cryptography

MDPI

Article

Public Key Protocols from Twisted-Skew Group Rings

Javier de la Cruz [1,*], Edgar Martínez-Moro [2], Steven Muñoz-Ruiz [3] and Ricardo Villanueva-Polanco [4]

[1] Department of Mathematics and Statistics, Universidad del Norte, Barranquilla 081007, Colombia
[2] Institute of Mathematics, Universidad de Valladolid, 47011 Valladolid, Spain; edgar.martinez@uva.es
[3] Department of Mathematics, University of Miami, Coral Gables, FL 33146, USA; sxm2927@miami.edu
[4] Cryptography Research Center, Technology Innovation Institute, Abu Dhabi P.O. Box 9639, United Arab Emirates; ricardo.polanco@tii.ae
* Correspondence: jdelacruz@uninorte.edu.co

Abstract: This article studies some algebraic structures known as twisted-skew group rings in the context of public key cryptography. We first present some background related to these structures to then specifically introduce particular twisted-skew group rings and show how to utilize them as the underlying algebraic structure to build cryptographic protocols. We closely follow an incremental-like methodology to construct these protocols by putting parts together. As as result, we first introduce a key-agreement protocol and then generalize it to a group key-agreement protocol. We then proceed to construct a probabilistic public key encryption from our two-party key agreement and, finally, introduce a key-encapsulation mechanism from a well-known generic construction applied to probabilistic public encryption. Furthermore, we provide an in-depth security analysis for each cryptographic construction under new related algebraic assumptions and supply a proof-of-concept implementation for various candidate chosen groups.

Keywords: twisted-skew group ring; key agreements protocol; key-encapsulation mechanism; public key scheme

Citation: de la Cruz, J.; Martínez-Moro, E.; Muñoz-Ruiz, S.; Villanueva-Polanco, R. Public Key Protocols from Twisted-Skew Group Rings. *Cryptography* **2024**, *8*, 29. https://doi.org/10.3390/cryptography8030029

Academic Editor: Josef Pieprzyk

Received: 3 April 2024
Revised: 10 June 2024
Accepted: 13 June 2024
Published: 5 July 2024

1. Introduction

While the increasingly close possibility of bringing about quantum technology for massive use in the coming years approaches, which will render current public key schemes insecure, the cryptographic community has devoted efforts to design, implement, and deploy quantum-safe public key primitives that replace current public key algorithms. Thus far, many candidates have been proposed via standardization calls for proposals and independent and individual efforts [1–3]. Those candidates may roughly be classified into several categories or groups, namely lattice-based schemes, code-based schemes, isogeny schemes, MPC-in-the-Head schemes, multivariate schemes, and symmetric-based schemes [1–3].

Nevertheless, recent papers [4–7] propose different, promising cryptographic schemes based on group ring generalizations, which seem to be quantum-secure [8]. The research articles [5–7] introduce cryptographic protocols whose security hinges on algebraic problems defined on the structure of a twisted dihedral group algebra, while [4] presents constructions that are supported on a skew dihedral group ring structure. Specifically, Ref. [7] proposes a two-cocycle α_λ to form the twisted algebra $\mathbb{F}_q^{\alpha_\lambda} D_{2n}$ for a non-square λ in the field $\mathbb{F}_q$, where $D_{2n} = \langle x, y : x^n = y^2 = 1, yxy^{-1} = x^{-1} \rangle$ is the dihedral group of order $2n$. More precisely, the two-cocycle $\alpha_\lambda : D_{2n} \times D_{2n} \longrightarrow \mathbb{F}_q^*$ is defined by $\alpha_\lambda(g,h) = \lambda$ for $g = x^i y$, $h = x^j y$ with $i, j \in \{0, \ldots, n-1\}$ and $\alpha_\lambda(g,h) = 1$ otherwise. Furthermore, following an incremental-like methodology as employed by us in this paper, the authors of [7] introduce a key exchange protocol, a probabilistic public key scheme, and a key-encapsulation mechanism over the twisted algebra $\mathbb{F}_q^{\alpha_\lambda} D_{2n}$. Furthermore, their constructions and their proof-of-concept implementation are enhanced by exploiting the properties of the twisted

algebra. On the other hand, Ref. [4] proposes cryptographic protocols supported on an algebraic structure that is called the skew dihedral group ring. This algebraic platform, denoted by $\mathbb{F}_{q^2}^{\theta_\sigma} D_{2n}$, is formed of the dihedral group D_{2n} and the group homomorphism described by $\theta_\sigma(g) = \sigma$, where $\sigma(a) = a^q$ for all $a \in \mathbb{F}_{q^2}$, for $g = x^i y$, $i \in \{0, \ldots, n-1\}$, and $\theta_\sigma(g) = 1$ otherwise. Furthermore, using an incremental-like methodology, the authors of [4] similarly propose a key-exchange protocol, a probabilistic public key scheme, and a key-encapsulation mechanism over the skew dihedral group ring $\mathbb{F}_{q^2}^{\theta_\sigma} D_{2n}$.

Concerning other related works, the research articles [9,10] investigate ideals as codes in twisted-skew group rings. In particular, they characterize all linear codes that are twisted-skew group codes in terms of their automorphism group.

Our main contribution is generalizing previous approaches [4–7] in the sense that we present a novel algebraic structure, which is a twisted-skew group ring and generalizes those in [4–7], and exhibit various cryptographic protocols supported on this structure. This structure features a two-cocycle α_λ and a group homomorphism θ_σ. We particularly consider G to be a finite group of even order and $N \leq G$ such that $|N| = |G|/2 = n$, i.e., $[G : N] = 2$, and hence, $G = N \cup Ny$ with $y \in G \setminus N$. For $\lambda \in \mathbb{F}_q^*$, the map $\alpha_\lambda : G \times G \to \mathbb{F}_q^*$ as $\alpha(g, h) = \lambda$ if $g \notin N$ and $h \notin N$ and $\alpha_\lambda(g, h) = 1$ otherwise is a two-cocycle of the group G over $\mathbb{F}_q$. Also, the map $\theta_\sigma : G \to \mathrm{Gal}(\mathbb{F}_{q^2}, \mathbb{F}_q)$ defined by $\theta_\sigma(g) = \sigma$ if $g \notin N$ and $\theta_\sigma(g) = 1$ otherwise is a group homomorphism. We then define the twisted-skew group ring $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$, over which we build cryptographic constructions. By closely following an incremental-like methodology as previously used in [4,7], we construct a key-agreement protocol and then generalize it to a group key-agreement protocol. We then proceed to build a probabilistic public key encryption from our two-party key agreement and, finally, introduce a key-encapsulation mechanism from a well-known generic construction applied to the probabilistic public encryption. Furthermore, we provide an in-depth security analysis for each cryptographic construction under new related algebraic assumptions and supply a proof-of-concept implementation for various candidate chosen groups.

The outline of the paper is as follows. In Section 2, we will present background material and formally introduce the twisted-skew group ring over which we will build our cryptographic protocols. Section 3 will formally introduce our intractability assumptions. In particular, we will formally define attack games for the algebraic problems on which the security of our cryptographic constructions rely. Section 4 first gives a detailed account of our two-party key-agreement protocol together with its corresponding security analysis and then focuses on its generalization along with the corresponding security analysis. Section 5 will delineate a probabilistic public key-encryption scheme derived from our two-party key-agreement protocol and the corresponding security analysis. Section 6 will portray our key-encapsulation mechanism derived from the probabilistic public key-encryption scheme from Section 5. In Section 7, we will describe the pseudo-code of our proof-of-concept Python implementation for our cryptographic constructions and conclude this section by hinting at potential applications of our protocols. Finally, Section 8 will conclude our work and outline future research directions.

2. A Twisted-Skew Group Ring

Let G be a finite group and $\mathbb{F}_q$ be the finite field of order $q = p^m$ and characteristic p. We denote the automorphism group of $\mathbb{F}_q$ by $\mathrm{Aut}(\mathbb{F}_q)$, and $\mathrm{Gal}(\mathbb{F}_{q^k}, \mathbb{F}_q) \leq \mathrm{Aut}(\mathbb{F}_q)$ always denotes the group of all automorphisms of $\mathbb{F}_{q^k}$ that fix $\mathbb{F}_q$, which is called the Galois group of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. A well-known result is that the Galois group $\mathrm{Gal}(\mathbb{F}_{q^k}, \mathbb{F}_q)$ is a cyclic group of order k and that the Frobenius automorphism σ of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$ is a generator, which is defined as $\sigma(a) = a^q$ for all $a \in \mathbb{F}_{q^k}$. Moreover, by $\mathrm{Hom}(G, \mathrm{Aut}(\mathbb{F}_{q^k}))$ and $\mathrm{Hom}(G, \mathrm{Gal}(\mathbb{F}_{q^k}, \mathbb{F}_q))$, we denote the set of group homomorphisms from G to $\mathrm{Aut}(\mathbb{F}_{q^k})$

and $\mathrm{Gal}(\mathbb{F}_{q^k}, \mathbb{F}_q)$, respectively. Additionally, the map $\alpha : G \times G \to \mathbb{F}_q \setminus \{0\}$ is called a two-cocycle of G if $\alpha(1,1) = 1$ and

$$\alpha(g, hk)\alpha(h,k) = \alpha(gh, k)\alpha(g,h)$$

for all $g, h, k \in G$. Let $Z^2(G, \mathbb{F}_q)$ denote the set of all two-cocycles of G.

We say that the cocycle α is stabilized by the group $\theta(G) \leq \mathrm{Aut}(\mathbb{F}_q)$, if

$$\theta(g)\alpha(x,y) = \alpha(x,y) \tag{1}$$

for all $g, x, y \in G$.

For $\alpha, \beta \in \mathrm{Z}^2(\mathrm{G}, \mathbb{F}_q^*)$, we define $\alpha\beta \in \mathrm{Z}^2(\mathrm{G}, \mathbb{F}_q^*)$ as $\alpha\beta(g,h) = \alpha(g,h)\beta(g,h)$ for all $g, h \in G$. With this operation, $\mathrm{Z}^2(\mathrm{G}, \mathbb{F}_q^*)$ becomes a multiplicative Abelian group. If $\beta : G \longrightarrow \mathbb{F}_q^*$ is a map such that $\beta(1) = 1$, the *coboundary* $\partial\beta$ defined by $\partial\beta(g,h) = \beta(g)^{-1}\beta(h)^{-1}\beta(gh)$ for all $g, h \in G$ is in $\mathrm{Z}^2(\mathrm{G}, \mathbb{F}_q^*)$. We denote the set of all coboundaries of G by $\mathrm{B}^2(G, \mathbb{F}_q^*)$, which forms a subgroup of the group $\mathrm{Z}^2(\mathrm{G}, \mathbb{F}_q^*)$. Given a two-cocycle $\alpha \in \mathrm{Z}^2(\mathrm{G}, \mathbb{F}_q^*)$, its coset is denoted by $[\alpha] := \alpha\mathrm{B}^2(G, \mathbb{F}_q^*)$. Additionally, we call the quotient group

$$\mathrm{H}^2(\mathrm{G}, \mathbb{F}_q^*) = \mathrm{Z}^2(\mathrm{G}, \mathbb{F}_q^*)/\mathrm{B}^2(\mathrm{G}, \mathbb{F}_q^*)$$

the second cohomology group of G with values in $\mathbb{F}_q^*$.

Definition 1 (See [10]). *Let G be a finite multiplicative group; let $\alpha \in Z^2(G, \mathbb{F}_q)$ be a two-cocycle of G; let $\theta \in \mathrm{Hom}(G, \mathrm{Aut}(\mathbb{F}_q))$ be a group homomorphism. The twisted-skew group ring $\mathbb{F}_q^{\theta,\alpha}G$ is the set of all formal sums $\sum_{g\in G} a_g\overline{g}$, where $a_g \in \mathbb{F}_q$, with the following twisted-skew multiplication:*

$$a_g\overline{g} \cdot b_h\overline{h} = a_g(\theta(g)(b_h))\alpha(g,h)\overline{gh}.$$

In [10], it is proven that, if the cocycle α is stabilized by $\theta(G)$, then $\mathbb{F}_q^{\theta,\alpha}G$ is an associative ring with identity $\overline{1}$.

Note that $\mathbb{F}_q^{1,1}G$ is nothing else than the group algebra $\mathbb{F}_qG$, while $\mathbb{F}_q^{1,\alpha}G$ is the twisted group algebra $\mathbb{F}_q^{\alpha}G$, and $\mathbb{F}_q^{\theta,1}G$ is the skew group ring $\mathbb{F}_q^{\theta}G$ (see [4,9,10]). Moreover, by [10], Lemma 1.5, for $\theta \in \mathrm{Hom}(G, \mathrm{Aut}(\mathbb{F}_q))$, we have that $\mathbb{F}_q^{\theta,\alpha}G$ and $\mathbb{F}_q^{\theta,1}G = \mathbb{F}_q^{\theta}G$ are isomorphic, if α is a coboundary. In particular, $\mathbb{F}_q^{\alpha}G = \mathbb{F}_q^{1,\alpha}G \cong \mathbb{F}_q^{1,1}G = \mathbb{F}_qG$ as $\mathbb{F}_q$-algebras, if α is a coboundary.

