# GCKSign: Simple and Efficient Signatures from Generalized Compact Knapsack Problems[*]

Joo Woo[†] Kwangsu Lee[‡] Jong Hwan Park[§]

December 10, 2024

**Abstract**

In 2009, Lyubashevsky proposed a lattice-based signature scheme using the Schnorr-like identification and the Fiat-Shamir heuristic and proved its security under the collision resistance of a generalized compact knapsack (GCK) function. However, their security analysis requires the witness indistinguishability property, leading to significant inefficiency and an increase of sizes of public key and signature. To overcome the efficiency issue associated with the WI property, we introduce a new lattice-based assumption, called the target-modified one-wayness problem of the GCK function and show its reduction to well-known lattice-based problems. Additionally, we present a simple and efficient GCK-based signature scheme, GCKSign, whose security is based on the Module GCK-TMO problem in the random oracle model. GCKSign is a natural extension of Lyubashevsky's scheme in a module setting, but achieves considerable efficiency gains due to eliminating the witness indistinguishability property. As a result, GCKSign achieves approximately 3.4 times shorter signature size and 2.4 times shorter public key size at the same security level.

## 1 Introduction

The generalized compact knapsack (GCK) function [18] is defined with a ring-Short Integer Solution (SIS) instance over a polynomial-based ring $R_q$ for some modulus $q$. Specifically, for a random ring-SIS instance $\boldsymbol{a} = (a_1, \ldots, a_m) \in R_q^m$, the GCK function $F_{\boldsymbol{a}} : R_q^m \to R_q$ is computed as $t = F_{\boldsymbol{a}}(\boldsymbol{x}) = \sum_{i=1}^m a_i x_i$ for a domain element $\boldsymbol{x} \in R_q^m$ with short coefficients. In 2002, Micciancio [18] showed that the function $F_{\boldsymbol{a}}$ is one-way, assuming the worst-case hardness of some shortest independent vector problems (SIVP) on cyclic lattices in a ring. Furthermore, in 2006, researchers [15, 20] proved that the GCK function $F_{\boldsymbol{a}}$ is collision-resistant, assuming the worst-case hardness of some shortest vector problems (SVP) for ideal lattices in a ring. Additionally, the GCK function has the linearity property, which states that $F_{\boldsymbol{a}}(c_1\boldsymbol{x}_1 + c_2\boldsymbol{x}_2) = c_1F_{\boldsymbol{a}}(\boldsymbol{x}_1) + c_2F_{\boldsymbol{a}}(\boldsymbol{x}_2)$ for any ring elements $c_1, c_2 \in R_q$ and any domain elements $\boldsymbol{x}_1, \boldsymbol{x}_2 \in R_q^m$.

Based on the aforementioned properties of the GCK function, Lyubashevsky [13] proposed a GCK-based signature scheme in 2009, using the Schnorr-like identification protocol and the Fiat-Shamir transform [11]. The main idea of [13] is to set up a public key as $(\boldsymbol{a}, t = F_{\boldsymbol{a}}(\boldsymbol{s}))$ and its corresponding signing key as $\boldsymbol{s}$, where $\boldsymbol{s} \in R_q^m$ is a vector of short polynomials. The signing procedure begins by sampling a short

[*]This work was submitted to 'Korean Post-Quantum Cryptography Competition' (www.kpqc.or.kr).

[†]Korea University, Seoul, Korea. Email: woojoo0121@korea.ac.kr.

[‡]Sejong University, Seoul, Korea. Email: kwangsu@sejong.ac.kr.

[§]Sangmyung University, Seoul, Korea. Email: jhpark@smu.ac.kr.

vector $\boldsymbol{y} \in R_q^m$ (from a suitable distribution) and computing $v = \boldsymbol{a}\boldsymbol{y}(= F_{\boldsymbol{a}}(\boldsymbol{y}))$. The signer then computes $c = H(v, \mu)$, where $H$ is a hash function and $\mu$ is a message, and $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{s}c$. To ensure that the distribution of $\boldsymbol{z}$ is independent of the secret $\boldsymbol{s}$, preventing any leakage of information about $\boldsymbol{s}$, the signer employs a rejection rule on $\boldsymbol{z}$. Following this *rejection sampling*, a signature $(\boldsymbol{z}, c)$ is generated only when $\boldsymbol{z}$ is sampled from the (predefined $\boldsymbol{z}$'s) distribution centered at zero, rather than at $\boldsymbol{s}c$. Subsequently, the verifier checks whether $H(\boldsymbol{a}\boldsymbol{z} - \boldsymbol{t}c, \mu)$ is equal to $c$, and $\boldsymbol{z}$ is sufficiently small.

Although the GCK-based signature proposed by Lyubashevsky has a similar structure with the previous Schnorr signature, their security proof employed in [13] relies on the concept of *witness indistinguishability* (WI). This notion is essential for proving the security of [13] under the assumption of collusion resistance of the GCK function. WI refers to the property that when choosing an alternative (or multiple) signing key $\boldsymbol{s}'$ such that $\boldsymbol{t} = F_{\boldsymbol{a}}(\boldsymbol{s}) = F_{\boldsymbol{a}}(\boldsymbol{s}')$, the signature $(\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{s}c, c)$ is (statistically) indistinguishable from $(\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{s}'c, c)$. In the security analysis, all signatures are simulated using $\boldsymbol{s}$, but due to the WI property, the adversary cannot know the exact secret key $\boldsymbol{s}$. Thus, with high probability, it is expected that the adversary succeeds in forging a signature with the other $\boldsymbol{s}'$. In such a case, the rewinding technique [4] enables a reductionist to extract a pair of distinct inputs as a collision of $F_{\boldsymbol{a}}$. Since breaking the collision resistance of the GCK function implies the ability to solve a ring-SIS problem, the security of the GCK-based signature scheme [13] eventually relies on the hardness of a ring-SIS problem.

However, the WI property results in a significant efficiency drawback in [13]. This stems from the fact that the public key $t = F_{\boldsymbol{a}}(\boldsymbol{s})$ and the corresponding secret key $\boldsymbol{s}$ are generated in such a way as to guarantee collisions in $F_{\boldsymbol{a}}$. Specifically, $\boldsymbol{s}$ needs to be parametrized so that there exists another valid secret key $\boldsymbol{s}'$ such that $F_{\boldsymbol{a}}(\boldsymbol{s}) = F_{\boldsymbol{a}}(\boldsymbol{s}')$. This requirement mandates that each coefficient of the secret key $\boldsymbol{s}$ be sampled from a relatively large range, for example, $[-2047, 2047]$, resulting in large size of the signature $(\boldsymbol{z}, c)$. To overcome the efficiency issue associated with the WI property and the large bound on $\boldsymbol{s}$, Lyubashevsky [14] presented an alternative proof technique based on the *decisional* ring-SIS problem, where $t = F_{\boldsymbol{a}}(\boldsymbol{s})$ with a small bound on $\boldsymbol{s}$ is indistinguishable from a uniformly random $t$ in $R_q$. Consequently, a (real) GCK-based signature scheme is constructed using $\boldsymbol{s}$ with small coefficients, while a simulated signature scheme in the security proof is provided to the adversary using $\boldsymbol{s}'$ with large coefficients. Although the WI property with a small bound on $\boldsymbol{s}$ enables a more efficient GCK-based signature scheme, no known quantum reduction exists from worst-case lattice problems to the decisional ring-SIS problem in ideal lattices [14].

