A Transformation for Lifting Discrete Logarithm Based Cryptography to Post-Quantum Cryptography

Danilo Gligoroski*

June 9, 2023

Abstract

This is the second update of this report. In this update, we partially solve Open problem 2 and completely solve Open problem 3 from the previous version. By doing this we address one modification of the Panny's attack done by Nils Langius.

In the first update we introduced conditions for choosing the parameters that render the attacks (both classical and quantum algorithms attacks) proposed by Lorenz Panny in March 2023 on the first variant, inapplicable. For the classical attack, we prove that the discrete logarithms that he was basing his attack upon do not exist for the new parameters. For the quantum algorithm attacks where he proposed computing a basis of a three-dimensional lattice, as proposed in Kitaev's generalization of Shor's quantum algorithm, we prove that for our transformation, the rank of that lattice (the Abelian Stabiliser in Kitaev's terminology) has a rank one, which makes the Kitaev's quantum algorithm inapplicable.

In this paper we construct algebraic structures where rising to the non-associative power indices is no longer tied with the Discrete Logarithm Problem but with a variant of a problem that has been analysed in the last two decades and does not have a quantum polynomial algorithm that solves it. The problem is called Exponential Congruences Problem (ECP). By this, *we disprove* the claims presented in the ePrint report 2021/583 titled "Entropoids: Groups in Disguise" by Lorenz Panny that *"all instantiations of the entropoid framework should be breakable in polynomial time on a quantum computer."*

Additionally, we construct an Arithmetic for power indices and propose generic recipe guidelines that we call "Entropic-Lift" for transforming some of the existing classical cryptographic schemes that depend on the hardness of Discrete Logarithm Problem to post-quantum cryptographic schemes that will base their security on the hardness of the Entropoid variant of the Exponential Congruences Problem (EECP).

As concrete examples, we show how to transform the classical Diffie-Hellman key exchange, DSA and Schnorr signature schemes.

We also post several open problems (two of them now are solved) in relation to EECP and the "Entropic-Lift" transformation.

1 Mathematical preliminaries

We give here some basic definitions and prove some properties about the algebraic structures that we will use. For more definitions reader can consult any standard textbook on abstract algebra topics.

In further text, using multiplicative notation, we will assume that $(G, \cdot)$ is a finite commutative group with operation $\cdot$ and a unit element 1_G. Further, we will assume G has $|G| = q^2$ elements. We will also assume that q can be represented as a product of $\nu = 4$ (in one case it will be $\nu = 3$) distinct prime factors $q = q_1 \dots q_\nu$, (sorted in ascending order).

In general case the number of prime factors ν can be arbitrary, but for our design we will assume that $\nu = 4$ (or $\nu = 3$) and the biggest factor q_ν is as big as possible[1] (for example $q_\nu > 2^{260}$). Thus, we will work with a group G of order $|G| = (q_1\ q_2\ q_3\ q_4)^2$ or $|G| = (q_1\ q_2\ q_3)^2$. We will use a short notation $[\nu]$ to annotate the set $\{1, \dots \nu\}$.

The size of q_ν in this version of the document is mostly influenced by the fact that smaller instances of the initial variant of Entropic-Lift instance called SEQUOA [7] were successfully broken by Lorenz Panny and published in a form of a write-up for a CTF competition in [20]. In the meantime (May, June 2023) Nils Langius has updated Panny's attack [19], and launced an attack exloiting the fact that in the first variant

*Department of Information Security and Communication Technologies, Norwegian University of Science and Technology - NTNU

[1] As the analysis of the Entropic-Lift will deepen in the forthcoming period, this design choice of the size of q_ν might turn out to be too conservative, or too optimistic and will determine the parameters adjustments.

we used small "vanishing" prime factors for q_1 and q_2. We did not know how the choice of small factors q_1 and q_2 affects the security of the scheme, nor how to find generators with bigger q_1 and q_2, and that was stated as Open Problem 2 and Open problem 3. Now after solving open problems 2 and 3 we have an efficient algorithm for finding appropriate group generators for any choice of the prime factors.

Next assumption about the group G is that it is not cyclic, but is generated with two independent elements $g_1, g_2 \in G$, i.e. $\forall x \in G$, $\exists i, j \in \mathbb{Z}_q$, s.t. $x = g_1^i \cdot g_2^j$.

With other words we take that G is a direct product of two maximal cyclic subgroups $G_1 = \langle g_1 \rangle$ and $G_2 = \langle g_2 \rangle$ i.e.

$$G \cong G_1 \times G_2,$$

where the order of G_1 and G_2 is q i.e.

$$| \langle g_1 \rangle | = | \langle g_2 \rangle | = q.$$

From the properties of abelian groups, we have the following

Proposition 1. *For every $x \in G$, there are unique $i, j \in \mathbb{Z}_q$, s.t. $x = g_1^i \cdot g_2^j$.*

Definition 1. *An automorphism α on the group $(G, \cdot)$ is a bijective homomorphism of G to itself i.e.*

- *$\alpha : G \mapsto G$ is a bijection;*
- *α is homomorphism with the respect of the operation $\cdot$ i.e.*

$$\forall x, y \in G, \alpha(x \cdot y) = \alpha(x) \cdot \alpha(y).$$

Definition 2. *An involutive automorphism T on the group $(G, \cdot)$ is an automorphism that is also an involution i.e.*

$$T : G \mapsto G \text{ is bijection,}$$
$$\forall x, y \in G, \quad T(x \cdot y) = T(x) \cdot T(y), \text{ and}$$
$$\forall x \in G, \quad T(T(x)) = T^{(2)}(x) = x.$$

From T being automorphism, the following property holds:

Corollary 1. *For all $x \in G$,*

$$(T(x))^j = T(x^j).$$

Proposition 2. *If $h \in G$ is an element such that h and $T(h)$ are independent elements of order q, then for every $x \in G$ there is a unique pair $(i, j) \in \mathbb{Z}_q \times \mathbb{Z}_q$ such that*

$$x = h^i \cdot T(h^j). \tag{1}$$

Proof. Let us denote by $H_1 = \langle h \rangle$ the subgroup generated by h and $H_2 = \langle T(h) \rangle$ the subgroup generated by $T(h)$. Having h and $T(h)$ being independent means that $H_1 \cap H_2 = \{1_G\}$, $|H_1| = |H_2| = q$ and that $G \cong H_1 \times H_2$. Then the uniqueness of the pair of indices (i, j) follows from the Proposition 1.

□

Definition 3. *For every $x, y \in G$, let us define a non-commutative binary operation*

$$x \boxplus y \equiv x \cdot T(y).$$

We call the algebraic structure $\mathbb{E}_{q^2} = (G, \boxplus, \cdot)$ a finite Entropoid with q^2 elements.

We collect several properties of the operation $\boxplus$ in the following Proposition.

Proposition 3.

- *For $x, y \in G$, in general, $x \boxplus y \neq y \boxplus x$;*
- *The multiplicative unit 1_G acts as a a right zero in $(G, \boxplus)$ i.e. for all $x \in G$, $x \boxplus 1_G = x$;*
- *From the left, 1_G acts as the involution T on x i.e. for all $x \in G$, $1_G \boxplus x = T(x)$;*

Algorithm: Finding a generator of an Entropoid
Input: An Entropoid $\mathbb{E}_{q^2}$, and the prime factorization $q = q_1 \dots q_\nu$ **Output:** Generator g.
1. Choose a random element $g \in G$ 2. Compute the sets $\mathcal{B} = \{b \mid b = g^{(q/q_i)}$, for $i \in [\nu]\}$ and $\mathcal{B}_T = \{b \mid b = T(g)^{(q/q_i)}$, for $i \in [\nu]\}$. 3. If $1_G \in \mathcal{B}$ or $1_G \in \mathcal{B}_T$ then go to Step 1. 4. If $\mathcal{B} \cap \mathcal{B}_T \neq \emptyset$ then go to Step 1. 5. Return g.

Table 1: Finding a generator of and Entropoid.

- *To define a consistent operation $\boxminus$ that will act as "the opposite" operation to $\boxplus$ in $(G, \boxplus)$ we need to define it as: For all $x, y \in G$, $x \boxminus y \stackrel{def}{=} x \cdot T(y^{-1}) = x \boxplus (y^{-1})$. In that case, if $x \boxminus y = z$, then $x = z \boxplus y$.*

Definition 4. *We call an element $g \in G$* a generator of the Entropoid $\mathbb{E}_{q^2}$ *if g and $T(g)$ are independent elements of order q.*

We adapt Algorithm 4.80 from [14] that finds a generator of a cyclic group for finding a generator of an Entropoid.

Lemma 1. *If g is an output of Algorithm defined in Table 1, then it is a generator for the Entropoid $\mathbb{E}_{q^2}$.*

Proof. Since $G \cong G_1 \times G_2$ where the order of G_1 and G_2 is q the computation of the set $\mathcal{B}$ in Step 2, and the check in Step 3 if 1_G belongs to $\mathcal{B}$ ensure that g is a generator of a maximal cyclic subgroup of order q i.e. $|\langle g \rangle| = q$. Similar reasoning applies for the set $\mathcal{B}_T$, that ensures $T(g)$ is a generator of a maximal cyclic subgroup of order q i.e. $|\langle T(g) \rangle| = q$.

The final check in Step 4 checks if g and $T(g)$ are independent elements which from Definition 4 is the necessary condition for g to be a generator for the Entropoid $\mathbb{E}_{q^2}$. □

If we know the prime factorization of the order of a group, another useful algorithm is for finding the multiplicative order of an arbitrary element h of that group. Let us denote that algorithm with $order(h)$. In our particular case $\mathbb{E}_{q^2} = (G, \boxplus, \cdot)$, the order of G is q^2, where $q = q_1 \dots q_\nu$, and since G is not a cyclic, but generated by two independent elements, the order of every element $h \in G$ is not bigger than q, i.e. $order(h) \leq q$. Let us assume that all prime factors of q are distinct, and let us denote by $Divisors(q) = \{d_1 \equiv 1, d_2, d_3, \dots, d_{2^\nu}\}$ the set of all divisors of q. Additionally, let us take that the set $Divisors(q)$ is sorted in ascending order.

Algorithm: $order(h)$
Input: A group G with q^2 elements, the prime factorization $q = q_1 \dots q_\nu$, the sorted set $Divisors(q)$ and an element $h \in G$ **Output:** Integer $d \in Divisors(q)$, where $h^d = 1_G$, and $h^i \neq 1_G, \forall i,$ such that $0 < i < d$.
1. For each $d \in Divisors$: If $h^d = 1_G$: Return d

Table 2: Finding the multiplicative order of $h \in G$. An implicit assumption is that the For loop checks the elements from *Divisors* set in ascending order.