Definition 2 (See [10]). *For an element $a = \sum_{g\in G} a_g\overline{g} \in \mathbb{F}_q^{\theta,\alpha}G$, we define its adjunct as*

$$\widehat{a} := \varphi(a) = \sum_{g\in G} \theta(g^{-1})(a_g)\alpha(g, g^{-1})\overline{g^{-1}}.$$

In the sequel, we will always consider that G is a finite group of even order and $N \leq G$ such that $|N| = |G|/2 = n$, i.e., $[G : N] = 2$, and hence, $G = N \cup Ny$ with $y \in G \setminus N$. Also, we assume the elements of G are ordered according to some fixed order. In particular, $N = \{n_0, n_1, \ldots, n_{n-1}\}$ and $G = \{n_i y^j \mid i \in \{0, 1, \ldots, n-1\}, j \in \{0,1\}\}$. Some possible groups G satisfying the previous conditions are as follows:

1. Dihedral group: A presentation of the dihedral group D of order $2n$ is given by

$$D = \langle x, y : x^n = y^2 = 1, yxy^{-1} = x^{-1}\rangle,$$

where $N = \langle x^i\rangle$ and $|N| = n$.

2. Quasidihedral group: A presentation of the quasidihedral group $\mathfrak{G}$ of order $2^{\mathfrak{n}}$ is given by
$$\mathfrak{G} = \langle x, y : x^{2^{\mathfrak{n}-1}} = y^2 = 1, yxy = x^{2^{\mathfrak{n}-2}-1} \rangle,$$
where $N = \langle x^i \rangle$ and $|N| = n = 2^{\mathfrak{n}-1}$.
3. Modular maximal-cyclic group: A presentation of the modular maximal-cyclic group $\mathfrak{M}$ of order $2^{\mathfrak{n}}$ is given by
$$\mathfrak{M} = \langle x, y : x^{2^{\mathfrak{n}-1}} = y^2 = 1, yxy = x^{2^{\mathfrak{n}-2}+1} \rangle,$$
where $N = \langle x^i \rangle$ and $|N| = n = 2^{\mathfrak{n}-1}$.
4. Generalized quaternion group: A presentation of the generalized quaternion group $\mathfrak{Q}$ of order $2^{\mathfrak{n}}$ is given by
$$\mathfrak{Q} = \langle x, y : x^{2^{\mathfrak{n}-1}} = y^4 = 1, x^{2^{\mathfrak{n}-2}} = y^2, yxy^{-1} = x^{-1} \rangle,$$
where $N = \langle x^i \rangle$ and $|N| = n = 2^{\mathfrak{n}-1}$.

Lemma 1. *Let $y \in G \setminus N$. Then, we have the following:*

1. *$\mathbb{F}_q^{\theta,\alpha}G$ is a free $\mathbb{F}_q^{\theta,\alpha}N$-module with basis $\{1, y\}$. Therefore, $\mathbb{F}_q^{\theta,\alpha}G = \mathbb{F}_q^{\theta,\alpha}N \oplus \mathbb{F}_q^{\theta,\alpha}Ny$ as the direct sum of $\mathbb{F}_q$-vector spaces.*
2. *$\mathbb{F}_q^{\theta,\alpha}N \cong \mathbb{F}_q^{\theta,\alpha}Ny$ as $\mathbb{F}_q^{\theta,\alpha}N$-modules.*
3. *For $a \in \mathbb{F}_q^{\theta,\alpha}Ny$, $ab \in \mathbb{F}_q^{\theta,\alpha}N$ if $b \in \mathbb{F}_q^{\theta,\alpha}Ny$ or $ab \in \mathbb{F}_q^{\theta,\alpha}Ny$ if $b \in \mathbb{F}_q^{\theta,\alpha}N$.*
4. *If $a \in \mathbb{F}_q^{\theta,\alpha}N$, then $\widehat{a} \in \mathbb{F}_q^{\theta,\alpha}N$.*
5. *If $a \in \mathbb{F}_q^{\theta,\alpha}Ny$, then $\widehat{a} \in \mathbb{F}_q^{\theta,\alpha}Ny$.*

Proof. Let $y \in G \setminus N$:

1. Since $\{1, y\}$ is a transversal of N, the assertion follows.
2. Consider the map $\sigma : \mathbb{F}_q^{\theta,\alpha}N \to \mathbb{F}_q^{\theta,\alpha}Ny$ given by $\sigma(g) = gy$ for all $g \in N$, then σ is an $\mathbb{F}_q^{\theta,\alpha}N$-module isomorphism.
3. Since $[G : N] = 2$, then $gh \in N$ if and only if $g, h \in N$ or $g, h \in Ny$. Therefore, the assertion follows.
4. If $g \in N$, then $g^{-1} \in N$ since N is a subgroup. Hence, the assertion follows.
5. If $g \in Ny$, then $g^{-1} \in Ny$ since $[G : N] = 2$. Hence, the assertion follows.

□

Definition 3. *We define the (θ, α)-reversible subspace of $\mathbb{F}_q^{\theta,\alpha}Ny$ as the vector subspace $\Gamma_{\theta,\alpha} = \{a = \sum_{i=0}^{n-1} a_i \overline{n_i y} \in \mathbb{F}_q^{\theta,\alpha}Ny : a_i = a_{[-i]_n}$ for all $i = 1, 2, \ldots, n-1\}$.*

The following lemma introduces the two-cocycle α_λ on the group G, for a given element λ in $\mathbb{F}_q^*$.

Lemma 2. *Let $\lambda \in \mathbb{F}_q^*$. Then, the map $\alpha_\lambda : G \times G \to \mathbb{F}_q^*$ defined by $\alpha(g, h) = \lambda$ if $g \notin N$ and $h \notin N$ and $\alpha_\lambda(g, h) = 1$ otherwise is a two-cocycle of the group G over $\mathbb{F}_q$.*

Proof. By definition, $\alpha_\lambda(g, h)(1, 1) = 1$. Therefore, α_λ is a two-cocycle if $\alpha_\lambda(g, h)\alpha_\lambda(gh, k) = \alpha_\lambda(g, hk)\alpha_\lambda(h, k)$ for all $g, h, k \in G$. Let us first assume $g \in N$ and $h, k \in G$, then a straightforward calculation shows that $\alpha_\lambda(g, h)\alpha_\lambda(gh, k) = \alpha_\lambda(g, hk)\alpha_\lambda(h, k)$ holds. On the other hand, if $g \in Ny$, then a straightforward calculation also shows that $\alpha_\lambda(g, h)\alpha_\lambda(gh, k) = \alpha_\lambda(g, hk)\alpha_\lambda(h, k)$ holds. □

Lemma 3. *Let α_λ be the two-cocycle defined in Lemma 2:*

1. *If λ is a square in $\mathbb{F}_q^*$, then α_λ is a coboundary.*

2. *If λ is a non-square in $\mathbb{F}_q^*$ and there exists $g \in Ny$ such that $g^2 = 1$, then α_λ cannot be a coboundary.*
3. *If λ_1, λ_2 are non-squares in $\mathbb{F}_q^*$, then α_{λ_1} and α_{λ_2} are congruent.*

Proof.

1. Suppose λ is a square in $\mathbb{F}_q^*$, then there exists $t \in \mathbb{F}_q$ such that $t^2 = \lambda$. Let us define $\beta(g) = 1$ if $g \in N$, or else $\beta(g) = t^{-1}$. We have $\alpha_\lambda(g,h) = \beta(g)^{-1}\beta(h)^{-1}\beta(gh)$ for all $g, h \in G$, since $gh \in N$ if and only if $g, h \in N$ or $g, h \in Ny$.
2. Suppose there exists a function β such that $\beta(1) = 1$ and $\alpha_\lambda(g,h) = \beta(g)^{-1}\beta(h)^{-1}\beta(gh)$. Therefore, $\alpha_\lambda(g,g) = \lambda = \beta(g)^{-1}\beta(g)^{-1}\beta(g^2) = [\beta(g)^{-1}]^2$, a contradiction.
3. Let ξ be a primitive element of $\mathbb{F}_q^*$. Since λ_1, λ_2 are non-squares in $\mathbb{F}_q^*$, then $\lambda_1 = \xi_1^k$ and $\lambda_2 = \xi_2^k$ with k_1 and k_2 being odd. Therefore, $\lambda_1 = \xi_1^k = \lambda_2\xi^{k_3}$, where k_3 is even, i.e., ξ^{k_3} is a square in $\mathbb{F}_q$. Let us define $\beta(g) = 1$ if $g \in N$ and $\beta(g) = \xi^{k_3/2}$ otherwise. We, therefore, have $\alpha_{\lambda_1}(g,h) = \alpha_{\lambda_2}(g,h)\beta(g)\beta(h)\beta(gh)^{-1}$ for all $g, h \in G$, since $gh \in N$ if and only if $g, h \in N$ or $g, h \in Ny$.

□

Remark 1. *Note that, since $(\lambda^{2^{m-1}})^2 = \lambda$, for all $\lambda \in \mathbb{F}_{2^m}$ and $m \in \mathbb{N}$, then $\mathbb{F}_{2^m}G$ and $\mathbb{F}_q^{\theta,\alpha_\lambda}G$ are isomorphic. By this, we will assume that char $\mathbb{F}_q \neq 2$.*

Let $\mathbb{F}_{q^2}$ be a quadratic extension of $\mathbb{F}_q$. From now on, we only will take into consideration a quadratic extension of $\mathbb{F}_q$ since the ambient space over which we define our cryptographic constructions is the twisted-skew group ring $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$, where $\theta_\sigma \in \mathrm{Hom}(G, \mathrm{Aut}(\mathbb{F}_{q^2}))$ is a group homomorphism. We next introduce θ_σ.

Lemma 4. *Let $\sigma \in \mathrm{Gal}(\mathbb{F}_{q^2}, \mathbb{F}_q))$ be the Frobenius automorphism of $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$. Then, the map $\theta_\sigma : G \to \mathrm{Gal}(\mathbb{F}_{q^2}, \mathbb{F}_q)$ defined by $\theta_\sigma(g) = \sigma$ if $g \notin N$ and $\theta_\sigma(g) = 1$ otherwise is a group homomorphism.*

Proof. Let $g, h \in G$ and $\gamma \in \mathbb{F}_{q^2}$. There are four cases to check:

1. $g, h \in N$ is easy to check.
2. $g \in N$ y $h \notin N, \theta(gh)(\gamma) = \sigma(\gamma) = \mathrm{Id}(\sigma(\gamma)) = \theta(g)\theta(h)(\gamma)$.
3. $h \in N$ y $g \notin N, \theta(gh)(\gamma) = \sigma(\gamma) = \sigma(\mathrm{Id}(\gamma)) = \theta(g)\theta(h)(\gamma)$.
4. $g, h \notin N, \theta(gh)(\gamma) = \gamma = \sigma(\sigma(\gamma)) = \theta(g)\theta(h)(\gamma)$.

□

Remark 2. *θ_σ relies on the Frobenius automorphism σ. Moreover, since $\mathrm{Gal}(\mathbb{F}_{q^2}, \mathbb{F}_q) \leq \mathrm{Aut}(\mathbb{F}_{q^2})$, then $\theta_\sigma \in \mathrm{Hom}(G, \mathrm{Gal}(\mathbb{F}_{q^2}, \mathbb{F}_q))$. Furthermore, if $\lambda \in \mathbb{F}_q \subset \mathbb{F}_{q^2}$, then $\left(\lambda^{\frac{q+1}{2}}\right)^{(q-1)} = \lambda^{\frac{q^2-1}{2}} = 1$, i.e., it is a square, and also, α_λ is stabilized by the group $\theta_\sigma(G)$; therefore, the twisted-skew group ring $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$ is isomorphic to the skew group ring $\mathbb{F}_{q^2}^{\theta_\sigma}G$ introduced in [4]. However, if $\lambda \in \mathbb{F}_{q^2} \setminus \mathbb{F}_q$, then the cocycle α_λ is not stabilized by the group $\theta_\sigma(G)$, and so, the twisted-skew multiplication is not necessarily associative. In fact, for none of the possible four groups G we consider the twisted-skew multiplication is associative.*

Lemma 5. *Let α_λ be the two-cocycle defined in Lemma 2 and $\theta_\sigma \in Hom(G, \mathrm{Gal}(\mathbb{F}_{q^2}, \mathbb{F}_q))$ the group homomorphism defined in Lemma 4. Then, we have the following:*

1. *$ab = ba$ for $a, b \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}N$.*
2. *$a\hat{b} = b\hat{a}$ for $a, b \in \Gamma_{\theta_\sigma,\alpha_\lambda}$.*
3. *$\hat{a}b = \hat{b}a$ for $a, b \in \Gamma_{\theta_\sigma,\alpha_\lambda}$.*
4. *$((ah)\gamma) = (a(h\gamma))$ for $a \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}N, h \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G, \gamma \in \Gamma_{\theta_\sigma,\alpha_\lambda}$.*

Proof.