Another approach to circumvent the WI property with large coefficients of $\boldsymbol{s}$ is to construct the public key $t$ using the Learning with Errors (LWE) problem. In this approach, $t$ is computed as $t = F_{\boldsymbol{a}}(\boldsymbol{s}) + e$, where $e$ is a polynomial sampled from a narrow distribution. Based on the decisional ring-LWE problem, which is to distinguish between a ring-LWE instance and a uniformly random one, the security proof of a ring-LWE-based signature scheme can be established using a similar argument as that of the decisional ring-SIS problem. Importantly, this approach provides improved security against quantum adversaries because there is a reduction from worst-case lattice problems to the decisional ring-LWE problems [16]. Leveraging this idea, Güneysu et al. [12] proposed a ring-LWE-based signature scheme where a signature consists of three components $(c, \boldsymbol{z}_1, z_2)$, where $\boldsymbol{z}_1 = \boldsymbol{y}_1 + c\boldsymbol{s}$ and $z_2 = y_2 + ce$. Bai and Galbraith [3] reduced the signature size of [12] by omitting $z_2$ and introducing a *compression* technique to compensate for the correctness error resulted from $e$. In 2018, Ducas et al. [10] proposed a signature scheme called Dilithium, which can be viewed as a generalization of [3] based on Module-LWE problems. Additionally, [10] presented a distinct proof technique that does not rely on the WI property used previously. They employed two signature forgeries (by rewinding an adversary) to directly solve a ring-SIS problem concerning $(\boldsymbol{A}||\boldsymbol{t}||1)$ rather than $(\boldsymbol{A}||1)$. Many other signatures based on [13] were proposed [7, 9] recently and their security proof also follow the way of Dilithium.

## 1.1 Our Contribution

From what has been shown so far, while significant progress has been made in creating efficient lattice-based signatures, the following question still arises:

*Can we prove the security of GCK-based signature without using the WI property ?*

The goal of this paper is to give a positive answer to the above problem. To achieve this, we define a relaxed version of the one-wayness problem of the GCK function, called the target-modified one-wayness (TMO) problem. In essence, the TMO problem is to solve the one-wayness problem of the GCK function *approximately* rather than exactly: given $\boldsymbol{a} = (a_1, \ldots, a_m) \in R_q^m$ and $\boldsymbol{t} = F_{\boldsymbol{a}}(\boldsymbol{s}) \in R_q$ for some $\boldsymbol{s} \in R_q^m$, find short polynomials $(\boldsymbol{x} = (x_1, \ldots, x_m), c) \in R_q^m \times R_q$ such that $F_{\boldsymbol{a}}(\boldsymbol{x}) = c\boldsymbol{t}$. To provide confidence in the TMO problem, we show that the TMO problem is reduced to both the one-wayness and the collision-resistance problems of the GCK function.

Instead of proving the original GCK-based signature scheme [13], we present a more efficient GCK-based signature scheme called *GCKSign*, which is a natural extension of [13] to the Module-GCK function. We adopt a matrix form of $\boldsymbol{A} \in R_q^{k\times\ell}$ and $\boldsymbol{s} \in R_q^{\ell\times 1}$ and define a Module-GCK function as $\boldsymbol{t} = F_{\boldsymbol{A}}(\boldsymbol{s}) = \boldsymbol{A}\boldsymbol{s} \in R_q^{k\times 1}$. Similar to the TMO problem, we establish a Module GCK-TMO problem, and show its reduction to Module-SIS, LWE problems. Notably, we prove that GCKSign is secure in the random oracle model under the Module GCK-TMO problem without relying on the WI property. By eliminating this property, we achieve significant efficiency improvements. In detail, GCKSign achieves a signature size that is about 3.4 times shorter and a public-key size that is about 2.4 times shorter at the same security level, compared to [13]. In Section 5, we provide a concrete security and efficiency comparison between [13] and GCKSign.

## 2 Preliminaries

We begin by defining the syntax and security of digital signature. We also define two computational hardness problems related to the GCK function.

### 2.1 Notation

For a modulus $q \in \mathbb{N}$, $\mathbb{Z}_q$ denotes a quotient group with respect to addition modulo $q$. Let $R$ and $R_q$ respectively be the rings $\mathbb{Z}[x]/(x^n+1)$ and $\mathbb{Z}_q[x]/(x^n+1)$, where $n$ is a power of two. Vectors with entries in $R_q$ are denoted by bold lowercase letters, for example, $\boldsymbol{a} = (a_1, \ldots, a_m) \in R_q^m$ where $a_1, \ldots, a_m \in R_q$ for some positive integer $m$. $R_{n,h}$ denotes a subset of $R_q$, consisting of polynomials with coefficients of which only a fixed number of $h$ is $-1$ and 1, and all other coefficients are zero. We notice that $|R_{n,h}| = 2^h \times \binom{n}{h}$. For a positive integer $x$, $R_{[-x,x]}$ denotes a set of $R_q$, consisting of polynomials with coefficients between $[-x, x]$. The notation $\|\cdot\|_\infty$ refers to the infinity norm.

### 2.2 Digital Signature

**Definition 2.1.** A digital signature (DS) scheme for a message space $\mathcal{M}$ consists of three algorithms: *KeyGen, Sign*, and Verify such that:

- KeyGen$(\lambda)$: The key generation algorithm takes as input a security parameter $\lambda$ and outputs a pair of keys $(pk, sk)$. These keys are called the public key and the private key.

- $\mathsf{Sign}(sk, \mu)$: The signing algorithm takes as input the private key $sk$ and a message $\mu \in \mathcal{M}$, and then outputs a signature $\sigma$.
- $\mathsf{Verify}(pk, \mu, \sigma)$: The verification algorithm takes as input the public key $pk$, a message $\mu$ and a signature $\sigma$, and then outputs 1 if the signature is valid or 0 otherwise.