Definition 5. *Let the two-dimensional exponents $X = (x_1, x_2)$ be called* power indices. *For every $g \in G$ we define exponentiation with the power index $X = (x_1, x_2)$ as:*

$$g^X = g^{(x_1, x_2)} = g^{x_1} \cdot T(g^{x_2}).$$

Lemma 2 (Arithmetic of power indices)**.** *Let $X = (x_1, x_2)$ and $Y = (y_1, y_2)$ be two power indices, and let us define the following operations (where mod q acts component-wise):*

addition: $X + Y \equiv (x_1, x_2) + (y_1, y_2) = ((x_1 + y_1),\ (x_2 + y_2)) \mod q$

subtraction: $X - Y \equiv (x_1, x_2) - (y_1, y_2) = ((x_1 - y_1),\ (x_2 - y_2)) \mod q$

multiplication: $XY \equiv (x_1, x_2) \times (y_1, y_2) = ((x_1y_1 + x_2y_2),\ (x_1y_2 + x_2y_1)) \mod q$

division: $\frac{X}{Y} \equiv \frac{(x_1,x_2)}{(y_1,y_2)} = \left(\frac{x_1y_1 - x_2y_2}{y_1^2 - y_2^2},\ \frac{x_2y_1 - x_1y_2}{y_1^2 - y_2^2}\right) \mod q$, *assuming the following condition* $\mathrm{GCD}(y_1^2 - y_2^2, q) = 1$

Then, for every $h \in G$ the following relations hold:

$$h^X \cdot h^Y = h^{(X+Y)},$$

$$h^X \cdot (h^Y)^{-1} = h^{(X-Y)},$$

$$(h^X)^Y = h^{(XY)}.$$

For the division operation the following computational problem is solved: Given, h, X and Y find a power index Z such that

$$h^Z = h_1 \quad \textit{and} \quad h^X = h_1^Y.$$

Proof. While it may seem that the arithmetic expressions are complicated, their consistency can be checked with simple manipulations with algebraic expressions and taking in consideration Definition 2 and Corollary 1. □

Lemma 3. *Let the power index $S = (s_1, s_2)$ is such that* $\mathrm{GCD}(s_1^2 - s_2^2, q) = 1$*. Then the mapping $\sigma : G \mapsto G$ defined as $\sigma(x) = x^S$ is an automorphism of G.*

Proof. The mapping σ is homomorphism since for all $x, y \in G$ we have $\sigma(x \cdot y) = (x \cdot y)^S = (x \cdot y)^{s_1} \cdot T((x \cdot y)^{s_2}) = x^{s_1} \cdot y^{s_1} \cdot T(x^{s_2} \cdot y^{s_2}) = x^{s_1} \cdot y^{s_1} \cdot T(x^{s_2}) \cdot T(y^{s_2}) = x^{s_1} \cdot T(x^{s_2}) \cdot y^{s_1} \cdot T(y^{s_2}) = \sigma(x) \cdot \sigma(y)$.

We prove that σ is injection as follows. Let $\sigma(x_1) = \sigma(x_2)$ i.e. $x_1^{(s_1,s_2)} = x_2^{(s_1,s_2)}$. Let us set $h = \frac{x_1}{x_2} = x_1 \cdot x_2^{-1}$ and $X = (1, 0)$. From the condition that $\mathrm{GCD}(s_1^2 - s_2^2, q) = 1$ it follows that there is a power index $Z = \frac{X}{S} = \left(\frac{s_1}{s_1^2 - s_2^2}, \frac{-s_2}{s_1^2 - s_2^2}\right)$, such that $h_1 = h^Z$ and $h^X = h_1^S$. Computing fist h_1 we have: $h_1 = \left(\frac{x_1}{x_2}\right)^Z = \left(\frac{x_1}{x_2}\right)^{\frac{s_1}{s_1^2 - s_2^2}} \cdot T\left(\left(\frac{x_1}{x_2}\right)^{\frac{-s_2}{s_1^2 - s_2^2}}\right)$. Then replacing h and h_1 in $h^X = h_1^S$ we have:

$$\left(\frac{x_1}{x_2}\right)^{(1,0)} = \left(\left(\frac{x_1}{x_2}\right)^{\frac{s_1}{s_1^2 - s_2^2}} \cdot T\left(\left(\frac{x_1}{x_2}\right)^{\frac{-s_2}{s_1^2 - s_2^2}}\right)\right)^{(s_1,s_2)}$$

which is equivalent to

$$\left(\frac{x_1}{x_2}\right)^{(1,0)} = \left(\left(\frac{x_1}{x_2}\right) \cdot T\left(\left(\frac{x_1}{x_2}\right)\right)\right)^{(1,0)}$$

which further reduces to

$$\frac{x_1}{x_2} = \frac{x_1}{x_2} \cdot T\left(\frac{x_1}{x_2}\right)$$

and further to

$$1_G = T\left(\frac{x_1}{x_2}\right).$$

Since T is involutive automorphism, $T(1_G) = 1_G$, and by applying T on both sides of the last equality we have

$$1_G = \frac{x_1}{x_2},$$

i.e. $x_1 = x_2$. □

In its most general form, Exponential Congruences Problem (ECP) is seeking for a solution of the equation

$$ag_1^{x_1} + bg_2^{x_2} = c \tag{2}$$

in an algebraic structure defined over two operations $(G, +, *)$, with q elements i.e., $|G| = q$, where $a, b, g_1, g_2, c \in G$, g_1 and g_2 have respectively orders s and t i.e., $|\langle g_1 \rangle| = s$ and $|\langle g_2 \rangle| = t$, and where $s, t < q$ but $st \geq q$ [26, 29].

Definition 6. *Let g be the generator of the Entropoid $\mathbb{E}_{q^2}$, and let the DLP be a computationally hard problem over the cyclic subgroup $(G_1, \cdot)$ generated by g i.e. $G_1 = \langle g \rangle$. With other words for a given $y \in G_1$, where $y = g^x$ we assume that there is no (classical) polynomial time algorithm that finds $x \in \mathbb{Z}_q$. **Entropic-Lift** of the DLP is the transformation that replaces the exponents $x \in \mathbb{Z}_q$ with two-dimensional power indices $X = (x_1, x_2)$ where $x_1, x_2 \in \mathbb{Z}_q$. More concretely, for a given $y \in G$, where $y = g^X$ find the power index $X = (x_1, x_2)$.*

Lemma 4. *The elevated DLP is a simplified Exponential Congruence Problem in the Entropoid $\mathbb{E}_{q^2}$, where $a = b = 1$ and $g_1 = g_2 = g$ i.e. has the following form:*

$$y = g^{x_1} \boxplus g^{x_2} \tag{3}$$

Proof. The elevated DLP is the following problem: for a given $y \in G$ it seeks to find $x_1, x_2 \in \mathbb{Z}_q$ s.t. $y = g^{(x_1, x_2)}$. Directly from the definition of exponentiation with power indices in $\mathbb{E}_{q^2}$ we have that $y = g^{x_1} \boxplus g^{x_2}$. The most important part is the fact that g and $T(g)$ are independent, which prevents neither g nor $T(g)$ to be represented as powers of each other. That prevents a collapse of the two-dimensionality of the power indices to a one-dimensional case, which would be the Discrete Logarithm Problem. The obtained variant of the Exponential Congruence expression is indeed a simplified variant of the general ECP where $a = b = 1$ and $g_1 = g_2 = g$. □

Definition 7 (Entropic-Lift of a cryptographic scheme)**.** *Let g be the generator of the Entropoid $\mathbb{E}_{q^2}$, and let $\mathcal{S}(g, \mathcal{A})$ is a cryptographic scheme with a set of algorithms $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\nu\}$ that bases its security on DLP. Let DLP be a computationally hard problem over the cyclic subgroup $(G_1, \cdot)$ generated by g. Let the set of algorithms $\mathcal{A}$ use in total μ exponent variables denoted with $x^{(i)} \in \mathbb{Z}_q$, where $i \in \{1, \ldots \mu\}$. **Entropic-Lift** of the scheme $\mathcal{S}$ is the transformation that replaces all used exponent variables $x^{(i)} \in \mathbb{Z}_q$ in algorithms $\mathcal{A}$, with two-dimensional power indices $X^{(i)} = (x_1^{(i)}, x_2^{(i)})$, $i \in \{1, \ldots \mu\}$. The replacements include all expressions: exponential expressions and arithmetic expressions with ordinary indices from $\mathbb{Z}_q$.*

2 The rationale for using involutive automorphisms $T : G \mapsto G$

The first attempt to use entropic non-commutative and non-associative quasigroups [5] was cryptanalyzed by Panny in [18].

The design idea was to define a general class of groupoids $(G, *)$ (sets G with a binary operation $*$ that is both non-commutative and non-associative) that are *"Entropic"* i.e. for every four elements x, y, z and w, a pseudo-associativity law is satisfied:

$$(x * y) * (z * w) = (x * z) * (y * w).$$

In order to compute the powers x^a where $a \in \mathbb{Z}^+$, of elements $x \in G$, due to the non-associativity, the exact bracketing shape a_s is also required to be known, and Etherington called those general exponentiation indices *power indices*. They can be denoted as pairs $\mathbf{A} = (a, a_s)$. Further, for those entropic groupoids Etherington showed that they satisfy the *"Palintropic"* property i.e., $x^{\mathbf{AB}} = (x^{\mathbf{A}})^{\mathbf{B}} = (x^{\mathbf{B}})^{\mathbf{A}} = x^{\mathbf{BA}}$. Those relations are exactly the Diffie-Hellman key exchange protocol relations used with groups. The work [5] proposed a succinct notation of the exponentially large non-associative power indices. However, the instances proposed there and later in [6] were successfully cryptanalyzed by Panny in [18].

Panny intelligently used a theorem proved by Murdoch [16], Toyoda [28] and Bruck [3]:

Theorem 1 (Theorem 1 in [18])**.** *For every entropic quasigroup $(G, *)$, there exist an abelian group $(G, \cdot)$, commutative automorphisms σ, τ of $(G, \cdot)$, and an element $c \in G$, such that*

$$x * y = x^\sigma \cdot y^\tau \cdot c \quad .$$

Two correct conclusions in the Panny's cryptanalysis were given:

1. "the composition law in *any* entropic quasigroup comes from a multiplication in an abelian group that is twisted by automorphisms and translated by a constant."