1. Let $a = \sum_{i=0}^{n-1} a_i\overline{n_i} \in \mathbb{F}_q^{\theta_\sigma,\alpha_\lambda} N$ and $b = \sum_{i=0}^{n-1} b_i\overline{n_i} \in \mathbb{F}_q^{\theta_\sigma,\alpha_\lambda} N$.

$$ab = \sum_{i=0}^{n-1} a_i\overline{n_i} \sum_{j=0}^{n-1} b_j\overline{n_j} = \sum_{k\in N}\left(\sum_{n_in_j=k} a_ib_j\right)\overline{k} \tag{2}$$

$$ba = \sum_{i=0}^{n-1} b_i\overline{n_i} \sum_{j=0}^{n-1} a_j\overline{n_j} = \sum_{k\in N}\left(\sum_{n_in_j=k} b_ia_j\right)\overline{k} \tag{3}$$

Note that the second sum of Equations (2) and (3) follows from the definitions of θ_σ and α_λ. Therefore, $ab = ba$.

2. Let $a = \sum_{i=0}^{n-1} a_i\overline{n_iy} \in \Gamma_{\theta_\sigma,\alpha_\lambda}$ and $b = \sum_{i=0}^{n-1} b_i\overline{n_iy} \in \Gamma_{\theta_\sigma,\alpha_\lambda}$. Then, $a\hat{b}$ can be expressed as

$$\sum_{i=0}^{n-1} a_i\overline{n_iy} \sum_{j=0}^{n-1} \theta((n_jy)^{-1})(b_j)\alpha_\lambda(n_jy,(n_jy)^{-1})\overline{(n_jy)^{-1}} = \sum_{k\in N}\left(\sum_{(n_iy)(n_jy)=k} a_ib_j\lambda^{q+1}\right)\overline{k} \tag{4}$$

and $b\hat{a}$ as

$$\sum_{i=0}^{n-1} b_i\overline{n_iy} \sum_{j=0}^{n-1} \theta((n_jy)^{-1})(a_j)\alpha_\lambda(n_jy,(n_jy)^{-1})\overline{(n_jy)^{-1}} = \sum_{k\in N}\left(\sum_{(n_iy)(n_jy)=k} b_ia_j\lambda^{q+1}\right)\overline{k} \tag{5}$$

The first sum of Equations (4) and (5) follows from the definitions and that, for all $\omega \in Ny$, $\alpha(\omega,\omega^{-1}) = \lambda$. The second sum of Equations (4) and (5) follows from θ_σ being a homomorphism, and thus, $\theta_\sigma(g)(\theta_\sigma(h)(a)) = \theta_\sigma(gh)(a) = a$ for $a \in \mathbb{F}_{q^2}$ and $\theta_\sigma(g)(\lambda) = \sigma(\lambda) = \lambda^q$.
Since $a, b \in \Gamma_{\theta_\sigma,\alpha_\lambda}$, then $a_i = a_{[-i]_n}$ and $b_j = b_{[-j]_n}$ for $i, j \in \{0,1,\ldots,n-1\}$. Therefore, the (i,j)-th term $a_ib_j\lambda^{q+1}$ of $\sum_{(n_iy)(n_jy)=k} a_ib_j\lambda^{q+1}$ in (4) coincides with the $([-j]_n,[-i]_n)$-th term $b_{[-j]_n}a_{[-i]_n}\lambda^{q+1}$ of $\sum_{(n_iy)(n_jy)=k} b_ia_j\lambda^{q+1}$ in (5), which implies the equality.

3. Let $a = \sum_{i=0}^{n-1} a_i\overline{n_iy} \in \Gamma_{\theta_\sigma,\alpha_\lambda}$ and $b = \sum_{i=0}^{n-1} b_i\overline{n_iy} \in \Gamma_{\theta_\sigma,\alpha_\lambda}$. Then, we can write $\hat{a}b$ as

$$\begin{aligned}\sum_{i=0}^{n-1} \theta((n_iy)^{-1})(a_i)\alpha_\lambda(n_iy,(n_iy)^{-1})\overline{(n_iy)^{-1}} \sum_{j=0}^{n-1} b_j\overline{n_jy} &= \sum_{i=0}^{n-1} a_i^q\lambda\overline{(n_iy)^{-1}} \sum_{j=0}^{n-1} b_j\overline{n_jy} \\ &= \sum_{k\in N}\left(\sum_{(n_iy)^{-1}(n_jy)=k} a_i^qb_j^q\lambda^2\right)\overline{k}\end{aligned} \tag{6}$$

and $\hat{ba}$ is

$$\sum_{i=0}^{n-1} \theta((n_i y)^{-1})(b_i)\alpha_\lambda(n_i y,(n_i y)^{-1})\overline{(n_i y)^{-1}} \sum_{j=0}^{n-1} a_j \overline{n_j y}$$
$$= \sum_{i=0}^{n-1} b_i^q \lambda \overline{(n_i y)^{-1}} \sum_{j=0}^{n-1} a_j \overline{n_j y}$$
$$= \sum_{k\in N} \left(\sum_{(n_i y)^{-1}(n_j y)=k} b_i^q a_j^q \lambda^2 \right) \overline{k} \quad (7)$$

The last sum of Equations (6) and (7) follows from the definitions, $\alpha(\omega,\omega^{-1}) = \lambda$ and $\theta_\sigma(\omega)(a) = a^q$ for $\omega \in Ny, a \in \mathbb{F}_q^2$.
Since $a, b \in \Gamma_{\theta_\sigma,\alpha_\lambda}$, then $a_i = a_{[-i]_n}$ and $b_j = b_{[-j]_n}$ for $i, j \in \{0,1,\ldots,n-1\}$. Therefore, the (i,j)-th term $a_i^q b_j^q \lambda^2$ of $\sum_{(n_i y)(n_j y)=k} a_i^q b_j^q \lambda^2$ in (4) coincides with the $([-j]_n,[-i]_n)$-th term $b_{[-j]_n}^q a_{[-i]_n}^q \lambda^2$ of $\sum_{(n_i y)(n_j y)=k} b_i^q a_j^q \lambda^2$ in (5), which implies the equality.

□

3. Intractability Assumptions

This section will describe some attack games concerning the algebraic problems on which the security of our cryptographic constructions lies [4,7,11,12]. Before giving a detailed account of them, we will introduce some notation that we will use for the remaining part of this paper:

1. Let G be a finite group of even order and $N \leq G$ such that $|N| = |G|/2 = n$.
2. Let p be a prime number and $q = p^m$ for some $m \in \mathbb{N}$. Let $\mathbb{F}_{q^2}$ be the quadratic extension of $\mathbb{F}_q$.
3. The two-cocycle α_λ is instantiated by selecting λ such that it is a non-square in $\mathbb{F}_{q^2}$. Additionally, θ_σ is chosen as specified by Lemma 4.
4. We set $\mathtt{h} = \mathtt{h}_1 + \mathtt{h}_2$ as a public element, where $\mathtt{h}_1 \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} N$ and $\mathtt{h}_2 \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} Ny$ are random non-zero elements.
5. We denote the secret key space by $\mathcal{SK} = \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} N \times \Gamma_{\theta_\sigma,\alpha_\lambda}$. Given a secret key $\mathtt{sk} = (\mathtt{a},\gamma) \in \mathcal{SK}$, we denote $(\mathtt{a},\widehat{\gamma})$ by $\widehat{\mathtt{sk}}$. Besides, we define $\psi : \mathcal{SK} \times \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G \longrightarrow \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$ as $\psi(\mathtt{sk},h) = \mathtt{a}h\gamma$.

Game 1 (Twisted-Skew Product Decomposition)**.** *Let $\mathcal{A}$ be an efficient adversary. We define the Twisted-Skew Product Decomposition (TSPD) Attack Game as shown by Algorithm 1.*

Algorithm 1 defines the Twisted-Skew Product Decomposition (TSPD) Attack Game

The challenger $\mathcal{C}$ executes
1: $(\mathtt{a},\gamma) \xleftarrow{R} \mathcal{SK}$;
2: $\mathtt{pk} \leftarrow \psi((\mathtt{a},\gamma),\mathtt{h})$;
3: $(\widetilde{\mathtt{a}},\widetilde{\gamma}) \leftarrow \mathcal{A}(\mathtt{pk})$;
4: **return** $[[\widetilde{\mathtt{a}}\mathtt{h}\widetilde{\gamma} = \mathtt{a}\mathtt{h}\gamma]]$;

In the TSPD attack game, $[[\widetilde{\mathtt{a}}\mathtt{h}\widetilde{\gamma} = \mathtt{a}\mathtt{h}\gamma]]$ denotes a Boolean value, which is 1 when $\widetilde{\mathtt{a}}\mathtt{h}\widetilde{\gamma} = \mathtt{a}\mathtt{h}\gamma$, or 0 otherwise. We define E_1 as the event that the TSPD attack game outputs 1 after $\mathcal{A}$ plays it for $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$. Furthermore, we define $\mathcal{A}$'s advantage in solving the TSPD problem for $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$ as the probability of E_1 and denote it by $\mathrm{TSPDadv}[\mathcal{A},\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G]$.

Definition 4 (Twisted-Skew Product Decomposition Assumption)**.** *We say that the TSPD assumption holds for* $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$ *if, for all efficient adversaries* $\mathcal{A}$*, the quantity* $TSPDadv[\mathcal{A},\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G]$ *is negligible.*

Game 2 (Computational Twisted-Skew Product)**.** *Let* $\mathcal{A}$ *be an efficient adversary. We define the Computational Twisted-Skew Product (CTSP) Attack Game as shown by Algorithm 2.*

Algorithm 2 defines the Computational Twisted-Skew Product Attack Game

The challenger $\mathcal{C}$ executes

1: $(\mathtt{a}_1,\gamma_1) \xleftarrow{R} \mathcal{SK}$;
2: $(\mathtt{a}_2,\gamma_2) \xleftarrow{R} \mathcal{SK}$;
3: $\mathtt{pk}_1 \leftarrow \psi((\mathtt{a}_1,\gamma_1),\mathtt{h})$;
4: $\mathtt{pk}_2 \leftarrow \psi((\mathtt{a}_2,\gamma_2),\mathtt{h})$;
5: $\mathtt{k} \leftarrow \psi((\mathtt{a}_2,\widehat{\gamma_2}),\mathtt{pk}_1)$;
6: $\widetilde{\mathtt{k}} \leftarrow \mathcal{A}(\mathtt{pk}_1,\mathtt{pk}_2)$;
7: **return** $[[\widetilde{\mathtt{k}}=\mathtt{k}]]$;

We define E_2 as the event that the CTSP attack game outputs 1 after $\mathcal{A}$ plays it for $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$. Moreover, we define $\mathcal{A}$'s advantage in solving the CTSP problem for $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$ as the probability of E_2 and denote it by $\mathrm{CTSPadv}[\mathcal{A},\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G]$.

Definition 5 (Computational Twisted-Skew Product Assumption)**.** *We say that the CTSP assumption holds for* $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$ *if, for all efficient adversaries* $\mathcal{A}$*, the quantity* $CTSPadv[\mathcal{A},\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G]$ *is negligible.*

Lemma 6. *If the TSPD assumption does not hold for* $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$*, then the CTSP assumption does not hold for* $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$*.*

Proof. Since the TSPD assumption does not hold for $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$, then there exists an efficient adversary $\mathcal{B}$ that can win the TSPD attack game with non-negligible probability ρ, i.e., $\mathcal{B}$ can output $(\widetilde{\mathtt{a}},\widetilde{\gamma}) \in \mathcal{SK}$ such that $\widetilde{\mathtt{a}}\mathtt{h}\widetilde{\gamma} = \mathtt{a}\mathtt{h}\gamma = \mathtt{pk}$ with non-negligible probability ρ.