**Definition 2.2** (Existential unforgeability)**.** Let $\mathsf{DS}$ = ($\mathsf{KeyGen}$, $\mathsf{Sign}$, $\mathsf{Verify}$) be a digital signature scheme. The existential unforgeability against chosen-message attacks (UF-CMA) is defined via the following experiment $\mathbf{UF\text{-}CMA}^{\mathcal{A}}_{DS}(\lambda)$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

1. $\mathcal{C}$ runs the key generation algorithm to get $(pk, sk)$ and gives $pk$ to $\mathcal{A}$.
2. $\mathcal{A}$ queries a signing oracle with a message $\mu$. Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ queried.
3. Finally, $\mathcal{A}$ outputs a signature $\sigma^*$ and a message $\mu^*$. $\mathcal{C}$ returns 1 if (1) $\mathsf{Verify}(pk, \mu^*, \sigma^*) = 1$ and (2) $\mu^* \notin \mathcal{Q}$, and otherwise returns 0 as the output of the game.

The advantage of $\mathcal{A}$ for breaking the UF-CMA security of DS is defined as $\mathsf{Adv}^{\text{UF-CMA}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathsf{UF\text{-}CMA}^{\mathcal{A}}_{\mathsf{DS}} \Rightarrow 1]$. We say that a DS scheme is UF-CMA secure if for any polynomial-time adversary $\mathcal{A}$, we have $\mathsf{Adv}^{\text{UF-CMA}}_{\mathsf{DS}}(\mathcal{A}) \leq \epsilon(\lambda)$, where $\epsilon$ is a negligible function for the security parameter $\lambda$.

### 2.3 GCK Hardness Problems

**Definition 2.3** (GCK Function [18])**.** For a ring $R_q$, a subset $S \subset R_q$, an integer $m \geq 1$, and a randomly and independently chosen $\boldsymbol{a} = (a_1, \ldots, a_m) \in R_q^m$, the GCK function $F_{\boldsymbol{a}} : S^m \to R_q$ is defined as follows:

$$F_{\boldsymbol{a}}(\boldsymbol{x}) = \sum_{i=1}^{m} a_i x_i \tag{1}$$

for $\boldsymbol{x} \in S^m \subset R_q^m$, where $\sum_{i=1}^{m} a_i x_i$ is computed using the ring multiplication and addition operations.

In this paper, we specify the subset $S$ as $S = R_{[-\beta,\beta]}$ for some integer $\beta$.

**Definition 2.4** (One-Wayness of GCK function [18])**.** A GCK function is one-way (OW) if for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, it is easy to compute, but computationally hard to invert the GCK function: given a pair $(\boldsymbol{a}, t = F_{\boldsymbol{a}}(\boldsymbol{x}))$ for a randomly chosen $\boldsymbol{a} \in R_q^m$ and $\boldsymbol{x} \in R^m_{[-\beta,\beta]}$, find $\boldsymbol{x}$ in the domain such that $F_{\boldsymbol{a}}(\boldsymbol{x}) = t$. For integers $n, m, q, \beta \in \mathbb{N}$, we define $\mathsf{Adv}^{\mathsf{OW}}_{n,m,q,\beta}$ to be the advantage of an algorithm $\mathcal{A}$ in solving the OW problem of a GCK function over the ring $R_q$.

**Definition 2.5** (Collision-Resistance of GCK function [15, 20])**.** A GCK function is collision-resistant (CR) if for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, it is computationally hard to find a collision of a GCK function: given a randomly chosen $\boldsymbol{a} \in R_q^m$, find distinct $\boldsymbol{x}, \boldsymbol{x}' \in R^m_{[-\beta,\beta]}$ such that $F_{\boldsymbol{a}}(\boldsymbol{x}) = F_{\boldsymbol{a}}(\boldsymbol{x}')$. For integers $n, m, q, \beta \in \mathbb{N}$, we define $\mathsf{Adv}^{\mathsf{CR}}_{n,m,q,\beta}$ to be the advantage of an algorithm $\mathcal{A}$ in solving the CR problem of a GCK function over the ring $R_q$.

# 3 Main Results

## 3.1 GCK-TMO Problem

Now, we define a new GCK-related problem, called TMO problem of the GCK function.

**Definition 3.1** (Target-Modified One-wayness of GCK function)**.** For integers $n, m, q, \alpha, \beta \in \mathbb{N}$, the TMO problem is defined as follows: given $\boldsymbol{a} \in R_q^m$ and $t \in R_q$, find $\boldsymbol{x} \in R_q^m$ and $c \in R_q$ such that $\|c\|_\infty \leq \alpha,\ \|\boldsymbol{x}\|_\infty \leq \beta$ satisfying

$$F_{\boldsymbol{a}}(\boldsymbol{x}) = ct. \tag{2}$$

The TMO problem is a modified OW problem of a GCK function, obtained by changing the original target $t$ into a new $ct$. The important point is that $c \in R_q$ should be short, meaning that $\|c\|_\infty \leq \alpha$ for a small integer $\alpha$, and can also be chosen by a solver as desired. Intuitively, the TMO problem becomes trivial if $c$ is chosen freely in $R_q$, because for a short $\boldsymbol{x}$ such that $\|\boldsymbol{x}\|_\infty \leq \beta$, $F_{\boldsymbol{a}}(\boldsymbol{x}) = t'$ is firstly computed and then $c$ is obtained via $c = t't^{-1}$ (if $t^{-1}$ exists). Also, the OW problem of a GCK function can be viewed as a special case of the TMO problem by issuing $(\boldsymbol{x}, c = 1)$ as a solution. For integers $n, m, q, \alpha, \beta \in \mathbb{N}$, we define $\mathsf{Adv}^{\mathrm{TMO}}_{n,m,q,\alpha,\beta}$ to be the advantage of an algorithm $\mathcal{A}$ solving the TMO problem over the ring $R_q$.

### 3.1.1 Computational Hardness of the TMO Problem

By deriving an upper bound on $\mathsf{Adv}^{\mathrm{TMO}}_{n,m,q,\alpha,\beta}$ from the following reduction, we demonstrate that the TMO problem is at least as hard as the CR and OW problems of a GCK function. For our reduction, we require a special form of $(n, q)$ that determines a ring $R_q = \mathbb{Z}_q/(x^n + 1)$, in order to guarantee that a short $c \in R_q$ has an inverse. More precisely, we require that $n$ and $\rho$ are power-of-2 integers such that $n \geq \rho$, and $q$ is a prime such that $q \equiv 2\rho + 1 \pmod{4\rho}$. We then use the result [17, Corollary 1.2.] that, for any short $c \in R_q$ such that $\|c\|_\infty < (1/\sqrt{\rho}) \times q^{1/\rho}$, $c$ has an inverse in $R_q$ with probability 1.