2. "any non-associative power of an element $x \in G$ can in fact be written as a product combination in $(G, \cdot)$ of elements of the form x^{ψ} and c^{γ} where $\psi, \gamma \in \langle \sigma, \tau \rangle$."

The second conclusion was supported by the following Lemma:

Lemma 5. *For a binary operation $x * y = x^{\sigma} \cdot y^{\tau} \cdot c$ as in Theorem 1 and any non-associative exponent $\mathbf{A}$, there exists $\gamma \in \mathbb{Z}[\sigma, \tau]$ such that for all $x \in G$*

$$x^{\mathbf{A}} = x^{1+(\sigma+\tau-1)\gamma} \cdot c^{\gamma} \quad . \tag{4}$$

Moreover, if (4) holds for some $x = g \in G$, then (4) holds for all $x \in \langle g \rangle_*$.

Luckily (for the concept of Entropoid cryptography), Panny made one implicit assumption that the commutative automorphisms σ, τ of $(G, \cdot)$ are defined exclusively with the group operation of $(G, \cdot)$ and one element $g \in G$. That assumption led to the following two incomplete conclusions

1. "The classification of finite abelian groups implies that there exists a small subset of such elements that suffices to span the entire subquasigroup $\langle g \rangle_*$ generated by $g \in G$, and again, recovery of the exponents corresponding to Alice's private-key operation consists of a multidimensional discrete-logarithm computation (which is polynomial-time quantumly)."

2. "Therefore, all instantiations of the entropoid framework where a representation of $*$ using $\cdot$ and σ, τ, c can be found efficiently (cf. Section 2.2) should be breakable in polynomial time on a quantum computer."

With other words, assuming that automorphisms σ, τ of $(G, \cdot)$ are exclusively defined with the group operation $\cdot$ and the generator g (basically taking that σ and τ are some fixed integer exponents), indeed the statements in Lemma 1 suggest that the secret power index $\mathbf{A}$ of Alice can be represented in a form that depends on another unknown fixed integer γ, and thus the nature of the problem remains the same: solving the Discrete Logarithm Problem.

However, Theorem 1 does not specify the nature of the automorphisms σ, τ of $(G, \cdot)$. For that matter, the group of all automorphisms $\mathbf{Aut}(G)$ can be very reach, and we can definitely find automorphisms that bijectively and homomorphically (regarding the group operation $\cdot$) are mapping the elements of G to G, but they can not be represented as fixed number of applications exclusively of the internal operation $\cdot$ on g. We used one such involutive automorphism in Definition 2.

In that case, the relation (4) still holds, but the recovery of the exponents corresponding to Alice's private-key operation becomes a search for a power index $\gamma = (\gamma_1, \gamma_2)$. That problem as we showed in previous section reduces to the problem of computing exponential congruences for which there is no polynomial-time quantum algorithm.

Moreover, now we do not need to hide the abelian group $(G, \cdot)$, nor the automorphisms σ and τ. Concretely, as in Lemma 3 we select two power indices $S = (s_1, s_2)$ and $U = (u_1, u_2)$ where $\mathrm{GCD}(s_1^2 - s_2^2, q) = 1$ and $\mathrm{GCD}(u_1^2 - u_2^2, q) = 1$ to define two commuting automorphisms $\sigma, \tau : G \mapsto G$. Then we can take some $c \in G$, and we can again define a non-commutative and non-associative operation

$$x * y = x^{\sigma} \cdot y^{\tau} \cdot c \quad .$$

Then we can apply the techniques from [5] to compute non-associative powers.

As a conclusion of this section, we want to give the following

Remark: It turns out that now, with the transformation Entropic-Lift from Definition 7 we actually do not need the entropic quasigroups and the associated techniques for computing non-associative powers, since we can directly use the arithmetic for the power indices from Lemma 2. Additionally we can try to apply the **Entropic-Lift** recipe for many existing classical cryptographic schemes.

3 A suitable instance for a concrete Entropoid structure

For our concrete instantiation of the group $(G, \cdot)$ with $|G| = q^2$ elements we will use a subgroup of the multiplicative group $(G, \cdot) \equiv C(n, 2)$ of all non-singular $n \times n$ circulant matrices over $\mathbb{F}_2$. Circulant $n \times n$ matrices over any field form a ring with the operations matrix addition and matrix multiplication. That

ring is isomorphic with the quotient ring of polynomials $R = \mathbb{F}_2[x]/(x^n - 1)$. For further reading about circulant matrices I suggest for example [11, 13] and the references there.

Let us introduce the following notations. Let n be a prime number such that the polynomial $x^n - 1$ with coefficients defined over $\mathbb{F}_2$ has the following factorization to irreducible polynomials:

$$x^n - 1 = f_0(x) f_1(x) f_2(x),$$

where $f_0(x) = x - 1$, $deg(f_1) = deg(f_2) = \frac{n-1}{2} = k$ and where polynomials $f_1(x)$ and $f_2(x)$ are mutually reversible to each other i.e.

$$\begin{cases} f_1(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-2} x^{k-2} + a_{k-1} x^{k-1} + a_k x^k, \\ f_2(x) = a_k + a_{k-1} x + a_{k-2} x^2 + \ldots + a_2 x^{k-2} + a_1 x^{k-1} + a_0 x^k, \end{cases}$$

or

$$f_1(x) = x^k \ f_2(1/x).$$

The following result is known about the number of elements of the group $C(n, 2)$.

Proposition 4 (see Corollary 13.2.34, p.505 of [15]). *Assume 2 and n are co-prime. Then*

$$|C(n, 2)| = \prod_{j=0}^{r-1} (2^{m_j} - 1), \tag{5}$$

where $m_0, \ldots, m_{r-1}$ are the degrees of the irreducible factors of $x^n - 1$ over $\mathbb{F}_2$.

In our particular case the expression (5) reduces to the following one:

$$|C(n, 2)| = (2^k - 1)^2. \tag{6}$$

Let $q_0 = 2^k - 1 = p_1 \ p_2 \ \ldots p_\mu$ be the prime factorization of $q_0 = 2^k - 1$, where $p_1, \ldots, p_\mu$ are primes sorted in ascending order. We will be interested in instances where the number of prime factors μ is relatively small, but the biggest factor p_μ is a big prime number (for example $p_\mu > 2^{260}$). Then we will define $q_1 = p_{\mu-3}$, $q_2 = p_{\mu-2}$, $q_3 = p_{\mu-1}$ and $q_4 = p_\mu$, and we will be interested to work in a subgroup of R with an order $q = q_1 \ q_2 \ q_3 \ q_4$. **Note:** In one case we will use only three prime factors i.e. $q = q_1 \ q_2 \ q_3$ where $q_3 = p_\mu$. The reasons for making this exception are explained in the Section 6: "Open research questions about Entropic-Lift".

In Table 3 we give 7 instances for different values of n. Note that the green highlighted values $1303, 1511$ and 2423 are proposed to be instances that offer at least the security of NIST's Level 1, 3 and 5. In the software package for the NIST submission there are also four more instances as challenges. The instance with $n = 103$ defines q as a small number, and it would be an easy challenge, while for challenges with $n = 271, 367, 463$, I hope that the cryptology community will give a significant feedback and analysis.

n	q_1 or $\log_2(q_1)$	q_2 or $\log_2(q_2)$	q_3 or $\log_2(q_3)$	q_4 or $\log_2(q_4)$	Note
103	103	2143	11119	131071	Small n with smooth q
271	23311	262657	348031	45.5061	An n for a challenge
367	367	55633	$2^{61} - 1$	94.9093	An n for a challenge
463	463	599479	59.0120	111.9404	An n for a challenge
1303	72.4086	78.3243	89.9449	260.5034	An n for NIST Level 1
1511	33.8112	48.0261	82.5920	506.6470	An n for NIST Level 3
2423	58.5731	77.0392	961.1956		An n for NIST Level 5

Table 3: We omit the exact numerical values for all factors of q in order to present a compact table. For example for $n = 271$ we have that $q = q_1 \ q_2 \ q_3 \ q_4$ where $q_1 = 23311$, $q_2 = 262657$, $q_3 = 348031$ and $q_4 = 49971617830801 \approx 2^{45.5061}$. The table shows some concrete instances for prime numbers n. Integer values for q_i are the exact values, while floating point numbers are power of 2 approximations.

For the involutive automorphism $T : C \mapsto C$ it turns out that the operation of matrix transposition of elements in C is a suitable operation. It is automorphism, and it is involution. When elements $a \in C$ are being presented as polynomials

$$a = a_0 + a_1 x + a_2 x^2 + \ldots a_{n-2} x^{n-2} + a_{n-1} x^{n-1},$$

the transposition of a is denoted as a^T and

$$a^T = a_0 + a_{n-1}x + a_{n-2}x + \ldots + a_2x^{n-2} + a_1x^{n-1}.$$

Working with elements in the quotient ring $R = \mathbb{F}_2[x]/(x^n - 1)$ let us make the following casting conversion convention between the set of integers $\mathbb{Z}_{2^n} = \{0, 1, \ldots, 2^n - 1\}$ and the elements or R: For every $i \in \mathbb{Z}_{2^n}$ let $i.bits() = [a_0, a_1, a_2, \ldots, a_{n-2}, a_{n-1}]$ be the little-endian binary representation of of i. That means $i = a_0 + a_1 \times 2 + a_2 \times 2^2 + \ldots + a_{n-2} \times 2^{n-2} + a_{n-1} \times 2^{n-1}$. In that case, we say that $a = R(i.bits())$ iff

$$a = a_0 + a_1x + a_2x^2 + \ldots a_{n-2}x^{n-2} + a_{n-1}x^{n-1}.$$

For the following definitions all the credits go to Lorenz Panny and his attack described in [20]. With the irreducible polynomials $f_1(x)$ and $f_2(x)$ let us define two projection maps π_1 and π_2 that are homomorphisms from R to $\mathbb{F}_{2^k}$ as follows:

$$\begin{array}{rcl} \pi_1 : R & \rightarrow & \mathbb{F}_2/f_1(x), \quad \text{where field} \quad \mathbf{F}_1 = \mathbb{F}_2/f_1(x) \cong \mathbb{F}_{2^k} \\ \pi_2 : R & \rightarrow & \mathbb{F}_2/f_2(x), \quad \text{where field} \quad \mathbf{F}_2 = \mathbb{F}_2/f_2(x) \cong \mathbb{F}_{2^k} \end{array}$$