We now construct an efficient adversary $\mathcal{A}$ that plays and wins the CTSP attack game with non-negligible probability ρ. $\mathcal{A}$ simply uses $\mathcal{B}$ as the subroutine. Upon receiving $\mathtt{pk}_1$ and $\mathtt{pk}_2$ from its challenger, $\mathcal{A}$ calls $\mathcal{B}$ upon the input either $\mathtt{pk}_1$ or $\mathtt{pk}_2$. In either case, if $\mathcal{B}$ succeeds in returning a $(\widetilde{\mathtt{a}},\widetilde{\gamma}) \in \mathcal{SK}$ such that $\widetilde{\mathtt{pk}} = \widetilde{\mathtt{a}}\mathtt{h}\widetilde{\gamma} = \mathtt{a}_b\mathtt{h}\gamma_b = \mathtt{pk}_b$ $(b \in \{1,2\})$, then $\mathcal{A}$ will calculate $\widetilde{\mathtt{k}} = \widetilde{\mathtt{a}}\mathtt{pk}_{\bar{b}}\widehat{\widetilde{\gamma}}$, with $\bar{b} = 3-b$, and return $\widetilde{\mathtt{k}}$ to its challenger. Because of the choice of θ_σ and α_λ and Lemma 5, then

$$\widetilde{\mathtt{k}} = \widetilde{\mathtt{a}}\mathtt{pk}_{\bar{b}}\widehat{\widetilde{\gamma}} = \widetilde{\mathtt{a}}\mathtt{a}_{\bar{b}}\mathtt{h}\gamma_{\bar{b}}\widehat{\widetilde{\gamma}} = \mathtt{a}_{\bar{b}}\widetilde{\mathtt{a}}\mathtt{h}\widetilde{\gamma}\widehat{\gamma_{\bar{b}}} = \mathtt{a}_{\bar{b}}\widetilde{\mathtt{pk}}\widehat{\gamma_{\bar{b}}} = \mathtt{a}_{\bar{b}}\mathtt{pk}_b\widehat{\gamma_{\bar{b}}} = \mathtt{k}$$

In conclusion, $\mathcal{A}$ is an efficient adversary and may succeed in computing the correct $\mathtt{k}$ in the CTSP attack game with non-negligible probability ρ. □

Game 3 (Decisional Twisted-Skew Product)**.** *Let* $\mathcal{A}$ *be an efficient adversary. We define the Decisional Twisted-Skew Product (DTSP) Attack Game by two experiments indexed by a bit* b *as shown by Algorithm 3.*

Algorithm 3 defines the Decisional Twisted-Skew Product Attack Game

For Experiment b, the challenger $\mathcal{C}$ executes

1: $(\mathtt{a}_1, \gamma_1) \xleftarrow{R} \mathcal{SK}$;
2: $(\mathtt{a}_2, \gamma_2) \xleftarrow{R} \mathcal{SK}$;
3: $(\mathtt{a}_3, \gamma_3) \xleftarrow{R} \mathcal{SK}$;
4: $\mathtt{pk}_1 \leftarrow \psi((\mathtt{a}_1, \gamma_1), \mathtt{h})$;$\mathtt{pk}_2 \leftarrow \psi((\mathtt{a}_2, \gamma_2), \mathtt{h})$;
5: $\mathtt{k}_0 \leftarrow \psi((\mathtt{a}_2, \widehat{\gamma_2}), \mathtt{pk}_1)$; $\mathtt{k}_1 \leftarrow \psi((\mathtt{a}_3, \gamma_3), \mathtt{h})$;
6: $\widetilde{\mathtt{b}} \leftarrow \mathcal{A}(\mathtt{pk}_1, \mathtt{pk}_2, \mathtt{k}_\mathtt{b})$
7: **return** $[[b = \widetilde{\mathtt{b}}]]$;

Remark 3. *Game 3 defines two experiments indexed by a random bit b chosen by the challenger. Therefore, the challenger returns either* $(\mathtt{pk}_1, \mathtt{pk}_2, \mathtt{k}_0)$ *or* $(\mathtt{pk}_1, \mathtt{pk}_2, \mathtt{k}_1)$ *to the adversary* $\mathcal{A}$*, depending on the experiment the challenger is playing, i.e., the challenger gives* $(\mathtt{pk}_1, \mathtt{pk}_2, \mathtt{k}_\mathtt{b})$ *to* $\mathcal{A}$*. We denote the experiment b by* $\mathtt{DTSP}(b)$*.*

We define $W_\mathtt{b}$ as the event that $\mathcal{A}$ outputs the bit 1 after playing the experiment $\mathtt{b}$ in the DTSP attack game for $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$. Furthermore, we define $\mathcal{A}$'s advantage in solving the DTSP problem for $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$ as $|\Pr[W_0] - \Pr[W_1]|$ and denote it by $\mathrm{DTSPadv}[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G]$.

Definition 6 (Decisional Twisted-Skew Product Assumption)**.** *We say that the DTSP assumption holds for* $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$ *if, for all efficient adversaries* $\mathcal{A}$*, the quantity* $DTSPadv[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G]$ *is negligible.*

Lemma 7. *If the CTSP assumption does not hold for* $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$*, then the DTSP assumption does not hold for* $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$*.*

Proof. Since the CTSP assumption does not hold for $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$, then there exists an efficient adversary $\mathcal{B}$ that outputs $\widetilde{\mathtt{k}} \in \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$ such that $\widetilde{\mathtt{k}} = \mathtt{k}$ after being given public keys $\mathtt{pk}_1$ and $\mathtt{pk}_2$ with non-negligible probability.

We now construct an efficient adversary $\mathcal{A}$ that plays and wins the DTSP attack game with non-negligible probability. This adversary $\mathcal{A}$ uses $\mathcal{B}$ as the subroutine. In particular, upon receiving $(\mathtt{pk}_1, \mathtt{pk}_2, \mathtt{k}_\mathtt{b})$ from its challenger, $\mathcal{A}$ calls $\mathcal{B}$ upon the input $(\mathtt{pk}_1, \mathtt{pk}_2)$. If $\mathcal{B}$ solves this instance of the CTSP problem for given $\mathtt{pk}_1$ and $\mathtt{pk}_2$ and returns $\tilde{\mathtt{k}}$ to $\mathcal{A}$, then $\mathcal{A}$ compares $\tilde{\mathtt{k}}$ and $\mathtt{k}_\mathtt{b}$ to see whether they are equal. If so, then $\mathcal{A}$ returns 0, or 1 otherwise.

In summary, $\mathcal{A}$ is an efficient adversary and may succeed in winning the DTSP attack game with non-negligible probability. □

The Hardness of the TSPD Problem

The TSPD problem is similar to both the Dihedral Product Decomposition (DPD) and Skew Dihedral Product Decomposition (SDPD) problems. The former was introduced in [5], then formalized in [7] and extended in [6], while the latter was introduced in [4] as an extension of the former. The key difference between both is that the latter is defined over the Dihedral Skew Group Ring $\mathbb{F}_{q^2}^{\theta_\sigma} D_{2n}$, which is structurally different from the algebra $\mathbb{F}_q^{\alpha_\lambda} D_{2n}$ over which the former is defined.

We remark that the algorithmic analysis presented for the DPD problem over $\mathbb{F}_q^{\alpha_\lambda} D_{2n}$ in [7] can be adjusted easily to both the SDPD and TSPD problems. Furthermore, note that, if λ is non-square, then the twisted-skew multiplication defined over $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$ is not associative, which motives the claim that the TSPD problem is defined over a less-structured algebraic structure.

4. Key-Agreement Protocols from Twisted-Skew Group Rings

This section introduces key-agreement protocols using two-sided multiplications over a twisted-skew group ring $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$.

We note that previous research papers have introduced similar key-exchange protocols using two-sided semi-group actions or matrices over groups [13–17]. However, our approach may be seen as an alternative proposal, since it follows in design and generalizes the works presented in [4,6,7], which propose key-agreement protocols using two-sided multiplications over dihedral twisted (skewed) group rings.

4.1. Choice of Parameters

Here, we will outline our choice of parameters. We remark that Section 7.7 will provide more details regarding specific values assigned to some of them:

1. Let G be a finite group of even order and $N \leq G$ such that $|N| = |G|/2 = n$. In particular, we may select a group from the set $\{D, \mathfrak{G}, \mathfrak{M}, \mathfrak{Q}\}$.
2. Let p be a prime number and $q = p^m$ for some $m \in \mathbb{N}$. Let $\mathbb{F}_{q^2}$ be the quadratic extension of $\mathbb{F}_q$.
3. The two-cocycle α_λ is instantiated by selecting λ such that it is a non-square in $\mathbb{F}_{q^2}$. Additionally, θ_σ is chosen as specified by Lemma 4.
4. We set $\mathtt{h} = \mathtt{h}_1 + \mathtt{h}_2$ as a public element, where $\mathtt{h}_1 \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}N$ and $\mathtt{h}_2 \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}Ny$ are random non-zero elements.
5. We denote the secret key space by $\mathcal{SK} = \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}N \times \Gamma_{\theta_\sigma,\alpha_\lambda}$. Given a secret key $\mathtt{sk} = (\mathtt{a}, \gamma) \in \mathcal{SK}$, we denote $(\mathtt{a}, \widehat{\gamma})$ by $\widehat{\mathtt{sk}}$. Besides, we define $\psi : \mathcal{SK} \times \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G \longrightarrow \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$ as $\psi(\mathtt{sk}, h) = \mathtt{a}h\gamma$.

4.2. Two-Party Key-Agreement Protocol

We begin by introducing some notation. A set of identifiers is denoted by $\mathcal{ID} = \{0,1\}^+$. The identifier for the principal P_i is denoted by $\mathtt{ID}_i \in \mathcal{ID}$. A session id is denoted by a bit string s. The two-party key-agreement protocol between P_i and P_j runs as shown in Algorithm 4.

Algorithm 4 Describes our two-party key-exchange protocol

1: Upon the input $(\mathtt{ID}_i, \mathtt{ID}_j, s)$, the principal P_i randomly selects a secret pair $\mathtt{sk}_i = (\mathtt{a}_i, \gamma_i) \xleftarrow{R} \mathcal{SK}$, then generates the corresponding public key by invoking $\mathtt{pk}_i = \psi(\mathtt{sk}_i, \mathtt{h})$, and finally, transmits $(\mathtt{ID}_i, s, \mathtt{pk}_i)$ to P_j;
2: After receiving $(\mathtt{ID}_i, s, \mathtt{pk}_i)$ from P_i, the principal P_j randomly selects a secret pair $\mathtt{sk}_j = (\mathtt{a}_j, \gamma_j) \xleftarrow{R} \mathcal{SK}$, then generates the corresponding public key by executing $\mathtt{pk}_j = \psi(\mathtt{sk}_j, \mathtt{h})$, and transmits $(\mathtt{ID}_j, s, \mathtt{pk}_j)$ to P_i. Additionally, P_j computes $\mathtt{k}_j = \psi(\widehat{\mathtt{sk}_j}, \mathtt{pk}_i)$, securely erases $(\mathtt{a}_j, \gamma_j)$, and outputs the key $\mathtt{k}_j$ under the session id s;
3: After receiving $(\mathtt{ID}_j, s, \mathtt{pk}_j)$ from P_j, the principal P_i calls $\mathtt{k}_i = \psi(\widehat{\mathtt{sk}_i}, \mathtt{pk}_j)$ to obtain the shared key $\mathtt{k}_i$ under the session id s, then securely erases $(\mathtt{a}_i, \gamma_i)$, and outputs the key $\mathtt{k}_i$ under the session id s.

Security Analysis of Our Two-Party Key-Exchange Protocol

This section is devoted to providing an analysis of our two-party key-exchange protocol in the authenticated links adversarial model [18–20]. We will present a description of this security model for completeness:

1. Let us denote by $P = \{P_1, P_2, \ldots, P_n\}$ a finite set of principals.
2. Let $\mathcal{A}$ be an adversary controlling all messages between two principals; however, we have the following:

- $\mathcal{A}$ is not permitted to insert or alter messages, except for those messages transmitted by corrupted principals or sessions.
- $\mathcal{A}$ may opt for not forwarding a message at all. However, in case $\mathcal{A}$ decides to forward a message m, then $\mathcal{A}$ should transmit it to its right destination, only on one occasion and refraining from making partial or minor changes to it.
- Principals freely transfer the possession of their egress messages to $\mathcal{A}$, who has the power to influence their transmission by means of the `Send` query. $\mathcal{A}$ may make a principal P_i operative by `Send` queries, i.e., the adversary holds the power or ability to create protocol sessions, which occur within each principal. Two sessions, with ids s_1 and s_0, respectively, are said to be matching when egress messages from one side are the ingress messages from the other side, and vice versa.
  Additionally, $\mathcal{A}$ is given the ability to make queries to the following oracles:
  - `SessionStateReveal` oracle.
    Whenever $\mathcal{A}$ queries it for a clearly identified session id s within some principal P_i, then $\mathcal{A}$ will acquire the contents of the specified session id s within P_i, including any secret information. This event will be registered and produce no further output.
  - `SessionKeyReveal` oracle.
    Whenever $\mathcal{A}$ queries it for a clearly identified session id s, then $\mathcal{A}$ will acquire the session key for the specified session id s, as long as s has an associated session.
  - `Corrupt` oracle.
    Whenever $\mathcal{A}$ queries it for a clearly identified principal P_i, then $\mathcal{A}$ will assume control of the principal P_i, i.e., $\mathcal{A}$ will have the opportunity to obtain all information in P_i's memory, which includes long-lived keys and any session-specific, remaining data. A corrupted principal will yield no further output.
- $\mathcal{A}$ may query the `test` oracle on one occasion and at any point for a completed, fresh, unexpired session id s. When queried on input s, the `test` oracle randomly picks a bit, $b \xleftarrow{R} \{0,1\}$, then it will return the session key associated with the specified session id s if $b = 0$. Otherwise, it will return a random value in the key space. Additionally, $\mathcal{A}$ can issue subsequent queries to the other oracles as desired, but it cannot expose the test session. At any further point, the adversary will try to guess b.
- We denote the probability of the event that $\mathcal{A}$ correctly guesses b by $\mathrm{Guess}[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G]$, and define the advantage as

$$\mathrm{SKAdv}[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G] = |\mathrm{Guess}[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G] - 1/2|.$$

Theorem 1. *Let $\mathcal{A}$ be an efficient adversary in the authenticated links adversarial model (AM). If the DTSP assumption holds for $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$, then our key exchange protocol is session key-secure in this setting, i.e., the two-party key-agreement protocol satisfies the following:*

1. *If two principals engage in a protocol session s, are not corrupted during the execution of it, and complete it successfully, then each principal will compute matching keys.*
2. *$SKAdv[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G]$ is negligible.*

Proof. The following proof is essentially an adaptation of the proofs given for each key-exchange protocol presented, respectively, in [4,7]:

1. Let us assume that two principals P_i and P_j engage in a protocol session s, are not corrupted during the execution of it, and complete it successfully. We claim that each

principal will compute matching keys. By Lemma 5 and the choice of θ_σ and α_λ, the claim follows. Indeed,

$$\mathtt{k}_i = \mathtt{a}_i \mathtt{pk}_j \widehat{\gamma}_i = \mathtt{a}_i \mathtt{a}_j \mathtt{h} \gamma_j \widehat{\gamma}_i = \mathtt{a}_j \mathtt{a}_i \mathtt{h} \gamma_i \widehat{\gamma}_j = \mathtt{a}_j \mathtt{pk}_i \widehat{\gamma}_j = \mathtt{k}_j.$$

2. We will prove this statement by way of contradiction. Let us suppose $\mathcal{A}$ is an adversary against our protocol such that $\mathrm{SKAdv}[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G]$ is non-negligible. Let $\mathfrak{B}$ be an upper bound on the number of session calls by $\mathcal{A}$ in any interaction.
We now present a distinguisher $\mathcal{D}$ for the DTSP problem, depicted by Algorithm 5.