**Theorem 3.2.** Let $n$ and $\rho$ be power-of-2 integers such that $n \geq \rho$, and $q$ is a prime such that $q \equiv 2\rho + 1 \pmod{4\rho}$. For integers $m, \alpha, \beta, \gamma \in \mathbb{N}$ satisfying $\alpha < (1/\sqrt{\rho}) \times q^{1/\rho}$, $(2\beta + 1)^{mn} \gg q^n$ and $n\alpha\gamma \leq \beta$, it holds that $\mathsf{Adv}^{\mathrm{TMO}}_{n,m,q,\alpha,\beta} \leq \mathsf{Adv}^{\mathrm{CR}}_{n,m,q,\beta} + \mathsf{Adv}^{\mathrm{OW}}_{n,m,q,\gamma}$.

*Proof.* Suppose there is an algorithm $\mathcal{A}$ that solves the TMO problem with advantage $\mathsf{Adv}^{\mathrm{TMO}}_{n,m,q,\alpha,\beta}$ for any $\alpha$ such that $\alpha < (1/\sqrt{\rho}) \times q^{1/\rho}$. Recall that $\mathcal{A}$ takes $(\boldsymbol{a}, t)$ as input and tries to find a pair $(\boldsymbol{x}, c)$ such that $F_{\boldsymbol{a}}(\boldsymbol{x}) = ct$, satisfying the condition that $\|c\|_\infty \leq \alpha,\ \|\boldsymbol{x}\|_\infty \leq \beta$. According to [17, Corollary 1.2.], any $c \in R_q$ that satisfies $\|c\|_\infty < (1/\sqrt{\rho}) \times q^{1/\rho}$ has an inverse in $R_q$. Since $\alpha$ is less than $(1/\sqrt{\rho}) \times q^{1/\rho}$, the short polynomial $c$ always has an inverse in $R_q$. With $(\boldsymbol{x}, c)$ that $\mathcal{A}$ outputs, we set $\boldsymbol{z} = \boldsymbol{x}c^{-1}$ by considering $c^{-1}$ as a scalar. Since the GCK function $F_{\boldsymbol{a}}$ is linear, we see that $F_{\boldsymbol{a}}(\boldsymbol{z}) = F_{\boldsymbol{a}}(\boldsymbol{x}c^{-1}) = c^{-1}F_{\boldsymbol{a}}(\boldsymbol{x}) = c^{-1}ct = t$. For the integer $\gamma \in \mathbb{N}$ satisfying $\gamma \leq \beta/n\alpha$, we consider two cases as follows:

**Case 1**: $\|\boldsymbol{x}c^{-1}\|_\infty > \gamma$.
**Case 2**: $\|\boldsymbol{x}c^{-1}\|_\infty \leq \gamma$.

Obviously, $\mathcal{A}$'s output $(\boldsymbol{x}, c)$ belongs to either case 1 or case 2. In case 1, we show that an algorithm $\mathcal{B}$ uses $\mathcal{A}$ to solve the CR problem of a GCK function. The assumption that $(2\beta + 1)^{mn} \gg q^n$ guarantees that it is presumably feasible to find a pair of collision with respect to any element in $R_q$. Given $\boldsymbol{a} \in R_q^m$ as input, $\mathcal{B}$ does as follows:

1. Choose a random $\boldsymbol{z'} \in R^m_{[-\gamma,\gamma]}$.

2. Compute $t = F_{\boldsymbol{a}}(\boldsymbol{z'})$.
3. Run $\mathcal{A}$ on input $(\boldsymbol{a}, t)$ and get $(\boldsymbol{x}, c)$ from $\mathcal{A}$.
4. Compute $\boldsymbol{x'} = c\boldsymbol{z'}$ and output $(\boldsymbol{x}, \boldsymbol{x'})$ as a solution.

For $(\boldsymbol{x}, \boldsymbol{x'})$ to be a solution of the CR problem, we need to show that $F_{\boldsymbol{a}}(\boldsymbol{x}) = F_{\boldsymbol{a}}(\boldsymbol{x'})$, $\|\boldsymbol{x}\|_\infty \leq \beta$, $\|\boldsymbol{x}'\|_\infty \leq \beta$, and $\boldsymbol{x} \neq \boldsymbol{x'}$. First, since $(\boldsymbol{x}, c)$ is a solution for the TMO problem, it means that $F_{\boldsymbol{a}}(\boldsymbol{x}) = ct$. Also, we set $t = F_{\boldsymbol{a}}(\boldsymbol{z'})$ in step 2. By the linearity property of GCK function, it holds that $ct = cF_{\boldsymbol{a}}(\boldsymbol{z'}) = F_{\boldsymbol{a}}(c\boldsymbol{z'}) = F_{\boldsymbol{a}}(\boldsymbol{x'})$. Thus, $F_{\boldsymbol{a}}(\boldsymbol{x}) = ct = F_{\boldsymbol{a}}(\boldsymbol{x'})$. Secondly, it holds that $\|\boldsymbol{x}\|_\infty \leq \beta$ because $\boldsymbol{x}$ is the solution of the TMO problem, and also $\|\boldsymbol{x'}\|_\infty \leq \beta$ because (1) $\gamma \leq \beta/n\alpha$ (by assumption) and (2) $\|\boldsymbol{x'}\|_\infty \leq n \times \|c\|_\infty \times \|\boldsymbol{z'}\|_\infty \leq n\alpha\gamma \leq \beta$. Lastly, we can see that $\boldsymbol{x} \neq \boldsymbol{x'}$. Note that in case 1 where $\boldsymbol{z}$ is computed as $\boldsymbol{x}c^{-1}$, we see that $\boldsymbol{z} \neq \boldsymbol{z}'$ because of the fact that $\|\boldsymbol{z}\|_\infty > \gamma$ (in case 1) and $\|\boldsymbol{z'}\|_\infty \leq \gamma$ (in step 1). This inequality means that $\boldsymbol{x}c^{-1} \neq \boldsymbol{x}'c^{-1}$, resulting in $\boldsymbol{x} \neq \boldsymbol{x}'$.

Next, in case 2, we show that there is another algorithm $\mathcal{C}$ that uses $\mathcal{A}$ to solve the OW problem of a GCK function. Given $(\boldsymbol{a}, t)$ as input, $\mathcal{C}$ does as follows:

1. Run $\mathcal{A}$ on input $(\boldsymbol{a}, t)$ and get $(\boldsymbol{x}, c)$ from $\mathcal{A}$.
2. Output $\boldsymbol{z} = c^{-1}\boldsymbol{x}$ as a solution.

The reason why $\boldsymbol{z}$ is a solution for the OW problem is that it holds that $F_{\boldsymbol{a}}(\boldsymbol{z}) = t$ and $\|\boldsymbol{z}\| \leq \gamma$ by the condition of case 2.