For an element $g \in R$, $q = q_1\ q_2\ q_3\ q_4$ and the set $Divisors(q)$ let us define four sets:

$$\mathcal{B}_1(g) = \{b \mid b = \pi_1(g)^d, \text{ for } d \in Divisors(q)\},$$

$$\mathcal{B}_{1T}(T(g)) = \{b \mid b = \pi_1(T(g))^d, \text{ for } d \in Divisors(q)\},$$

$$\mathcal{B}_2(g) = \{b \mid b = \pi_2(g)^d, \text{ for } d \in Divisors(q)\},$$

$$\mathcal{B}_{2T}(T(g)) = \{b \mid b = \pi_2(T(g))^d, \text{ for } d \in Divisors(q)\}.$$

Now, we are interested in finding generators $g \in R$ that satisfy the following conditions (7):

$$\exists\ g^{-1} \in R \tag{7a}$$
$$order(g) = q \tag{7b}$$
$$order(T(g)) = q \tag{7c}$$
$$g \text{ and } T(g) \text{ are mutually independent} \tag{7d}$$

$$order(\pi_1(g)) < q \quad \text{and} \quad order(\pi_1(g)) \mid q \tag{7e}$$
$$order(\pi_1(T(g))) < q \quad \text{and} \quad order(\pi_1(T(g))) \mid q \tag{7f}$$
$$order(\pi_1(g)) \neq order(\pi_1(T(g))) \tag{7g}$$
$$\mathcal{B}_1(g) \cap \mathcal{B}_{1T}(T(g)) = \{1\}, 1 \in \mathbb{F}_{2^k} \tag{7h}$$

$$order(\pi_2(g)) < q \quad \text{and} \quad order(\pi_2(g)) \mid q \tag{7i}$$
$$order(\pi_2(T(g))) < q \quad \text{and} \quad order(\pi_2(T(g))) \mid q \tag{7j}$$
$$order(\pi_2(g)) \neq order(\pi_2(T(g))) \tag{7k}$$
$$\mathcal{B}_2(g) \cap \mathcal{B}_{2T}(T(g)) = \{1\}, 1 \in \mathbb{F}_{2^k} \tag{7l}$$

$$\begin{array}{c} \text{if } order(\pi_1(T(g))) < order(\pi_1(g)) \text{ then } order(\pi_2(g)) < order(\pi_2(T(g))), \\ \text{or} \\ \text{if } order(\pi_1(g)) < order(\pi_1(T(g))) \text{ then } order(\pi_2(T(g))) < order(\pi_2(g)), \end{array} \tag{7m}$$

The algorithm given in Table 4 is a simple (and very inneficient) search algorithm for finding a generator that satisfies the conditions (7).

Lemma 6. *If $g \in R$ is an element returned by the algorithm defined in Table 4 then g is a generator of an Entropoid $\mathbb{E}_{q^2}$.*

Proof. The relation $g \leftarrow h^{q_0/q}$ from the Repeat-Until loop of the algorithm ensures that g has an order not bigger than q. Then satisfying the conditions (7b) – (7d) ensures that g is a generator of an Entropoid $\mathbb{E}_{q^2}$. □

Algorithm: Finding a generator $g \in R$ that satisfies the conditions (7)
Input: Starting value i_{start}, q_0, $Divisors(q_0)$, q and $Divisors(q)$ **Output:** Element $g \in R$ that satisfies the conditions (7)
1. $i \leftarrow i_{start} - 1$ 2. Repeat $i \leftarrow i + 1$ $h \leftarrow R(i.bits())$ $g \leftarrow h^{q_0/q}$ Until g satisfies conditions (7) 3. Return g, i

Table 4: A simple search algorithm for finding a generator satisfying the conditions (7). The sequential search starts from some submitted value i_{start}.

Remark: It might happen that there is a smaller subset of the conditions (7) that will give equivalent algebraic structures, but finding that smaller set of conditions is left as a future work.

Two important questions raise naturally:

1. Do $g \in R$ as output of the Algorithm in Table 4 exist?
2. What is the expected running time of the Algorithm in Table 4?

We do not know the answer of the second question, but we observe that for smaller values of q_1 and q_2 the algorithm finds the value i faster. For some small values of q_1 and q_2 there is a Table 5 in the previous version of this document.

Before we describe an efficient algorithm for finding generators that satisfy the conditions (7), let us introduce the following definition.

Definition 8. *Let g be a generator of $(G, \cdot)$, $order(g) = q = q_1\ q_2\ q_3\ q_4$, where $|G| = q^2$, and where G is the subgroup of $C(n, 2)$ which has q_0^2 elements, where $q_0 = 2^k - 1 = p_1\ p_2\ \ldots p_\mu$, as defined in this document. For a factor q_i of q we say it is a* vanishing prime factor with respect to the projection $\pi_j, j \in \{1, 2\}$ *of g if $q_i \not| \ order(\pi_j(g))$ i.e. it vanishes as a prime factor of $order(\pi_j(g))$.* A vanishing set of prime factors with the respect of the generator g *or shortly* a vanishing set $V = \{q_{i_1}, q_{i_2}\}$ *is any two-element subset of $\{q_1, q_2, q_3, q_4\}$ where both q_{i_1} and q_{i_2} are vanishing prime factors either for π_1 or for π_2.*

The algorithm `SEQUOA-generator` given in Table 5 is an efficient algorithm for finding a generator satisfying the conditions (7) for a given vanishing set V. We can now choose any vanishing set, but for our instances we choose $V = \{q_1, q_4\}$, i.e. consisting of the smallest and the biggest factor of q. The sequential search starts from some submitted value i_{start}.

4 Analysis of known attacks

4.1 Attacks on the Exponential Congruences Problem (ECP)

In this section we adapt the known algorithms for solving ECP given in [26, 29].

Let us first highlight the differences with the ECP addressed in the open literature and the ECP in this work:

1. In [26, 29], the equation (2) is defined over finite field $\mathbb{F}_q$, where $q = p^k$, p a prime number and the number of elements in the multiplicative group $\mathbb{F}_q^*$ is $q - 1 = p^k - 1$. In our case the equation is over a ringoid structure called Entropoid, $\mathbb{E}_{q^2} = (G, \boxplus, \cdot)$ where $(G, \cdot)$ is a group with $q^2 = q_1^2 \ldots q_\nu^2$ elements, $q_1, \ldots, q_\nu$ prime numbers, and the operation $\boxplus$ is defined with an automorphism $T : G \mapsto G$ that can not be expressed as an exponentiation in G.
2. In [26, 29], g_1 and g_2 are in general different (but the authors also discuss the situations where $g_1 = g_2$) and have respectively orders s and t i.e., $|\langle g_1 \rangle| = s$ and $|\langle g_2 \rangle| = t$, and where $s, t < q$ but $st \geq q$, while in our case $g_1 = g_2 = g$ and $|\langle g \rangle| = q$.
3. Depending on chosen g_1 and g_2, in [26, 29] the equation can have zero, one or many solutions (x_1, x_2), while in our case for g generator of the Entropoid, there exists one unique solution (x_1, x_2).

Lemma 7 (adaptation of Theorem 1 in [29])**.** *Let g be a generator of $\mathbb{E}_{q^2}$. Further, let $y \in G$ be given such that it is a solution of the equation $y = g^{x_1} \boxplus g^{x_2}$, for some $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$. One can find the solution (x_1, x_2) in deterministic time $O(q^{3/2}(\log q))$.*

Algorithm: `SEQUOA-generator`. Finding a generator $g \in R$ that satisfies the conditions (7)

Input: Starting value i_{start}, q_0, $Divisors(q_0)$, q, $Divisors(q)$, vanishing set of factors $V = \{V_0, V_1\}$
Output: Element $g \in R$ that satisfies the conditions (7)

1. $i \leftarrow i_{start} - 1$
2. Repeat
   $i \leftarrow i + 1$
   $h \leftarrow R(i.bits())$

   # Make sure that t in R has order not bigger than q
   $t \leftarrow h^{q_0/q}$

   # (going from R to $\mathbf{F}_1$) Project t with π_1 into $\mathbf{F}_1$
   $t_{\pi_1} \leftarrow \pi_1(t)$

   # Make sure to produce an element in $\mathbf{F}_1$
   # with an order that does not have V_1 as its factor
   $t_1 \leftarrow (t_{\pi_1})^{V_1}$

   # Come back to the ring R and update the value of t
   # by subtracting t_{π_1} and adding t_1
   $t \leftarrow t - R(t_{\pi_1}) + R(t_1)$

   # Find out what is the multiplicative order of the latest value of t
   $ord_t \leftarrow order(t)$

   # Make sure that t has order not bigger than q
   $t \leftarrow t^{ord_t/q}$

   # We have the first generator $g_1 \in R$ that has
   # V_1 as its vanishing factor
   $g_1 \leftarrow t$

   # (going from R to $\mathbf{F}_2$) Project t with π_2 into $\mathbf{F}_2$
   $t_{\pi_2} \leftarrow \pi_2(t)$

   # Make sure to produce an element in $\mathbf{F}_2$
   # with an order that does not have V_0 as its factor
   $t_2 \leftarrow (t_{\pi_1})^{V_0}$

   # Come back to the ring R and update the value of t
   # by subtracting t_{π_2} and adding t_2
   $t \leftarrow t - R(t_{\pi_2}) + R(t_2)$

   # Find out what is the multiplicative order of the latest value of t
   $ord_t \leftarrow order(t)$

   # Make sure that t has order not bigger than q
   $t \leftarrow t^{ord_t/q}$

   # We have the second generator $g_2 \in R$ that has
   # V_0 as its vanishing factor
   $g_2 \leftarrow t$

   # Construct a candidate generator g by multiplying powers of g_1 and g_2
   $g \leftarrow (g_1)^{V_0} \cdot (g_2)^{V_1}$

   Until g satisfies conditions (7)
3. Return g, i

Table 5: `SEQUOA-generator` is an efficient algorithm for finding a generator satisfying the conditions (7) for a given vanishing set V.