Algorithm 5 depicts distinguisher $\mathcal{D}$ for the DTSP problem

function $\mathcal{D}(h, \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G, \mathtt{pk}_1, \mathtt{pk}_2, \mathtt{k})$
  $\mathfrak{s} \xleftarrow{R} \{1, \ldots, \mathfrak{B}\}$;
  Let $\mathcal{A}$ interact with principals $P_1, \ldots, P_n$ with identifiers $\mathtt{ID}_1, \ldots, \mathtt{ID}_n$, except for the $\mathfrak{s}$-th session. For the $\mathfrak{s}$-th session, let P_i, P_j interact with each other, and run Algorithm 4. In particular, P_i will transmit $(\mathtt{ID}_i, \mathfrak{s}, \mathtt{pk}_i = \mathtt{a}_i \mathtt{h} \gamma_i)$ to P_j, while P_j will send $(\mathtt{ID}_j, \mathfrak{s}, \mathtt{pk}_j = \mathtt{a}_j \mathtt{h} \gamma_j)$ to P_i;
  if $\mathcal{A}$ has chosen the $\mathfrak{s}$-th session as its test session **then**
    Return $\mathtt{k}$ to $\mathcal{A}$ as the response to its `test` oracle query;
    $\mathtt{b} \leftarrow \mathcal{A}(\mathtt{k})$;
  else
    $\mathtt{b} \xleftarrow{R} \{0, 1\}$;
  end if
  return $\mathtt{b}$
end function

We now analyze the distinguisher $\mathcal{D}$ in two cases:

(a) Let us assume that $\mathcal{A}$ randomly chooses the $\mathfrak{s}$-th one as its test session. This implies that $\mathcal{A}$ will receive either $\mathtt{k}_0$ or $\mathtt{k}_1$. This is a result of the DTSP challenger, because upon request, it always picks a random bit b and then returns $(\mathtt{pk}_1, \mathtt{pk}_2, \mathtt{k}_b)$ to $\mathcal{D}$. Therefore, the probability of correctly distinguishing them by $\mathcal{A}$ is $1/2 + \epsilon$ with non-negligible ϵ, since, by assumption, $\mathrm{SKAdv}[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G]$ is non-negligible.

(b) If $\mathcal{A}$ does not pick the $\mathfrak{s}$-th one as its test session, then, by design, $\mathcal{D}$ returns a random bit, and therefore, the distinguishing probability is $1/2$.

Let E_r be the event of $\mathcal{A}$ selecting the test session as the $\mathfrak{s}$-th session. Hence, the probabilities of $E_\mathfrak{s}$ and $\bar{E}_\mathfrak{s}$ are $1/\mathfrak{B}$ and $1 - 1/\mathfrak{B}$, respectively. Consequently, the overall probability for $\mathcal{D}$ to win the DTSP game is

$$1/(2\mathfrak{B}) + \epsilon/\mathfrak{B} + 1/2 - 1/(2\mathfrak{B}) = 1/2 + \epsilon/\mathfrak{B},$$

which is non-negligible.

□

4.3. Generalizing Our Two-Party Key-Agreement Protocol

Throughout this section, we will focus on generalizing our two-party key-exchange protocol to a group key-exchange protocol [21–23].

Let us assume there are $\eta > 0$ principals whose respective identifiers are $ID_1, \ldots, ID_\eta$. Thus, the generalized protocol runs as follows:

1. The principal ID_1 randomly generates a secret pair $\mathtt{sk}_1 = (\mathtt{a}_1, \gamma_1) \xleftarrow{R} \mathcal{SK}$ and then transmits the list:
$$\mathcal{M}_1 = [m_1^1 = h, m_1^2 = \psi(\mathtt{sk}_1, \mathtt{h}) = a_1 h \gamma_1]$$
to the principal ID_2.

2. For $i \in \{2, \ldots, \eta - 1\}$, the principal ID_i randomly generates its secret pair $\mathtt{sk}_i = (\mathtt{a}_i, \gamma_i) \xleftarrow{R} \mathcal{SK}$ and sets $a_i = \mathtt{sk}_i$ and $b_i = \widehat{\mathtt{sk}_i}$ if i is even, or else, it sets $a_i = \widehat{\mathtt{sk}_i}$ and $b_i = \mathtt{sk}_i$. This principal constructs a list $\mathcal{M}_i$ and transmits it to the principal ID_{i+1}, where $\mathcal{M}_i$ contains $i+1$ messages enumerated as $m_i^j = \psi(a_i, m_{i-1}^j)$ for $j \in \{1, \ldots, i-1\}$, $m_i^i = m_{i-1}^i$ and $m_i^{i+1} = \psi(b_i, m_{i-1}^i)$.
3. The principal ID_η randomly generates its secret pair $\mathtt{sk}_\eta = (\mathtt{a}_\eta, \gamma_\eta) \xleftarrow{R} \mathcal{SK}$, then sets $a_\eta = \mathtt{sk}_\eta$ and $b_\eta = \widehat{\mathtt{sk}_\eta}$ if η is even. Otherwise, it sets $a_\eta = \widehat{\mathtt{sk}_\eta}$ and $b_\eta = \mathtt{sk}_\eta$.
This principal then constructs a list $\mathcal{M}_\eta$ containing η messages enumerated as $m_\eta^j = \psi(a_\eta, m_{\eta-1}^j)$ for $j \in \{1 \ldots, \eta - 1\}$ and $m_\eta^\eta = m_{\eta-1}^\eta$ and then broadcasts $\mathcal{M}_\eta$ to all other principals. Finally, this principal computes the shared key $\mathtt{k} = \psi(b_\eta, m_\eta^\eta = m_{\eta-1}^\eta)$.
4. For $i \in \{1, \ldots, \eta - 1\}$, each principal with identifier ID_i finally computes the shared key $\mathtt{k}$ by calculating either $\mathtt{k} = \psi(sk_i, m_\eta^i)$ if η is odd or $\mathtt{k} = \psi(\widehat{sk_i}, m_\eta^i)$ otherwise.

Theorem 2. *Our group key exchange protocol satisfies the following:*

1. *If η uncorrupted principals having identifiers $ID_1, \ldots, ID_\eta$, respectively, complete a run of the group key-agreement protocol successfully, then each principal will output the same shared key.*
2. *If the DTSP assumption holds for $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$, then an efficient adversary $\mathcal{A}$ will not be able to distinguish the shared group key from an arbitrary element in $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$.*

Proof.

1. This first statement will be proven by induction on the number of principals η.
Base case:
Let us set η to 2. For this case, the principal ID_1 computes $\mathcal{M}_1 = \{h, a_1 h \gamma_1\}$ and transmits it to the principal ID_2. Upon receiving $\mathcal{M}_1$, the principal ID_2 computes $\mathcal{M}_2 = [a_2 h \gamma_2]$ and $k = a_2 a_1 h \gamma_1 \widehat{\gamma_2}$. Finally, it transmits $\mathcal{M}_2$ back to the principal ID_1, which computes $k = a_2 a_1 h \gamma_1 \widehat{\gamma_2}$.
Inductive case:
By induction hypothesis, we assume that, if η uncorrupted principals run the protocol, each will successfully obtain the corresponding shared key k_η.
We will prove that if $\eta + 1$ uncorrupted principals run the protocol, each will successfully obtain the corresponding shared key $k_{\eta+1}$.
Assume now that there are $\eta + 1$ principals running the protocol. By the induction hypothesis, the principal ID_η receives the correct $\mathcal{M}_{\eta-1}$ from the principal $ID_{\eta-1}$ and, hence, correctly computes $\mathcal{M}_\eta$ and forwards it to the principal $ID_{\eta+1}$. This principal constructs the list $M_{\eta+1}$ by calculating the following:
   (a) $\mathcal{M}_{\eta+1} = \{m_{\eta+1}^i = \psi(sk_{\eta+1}, m_\eta^i)$ for $i \in \{1, 2, \ldots, \eta\}\}$ if $\eta + 1$ is even. This principal then transmits $\mathcal{M}_{\eta+1}$ to the other principals.
   Upon receiving $\mathcal{M}_{\eta+1}$ from the principal $ID_{\eta+1}$, each principal ID_i computes

$$\begin{aligned} k_{\eta+1} = \psi(\widehat{sk_i}, m_{\eta+1}^i) &= a_i m_{\eta+1}^i \widehat{\gamma_i} \\ &= a_i \psi(sk_{\eta+1}, m_\eta^i) \widehat{\gamma_i} = a_i a_{\eta+1} m_\eta^i \gamma_{\eta+1} \widehat{\gamma_i} \\ &= a_{\eta+1} k_\eta \widehat{\gamma_{\eta+1}}, \end{aligned}$$

   for $i \in \{1, \ldots, \eta\}$, while the principal $ID_{\eta+1}$ computes

$$\begin{aligned} k_{\eta+1} = \psi(\widehat{sk_{\eta+1}}, m_\eta^{\eta+1}) &= a_{\eta+1} m_\eta^{\eta+1} \widehat{\gamma_{\eta+1}} \\ &= a_{\eta+1} \psi(sk_\eta, m_{\eta-1}^\eta) \widehat{\gamma_{\eta+1}} = a_{\eta+1} k_\eta \widehat{\gamma_{\eta+1}}. \end{aligned}$$

(b) $\mathcal{M}_{\eta+1} = \{m^i_{\eta+1} = \psi(\widehat{sk_{\eta+1}}, m^i_\eta)$ for $i \in \{1, 2, \ldots, \eta\}\}$ if $\eta + 1$ is odd. This principal then transmits $\mathcal{M}_{\eta+1}$ to the other principals.
Upon receiving $\mathcal{M}_{\eta+1}$ from the principal $ID_{\eta+1}$, each principal ID_i computes

$$\begin{aligned} k_{\eta+1} = \psi(sk_i, m^i_{\eta+1}) &= a_i m^i_{\eta+1}\gamma_i \\ &= a_i\psi(\widehat{sk_{\eta+1}}, m^i_\eta)\gamma_i = a_i a_{\eta+1} m^i_\eta \widehat{\gamma_{\eta+1}}\gamma_i \\ &= a_{\eta+1}k_\eta\gamma_{\eta+1}, \end{aligned}$$

for $i \in \{1, \ldots, \eta\}$, while the principal $ID_{\eta+1}$ computes

$$\begin{aligned} k_{\eta+1} = \psi(sk_{\eta+1}, m^{\eta+1}_\eta) &= a_{\eta+1} m^{\eta+1}_\eta \gamma_{\eta+1} \\ &= a_{\eta+1}\psi(\widehat{sk_\eta}, m^\eta_{\eta-1})\gamma_{\eta+1} = a_{\eta+1}k_\eta\gamma_{\eta+1}. \end{aligned}$$

This concludes the proof of this first statement.

2. To prove this second statement, we can easily adapt the proof of Theorem 1 in [6] to this setting.

□

5. An Encryption Scheme from Twisted-Skew Group Rings

This section focuses on presenting a public key-encryption scheme that is procured from the two-party key-agreement protocol analyzed in Section 4.2. Our derivation approach mimics a well-known generic approach previously employed to obtain a probabilistic encryption scheme from instances of a key-exchange protocol [4,5,7,12,24] as, for instance, ElGamal encryption [25] being obtained from instances of the Diffie–Hellman protocol [26].