As a result, the ability for $\mathcal{A}$ to solve the TMO problem is transferred to that of solving the CR or OW problem of a GCK function. We see that $\mathsf{Adv}^{\text{TMO(case 1)}}_{n,m,q,\alpha,\beta} \leq \mathsf{Adv}^{\text{CR}}_{n,m,q,\beta}$ and $\mathsf{Adv}^{\text{TMO(case 2)}}_{n,m,q,\alpha,\beta} \leq \mathsf{Adv}^{\text{OW}}_{n,m,q,\gamma}$ in case 2. This completes the proof. $\square$

### 3.1.2 Extension to Module GCK-TMO Problem

The GCK function can be extended to a module setting where $F_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ for $\boldsymbol{A} \in R_q^{k\times\ell}$ and $\boldsymbol{x} \in R^{\ell\times 1}_{[-\beta,\beta]}$. Accordingly, the previous OW, CR, and TMO problems of GCK function can each be addressed with similarly defined problems in a module setting. In particular, we define a module version of TMO problem as below.

**Definition 3.3** (Module GCK-TMO Problem)**.** For integers $n, k, \ell, q, \alpha, \beta \in \mathbb{N}$, the Module GCK-TMO problem is defined as follows: given $\boldsymbol{A} \in R_q^{k\times\ell}$ and $\boldsymbol{t} \in R_q^{k\times 1}$, find $(\boldsymbol{x}, c) \in R_q^{\ell\times 1} \times R_q$ such that $\|c\|_\infty \leq \alpha,\ \|\boldsymbol{x}\|_\infty \leq \beta$ satisfying

$$F_{\boldsymbol{A}}(\boldsymbol{x}) = c\boldsymbol{t}. \tag{3}$$

Similarly, we define $\mathsf{Adv}^{\text{M-TMO}}_{n,k,\ell,q,\alpha,\beta}$ to be the advantage of an algorithm $\mathcal{A}$ solving the TMO problem of Module-GCK function over the ring $R_q$. With two newly defined advantages $\mathsf{Adv}^{\text{M-CR}}_{n,k,\ell,q,\beta}$ and $\mathsf{Adv}^{\text{M-OW}}_{n,k,\ell,q,\gamma}$ (for some positive integer $\gamma$) regarding Module-GCK function, we can prove the following theorem.

**Theorem 3.4.** Let $n$ and $\rho$ be power-of-2 integers such that $n \geq \rho$, and $q$ is a prime such that $q \equiv 2\rho + 1 \pmod{4\rho}$. For integers $k, \ell, \alpha, \beta, \gamma \in \mathbb{N}$ satisfying $\alpha < (1/\sqrt{\rho}) \times q^{1/\rho}$, $(2\beta+1)^{n\ell} \gg q^{nk}$ and $n\alpha\gamma \leq \beta$, it holds that $\mathsf{Adv}^{\text{M-TMO}}_{n,k,\ell,q,\alpha,\beta} \leq \mathsf{Adv}^{\text{M-CR}}_{n,k,\ell,q,\beta} + \mathsf{Adv}^{\text{M-OW}}_{n,k,\ell,q,\gamma}$.

The proof can be done by the same argument as in the proof of Theorem 1 so we omit the proof in this paper.

## 3.2 Proposed GCKSign Scheme

### 3.2.1 Construction

For the security parameter $\lambda$, GCKSign generates the public parameters, *params*, as follows: choose an integer $n$ such that $n = 2^a$ for an integer $a \in \mathbb{N}$ (indeed, we set $n = 256$ for all parameter sets), and choose a prime modulus $q$ and positive integers $k, \ell, B, h, L_s$ and $\eta$. Then, *params* is given by $(n, q, k, \ell, B, h, L_s, \eta)$. Also, GCKSign requires a hash functions $H : \{0,1\}^* \to \{0,1\}^{\ell_1}$ and an encoding function $Encode : \{0,1\}^{\ell_1} \to R_{n,h}$. It is assumed that *params* and two functions ($H$, $Encode$) are used for all algorithms in GCKSign.

The key generation, signing, and verification algorithms of GCKSign are described as follows:

**KeyGen**. This algorithm first chooses random 256-bit seeds $\text{seed}_{\boldsymbol{A}}$ and $\text{seed}_{\boldsymbol{s}}$. It samples public polynomials $\boldsymbol{A}$ uniformly at random over $R_q^{k\times\ell}$ by expanding $\text{seed}_{\boldsymbol{A}}$ and secret polynomials $s_1, \dots, s_\ell$ uniformly at random over $R_{[-\eta,\eta]}$ by expanding $\text{seed}_{\boldsymbol{s}}$. Next, it computes a polynomial $\boldsymbol{t} = F_{\boldsymbol{A}}(\boldsymbol{s}) = \boldsymbol{A} \cdot \boldsymbol{s})$. Finally, it outputs a public key $pk = (\boldsymbol{t}, \text{seed}_{\boldsymbol{A}})$ and a secret key $sk = (\boldsymbol{s}, \text{seed}_{\boldsymbol{A}})$.

**Algorithm 1:** KeyGen

```
    Input  : security parameter $\lambda$
    Output: public key $pk = (\boldsymbol{t}, \text{seed}_{\boldsymbol{A}})$, and secret key $sk = (\boldsymbol{s}, \text{seed}_{\boldsymbol{A}})$
1   $\text{seed}_{\boldsymbol{A}}, \text{seed}_{\boldsymbol{s}} \leftarrow \{0,1\}^{256}$;
2   $\boldsymbol{A} \leftarrow \text{sample}_A(\text{seed}_{\boldsymbol{A}}) \in R_q^{k\times\ell}$;
3   $\boldsymbol{s} \leftarrow \text{sample}_s(\text{seed}_{\boldsymbol{s}}) \in R_{[-\eta,\eta]}^{\ell\times 1}$;
4   $\boldsymbol{t} \leftarrow F_{\boldsymbol{A}}(\boldsymbol{s}) = \boldsymbol{A} \cdot \boldsymbol{s} \mod q$;
5   $pk \leftarrow (\boldsymbol{t}, \text{seed}_{\boldsymbol{A}})$;
6   $sk \leftarrow (\boldsymbol{s}, \text{seed}_{\boldsymbol{A}})$;
7   return $(pk, sk)$;
```

**Sign**. This algorithms first regenerates the public polynomials $\boldsymbol{A}$ from $\text{seed}_{\boldsymbol{A}}$. It chooses a random 256-bit seed $\text{seed}_{\boldsymbol{y}}$. It samples polynomials $\boldsymbol{y}$ uniformly at random over $R_{[-B,B]}^{\ell\times 1}$ by using $\text{seed}_{\boldsymbol{y}}$. Next, it computes $\boldsymbol{v} = \boldsymbol{A} \cdot \boldsymbol{y} \mod q$. It obtains $\hat{c}$ by computing the hash function $H(\boldsymbol{v}, \mu)$ together with the message $\mu$. It obtains a sparse polynomial $c \in R_{n,h}$ by running $Encode(\hat{c})$ and computes $\boldsymbol{z} = \boldsymbol{y} + c \cdot \boldsymbol{s}$. If $\boldsymbol{z} \notin R_{[-B+L_s,B-L_s]}^{\ell\times 1}$, then it goes to the step that samples $y$ and repeats the subsequent steps. Finally, it outputs a signature $\sigma = (\boldsymbol{z}, \hat{c})$.