Proof. For every $x_2 \in \{0, 1, \ldots, q-1\}$ we evaluate $y \boxminus g^{x_2}$ and then we try to compute its discrete logarithm to base g, that is an integer x_1 with $g^{x_1} = y \boxminus g^{x_2}$. A deterministic algorithm for this problem is the Shanks' Baby Step-Giant Step method [23] which runs in time $O(q^{1/2}(\log q))$ and space $O(q^{1/2})$. In our case the Baby Step-Giant Step method will give us either a solution, or will return that there is no solution for that particular x_2. Combining the run time to go trough all cases for x_2 and the time to solve each instance of the discrete logarithm problem gives us the total worst case complexity of finding the solution of the equation (3) which is $O(q\ q^{1/2}(\log q)) = O(q^{3/2}(\log q))$. □

n	i	$order(\pi_1(g))$	$order(\pi_1(T(g)))$	$order(\pi_2(g))$	$order(\pi_2(T(g)))$
103	1013	$q_1\ q_2\ q_3$	$q_2\ q_3\ q_4$	$q_2\ q_3\ q_4$	$q_1\ q_2\ q_3$
271	1001	$q_1\ q_2\ q_3$	$q_2\ q_3\ q_4$	$q_2\ q_3\ q_4$	$q_1\ q_2\ q_3$
367	1006	$q_1\ q_2\ q_3$	$q_2\ q_3\ q_4$	$q_2\ q_3\ q_4$	$q_1\ q_2\ q_3$
463	1002	$q_1\ q_2\ q_3$	$q_2\ q_3\ q_4$	$q_2\ q_3\ q_4$	$q_1\ q_2\ q_3$
1303	1002	$q_1\ q_2\ q_3$	$q_2\ q_3\ q_4$	$q_2\ q_3\ q_4$	$q_1\ q_2\ q_3$
1511	1002	$q_1\ q_2\ q_3$	$q_2\ q_3\ q_4$	$q_2\ q_3\ q_4$	$q_1\ q_2\ q_3$
2423	1002	$q_1\ q_2$	$q_2\ q_3$	$q_2\ q_3$	$q_1\ q_2$

Table 6: The column i presents the values for which the generator g was computed with the algorithm `SEQUOA-generator`. The serarch for i was started from $i = 1001$.

Corollary 2. *There is a randomized algorithm for finding a solution of the equation $y = g^{x_1} \boxplus g^{x_2}$ that takes $O(q_\nu^{5/2})$ group operations, where $q = q_1 \dots q_\nu$ and the largest prime factor is $q_\nu > q^{1/2}$.*

Proof. We replace the complexity $O(q^{1/2}(\log q))$ of the Baby Step-Giant Step method in Lemma 7, with a randomized algorithm for computing the discrete logarithm that takes $\Omega(\sqrt{q_\nu})$ group operations proposed in Shoup's work [25]. That makes the total running time for solving (2) to be $O(q\ q_\nu^{1/2}) < O(q_\nu^2\ q_\nu^{1/2}) = O(q_\nu^{5/2})$.

□

Lemma 8 (adaptation of Theorem 3 in [29])**.** *Let g be a generator of $\mathbb{E}_{q^2}$, and let $y \in G$ be given such that it is a solution of the equation $y = g^{x_1} \boxplus g^{x_2}$, for some $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$. One can find the solution (x_1, x_2) on a quantum computer in time $O(q^{1/2}(\log\log q))$.*

Proof. For every $x_2 \in \{0, 1, \dots, q-1\}$ we evaluate $y \boxminus g^{x_2}$ and then we use Shor's algorithm [24] to compute its discrete logarithm to base g, that is an integer x_1 with $g^{x_1} = y \boxminus g^{x_2}$ or to report that no such x_1 exists. The expected number of attempts before Shor's algorithm gives us the the answer is $O(\log\log q)$. Let denote by $\mathcal{S}(x_2)$ the subroutine that implements Shor's quantum circuit.

We now use Grover's search algorithm [8] over the subroutine $\mathcal{S}(x_2)$ and the search space $x_2 \in \{0, 1, \dots, q-1\}$. The whole running time is then $O(q^{1/2}(\log\log q))$. □

4.2 Panny's attacks on the first instance of the Entropic-Lift - SEQUOA

4.2.1 Classical attack

Let us first address the classical attack on the initial version of the Entropic-Lift with the instance named **SEQUOA** [7] (March 2023 version of this report). Lorenz Panny described his attack in [20]. A brief summary is as follows:

Let g be a generator of an Entropoid structure $\mathbb{E}_{q^2} = (G, \boxplus, \cdot)$, where the order of G is q^2 and $q = 2^{\frac{n-1}{2}} - 1$.

Let the private key is (a_1, a_2), and let $y = g^{(a_1,a_2)} = g^{a_1} T(g^{a_2})$ be the public key. If we apply the projections π_1 and π_2 on both sides of the equation $y = g^{a_1} T(g^{a_2})$ we get the following system:

$$\pi_1(y) = \pi_1(g)^{a_1} \pi_1(T(g))^{a_2} \tag{8a}$$
$$\pi_2(y) = \pi_2(g)^{a_1} \pi_2(T(g))^{a_2} \tag{8b}$$

We want to express all terms of the projection π_1 in (8a) as exponents of $\pi_1(g)$, and all terms of the projection π_2 in (8b) as exponents of $\pi_2(g)$. More concretely by finding four discrete logarithms in finite fields isomorphic to $\mathbb{F}_{2^k}$

$$\pi_1(y) = \pi_1(g)^{u_1} \tag{9a}$$
$$\pi_1(T(g)) = \pi_1(g)^{c_1} \tag{9b}$$
$$\pi_2(y) = \pi_2(g)^{u_2} \tag{9c}$$
$$\pi_2(T(g)) = \pi_2(g)^{c_2} \tag{9d}$$

we obtain the following system

$$\pi_1(g)^{u_1} = \pi_1(g)^{a_1 + a_2 c_1}$$
$$\pi_2(g)^{u_2} = \pi_2(g)^{a_1 + a_2 c_2}$$

or the following linear system

$$u_1 = a_1 + a_2 c_1 \mod q \tag{11a}$$
$$u_2 = a_1 + a_2 c_2 \mod q \tag{11b}$$

We now analyse the described attack in the case when g is a generator computed by the Algorithm in Table 4.

Lemma 9. *If g is a generator computed by the Algorithm in Table 4, then the equations (9b) and (9d) do not have solutions.*

Proof. Let us suppose that there exist solutions c_1 and c_2 of (9b) and (9d).

Since $\pi_1(T(g)) = \pi_1(g)^{c_1}$, it follows that $order(\pi_1(T(g))) \leq order(\pi_1(g))$. Due to the condition (7g) it must be $order(\pi_1(T(g))) < order(\pi_1(g))$.

Similarly, from $\pi_2(T(g)) = \pi_2(g)^{c_2}$ we conclude that $order(\pi_2(T(g))) < order(\pi_2(g))$.

Thus we have

$$\begin{cases} order(\pi_1(T(g))) < order(\pi_1(g)) \\ \text{and} \\ order(\pi_2(T(g))) < order(\pi_2(g)). \end{cases}$$

But this violates the condition (7m).

□

4.2.2 Quantum attack

Panny's quantum attack on SEQUOA in [20] is in a form of a brief idea how that attack would be designed.

His main argument is the fact that involution $T : R \to R$ is automorphism i.e. $T(h^b) = T(h)^b$ for any h and b. From there, Panny infers that having the public key $y = g^{a_1} T(g)^{a_2}$, a basis of a period latice Λ of the map

$$(\mathbb{Z}^3, +) \longrightarrow R, \text{ where } (u, v, w) \mapsto g^u T(g)^v y^w,$$

can be computed in quantum polynomial time using the approaches given by Boneh and Lipton in [2] or by Kitaev in [9] (these approaches generalize Shor algorithm for finding discrete logarithms). Then, finding a vector in Λ of the form $(a_1, a_2, -1)$ will recover the secret key (a_1, a_2).

We give arguments why neither Boneh-Lipton nor Kitaev methodologies are applicable on Entropic-Lift schemes with parameters chosen by the principles described in this paper.

4.2.2.1 Boneh-Lipton approach First we briefly describe the approach of Boneh and Lipton given in [2]. They introduced a function $h : \mathbb{Z} \to S$ that had a period q i.e. for any $x \in \mathbb{Z}$ the relation $h(x + q) = h(x)$ holds. Thus, the function h could be considered as a function $h : \mathbb{Z}_q \to S$. Then, they defined the property of "hidden linear layer over q" for a function $f : \mathbb{Z}^k \to S$. That is, there are integers $\alpha_2, \ldots, \alpha_k$ and a function h with a period q, such that

$$f(x_1, \ldots, x_k) = h(x_1 + \alpha_2 x_2 + \ldots + \alpha_k x_k)$$

for all integers $x_1, \ldots, x_k$. Under the restrictions that k is linear on $n = \log q$ and the maximal number of preimages is m for any element $z \in S$ i.e. $m = \max_{z \in S} |h^{-1}(z) \mod q|$ and is upper bounded by p, i.e., $m < p$, where p is the smallest prime divisor of q, they proved that there is a quantum algorithm that in random quantum polynomial time in n can recover the values $\alpha_2, \ldots, \alpha_k$ (all of them computed $\mod q$). They called value m the order of h. Additionally they constructed another random quantum polynomial time algorithm for recovering the period of the periodic function $h : \mathbb{Z} \to S$.

Then, they applied the developed quantum polynomial time algorithms to compute the discrete logarithm over an arbitrary group G as follows. The role of function h is given to a homomorphism $h : \mathbb{Z} \to G$. Given a value $\beta \in G$ where $\beta = h(\alpha)$ the goal is to compute the smallest positive integer x such that $\beta = h(x)$. Taking that a concrete element $h(1) \in G$ has some order d, leads to the conclusion that homomorphism h has a period d. Finding the period d can be performed by one of the developed quantum algorithms.

Next, they defined a function $f : \mathbb{Z}^2 \to G$ as $f(x, y) = h(x + \alpha y)$. Apparently, the function f has a hidden linear form over d of order 1. Additionally, having access to function h, the computation of f is efficiently doable

$$f(x, y) = h(x + \alpha y) = h(x)h(\alpha y) = h(x)h(\alpha)^y = h(x)\beta^y \ .$$

Then, invoking the quantum polynomial time algorithm for finding a value $\alpha \mod d$ solves the discrete logarithm problem.