Before presenting our probabilistic public key encryption, we first establish some notation. The public key space is denoted by $\mathcal{PK} = \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$; the message space is denoted by $\mathcal{M} = \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$; finally, the ciphertext space is denoted by $\mathcal{C} = \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$.

We now introduce the public key-encryption scheme $\mathcal{E} = (\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$. Algorithm 6 describes the function `Gen`; Algorithm 7 depicts the function `Enc`; finally, Algorithm 8 presents the function `Dec`.

Algorithm 6 generates a key pair

1: **function** GEN($\mathtt{h} \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$)
2: $(\mathtt{a}_1, \gamma_1) \xleftarrow{R} \mathcal{SK}$;
3: $\mathtt{sk} \leftarrow (\mathtt{a}_1, \gamma_1)$;
4: $\mathtt{pk} \leftarrow \psi(\mathtt{sk}, h)$;
5: **return** sk, pk;
6: **end function**

Algorithm 7 encrypts a message and returns a ciphertext

1: **function** ENC($\mathtt{m} \in \mathcal{M}, \mathtt{pk} \in \mathcal{PK}, \mathtt{r}_2 \in \mathcal{SK}, \mathtt{h} \in \mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda} G$)
2: $(\mathtt{a}_2, \gamma_2) \leftarrow \mathtt{r}_2$;
3: $\mathtt{c}_1 \leftarrow \psi((\mathtt{a}_2, \gamma_2), h)$;
4: $\mathtt{c}_2 \leftarrow \mathtt{m} + \psi((\mathtt{a}_2, \widehat{\gamma_2}), pk)$;
5: $\mathtt{c} \leftarrow (\mathtt{c}_1, \mathtt{c}_2)$;
6: **return** c;
7: **end function**

Algorithm 8 decrypts a ciphertext and returns a message

function $\texttt{Dec}(\texttt{c} \in \mathcal{C}, \texttt{sk} \in \mathcal{SK})$
  $(\texttt{c}_1, \texttt{c}_2) \leftarrow \texttt{c}$;
  $\texttt{k} \leftarrow \psi(\widehat{\texttt{sk}}, \texttt{c}_1)$;
  $\texttt{m} \leftarrow \texttt{c}_2 - \texttt{k}$;
  return $\texttt{m}$;
end function

Theorem 3. *Let h be a public element in $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$, and let $\mathcal{E}$ be the public key-encryption scheme:*

1. *For any message $m \in \mathcal{M}$, $r_2 \xleftarrow{R} \mathcal{SK}$ and $(pk, sk) \leftarrow Gen(h)$, it holds that*

$$m \leftarrow Dec(Enc(m, pk, r_2, h), sk)$$

2. *If the DTSP assumption holds for $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$, then $\mathcal{E}$ is semantically secure.*

Proof. The proofs of these two items are an adaptation of the proofs of Lemma 5.1 and Theorem 5.2, respectively, from [7]:

1. Given $\texttt{m} \in \mathcal{M}$, $\texttt{r}_2 \in \mathcal{SK}$ uniformly chosen at random and $(\texttt{pk}, \texttt{sk}) \leftarrow \texttt{Gen}(\texttt{h})$, then we have
$$(\texttt{c}_1 = \texttt{a}_2\texttt{h}\gamma_2, \texttt{c}_2 = \texttt{m} + \texttt{a}_2\texttt{pk}\widehat{\gamma}_2) \leftarrow \texttt{Enc}(\texttt{m}, \texttt{pk}, \texttt{r}_2, \texttt{h}).$$
By calling $\texttt{Dec}((\texttt{c}_1, \texttt{c}_2), \texttt{sk})$, then
$$\texttt{k} = \texttt{a}_1\texttt{c}_1\widehat{\gamma}_1 = \texttt{a}_1\texttt{a}_2\texttt{h}\gamma_2\widehat{\gamma}_1 = \texttt{a}_2\texttt{a}_1\texttt{h}\gamma_1\widehat{\gamma}_2 = \texttt{a}_2\texttt{pk}\widehat{\gamma}_2,$$
and therefore,
$$\texttt{c}_2 - \texttt{k} = \texttt{m} + \texttt{a}_2\texttt{pk}\widehat{\gamma}_2 - \texttt{a}_2\texttt{pk}\widehat{\gamma}_2 = \texttt{m}.$$
2. For this statement, we consider an efficient adversary $\mathcal{A}$ and define $\texttt{Game}_0$ as the semantic security (SS) attack game against $\mathcal{A}$. Algorithm 9 depicts $\texttt{Game}_0$.

Algorithm 9 depicts attack games defined for the proof of Theorem 3

function $\textsc{Game}_0$
  $((\texttt{a}_1, \gamma_1), \texttt{pk}_1) \xleftarrow{R} \texttt{Gen}(h)$;
  $(\texttt{m}_0, \texttt{m}_1) \leftarrow \mathcal{A}(\texttt{pk}_1)$;
  $\texttt{b} \xleftarrow{R} \{0, 1\}$;
  $(\texttt{a}_2, \gamma_2) \xleftarrow{R} \mathcal{SK}$; $\texttt{pk}_2 \leftarrow \texttt{a}_2 h\gamma_2$;
  $\texttt{k} \leftarrow \texttt{a}_2\texttt{pk}_1\widehat{\gamma_2}$;
  $\texttt{c} \leftarrow \texttt{m}_b + \texttt{k}$;
  $\tilde{\texttt{b}} \leftarrow \mathcal{A}(\texttt{pk}_1, \texttt{pk}_2, \texttt{c})$;
  return $[[\texttt{b} = \tilde{\texttt{b}}]]$;
end function

function $\textsc{Game}_1$
  $((\texttt{a}_1, \gamma_1), \texttt{pk}_1) \xleftarrow{R} \texttt{Gen}(h)$;
  $(\texttt{m}_0, \texttt{m}_1) \leftarrow \mathcal{A}(\texttt{pk}_1)$;
  $\texttt{b} \xleftarrow{R} \{0, 1\}$;
  $(\texttt{a}_2, \gamma_2) \xleftarrow{R} \mathcal{SK}$; $\texttt{pk}_2 \leftarrow \texttt{a}_2 h\gamma_2$;
  $(\texttt{a}_3, \gamma_3) \xleftarrow{R} \mathcal{SK}$; $\texttt{k} \leftarrow \texttt{a}_3 h\gamma_3$;
  $\texttt{c} \leftarrow \texttt{m}_b + \texttt{k}$;
  $\tilde{\texttt{b}} \leftarrow \mathcal{A}(\texttt{pk}_1, \texttt{pk}_2, \texttt{c})$;
  return $[[\texttt{b} = \tilde{\texttt{b}}]]$;
end function

Recall $[[\texttt{b} = \tilde{\texttt{b}}]]$ denotes a Boolean value. In particular, $[[\texttt{b} = \tilde{\texttt{b}}]]$ is 0 if both $\texttt{b}$ and $\tilde{\texttt{b}}$ differ, or 1 otherwise. In $\texttt{Game}_0$, $\mathcal{A}$ sends the challenger two messages $\texttt{m}_0, \texttt{m}_1 \in \mathcal{M}$ of the same length in bits.

Let us define S_0 as the event of $\texttt{Game}_0$ returning 1, then $\mathcal{A}$'s SS-advantage is given by $|Pr[S_0] - 1/2|$. The proof essentially will show that $|Pr[S_0] - 1/2|$ is negligible if the DTSP assumption holds. Let us define $\texttt{Game}_1$ by modifying $\texttt{Game}_0$ as shown in Algorithm 9.

Let us define S_1 as the event of $\texttt{Game}_1$ outputting 1. From $\texttt{Game}_1$, it is clear that $\texttt{b}$, $\texttt{pk}_1$, $\texttt{pk}_2$, and $\texttt{c}$ are mutually independent; so are $\texttt{b}$ and $\tilde{\texttt{b}} \leftarrow \mathcal{A}(\texttt{pk}_1, \texttt{pk}_2, \texttt{c})$, and hence, $Pr[S_1] = 1/2$.

We now demonstrate an adversary $\mathcal{B}$ that performs the DTSP attack game 3 defined in Section 3. In particular, the adversary $\mathcal{B}$ will assume the role of challenger for $\mathcal{A}$, and its part is as follows. $\mathcal{B}$ will first communicate with its own challenger from which it will receive the three-tuple $(\mathtt{pk}_1, \mathtt{pk}_2, \mathtt{k})$. It then forwards $\mathtt{pk}_1$ to $\mathcal{A}$. When it obtains $(\mathtt{m}_0, \mathtt{m}_1)$ from $\mathcal{A}$, then $\mathcal{B}$ chooses a bit $\mathtt{b}$ at random, computes $\mathtt{c} \leftarrow \mathtt{m}_b + \mathtt{k}$, and transmits $(\mathtt{pk}_1, \mathtt{pk}_2, \mathtt{c})$ to $\mathcal{A}$. Once $\mathcal{B}$ finally procures a final response bit $\tilde{\mathtt{b}}$ by $\mathcal{A}$, it returns $[[\mathtt{b} = \tilde{\mathtt{b}}]]$ in the DTSP attack game. Clearly, $\mathcal{B}$ is an efficient adversary, since $\mathcal{A}$ is also an efficient adversary.
Recall that $W_{\tilde{b}}$ is the event that $\mathcal{B}$ outputs 1 in game $\mathtt{DTSP}(\tilde{\mathtt{b}})$; thus, $\mathcal{B}$'s advantage for solving the DTSP problem for $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$ is given by

$$\mathrm{DTSPadv}[\mathcal{A}, \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G] = |\Pr[W_0] - \Pr[W_1]|.$$

A key observation here is the following.
On the one hand, whenever $\mathcal{B}$'s challenger is playing game $\mathtt{DTSP}(0)$, $\mathcal{A}$ is in turn playing $\mathtt{Game}_0$, since $\mathcal{B}$ obtains from its challenger

$$(\mathtt{pk}_1 = \mathtt{a}_1 h \gamma_1, \mathtt{pk}_2 = \mathtt{a}_2 h \gamma_2, \mathtt{k} = \mathtt{a}_2 \mathtt{pk}_1 \widehat{\gamma_2}).$$

Therefore, $Pr[W_0] = Pr[S_0]$.
On the other hand, whenever $\mathcal{B}$'s challenger is playing Game $\mathtt{DTSP}(1)$, $\mathcal{A}$ is in turn playing $\mathtt{Game}_1$, because $\mathcal{B}$ obtains from its challenger

$$(\mathtt{pk}_1 = \mathtt{a}_1 h \gamma_1, \mathtt{pk}_2 = \mathtt{a}_2 h \gamma_2, \mathtt{k} = \mathtt{a}_3 h \gamma_3).$$

Therefore, $Pr[W_1] = Pr[S_1]$.
By hypothesis, $|Pr[W_0] - Pr[W_1]|$ is negligible; therefore, $|Pr[S_0] - 1/2|$ is negligible, and the assertion follows.

□

6. A Key-Encapsulation Mechanism from Twisted-Skew Group Rings

In this section, we will focus on deriving a CCA-secure key-encapsulation mechanism from our probabilistic public key encryption $\mathcal{E}$. To accomplish this task, we will apply a generic transformation from [27] to $\mathcal{E}$. We will next describe this generic transformation a bit more.

Let $\mathtt{PKE} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ be a public key-encryption scheme with message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, and randomness space $\mathcal{R}$. Let $\mathtt{KeyLength} \in \mathbb{N}$ and $\mathcal{G} : \mathcal{M} \to \mathcal{R}$ and $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{\mathtt{KeyLength}}$ be hash functions. This transformation is a variant of the Fujisaki–Okamoto transformation with "implicit rejection" of inconsistent ciphertexts. Formally, it is defined as $\mathtt{KEM}^{\not\perp} = \mathtt{FO}^{\not\perp}(\mathtt{PKE}, \mathcal{G}, \mathcal{H}) := \mathrm{U}^{\not\perp}[\mathrm{T}[\mathtt{PKE}, \mathcal{G}], \mathcal{H}] = (\mathtt{Gen}, \mathtt{Encaps}, \mathtt{Decaps})$ (see [27] for more details). Algorithm 10 summarizes functions $(\mathtt{Gen}, \mathtt{Encaps}, \mathtt{Decaps})$ after applying the transformation to $\mathtt{PKE}$, converting it into a CCA-secure key-encapsulation mechanism. We remark that the proof that this generic transformation converts a public key encryption scheme into a CCA-secure key-encapsulation mechanism may be found in [27].