**Algorithm 2:** Sign

```
   Input  : message $\mu$, and secret key $sk = (\boldsymbol{s}, \text{seed}_{\boldsymbol{A}})$
   Output: signature $(\boldsymbol{z}, \hat{c})$
1  $\boldsymbol{A} \leftarrow \text{sample}_A(\text{seed}_{\boldsymbol{A}}) \in R_q^{k\times\ell}$;
2  $\text{seed}_{\boldsymbol{y}} \leftarrow \{0,1\}^{256}$;
3  $\boldsymbol{y} \leftarrow \text{sample}_y(\text{seed}_{\boldsymbol{y}}) \in R_{[-B,B]}^{\ell\times 1}$;
4  $\boldsymbol{v} = \boldsymbol{A} \cdot \boldsymbol{y} \mod q$;
5  $\hat{c} \leftarrow H(\boldsymbol{v}, \mu) \in \{0,1\}^{\ell_1}$;
6  $c \leftarrow Encode(\hat{c}) \in R_{n,h}$;
7  $\boldsymbol{z} \leftarrow \boldsymbol{y} + \boldsymbol{s} \cdot c$;
8  if $\boldsymbol{z} \notin R_{[-B+L_s,B-L_s]}^{\ell\times 1}$ then
9      goto step 2;
10 end
11 return $\sigma = (\boldsymbol{z}, \hat{c})$;
```

**Verify**. This algorithm first derives a polynomial c from $\hat{c}$ in the signature. It regenerates the public polynomials $\boldsymbol{A}$ from the seed $\text{seed}_{\boldsymbol{A}}$. Next, it computes $\boldsymbol{v} = \boldsymbol{A} \cdot \boldsymbol{z} - c \cdot \boldsymbol{t} \mod q$. The value $\boldsymbol{v}$ is used to compute the hash value $H(\boldsymbol{v}, \mu)$ together with the message $\mu$. It accepts the signature if the hash value matches the signature $\hat{c}$ and $\boldsymbol{z} \in R_{[-B+L_s,B-L_s]}^{\ell\times 1}$.

**Algorithm 3:** Verify

```
   Input  : message $\mu$, signature $\sigma = (\boldsymbol{z}, \hat{c})$, and public key $pk = (\boldsymbol{t}, \text{seed}_{\boldsymbol{A}})$
   Output: $\{1, 0\}$ // accept or reject signature
1  $c \leftarrow Encode(\hat{c}) \in R_{n,h}$;
2  $\boldsymbol{A} \leftarrow \text{sample}_A(\text{seed}_{\boldsymbol{A}}) \in R_q^{k\times\ell}$;
3  $v = \boldsymbol{A} \cdot \boldsymbol{z} - \boldsymbol{t} \cdot c \mod q$;
4  if $\boldsymbol{z} \notin R_{[-B+L_s,B-L_s]}^{\ell\times 1} \vee \hat{c} \neq H(\boldsymbol{v}, \mu)$ then
5      return 0;
6  end
7  return 1;
```

### 3.2.2 Correctness

The way that GCKSign works is almost the same as Schnorr signature scheme, except for (1) rejection sampling in the Sign algorithm and (2) the usage of the encoding function. For a correctly generated $\sigma = (\boldsymbol{z}, \hat{c})$, the first condition such that $\|\boldsymbol{z}\|_\infty < B - L_s$ holds, because it is the same as in the Sign algorithm, and the second condition is also guaranteed by the following equation

$$\boldsymbol{A}\boldsymbol{z} - \boldsymbol{t}c = \boldsymbol{A}(\boldsymbol{y} + \boldsymbol{s}c) - (\boldsymbol{A}\boldsymbol{s})c = \boldsymbol{A}\boldsymbol{y}. \quad (4)$$

## 3.3 Security Proof

**Theorem 3.5.** Assume that $\|c\boldsymbol{s}\|_\infty \le L_s$ and $H$ is modeled as a random oracle. GCKSign is UF-CMA secure in the random oracle model if the TMO problem of the Module-GCK function is hard. That is, for any PPT adversary $\mathcal{A}$ with $\mathsf{Adv}^{\text{UF-CMA}}_{\text{DS}}(\mathcal{A})$, making at most $q_h$ hash queries and at most $q_s$ signature queries, there exists a PPT algorithm $\mathcal{B}$ that solves the TMO problem with $\mathsf{Adv}^{\text{TMO}}_{n,k,\ell,q,2,2(B-L_s)}$, where

$$\mathsf{Adv}^{\text{UF-CMA}}_{\text{DS}}(\mathcal{A}) \le \frac{q_h q_s}{(2B+1)^{n\ell}} + \sqrt{q_h \mathsf{Adv}^{\text{TMO}}_{n,k,\ell,q,2,2(B-L_s)}} + \frac{q_h}{2^{\ell_1}}. \tag{5}$$

*Proof.* In order to prove the security of GCKSign, we define a sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2$, where $\mathbf{G}_0$ is the original UF-CMA security game defined in Chapter 2.2 and $\mathbf{G}_2$ is the final game in which the success probability of $\mathcal{A}$ can be easily bounded by the hardness of the TMO problem. For each game $\mathbf{G}_i$, we define an event $\delta_i$ where $\mathcal{A}$ successfully outputs a forgery in the game $\mathbf{G}_i$.

**Game $\mathbf{G}_0$.** In this game, a challenger $\mathcal{C}$ runs the key generation algorithm to get $(pk, sk)$ and gives $pk$ to $\mathcal{A}$. Whenever $\mathcal{A}$ asks a hash query $(\boldsymbol{v}, \mu)$, $\mathcal{C}$ gives the same answer to $\mathcal{A}$ if the query has been asked before. If not, $\mathcal{C}$ chooses a random $\hat{c} \in \{0,1\}^{\ell_1}$ and gives it to $\mathcal{A}$. Whenever $\mathcal{A}$ asks a signature query $\mu$, $\mathcal{C}$ runs the signing algorithm to get a signature $\sigma$ and gives $\sigma = (\boldsymbol{z}, \hat{c})$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $(\mu^*, \sigma^*)$ with the condition that $\mu^*$ was not queried. $\mathcal{C}$ returns $\mathsf{Verify}(pk, \mu^*, \sigma^*)$ as the output of the experiment. Thus, we get $\Pr[\delta_0] = \mathsf{Adv}^{\text{UF-CMA}}_{\text{PKS}}(\mathcal{A})$.

