i + \alpha_i \ell^i$;
end
Return K_n;

Algorithm 2: Adaptive attack using oracle $O(E, R, S, E')$.

Data: n, E, P_A, Q_A, P_B, Q_B, E_A, $\phi_A(P_B)$, $\phi_A(Q_B)$
Result: α

Set $K_0 \leftarrow 0$;
for $i \leftarrow 0$ **to** $n-3$ **do**
  Set $\alpha_i \leftarrow 0$;
  Choose random (b_1, b_2);
  Set $G_B \leftarrow \langle [b_1]P_B + [b_2]Q_B \rangle$;
  Set $E_B \leftarrow E/G_B$ and let $\phi_B : E \to E_B$ be the isogeny with kernel G_B;
  Set $(R, S) \leftarrow (\phi_B(P_A), \phi_B(Q_A))$;
  Set $E_{AB} \leftarrow E_A/\langle [b_1]\phi_A(P_B) + [b_2]\phi_A(Q_B) \rangle$;
  Set $\theta \leftarrow \sqrt{(1 + 2^{n-i-1})^{-1}} \pmod{2^n}$;
  Query the oracle on $\left(E_B, [\theta](R - [2^{n-i-1}K_i]S), [\theta][1 + 2^{n-i-1}]S, E_{AB}\right)$;
  if *Response is false* **then** $\alpha_i = 1$;
  Set $K_{i+1} \leftarrow K_i + 2^i \alpha_i$;
end
Brute force $\alpha_{n-2}, \alpha_{n-1}$ using E and E_A and $K_{n-2} = \alpha \pmod{2^{n-2}}$ to find α (this requires no oracle calls);
Return α;

Appendix B. Codes to help you get started

Here are some codes that you may use for testing your understanding of SIDH. They are hastily written and I do not recommend that you put them into use for a real application!

```
%%%%  This algorithm will return an SIDH prime given a particular security level.
%%%%  But if you are unlucky, the algorithm won't return a result.
def FindPrime ( sec_level ):
    TwoPower = 2*sec_level
```

```
    ThreePower = Ceil(2*sec_level/Log(2,3))
    for f in list(primes(4, 50)):
        p = 2^TwoPower * 3^ThreePower * f
        if p.is_prime() == True:
            return p, TwoPower, ThreePower, f
    return "Bad luck"

%%%%  Given a prime p, this algorithm returns a random supersingular
%%%%  elliptic curve by taking a ``random'' non-backtracking walk on
%%%%  the supersingular graph.
def Random_SS_Curve ( p ):
    L = GF(p^2)
    iter = Ceil(Log(2,p))
    E0 = EllipticCurve(j=L(1728))
    if not E0.is_supersingular():
        print 'VOOPSIE'
        return 0
    for i in range(0,iter):
        E0_2Tors = E0(0).division_points(2)
        E0_2Tors.remove(E0(0))
        for j in range( 0, len( E0_2Tors ) ):
            E1 = EllipticCurveIsogeny(E0, E0_2Tors[j]).codomain()
            if E0.j_invariant() != E1.j_invariant():
                E0 = E1
                break
    return E1

%%%%  Given parameters of the cryptosystem, this function returns
%%%%  generators of the 2^n and 3^m torsion groups.
def FindBasePoints ( E, TwoPower, ThreePower, f ):
    P_A = 3^ThreePower*f*E.random_element()
    while P_A.order() != 2^TwoPower:
        P_A = 3^ThreePower*f*E.random_element()
    Q_A = 3^ThreePower*f*E.random_element()
    while Q_A.order() != 2^TwoPower or pow(Q_A.weil_pairing(P_A,2^TwoPower),2^(TwoPower-1))==1:
        Q_A = 3^ThreePower*f*E.random_element()

    P_B = 2^TwoPower*f*E.random_element()
    while P_B.order() != 3^ThreePower:
        P_B = 2^TwoPower*f*E.random_element()
    Q_B = 2^TwoPower*f*E.random_element()
    while Q_B.order() != 3^ThreePower or pow(Q_B.weil_pairing(P_B,3^ThreePower),3^(ThreePower-1))==1:
        Q_B = 2^TwoPower*f*E.random_element()

    return [P_A, Q_A, P_B, Q_B]
```