Algorithm 10 depicts the CCA-secure key-encapsulation mechanism $(\texttt{Gen}, \texttt{Encaps}, \texttt{Decaps})$ from PKE

function GEN()
  $(\texttt{pk}, \texttt{sk}) \leftarrow \texttt{PKE.Gen}$;
  $\texttt{s} \xleftarrow{R} \mathcal{M}$;
  $\texttt{sk}' \leftarrow (\texttt{sk}, s)$;
  return $(\texttt{sk}', \texttt{pk})$;
end function

function ENCAPS(pk)
  $\texttt{m} \xleftarrow{R} \mathcal{M}$;
  $\texttt{c} \leftarrow \texttt{PKE.Enc}(\texttt{pk}, m, \mathcal{G}(m))$;
  $\texttt{k} \leftarrow \mathcal{H}(\texttt{m}, \texttt{c})$;
  return $(\texttt{k}, \texttt{c})$;
end function

function DECAPS($\texttt{sk}' = (\texttt{sk}, \texttt{s}), \texttt{c}$)
  $\texttt{m}' \leftarrow \texttt{PKE.Dec}(\texttt{sk}, \texttt{c})$;
  if $\texttt{c} = \texttt{PKE.Enc}(\texttt{pk}, \texttt{m}', \mathcal{G}(m'))$ **then**
    return $\mathcal{H}(\texttt{m}', \texttt{c})$;
  else
    return $\mathcal{H}(\texttt{s}, \texttt{c})$;
  end if
end function

To apply this transformation to our scheme $\mathcal{E}$, we proceed by establishing the following. Let $\mathcal{K} = \{0,1\}^{\texttt{KeyLength}}$ be the key space and $\texttt{BinaryRep}(x)$ be a function that returns the binary representation of x. Furthermore, recall that the randomness space is $\mathcal{SK} = \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} N \times \Gamma_{\theta_\sigma, \alpha_\lambda}$, the public key space is $\mathcal{PK} = \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$, the message space is $\mathcal{M} = \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$, and the ciphertext space is $\mathcal{C} = \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$. Finally, we will define the hash function $\mathcal{H}_1$ and $\mathcal{H}_2$ as follows:

- $\mathcal{H}_1 : \{0,1\}^* \longrightarrow \mathcal{SK}$ is a hash function, which, upon the input of a variable-length bit string $\texttt{x}$, returns $(\texttt{a}, \gamma) \in \mathcal{SK}$. Using the notation of [28], this function may be defined as $\mathcal{H}_1(\texttt{x}) = \texttt{SHAKE}_{256}(\texttt{x}, \zeta)$, where $\zeta = 2\lceil \log_2(q) \rceil (n + \lceil \frac{n}{2} \rceil)$ is the bit length of the output and $|N| = |G|/2 = n$. The bit string $\texttt{bitstring}$ returned by $\texttt{SHAKE}_{256}$ can be converted into a element in $\mathcal{SK}$ by carefully dividing the $\texttt{bit string}$ into two parts, the first of length $2\lceil \log_2(q) \rceil n$ bits and the second of length $2\lceil \log_2(q) \rceil \lceil \frac{n}{2} \rceil$ bits, each being employed to derive $\texttt{a}, \gamma$, respectively.
  Let $\texttt{m}_1, \texttt{m}_2 \in \mathcal{M}$, and we define $\mathcal{G}(\texttt{m}_1, \texttt{m}_2) := \mathcal{H}_1\big(\texttt{BinaryRep}(\texttt{m}_1) || \texttt{BinaryRep}(\texttt{m}_2)\big)$.
- $\mathcal{H}_2 : \{0,1\}^* \longrightarrow \mathcal{K}$ is a hash function that, upon the input of a variable-length bit string $\texttt{x}$, returns $\texttt{k} \in \mathcal{K}$. This function may be defined as $\mathcal{H}_2(\texttt{x}) = \texttt{SHAKE}_{256}(\texttt{p}_1 || \texttt{x}, \texttt{KeyLength})$, where $\texttt{KeyLength}$ is the bit length of the output and $\texttt{p}_1$ is a prepended fixed bit string to make it different from $\mathcal{H}_1$.
  Let $\texttt{m}, \texttt{c} \in \mathcal{M}$. We define $\mathcal{H}(\texttt{m}, \texttt{c}) := \mathcal{H}_2\big(\texttt{BinaryRep}(\texttt{m}) || \texttt{BinaryRep}(\texttt{c})\big)$.

After applying the generic transformation to $\mathcal{E}$, i.e., $\mathrm{U}^{\not\perp}[\mathrm{T}[\mathcal{E}, \mathcal{G}], \mathcal{H}]$, we obtain $\texttt{KEM} = (\texttt{KeyGen}, \texttt{Encaps}, \texttt{Decaps})$. Algorithm 11 describes the functions $\texttt{KeyGen}$, $\texttt{Encaps}$ and $\texttt{Decaps}$.

Algorithm 11 depicts the CCA-secure key-encapsulation mechanism $(\texttt{KeyGen}, \texttt{Encaps}, \texttt{Decaps})$ from $\mathcal{E}$

function KEYGEN(h)
  $(\texttt{pk}, \texttt{sk}) \leftarrow \mathcal{E}.\texttt{Gen}(\texttt{h})$;
  $\texttt{s} \xleftarrow{R} \mathcal{M}$;
  return $(\texttt{sk}, \texttt{s}, \texttt{pk})$;
end function

function ENCAPS(pk, h)
  $\texttt{m} \xleftarrow{R} \mathcal{M}$;
  $\texttt{r} \leftarrow \mathcal{G}(\texttt{m}, \texttt{pk})$;
  $\texttt{c} \leftarrow \mathcal{E}.\texttt{Enc}(\texttt{m}, \texttt{pk}, \texttt{r}, \texttt{h})$;
  $\texttt{K} \leftarrow \mathcal{H}(\texttt{m}, \texttt{c})$;
  return $(\texttt{K}, \texttt{c})$;
end function

function DECAPS($(\texttt{sk}, \texttt{s}, \texttt{pk}), \texttt{c}, \texttt{h}$)
  $\texttt{m}' \leftarrow \mathcal{E}.\texttt{Dec}(\texttt{c}, \texttt{sk})$;
  $\texttt{r}' \leftarrow \mathcal{G}(\texttt{m}', \texttt{pk})$;
  if $\texttt{c} = \mathcal{E}.\texttt{Enc}(\texttt{m}', \texttt{pk}, \texttt{r}', \texttt{h})$ **then**
    return $\mathcal{H}(\texttt{m}', \texttt{c})$;
  else
    return $\mathcal{H}(\texttt{s}, \texttt{c})$;
  end if
end function

7. Implementation of Our Cryptographic Constructions

The proof-of-concept implementation of our cryptographic constructions was coded in Python. The interested reader can see it on Google Colaboratory [29].

7.1. Group Representation

Recall that $G = N \cup Ny$, where $|N| = |G|/2 = n$. For our protocols, we only considered $G \in \{D, \mathfrak{G}, \mathfrak{M}, \mathfrak{Q}\}$. For any choice, $N = \langle x^i \rangle$ is a cyclic group, and thus, a group element $g \in G$ is of the form $g = x^i y^j$, which may be represented as an integer $\mathtt{g} = j \cdot n + i$, where $i \in \{0, \ldots, n-1\}$ and $j \in \{0, 1\}$.

The computation of the integer representation of either $g_1 \cdot g_2$ or g_1^{-1}, $g_1, g_2 \in G$, will hinge on the form of the group elements and the specific presentation of G. Note that, by exploiting each group presentation, explicit formulae can be derived to compute both $g_1 \cdot g_2$ and g_1^{-1} efficiently. The interested reader can see the implementation [29].

7.2. Two-Cocycle α_λ

The function $\mathtt{2cocycle}(k_1, k_2)$ takes two group element representations, k_1 and k_2, as the input, then the function returns λ if $n \leq k_1 < 2n$ and $n \leq k_2 < 2n$. Otherwise, it returns 1.

7.3. Homomorphism θ_σ

The function $\mathtt{homomorphism}(k_1)$ takes a group element representation, k_1, as the input, then this function returns a pointer to the function σ if $n \leq k_1 < 2n$. Otherwise, it returns a pointer to the identity function `I`. Algorithm 12 shows both functions.

Algorithm 12 presents functions involved in computing the homomorphism θ_σ

```
1:  function σ(a ∈ F_{q^2})
2:      [b_s, b_{s−1}, ..., b_0] ← BinaryRep(q);
3:      r ← getOneFromQuadraticField();
4:      for i ← s to 0 do
5:          r ← r · r;
6:          if b_i = 1 then
7:              r ← r · a;
8:          end if
9:      end for
10:     return r;
11: end function
```

```
1:  function I(a ∈ F_{q^2})
2:      return a
3:  end function
```

7.4. The Twisted-Skew Group Ring $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$

To represent an element $a = \sum_{i=0}^{n-1} a_i x^i + \sum_{i=0}^{n-1} a_{n+i} x^i y$ in the group ring $\mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} G$, we make use of an array of $2n$ field elements $\mathtt{a} = [\mathtt{a}_0, \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_{2n-1}]$, where $\mathtt{a}_i$ is the representation of the field element $a_i \in \mathbb{F}_{q^2}$. Algorithms 13 and 14 describe the addition and product operations, respectively.

Algorithm 13 computes the addition of two ring elements

```
1:  function ADDITION(a, b)
2:      c ← [0, ··· , 0];
3:      for (i ← 0; i < 2n; i ← i + 1) do
4:          c[i] ← a[i] + b[i];
5:      end for
6:      return c;
7:  end function
```

Algorithm 14 computes the product of two ring elements

```
1:  function PRODUCT(a, b)
2:      c ← [0, ··· , 0];
3:      for (i ← 0; i < 2n; i ← i + 1) do
4:          for (j ← 0; j < 2n; j ← j + 1) do
5:              k ← G.eval(i, j);
6:              outH ← homomorphism(i)(b[j]);
7:              out2c ← 2cocylce(i, j);
8:              fe ← a[i] · outH · out2c;
9:              c[k] ← c[k] + fe;
10:         end for
11:     end for
12:     return c;
13: end function
```

Addition and Product Costs

We now quantify the cost of Algorithms 13 and 14. Let us denote

- FA and FM as the costs of a field addition and a field multiplication respectively.
- GE and HC as bounds on the cost of calling $\texttt{G.eval}(i, j)$ and the number of field multiplications to compute $\texttt{homomorphism}(i)(b[j])$ respectively.
- $\texttt{C}_{\alpha_\lambda}$ as the constant cost of executing $\texttt{2cocylce}(i, j)$.

On the one hand, Algorithm 13 has a cost of $2n\texttt{FA}$ when computing a ring element c. On the other hand, Algorithm 14 has a cost of $4n^2(\texttt{FA} + (2 + \texttt{HC})\texttt{FM} + \texttt{GE} + \texttt{C}_{\alpha_\lambda})$.

7.5. Auxiliary Functions

As auxiliary functions, we implemented the following functions:

1. Algorithm 15 computes the adjunct of a ring element, and its cost is $2n(\texttt{GI} + (\texttt{HC} + 1)\texttt{FM} + \texttt{C}_{\alpha_\lambda})$, where GI is a bound on the cost of calling $\texttt{G.inverse}(i)$.
2. Functions for computing random elements in different sets are implemented. They are described in Algorithm 16.

Algorithm 15 computes the adjunct of a ring element

```
1:  function ADJUNCT(a)
2:      c ← [0, ··· , 0];
3:      for (i ← 0; i < 2n; i ← i + 1) do
4:          j ← inverse(i);
5:          f1 ← homomorphism(j)(a[i]);
6:          f2 ← 2cocylce(i, j);
7:          c[j] ← f1 · f2;
8:      end for
9:      return c
10: end function
```

Algorithm 16 presents functions for computing a random element in different sets

```
1: function getPublicElement()
2:     sw_1 ← False;
3:     while not sw_1 do
4:         a ← getRandomFG();
5:         i ← 0;
6:         sw_2 ← False;
7:         while i < n and not sw_2 do
8:             if a[i] ≠ 0 then
9:                 sw_2 ← True;
10:            end if
11:            i ← i + 1;
12:        end while
13:        i ← n;
14:        sw_3 ← False;
15:        while i < 2n and not sw_3 do
16:            if a[i] ≠ 0 then
17:                sw_3 ← True;
18:            end if
19:            i ← i + 1;
20:        end while
21:        sw_1 ← sw_2 and sw_3;
22:    end while
23:    return a;
24: end function
```

```
1: function getRandomfromT()
2:     c ← [0, ··· , 0];
3:     c[n] ← getRandomFieldElement();
4:     n_1 ← n/2;
5:     for (i ← 1; i ≤ n_1; i ← i + 1) do
6:         c[i + n] ← getRandomFieldElement();
7:         c[n + (n − i) mod n] ← c[i + n];
8:     end for
9:     return c;
10: end function
```

```
1: function getRandomFG()
2:     c ← [0, ··· , 0];
3:     for (i ← 0; i < 2n; i ← i + 1) do
4:         c[i] ← getRandomFieldElement();
5:     end for
6:     return c;
7: end function
```

```
1: function getRandomFH()
2:     c ← [0, ··· , 0];
3:     for (i ← 0; i < n; i ← i + 1) do
4:         c[i] ← getRandomFieldElement();
5:     end for
6:     return c;
7: end function
```

```
1: function getRandomFHy()
2:     c ← [0, ··· , 0];
3:     for (i ← n; i < 2n; i ← i + 1) do
4:         c[i] ← getRandomFieldElement();
5:     end for
6:     return c;
7: end function
```

7.6. Key Sizes

We next provide estimates for the memory sizes in bits required to store both a public key and a private key.