Let us see how Boneh-Lipton approach applies to SEQUOA. The first difference is that now the homomorphism function h is $h : \mathbb{Z}^2 \to G$ (more concretely for a given $g \in G$ $h_g : (a_1, a_2) \mapsto g^{a_1} T(g)^{a_2}$). The function h has a two-dimensional period (q, q) i.e. for any $x = (x_1, x_2) \in \mathbb{Z}^2$ the relation $h((x_1, x_2) + (q, q)) = h((x_1, x_2))$ holds. Thus, the function h could be considered as a function $h : \mathbb{Z}_q \times \mathbb{Z}_q \to G$. An analogous definition for a "hidden layer over (q, q)" for a function $f : (\mathbb{Z}_q \times \mathbb{Z}_q)^k \to G$ is that there are pairs $\alpha_2, \ldots, \alpha_k$ where $\alpha_i = (\alpha_{i,1}, \alpha_{i,2})$ and a function h with a period (q, q), such that

$$f((x_{1,1}, x_{1,2}), \ldots, (x_{k,1}, x_{k,2})) = h((x_{1,1}, x_{1,2}) + \alpha_2(x_{2,1}, x_{2,2}) + \ldots + \alpha_k(x_{k,1}, x_{k,2}))$$

for all pairs $(x_{1,1}, x_{1,2}), \ldots, (x_{k,1}, x_{k,2})$. Note that multiplications of pairs modulo q is defined in Lemma 2.

The main difficulty now for applying Boneh-Lipton algorithm for finding α_i is that they are two-dimensional vectors with two independent degrees of freedom. The proof technique of their main theorem first shows how to find one-dimensional α, which then is extended sequentially for cases with more alphas. In the case of **SEQUOA** the simplest case with one α_1 is that $\alpha_1 = (\alpha_{1,1}, \alpha_{1,2})$. Note that we can not simply replace the problem of finding $k-1$ two-dimensional vectors $\alpha_i = (\alpha_{i,1}, \alpha_{i,2})$ for $h : \mathbb{Z}_q \times \mathbb{Z}_q \to G$ with a problem of finding $2(k-1)$ one-dimensional values $\alpha_2, \ldots, \alpha_{2(k-1)}$.

4.2.2.2 Kitaev approach We now briefly describe the approach of Kitaev given in [9]. Kitaev introduced the Abelian Stabilizer Problem. It is defined for Abelian groups G acting on a finite set $M \subseteq \{0,1\}^n$. Additionally, since any finitely generated Abelian group is a homomorphic image of $\mathbb{Z}^k$ the group can be taken to be $G = \mathbb{Z}^k$. The pair (k, n) is called the size of the problem, and for any element $x \in M$ an action (a function) $F : \mathbb{Z}^k \times M \to M$ is defined such that

$$F(0, x) = x, \quad F(g + h, x) = F(g, F(h, x)), \quad \text{for any } g, h \in \mathbb{Z}^k,\ x \in M.$$

A stabilizer of an element $a \in M$ with respect to the function F is the set $St_F(a) = \{g \in \mathbb{Z}^k : F(g, a) = a\}$. The stabilizer $St_F(a)$ is a subgroup in $\mathbb{Z}^k$ and is isomorphic to $\mathbb{Z}^k$. It has a basis $(g_1, \ldots, g_k)$. Abelian Stabilizer Problem is to find a basis of $St_F(a)$.

How discrete logarithm is reduced to the ASP? The role of the set M is given to the ring of integers modulo q, and G is the group of invertible elements of M. Then, if $g_1, \ldots, g_k \in G$, the function $F : \mathbb{Z}^k \times M \to M$ is defined as the action

$$F \equiv F_{g_1,\ldots,g_k} : (m_1, \ldots, m_k, x) \mapsto g_1^{m_1} \cdots g_k^{m_k} x,$$

where $m_i \in \mathbb{Z},\ x \in M$.

More concretely, for q being a prime number, $\zeta \in G \cong \mathbb{Z}_{q-1}$ being a primitive element, and $g \in G$ an arbitrary element, we can find the discrete logarithm m such that $\zeta^m = g$ in the following way. We search for a basis of the stabilizer of 1 with the respect to $F_{\zeta,g}$. The stabilizer is the set $P = St_{F_{\zeta,g}}(1) = \{(m, r) \in \mathbb{Z}^2 : \zeta^m g^r 1 = 1\}$. Given a basis for the stabilizer $P \subseteq \mathbb{Z}^2$, it is easy to find an element $(m, -1) \in P$. Then, m is the solution to the discrete logarithm problem $\zeta^m = g$.

Kitaev's quantum algorithm that finds the basis of the stabilizer $St_F(a) = \{g \in \mathbb{Z}^k : F(g, a) = a\}$ for any instance (k, n, a, F) is an algorithm that produces random elements of $St_F(a)$ (actually random elements of homomorphically equivalent sets to $St_F(a)$). He showed that collecting $l = n + 4$ such random elements with high probability ensures finding a basis of $St_F(a)$.

Let us see how Kitaev approach applies to **SEQUOA**. The role of M we give to $R = \mathbb{F}_2[x]/(x^n - 1)$, and the action $F : \mathbb{Z}^k \times R \to R$ is defined with respect to g, $T(g)$ and y as

$$F \equiv F_{g,T(g),y} : (m_1, m_2, m_3, z) \mapsto g^{a_1} T(g)^{a_2} y^{a_3} z$$

where $m_i \in \mathbb{Z},\ g, T(g), y, z \in M$.

In this case the stabilizer of 1 is the set $P = St_{F_{g,T(g),y}}(1) = \{(m_1, m_2, m_3) \in \mathbb{Z}^3 : g^{m_1} T(g)^{m_2} y^{m_3} 1 = 1\}$. Once we find a basis for P, we can compute the vector $(m_1, m_2, -1) \in P$, and thus we have computed the discrete logarithm $(a_1, a_2) = (m_1, m_2)$ of $y = g^{a_1} T(g)^{a_2}$. But the following Theorem proves that collecting random elements of P will not help finding the vector $(m_1, m_2, -1) \in P$.

Theorem 2. *The rank of the set* $P = St_{F_{g,T(g),y}}(1) = \{(m_1, m_2, m_3) \in \mathbb{Z}^3 : g^{m_1} T(g)^{m_2} y^{m_3} 1 = 1\}$ *is 1.*

Proof. Let us start with the fact that $GCD(a_1, q) = 1$ and $GCD(a_2, q) = 1$. The consequence of that for the element $y = g^{a_1} T(g)^{a_2}$ is: $order(y) = q$ (elements can not have orders bigger than q). Let us now analyse the set $\Gamma = \{y^{m_3} : m_3 \in \{0, \ldots q-1\}\}$. For every element $z = y^{m_3} \in \Gamma$ due to Proposition 1 there exist unique pair (o_1, o_2) such that $g^{o_1} T(g)^{o_2} = y^{m_3}$. Let us look at the set of those triplets $O = \{(o_1, o_2, m_3) : g^{o_1} T(g)^{o_2} = y^{m_3}\}$. Apparently, $O \subseteq P$, and $O \cong P \mod q$. Moreover, since m_3

takes all the values from the range $\{0, \dots q-1\}$, it follows that the values o_1 must also take all the values from the range $\{0, \dots q-1\}$ (otherwise there would be a repetition for o_1 which would violate the condition $GCD(a_1, q) = 1$), and the same applies for o_2. Let us now look at the element $b_1 = (1, m'_2, m'_3) \in O$. Knowing b_1 we can generate all elements in O (and thus all elements in P) by simply multiplying it with every $\alpha \in \{0, \dots q-1\}$ i.e. $O = \{\alpha \times (1, m'_2, m'_3) \ : \ \alpha \in \{0, \dots q-1\}\}$. That proves that the rank of P is one. □

A direct consequence of Theorem 2 and adaptation of Lemma 8 is the following

Corollary 3. *One can find the vector* $b_1 = (1, m'_2, m'_3) \in O$ *on a quantum computer in time* $O(q^{1/2}(\log\log q))$.

Proof. For every $m'_3 \in \{0, 1, \dots, q-1\}$ we evaluate $g^{-1}y^{-m'_3}$ and then we use Shor's algorithm to compute its discrete logarithm to base g, that is an integer m'_2 with $g^{m'_2} = g^{-1}y^{-m'_3}$ or to report that no such m'_2 exists. The expected number of attempts before Shor's algorithm gives us the the answer is $O(\log\log q)$. Let denote by $\mathcal{S}(x_2)$ the subroutine that implements Shor's quantum circuit.

We then use Grover's search algorithm [8] over the subroutine $\mathcal{S}(x_2)$ and the search space $m'_2 \in \{0, 1, \dots, q-1\}$. The whole running time is then $O(q^{1/2}(\log\log q))$. □

We will demonstrate Theorem 2 for small $n = 71$. Beside the example given below, there is an accompanied Jupyter notebook `SEQUOA_n_71_demo_Kitaev_rank_1.ipynb` in the NIST submission package (in the folder `/Additional_Implementations/SageMath/`). To run the script you should have SageMath [27] installed as well as Jupyter [10] to run the SageMath script in a browser.

Example 1. For $n = 71$ we have that

$$\begin{cases} f_1(x) = & x^{35} + x^{33} + x^{28} + x^{27} + x^{26} + x^{25} + x^{24} + x^{17} + x^{13} + x^8 + x^7 + x^5 + x^4 + x + 1 \\ f_2(x) = & x^{35} + x^{34} + x^{31} + x^{30} + x^{28} + x^{27} + x^{22} + x^{18} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^2 + 1 \end{cases}$$

Thus, we work with the multiplicative group $C(71, 2)$ of the ring of non-singular 71×71 circulant matrices over $\mathbb{F}_2$, i.e. with the quotient ring of polynomials $R = \mathbb{F}_2[x]/(x^{71} - 1)$. According to expression (6) the number of elements in $C(71, 2)$ is $C(71, 2) = q_0^2$, where $q_0 = 2^{35} - 1$. The prime factorization of q_0 is: $q_0 = 31 \times 71 \times 127 \times 122921$.

For demonstration purpose, we will choose the three smallest factor to define the subgroup where we will work. Namely we choose $q = 31 \times 71 \times 127$.