**Game $\mathbf{G}_1$.** $\mathbf{G}_1$ is the same as $\mathbf{G}_0$ except that the signing queries are replaced by the MidSign algorithm (see Algorithm 4 below). Since two games $\mathbf{G}_0$ and $\mathbf{G}_1$ are the same except that the random hash value $\hat{c}$ is programmed before receiving the input $(v, \mu)$. $\mathcal{A}$ can not tell if the signing oracle was answered by the Sign algorithm or the MidSign algorithm.

**Algorithm 4:** MidSign

```
   Input  : message $\mu$, public key $(\boldsymbol{A}, \boldsymbol{t})$
   Output: signature $(\boldsymbol{z}, \hat{c})$
1  Choose $\boldsymbol{y}$ uniformly at random from $R^{\ell\times 1}_{[-B,B]}$;
2  Choose $\hat{c} \in R_{n,h}$ uniformly at random;
3  Compute $c = Encode(\hat{c})$;
4  Compute $\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}$. If $\|\boldsymbol{z}\|_\infty > B - L_s$, then
5      retry at step 1;
6  Compute $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{y}$;
7  Re-program the hash oracle $H(\cdot)$ so that
8      $H(\boldsymbol{v}, \mu) = \hat{c}$;
9  Return $\sigma = (\boldsymbol{z}, \hat{c})$;
```

Let $inconsistency$ be the event that any $\hat{c}$ is previously assigned to an input value $(\boldsymbol{v}, \mu)$ queried by $\mathcal{A}$. This event occurs when $\boldsymbol{v}$ (computed by $\mathcal{C}$) becomes one of values queried by $\mathcal{A}$, assuming that $\mu$ is the same message. Since $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{y} \in R^{k\times 1}_q$ and $\boldsymbol{y}$ is chosen uniformly at random from $R^{\ell\times 1}_{[-B,B]}$, the total number of $\boldsymbol{v}$ is $(2B+1)^{n\ell}$. For $q_h$ number of hash queries, the probability that any $\hat{c}$ becomes one of queried $\boldsymbol{v}$ values is at most $q_h/(2B+1)^{n\ell}$. Also, since $\mathcal{A}$ issues at most $q_s$ number of signature queries, the probability that $inconsistency$ happens is at most $q_h q_s/(2B+1)^{n\ell}$. Obviously, unless $inconsistency$ happens, $\mathbf{G}_1$ is the same as $\mathbf{G}_0$. Thus, we see that $|\Pr[\delta_1] - \Pr[\delta_0]| \le q_h q_s/(2B+1)^{n\ell}$.

**Game $\mathbf{G}_2$.** $\mathbf{G}_2$ is the same as $\mathbf{G}_1$ except that the signature queries are replaced by the SimSign algorithm (see Algorithm 5 below). $\mathbf{G}_2$ is the same as $\mathbf{G}_1$ except the way that $\boldsymbol{z}$ and $\boldsymbol{v}$ are generated. Though $\sigma = (\boldsymbol{z}, \hat{c})$ is generated in $\mathbf{G}_2$ without using $sk = \boldsymbol{s}$, it is infeasible for $\mathcal{A}$ to differentiate between $\mathbf{G}_1$ and $\mathbf{G}_2$ because of the zero-knowledge property of the relevant identification protocol.

**Algorithm 5:** SimSign

```
    Input  : message $\mu$, public key $(\boldsymbol{A}, \boldsymbol{t})$
    Output: signature $(\boldsymbol{z}, \hat{c})$
1   Choose $\boldsymbol{z}$ uniformly at random from $R^{\ell\times 1}_{[-B+L_s, B-L_s]}$;
2   Choose $\hat{c} \in R_{n,h}$ uniformly at random;
3   Compute $c = Encode(\hat{c})$;
4   Compute $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}$;
5   Re-program the hash oracle $H(\cdot)$ so that
6       $H(\boldsymbol{v}, \mu) = \hat{c}$;
7   Return $\sigma = (\boldsymbol{z}, \hat{c})$;
```

The remaining part is to check if $\mathcal{A}$ can distinguish between the two distributions of $(\boldsymbol{z}, \hat{c}, \boldsymbol{v})$ in $\mathbf{G}_1$ and $\mathbf{G}_2$, especially in terms of $\boldsymbol{v}$. In $\mathbf{G}_1$, $\boldsymbol{v}$ is computed as $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{y}$ for some $\boldsymbol{y} \in R^{\ell\times 1}_{[-B,B]}$, whereas in $\mathbf{G}_2$ $\boldsymbol{v}$ is computed as $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}$ for some $\boldsymbol{z} \in R^{\ell\times 1}_{[-B+L_s, B-L_s]}$. what needs to be checked here is that $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}$ in $\mathbf{G}_2$ can be expressed as $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{r}$ for some (unknown) $\boldsymbol{r} \in R^{\ell\times 1}_{[-B,B]}$. If so, the two distribution of $(\boldsymbol{z}, \hat{c}, \boldsymbol{v})$ in $\mathbf{G}_1$ and $\mathbf{G}_2$ are identical from the $\mathcal{A}$'s point of view. To guarantee this, we can set $\boldsymbol{r} = \boldsymbol{z} - c\boldsymbol{s}$ for some (unknown) secret key $\boldsymbol{s} \in R^{\ell\times 1}_{[-\eta,\eta]}$ such that $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$. Then, we see that $\boldsymbol{A}\boldsymbol{r} = \boldsymbol{A}(\boldsymbol{z} - c\boldsymbol{s}) = \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} = \boldsymbol{v}$. Since $\|c\boldsymbol{s}\|_\infty \le L_s$ (by assumption) and $\|\boldsymbol{z}\|_\infty \le B - L_s$ (by choosing $\boldsymbol{z}$), we get $\|\boldsymbol{r}\|_\infty \le \|\boldsymbol{z}\|_\infty + \|c\boldsymbol{s}\|_\infty \le B - L_s + L_s = B$, which is the same distribution as $\mathbf{G}_1$. As a result, we have that $\Pr[\delta_2] = \Pr[\delta_1]$.