A field element requires $\mathtt{NFE} = 2\lceil \log_2(q) \rceil$ bits. On the one hand, a public key $\mathtt{pk} \in \mathcal{PK}$ is a ring element, which can be stored as an array of $|G|$ field elements. Therefore, storing a public key requires $|G| \cdot \mathtt{NFE}$ bits.

On the other hand, a private key $(\mathtt{a}, \gamma) \in \mathcal{SK}$ is a pair of two ring elements. Therefore, storing a full private key requires $2 \cdot |G| \cdot \mathtt{NFE}$ bits. This number of bits can be decreased further if the form of the private key is exploited. Note that, since $(\mathtt{a}, \gamma) \in \mathbb{F}_{q^2}^{\theta_\sigma, \alpha_\lambda} N \times \Gamma_{\theta_\sigma, \alpha_\lambda}$, only $n + \lceil \frac{n}{2} \rceil$ field elements need storing, and hence, a compressed private key requires $(n + \lceil \frac{n}{2} \rceil) \cdot \mathtt{NFE}$ bits. For completeness, Algorithms 17 and 18 describe the process of compressing and decompressing a private key, respectively.

Algorithm 17 compresses a private key

function compressPrivateKey(sk $\in \mathcal{SK}$)
  $\mathtt{l} \leftarrow n + \lceil n/2 \rceil$;
  $\mathtt{c} \leftarrow [0_0, \cdots, 0_{l-1}]$;
  for $(i \leftarrow 0; i < n; i \leftarrow i + 1)$ **do**
    $\mathtt{c}[i] \leftarrow \mathtt{a}[i]$;
  end for
  for $(i \leftarrow n; i < \mathtt{l}; i \leftarrow i + 1)$ **do**
    $\mathtt{c}[i] \leftarrow \gamma[n + i]$;
  end for
  return c;
end function

Algorithm 18 decompresses a private key

function decompressPrivateKey(sk)
  $\mathtt{l} \leftarrow \lceil n/2 \rceil$;
  $\mathtt{a} \leftarrow [0_0, \cdots, 0_{2n-1}]$;
  $\gamma \leftarrow [0_0, \cdots, 0_{2n-1}]$;
  for $(i \leftarrow 0; i < n; i \leftarrow i + 1)$ **do**
    $\mathtt{a}[i] \leftarrow \mathtt{sk}[i]$;
  end for
  $\gamma[n] \leftarrow \mathtt{sk}[n]$;
  for $(i \leftarrow 1; i < \mathtt{l}; i \leftarrow i + 1)$ **do**
    $\gamma[n + i] \leftarrow \mathtt{sk}[n + i]$;
    $\gamma[2n - i] \leftarrow \mathtt{sk}[n + i]$;
  end for
  return $(\mathtt{a}, \gamma)$;
end function

7.7. Parameter Choices

In reference to our key-encapsulation mechanism, we suggest using the parameters displayed by Table 1, which supplies varying and increasing security levels. Table 1 displays four sets of parameters, where KeyLength $\in \{128, 192, 256\}$ denotes the output key length. The values displayed in the column labeled as "Level of Security in Bits" have been computed as proposed in [7]. The interested reader may see our implementation here [29].

Table 1. Proposed parameters.

p	m	n	Group	KeyLength (bits)	Level of Security in Bits
19	1	20	Dihedral	$\{128, 192, 256\}$	130
19	1	23	Dihedral	$\{128, 192, 256\}$	149
19	1	32	Any of the four candidate groups	$\{128, 192, 256\}$	207
19	1	64	Any of the four candidate groups	$\{128, 192, 256\}$	410

7.8. Potential Applications

We believe that our protocols might find applications in environments like the Internet of Things (IoT) for various reasons. One first reason is that they may potentially be implemented in constrained devices and the overhead of running them in those devices might be small. This viewpoint stems from observing that the algorithms involved in computing encryptions (or shared keys) are relatively simple, as evinced in this section. Secondly, the key sizes are relatively small compared to other schemes [3], which offers an advantage for storing purposes. Furthermore, we remark that the study on the deployability of post-quantum cryptographic algorithms on constrained devices is of current interest,

as evidenced in [30]. On the other hand, we also believe that it might be possible to derive password authentication key exchange (PAKE) protocols from our protocols. If so, these PAKE protocols are versatile and may be used in many scenarios, such as credential recovery, device paring, and end-to-end (E2E)-secure channels, as shown in [31]. However, our protocols per se might be adapted and used in some of those potential scenarios, particularly E2E-secure channels.

8. Conclusions

This paper introduced the twisted-skew group ring $\mathbb{F}_{q^2}^{\theta_\sigma,\alpha_\lambda}G$, where α_λ is a two-cocycle, θ_σ a group homomorphism, and G a finite group of even order with $N \leq G$ such that $|N| = |G|/2 = n$, i.e., $[G : N] = 2$, and hence, $G = N \cup Ny$ with $y \in G \setminus N$. Over this algebraic platform, we built several cryptographic constructions following a incremental-like methodology. In particular, we first introduced a two-party key-agreement protocol and its generalization. Additionally, we derived a probabilistic public key encryption from the two-party key-agreement protocol and key-encapsulation mechanism from the probabilistic public key encryption.

As a future research direction, it would be interesting to explore the possibility of constructing other key-exchange protocols from twisted-skew group rings, namely a password authentication key exchange protocol, which might be suitable in environments like the IoT.

Author Contributions: Conceptualization, J.d.l.C., E.M.-M., S.M.-R. and R.V.-P.; methodology, J.d.l.C., E.M.-M., S.M.-R. and R.V.-P.; software, R.V.-P.; validation, J.d.l.C., E.M.-M., S.M.-R. and R.V.-P.; formal analysis, J.d.l.C., E.M.-M., S.M.-R. and R.V.-P.; investigation, J.d.l.C., E.M.-M., S.M.-R. and R.V.-P.; resources, J.d.l.C.; writing—original draft preparation, J.d.l.C., E.M.-M. and R.V.-P.; writing—review and editing, J.d.l.C., E.M.-M. and R.V.-P.; supervision, J.d.l.C. and R.V.-P.; project administration, J.d.l.C.; funding acquisition, J.d.l.C. All authors have read and agreed to the published version of the manuscript.

Funding: The first author is grateful for the support of Fundación para la Promoción de la Investigación y la Tecnología del Banco de la República under project 4649, and the second author is partially supported by Grant TED2021-130358B-I00 funded by MCIN/AEI/10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR".

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. National Institute of Standards and Technology, NIST Post-Quantum Cryptography. Available online: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022 (accessed on 1 June 2024).
2. National Institute of Standards and Technology, Post-Quantum Cryptography: Digital Signature Schemes. Available online: https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures (accessed on 1 June 2024).
3. Dam, D.-T.; Tran, T.-H.; Hoang, V.-P.; Pham, C.-K.; Hoang, T.-T. A Survey of Post-Quantum Cryptography: Start of a New Race. *Cryptography* **2023**, *7*, 40. [CrossRef]
4. de la Cruz, J.; Martínez-Moro, E.; Villanueva-Polanco, R. Public Key Protocols over Skew Dihedral Group Rings. *Mathematics* **2022**, *10*, 3343. [CrossRef]
5. Gómez Olvera, M.D.; López Ramos, J.A.; Torrecillas Jover, B. Public Key Protocols over Twisted Dihedral Group Rings. *Symmetry* **2019**, *11*, 1019. [CrossRef]
6. Gómez Olvera, M.D.; López Ramos, J.A.; Torrecillas Jover, B. Secure Group Communications Using Twisted Group Rings. *Mathematics* **2022**, *10*, 2845. [CrossRef]
7. de la Cruz, J.; Villanueva-Polanco, R. Public key cryptography based on twisted dihedral group algebras. *Adv. Math. Commun.* **2024**, *18*, 857–877. [CrossRef]
8. Suo, J.; Wang, L.; Yang, S.; Zheng, W.; Zhang, J. Quantum algorithms for typical hard problems: A perspective of cryptanalysis. *Quantum Inf. Process.* **2020**, *19*, 178. [CrossRef]
9. de la Cruz, J.; Willems, W. Twisted group codes. *IEEE Trans. Inf. Theory* **2021**, *67*, 5178–5184. [CrossRef]
10. Behajaina, A.; Borello, M.; de la Cruz, J.; Willems, W. Twisted skew G-codes. *Des. Codes Cryptogr.* **2024**, *92*, 1803–1821. [CrossRef]

11. Shoup, V. Sequences of Games: A Tool for Taming Complexity in Security Proofs, Cryptology ePrint Archive, Report 2004/332. 2004. Available online: http://eprint.iacr.org/2004/332 (accessed on 1 December 2023).
12. Boneh, D.; Shoup, V. A Graduate Course in Applied Cryptography, Textbook. Available online: http://toc.cryptobook.us/book.pdf (accessed on 1 June 2024).
13. Lopez-Ramos, J.A.; Rosenthal, J.; Schipani, D.; Schnyder, R. An application of group theory in confidential network communications. *Math. Meth. Apply Sci.* **2018**, *41*, 2294–2298. [CrossRef]
14. Kahrobaei, D.; Koupparis, C.; Shpilrain, V. Public key exchange using matrices over group rings. *Groups Complex Cryptol.* **2013**, *5*, 97–115. [CrossRef]
15. Eftekhari, M. Cryptanalysis of Some Protocols Using Matrices over Group Rings. In *Progress in Cryptology—AFRICACRYPT 2017*; Joye, M., Nitaj, A., Eds.; AFRICACRYPT 2017; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10239.
16. Maze, G.; Monico, C.; Rosenthal, J. Public key cryptography based on semigroup actions. *Adv. Math. Commun.* **2007**, *1*, 489–507. [CrossRef]
17. Roman'kov, V. A General Encryption Scheme Using Two-Sided Multiplications with Its Cryptanalysis. *arXiv* **2017**, arXiv:1709.06282.
18. Bader, C.; Hofheinz, D.; Jager, T.; Kiltz, E.; Li, Y. Tightly-Secure Authenticated Key Exchange. In *Theory of Cryptography*; Dodis, Y., Nielsen, J.B., Eds.; TCC 2015; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9014.
19. Jager, T.; Kiltz, E.; Riepel, D.; Schäge, S. Tightly-Secure Authenticated Key Exchange, Revisited, Cryptology ePrint Archive: Report 2020/1279. 2020. Available online: https://eprint.iacr.org/2020/1279 (accessed on 3 June 2024).
20. Canetti, R.; Krawczyk, H. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology-EUROCRYPT 2001*; Pfitzmann, B., Ed.; EUROCRYPT 2001; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2045.
21. Steiner, M.; Tsudik, G.; Waidner, M. Diffie-Hellman key distribution extended to group communication. In Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS '96), New Delhi, India, 14–15 March 1996; Association for Computing Machinery: New York, NY, USA, 1996; pp. 31–37. [CrossRef]
22. Boyd, C.; Mathuria, A.; Stebila, D. *Protocols for Authentication and Key Establishment, Second Edition, Information Security and Cryptography*; Springer: Berlin/Heidelberg, Germany, 2019.
23. Steiner, M.; Tsudik, G.; Waidner, M. Key agreement in dynamic peer groups. *IEEE Trans. Parallel Distrib. Syst.* **2000**, *11*, 769–780. [CrossRef]
24. Jao, D.; De Feo, L. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In *Post-Quantum Cryptography*; Yang, B.Y., Ed.; PQCrypto 2011; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 7071.
25. ElGamal, T. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology*; Blakley, G.R., Chaum, D., Eds.; CRYPTO 1984, Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1984; Volume 196.
26. Diffie, W.; Hellman, M.E. New Directions in Cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]
27. Hofheinz, D.; Hövelmanns, K.; Kiltz, E. *A Modular Analysis of the Fujisaki-Okamoto Transformation*; Kalai, Y., Reyzin, L., Eds.; Theory of Cryptography; TCC 2017; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10677.
28. Dworkin, M.J. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Inf. Process. Stds. (NIST FIPS). 2015. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf (accessed on 3 June 2024).
29. de la Cruz, J.; Martínez-Moro, E.; Muñoz-Martinez, S.; Villanueva-Polanco, R. Implementation of Cryptographic Constructions Based on a Twisted-Skew Group Rings. Available online: https://colab.research.google.com/drive/1QA_hktpdTDVG9cPfkj4Cq2IVeKMGy68Y?usp=sharing (accessed on 3 June 2024).
30. Fitzgibbon, G.; Ottaviani, C. Constrained Device Performance Benchmarking with the Implementation of Post-Quantum Cryptography. *Cryptography* **2024**, *8*, 21. [CrossRef]
31. Hao, F.; van Oorschot, P.C. SoK: Password-Authenticated Key Exchange – Theory, Practice, Standardization and Real-World Lessons. In Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22), Nagasaki, Japan, 30 May–3 June 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 697–711. [CrossRef]