After running the Algorithm for finding an appropriate generator that will satisfy conditions (7) starting from $i = 1$, the algorithm will return the value $i = 632$, for which $g = R(i.bits())^{q_0/q}$. More concretely,

$$\begin{aligned} g = & x^{70} + x^{66} + x^{64} + x^{62} + x^{61} + x^{60} + x^{58} + x^{54} + x^{52} + x^{51} + \\ & + x^{50} + x^{46} + x^{43} + x^{42} + x^{40} + x^{38} + x^{33} + x^{32} + x^{31} + x^{29} + \\ & + x^{28} + x^{25} + x^{24} + x^{22} + x^{21} + x^{19} + x^{17} + x^{16} + x^{15} + x^{13} + \\ & + x^{12} + x^{11} + x^7 + x^6 + x^3 \end{aligned}$$

Let $a = (a_1, a_2) = (51205, 232032)$ be a secret power index. In that case

$$\begin{aligned} y = g^a = & x^{70} + x^{68} + x^{67} + x^{65} + x^{64} + x^{61} + x^{58} + x^{56} + x^{55} + x^{54} + \\ & + x^{52} + x^{51} + x^{49} + x^{41} + x^{40} + x^{39} + x^{36} + x^{28} + x^{26} + x^{24} + \\ & + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{17} + x^{13} + x^{11} + x^6 + x^4 + \\ & + x^3 + x^2 + 1 \end{aligned}$$

We are interested in the stabilizer of 1 which is the set of triplets $P = St_{F_{g,T(g),y}}(1) = \{(m_1, m_2, m_3) \in \mathbb{Z}^3 \ : \ g^{m_1}T(g)^{m_2}y^{m_3}1 = 1\}$. Even for these small parameters, if we nest three for loops, we would need around 2^{54} tests, which is too costly for a small laptop.

But we can start by fixing $m_1 = 1$ and by computing two lists

$$\begin{cases} all\_y\_powers & = [y^{-m_3} \ : \ \text{for } m_3 \in \mathbb{Z}_q] \\ all\_g\_gT\_produtcs & = [g * T(g)^{m_2} \ : \ \text{for } m_2 \in \mathbb{Z}_q]. \end{cases}$$

Then we can find the intersection between these two lists. According to Theorem 2, the intersection should have only one element, and we can determine the index of the intersection in the *all_y_powers* list.

Running that part of the script gives that the basic triplet is $b_1 = (1, 5731, 59219) = (1, m'_2, m'_3)$

We can then update the list $all\_g\_gT\_produtcs$ by multiplying all its elements by g and again find all intersection elements between the lists all_y_powers and $all\_g\_gT\_produtcs$. Again, the intersection should have only one element and that element would be $b_2 = (2, 11462, 118438)$. We can check that $b_2 = 2 \times b_1 \mod q$. In a similar manner we will find $b_3 = (3, 17193, 177657) = 3 \times b_1 \mod q$ and so on.

Also, having $b_1 = (1, 5731, 59219) = (1, m'_2, m'_3)$ we can compute $a_1 = (q - m'_3)^{-1} \mod q = 51205$ and $a_2 = m'_2 * a_1 \mod q = 232032$.

4.2.3 Panny-Langius attack

After publishing the second version of SEQUOA, Nils Langius sent me an email [12] with a description of an updated Panny's attack. We describe his attack briefly here (but with the terminology of vanishing sets introduced in this version).

Let g be a generator of $(G, \cdot)$, $order(g) = q = q_1\ q_2\ q_3\ q_4$, where $|G| = q^2$, and where G is the subgroup of $C(n, 2)$ which has q_0^2 elements, where $q_0 = 2^k - 1 = p_1\ p_2\ \ldots p_\mu$. For simplicity of notation let us take that the vanishing set with respect of g is $V = \{q_1, q_2\}$. Let $a = (a_1, a_2)$ be a secret power exponent, and let $y = g^{(a_1,a_2)}$ is a public value.

The Panny-Langius attack for recovering $a = (a_1, a_2)$ from the knowledge of g, y, $order(g) = q$ and $V = \{q_1, q_2\}$ goes like this:

1. Produce a new generator $g_{new} = g^{q_1 q_2}$ and a new $y_{new} = y^{q_1 q_2}$
2. Now g_{new} has order $order(g_{new}) = q_3 q_4$, and with big probability there is no vanishing set for g_{new} i.e. $order(g_{new}) = order(\pi_1(g_{new})) = order(\pi_1(T(g_{new})))$ (also a similar relation holds for π_2). Thus, the original Panny's attack can be applied on g_{new} and y_{new} to find (c_1, c_2) in $\mathbb{Z}_{q_3 q_4}$. Solving the discrete logarithms for Panny's attack in this case will be in a subgroup of R with $q_3 q_4$ elements. Panny's attack will recover $(b_1, b_2) = (a_1, a_2) \mod q_3 q_4$. The last relation can be written as follows:
$$\begin{cases} a_1 = & b_1 + k_1\ q_3 q_4, \text{ where } k_1 < q_1 q_2 \\ a_2 = & b_2 + k_2\ q_3 q_4, \text{ where } k_2 < q_1 q_2 \end{cases}$$
3. To recover $(a_1, a_2) \mod q$ we need to search trough a space of $O(q_1, q_2)$ elements.

For the previous instance of SEQUOA we did not know how to find Entropoid generators with arbitrary vanishing set. The simple search algorithm in Table 4 could find in a very slow manner only generators with elements in the vanishing set $V = \{q_1, q_2\}$ where the prime numbers q_1 and q_2 were very small.

On the other hand, now, with the algorithm `SEQUOA-generator` we can find generators with a vanishing set $V = \{q_1, q_4\}$ where $q_4 > 2^{260}$. This, addresses the Panny-Langius attack.

5 Examples of Entropic-Lift

Example 2. Entropic-Lift for classical Diffie-Hellman key exchange protocol is just a straightforward variables replacement. For the classical case, Alice and Bob agree on a finite cyclic group $(G, \cdot)$ of order n and a generating element $g \in G$. For the Entropic Diffie-Hellman, Alice and Bob agree on a finite Entropoid $\mathbb{E}_{q^2} = (G, \boxplus, \cdot)$ with q^2 elements and a generating element $g \in G$.

Example 3. Entropic-Lift for Schnorr signature scheme [22] is also a straightforward variables replacement. In the classical case all users agree on a finite cyclic group $(G, \cdot)$ of prime order q and a generating element $g \in G$, in which the Discrete Logarithm Problem is assumed to be hard. Also, all users agree on a cryptographic hash function $H : \{0, 1\}^* \mapsto \mathbb{Z}_q$.

In the Entropic Schnorr case, all users agree on a finite Entropoid $\mathbb{E}_{q^2} = (G, \boxplus, \cdot)$ with q^2 elements and a generating element $g \in G$, in which the Exponential Congruences Problem is assumed to be hard. All users also agree on a cryptographic hash function $H : \{0, 1\}^* \mapsto \mathbb{Z}_q \times \mathbb{Z}_q$.

In Table 9 we present a variant of Entropic Schnorr signature implemented in C++ and submitted for inclusion in Supercop [1] for testing and measurement. Three different instances are submitted to Supercop with prime number $n = 1303, 1511, 2423$. The design goal for this variant was to offer some of the desirable features for signature schemes named as "BUFF - Beyond UnForgeability Features" [4].

Additional motivation was to offer a randomized signature scheme which in a case of complete deterioration of the entropy pool for its randomness, the scheme will become a deterministic scheme. That part is the step "Set $k = (k_1, k_2) \leftarrow H(rand||PrivateKey||M)$, where $rand \xleftarrow{\$} \{0, 1\}^n$ is a sequence of at least n randomly generated bits" in the signing part.

A more detailed description about the concrete implementation of SEQUOA is given in the NIST submission package.

(a) Classical Diffie–Hellman key exchange

1. Alice picks a random natural number a where $1 < a < n$, and sends the element g^a of G to Bob.

2. Bob picks a random natural number b where $1 < b < n$, and sends the element g^b of G to Alice.

3. Alice computes the element $SharedKey = \left(g^b\right)^a = g^{ba}$ of G.

4. Bob computes the element $SharedKey = \left(g^a\right)^b = g^{ab}$ of G.

(b) Entropic Diffie–Hellman key exchange

1. Alice picks a random power index $a = (a_1, a_2)$ where $GCD(a_1, q) = 1$ and $GCD(a_2, q) = 1$, and sends the element g^a of G to Bob.

2. Bob picks a random power index $b = (b_1, b_2)$ where $GCD(b_1, q) = 1$ and $GCD(b_2, q) = 1$, and sends the element g^b of G to Alice.

3. Alice computes the element $SharedKey = \left(g^b\right)^a = g^{ba}$ of G.

4. Bob computes the element $SharedKey = \left(g^a\right)^b = g^{ab}$ of G.

Table 7: Classical and Entropic Diffie-Hellman key exchange protocol. The Entropic variant is just a straightforward variables replacement

(a) Classical Schnorr signature scheme

KeyGen

$PrivateKey \equiv x \xleftarrow{\$} \mathbb{Z}_q^*, \quad PublicKey \equiv y = g^x$

Sign a message M

Choose random $k \xleftarrow{\$} \mathbb{Z}_q^*$
$r = g^k$
$e = H(r||M)$
$s = k - xe$
$Signature = (s, e)$

Verify

$r_v = g^s y^e$;
$e_v = H(r_v||M)$
Return $True$ if $e_v = e$ else Return $False$

(b) Entropic Schnorr signature scheme

KeyGen

$PrivateKey \equiv x = (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$, where $GCD(x_1, q) = 1$ and $GCD(x_2, q) = 1$
$PublicKey \equiv y = g^x$

Sign a message M

Choose random $k = (k_1, k_2) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$, where $GCD(k_1, q) = 1$ and $GCD(k_2, q) = 1$
$r = g^k$
$e = H(r||M)$
$s = k - xe$
$Signature = (s, e)$

Verify

$r_v = g^s y^e$;
$e_v = H(r_v||M)$
Return $True$ if $e_v = e$ else Return $False$

Table 8: Classical and Entropic Schnorr signature scheme. The Entropic variant is just a straightforward variables replacement.

Example 4. Entropic-Lift for DSA scheme is not so straightforward. The definition of the scheme involves variables that in one part play a role in one group, and later they are interpreted as members of another group. That would cause incompatible arithmetic operations between power indices and group elements in the Entropoid structure.

If we face a situation of incompatible operation in the Entropoid case, as suggested in the Entropic-Lift recipe, we can try to replace the involved group element variable with a cryptographic hash of that variable that maps it to a power index, thus enabling a proper arithmetic operation between power indices. We should also re-evaluate the lifted scheme to check if that altered expression still makes sense and is secure.

We describe here the classical DSA, without much details of the bit sizes of some of the variables since they are not crucially important for the purpose of this example of Entropic-Lift transformation. For more details we redirect the user to see the detailed definition of DSA [21].

In the classical DSA case all users agree on an N-bit prime q and L-bit prime p such that $p-1$ is multiple of q. A generator g is chosen to be in a form of $g = h^{(p-1)/q} \bmod p$. Also, all users agree on a cryptographic hash function $H : \{0,1\}^* \mapsto \mathbb{Z}_q$.

KeyGen

$PrivateKey \equiv x = (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$, where $GCD(x_1, q) = 1$ and $GCD(x_2, q) = 1$
$PublicKey \equiv y = g^x$

Sign a message M
Set $k = (k_1, k_2) \leftarrow H(rand||PrivateKey||M)$, where $GCD(k_1, q) = 1$ and $GCD(k_2, q) = 1$ and where $rand \xleftarrow{\$} \{0,1\}^n$ is a sequence of at least n randomly generated bits.
$r = g^k$
$e = H(r||PublicKey||M)$
$s = k - xe$
$Signature = (s, e)$

Verify
$r_v = g^s y^e$;
$e_v = H(r_v||PublicKey||M)$
Return $True$ if $e_v = e$ else Return $False$

Table 9: SEQUOA variant of Entropic Schnorr signature scheme. This variant is BUFF friendly.

In the Entropic DSA case, all users agree on a finite Entropoid $\mathbb{E}_{q^2} = (G, \boxplus, \cdot)$ with q^2 elements and a generating element $g \in G$, in which the Entropoid Exponential Congruences Problem is assumed to be hard. All users also agree on a cryptographic hash function $H : \{0,1\}^* \mapsto \mathbb{Z}_q \times \mathbb{Z}_q$.

Notice that in the Entropic variant the arithmetic expression for s does not have the term $x \cdot r$, but $x \cdot H(r)$. That is because in this case $x \in \mathbb{Z}_q \times \mathbb{Z}_q$, while $r \in G$, so the multiplication $x \cdot r$ is not well defined. On the other hand, $H(r) \in \mathbb{Z}_q \times \mathbb{Z}_q$, so the expression $x \cdot H(r)$ is a valid arithmetic operation.

(a) Classical DSA scheme

KeyGen

$PrivateKey \equiv x \xleftarrow{\$} \mathbb{Z}_q^*$,
$PublicKey \equiv y = g^x \bmod p$

Sign a message M

Choose random $k \xleftarrow{\$} \mathbb{Z}_q^*$
$r = \left(g^k \bmod p\right) \bmod q$
$s = \left(k^{-1}\left(H(M) + x \cdot r\right)\right) \bmod q$
$Signature = (r, s)$

Verify
$w = s^{-1} \bmod q$;
$u_1 = H(M) \cdot w \bmod q$; $u_2 = r \cdot w \bmod q$
$v = (g^{u_1} y^{u_2} \bmod p) \bmod q$;
Return $True$ if $v = r$ else Return $False$

(b) Entropic DSA scheme

KeyGen

$PrivateKey \equiv x = (x_1, x_2) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$, where $GCD(x_1, q) = 1$ and $GCD(x_2, q) = 1$
$PublicKey \equiv y = g^x$

Sign a message M

Choose random $k \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$, where $GCD(k_1, q) = 1$ and $GCD(k_2, q) = 1$
$r = g^k$
$s = k^{-1}\left(H(M) + x \cdot H(r)\right)$
$Signature = (r, s)$

Verify
$v_1 = r^s$
$v_2 = g^{H(M)} \cdot y^{H(r)}$

Return $True$ if $v_1 = v_2$ else Return $False$

Table 10: Classical and Entropic DSA scheme.

6 Open research questions about Entropic-Lift

The biggest and most important question is

Open Problem 1. *Is the Entropoid variant of the Exponential Congruences Problem (EECP) as hard as the the original ECP?*

The arguments presented in this document support the conjecture that the answer is affirmative. But more analysis is needed.

All instances discussed in this document (see Table 3) work with groups of order $q = q_1\ q_2\ q_3\ q_4$ except for the case for $n = 2423$, where we work with a group of order $q = q_1\ q_2\ q_3$.

Open Problem 2 (partially solved)**.** *Does the used number of prime factors of q affects the security of the scheme? Can we use three prime factors (two small, and one big) without significant loss of security?*

The answer to the first question is: Yes. If the vanishing set $V = \{q_{i_1}, q_{i_2}\}$ consist of two small prime factors, then the Panny-Langius attack breaks the system with complexity $O(q_{i_1} q_{i_2})$ after computing several discrete logarithms in a subgroup of R with $q_3 q_4$ elements.

An affirmative answer to the second question could lead to reduction of the size of the signatures.

Remark: The following discussion about the next open problem remains here just for a reference and a context.

A related question can be posted about the size of q_1 and q_2. In Table 6 (the Table in the previous eprint submission) we showed the outputs for i by which we can compute the appropriate generators g. However, those cases are obtained when we used relatively small prime factors of q. On the other hand, if q_1 and q_2 are some of the bigger prime factors of q, then the simple sequential search takes too much time to find the values i.

Open Problem 3 (solved with the algorithm `SEQUOA-generator`)**.** *Is there an efficient algorithm for finding a generator $g \in R$ that satisfies the conditions (7), for big prime factors of q?*

Open Problem 4. *From the perspective of provable security, and specifically from the perspective of security of post-quantum cryptographic schemes, precisely formalize and analyze the potentials and limits of the Entropic-Lift transformation.*

7 Conclusions

We offered a construction of algebraic structures, where rising to the non-associative power indices is no longer tied with the Discrete Logarithm Problem, but with a variant of a problem that in the last two decades has been analyzed and does not have a quantum polynomial algorithm that solves it. The problem is called Exponential Congruences Problem.

We also developed Arithmetic for the power indices. As a result, we proposed a generic recipe guidelines that we named "Entropic-Lift" for transforming some of the existing classical cryptographic schemes that depend on the hardness of Discrete Logarithm Problem to post-quantum cryptographic schemes that will base their security on the hardness of the Entropoid Exponential Congruences Problem (EECP).

We demonstrated the Entropic-Lift on three concrete examples: transforming the classical Diffie-Hellman key exchange, Schnorr and DSA signature schemes.

We also posted several open problems in relation to Entropic-Lift transformation.

Acknowledgements

I would like to thank Nils Langius for sharing his attack and findings about SEQUOA.

I would like to thank Lorenz Panny for the time spent with me during EUROCRYPT 2023 in Lyon, France, on discussions about my ideas how to fix the previous version of SEQUOA. Although Lorenz was not happy with my approach, his feedback is most appreciated.

I would also like to thank my former PhD student Mattia Veroni for long discussions that we spent during the Summer of 2022 analyzing the Panny's attack on the previous Entropoid schemes. I thank Daniel Nager for sharing his preprint [17] in the beginning of November 2022, and I thank Daniel Nager and Danny Niu for comments that improved this and previous versions of this text.

References

[1] Daniel J. Bernstein and Tanja Lange. (editors), eBACS: ECRYPT Benchmarking of Cryptographic Systems. accessed 3 March 2023. https://bench.cr.yp.to.

[2] Dan Boneh and Richard J Lipton. Quantum cryptanalysis of hidden linear functions. In *Advances in Cryptology—CRYPTO'95: 15th Annual International Cryptology Conference Santa Barbara, California, USA, August 27–31, 1995 Proceedings 15*, pages 424–437. Springer, 1995.

[3] Richard H Bruck. Some results in the theory of quasigroups. *Transactions of the American Mathematical Society*, 55:19–52, 1944.

[4] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. Buffing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1696–1714. IEEE, 2021.

[5] Danilo Gligoroski. Entropoid Based Cryptography. 2021. https://eprint.iacr.org/2021/469.

[6] Danilo Gligoroski. Rebuttal to claims in section 2.1 of the eprint report 2021/583" entropoid-based cryptography is group exponentiation in disguise". *Cryptology ePrint Archive*, 2021.

[7] Danilo Gligoroski. A transformation for lifting discrete logarithm based cryptography to post-quantum cryptography. Cryptology ePrint Archive, Paper 2023/318, 2023. https://eprint.iacr.org/2023/318.

[8] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[9] A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.

[10] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. Jupyter notebooks - a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Scmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, Netherlands, 2016. IOS Press.

[11] Irwin Kra and Santiago R Simanca. On circulant matrices. *Notices of the AMS*, 59(3):368–377, 2012.

[12] Nils Langius. Private email communications, May, June 2023.

[13] Ayan Mahalanobis. The discrete logarithm problem in the group of non-singular circulant matrices. 2010.

[14] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[15] Gary L Mullen and Daniel Panario. *Handbook of finite fields*, volume 17. CRC press Boca Raton, 2013.

[16] DC Murdoch. Quasi-groups which satisfy certain generalized associative laws. *American Journal of Mathematics*, 61(2):509–522, 1939.

[17] Daniel Nager. On linearization attack of entropic quasigroups cryptography. Cryptology ePrint Archive, Paper 2022/1575, 2022. https://eprint.iacr.org/2022/1575.

[18] Lorenz Panny. Entropoids: Groups in disguise. 2021. https://eprint.iacr.org/2021/583.

[19] Lorenz Panny. Private email communications, May 2021.

[20] Lorenz Panny. hxp CTF 2022: sequoa writeup. https://hxp.io/blog/95/, 2023. Accessed: 2023-05-24.

[21] Shirley M Radack. Updated digital signature standard approved as federal information processing standard (fips) 186-3. 2009.

[22] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology—CRYPTO'89 Proceedings 9*, pages 239–252. Springer, 1990.

[23] Daniel Shanks. Class number, a theory of factorization, and genera. in 1969 number theory institute (proc. sympos. pure math., vol. xx, state univ. new york, stony brook, ny, 1969), 415–440. *Amer. Math. Soc., Providence, Rhode Island, USA*, 1971.

[24] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[25] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology—EUROCRYPT'97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997 Proceedings 16*, pages 256–266. Springer, 1997.

[26] Igor E Shparlinski. On finding solutions to exponential congruences. *Bulletin of the Australian Mathematical Society*, 99(3):388–391, 2019.

[27] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.7)*, 2023. `https://www.sagemath.org`.

[28] Kôshichi Toyoda. On Axioms of Linear Functions. *Proceedings of the Imperial Academy*, 17(7):221–227, 1941.

[29] Wim Van Dam and Igor E Shparlinski. Classical and quantum algorithms for exponential congruences. In *Theory of Quantum Computation, Communication, and Cryptography: Third Workshop, TQC 2008 Tokyo, Japan, January 30-February 1, 2008. Revised Selected Papers 3*, pages 1–10. Springer, 2008.