We now apply the generalized forking lemma of Bellare and Neven [4] to $\mathbf{G}_2$. With the probability at least $\epsilon'\big(\epsilon'/q_h - 1/2^{\ell_1}\big)$ where $\Pr[\delta_2] = \epsilon'$, we obtain two signatures $(\boldsymbol{z}, \hat{c})$ and $(\boldsymbol{z}', \hat{c}')$ for $\hat{c} \ne \hat{c}'$ such that

$$\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} = \boldsymbol{v} = \boldsymbol{A}\boldsymbol{z}' - c'\boldsymbol{t}.$$

$\mathcal{C}$ then simply sets $\boldsymbol{x} = \boldsymbol{z} - \boldsymbol{z}'$, $\tilde{c} = c - c'$, and outputs $(\boldsymbol{x}, \tilde{c})$ as the solution of the TMO problem, satisfying $\boldsymbol{A}\boldsymbol{x} = \tilde{c}\boldsymbol{t}$. We can check that $\|\boldsymbol{x}\|_\infty \le 2(B - L_s)$ and $\|\tilde{c}\|_\infty \le 2$. Thus, we have that $\epsilon'\big(\epsilon'/q_h - 1/2^{\ell_1}\big) \le \mathsf{Adv}^{\mathrm{TMO}}_{n,m,q,2,2(B-L_s)}$, which is further simplified as $\epsilon' \le \sqrt{q_h \mathsf{Adv}^{\mathrm{TMO}}_{n,m,q,2,2(B-L_s)}} + q_h/2^{\ell_1}$. This completes the proof. ☐

## 3.4 Parameter Settings

Table 1 presents several conditions necessary for choosing the parameters for GCKSign. The proof of Theorem 3.5 indicates that the TMO problem of the Module-GCK function is specified by the parameters as $\alpha = 2$ and $\beta = 2(B - L_s)$. Also, because the number of non-zero coefficients of $c$ is $h$, for any $\boldsymbol{s} \in R^{\ell\times 1}_{[-\eta,\eta]}$, it holds that $\|c\boldsymbol{s}\|_\infty \le h\eta$, allowing for setting $L_s = h\eta$. Based on these requirements, Table 2 proposes three sets of parameters for GCKSign. We set $\eta = 1$ as the bound for the secret key polynomial $\boldsymbol{s}$, and set $\rho = 8$ for all parameter sets in order to choose $q$ that meets the requirement (2) in Table 1.

Table 1: **Requirements for Parameter Selection**

| | Parameter Requirements |
|---|---|
| (1) | $n = 2^a$ for $a \in \mathbb{N}$ |
| (2) | $q \equiv 2\rho + 1 \pmod{4\rho}$ , $\rho \mid n$ and $\alpha \leq (1/\sqrt{\rho}) \times q^{1/\rho}$ |
| (3) | $n\alpha\gamma \leq \beta$ |
| (4) | $2^h \times \binom{n}{h} \geq 2^\lambda$ |
| (5) | $\|c\boldsymbol{s}\|_\infty \leq L_s$ |
| (6) | $(2\beta + 1)^{n\ell} \gg q^{nk}$ |

Table 2: **Parameter Sets for GCKSign**

| | 1 | 2 | 3 |
|---|---|---|---|
| $n$ | 256 | 256 | 256 |
| $q$ | $2^{25} - 463$ | $2^{26} - 111$ | $2^{27} - 79$ |
| $(k, \ell)$ | (2,5) | (3,8) | (7,17) |
| $h$ | 24 | 39 | 74 |
| $2^h\binom{n}{h}$ | 135 | 192 | 291 |
| $B$ | $2^{15} - 1$ | $2^{15} + 2^9 - 1$ | $2^{18} - 1$ |
| $\eta$ | 1 | 1 | 1 |
| $L_s$ | 24 | 39 | 74 |
| # of RS | 2.55 | 3.38 | 3.41 |
| LWE Hardness (Core-SVP) | | | |
| BKZ ($b$) | 245 | 476 | 1047 |
| Classical | 71 | 139 | 306 |
| Quantum | 64 | 126 | 277 |
| SIS Hardness (Core-SVP) | | | |
| BKZ ($b$) | 251 | 459 | 996 |
| Classical | 73 | 134 | 291 |
| Quantum | 66 | 121 | 264 |
| Performance (K cycle) | | | |
| KeyGen | 361 | 796 | 4,276 |
| Sign | 839 | 4,185 | 8,823 |
| Verify | 413 | 886 | 4,474 |

# of RS : number of rejection sampling. $b$ : BKZ block size.

### 3.5 Concrete Security Analysis

As shown above, the security of GCKSign relies on the TMO problem of Module-GCK function relative to $\mathsf{Adv}^{\text{TMO}}_{n,k,\ell,q,2,2(B-L_s)}$. By Theorem 3.4, the security is further reduced to the CR and OW problems of Module-GCK function relative to $\mathsf{Adv}^{\text{M-CR}}_{n,k,\ell,q,2(B-L_s)}$ and $\mathsf{Adv}^{\text{M-OW}}_{n,k,\ell,q,\gamma}$ where $\gamma$ is determined by the requirement (3) $\gamma \leq \beta/(n\alpha)$ in Table 1. Also, we use the fact the CR problem of Module-GCK function is reduced to Module-SIS problem relative to $\mathsf{Adv}^{\text{M-SIS}}_{n,k,\ell,q,2(B-L_s)}$, and (using the Hermite normal form) the OW problem of Module-GCK function is reduced to Module-LWE problem relative to $\mathsf{Adv}^{\text{M-LWE}}_{n,k,(\ell-k),q,\gamma}$. Strictly speaking, $4(B-L_s)$ is more correct than $2(B-L_s)$, but we follow the same analysis as Dilithium [10] to analyze the security of Module-SIS. Regarding the choice of $\gamma$, especially, $\gamma$ can be chosen from any positive integer satisfying $\gamma \leq 2(B-L_s)/(2^8 \times 2)$, but we set $\gamma = 1$ for the Module-LWE problem. This is because a smaller ratio of $q/\gamma$ (i.e., modulus-to-noise) generally provides stronger concrete security against known lattice attacks [19], and thus decreasing $\gamma$ is more advantageous for an adversary solving Module-LWE problem when $q$ is fixed. Eventually, the concrete security of GCKSign is estimated by the best-known lattice attacks against Module- {SIS, LWE} problems relative to $\mathsf{Adv}^{\text{M-SIS}}_{n,k,\ell,q,2(B-L_s)}$ and $\mathsf{Adv}^{\text{M-LWE}}_{n,k,(\ell-k),q,1}$, respectively. The existence of a solution with respect to $\gamma = 1$ is guaranteed by the key generation such that $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$ with $\boldsymbol{A} \in R^{k\times\ell}$ and $\boldsymbol{s} \in R^{\ell\times 1}_{[-1,1]}$.

To analyze the concrete security of the above Module- {SIS, LWE} problems, we use the BKZ lattice reduction algorithm [6] as the best-known lattice attacks. There are a variety of approaches to measure the running time of BKZ [1, 2, 6]. In general, an SVP (Shortest Vector Problem) solver is the main building block of the BKZ algorithm. Regarding the number of SVP oracle calls that the BKZ algorithm makes, the Core-SVP model [2] assumes that an SVP oracle is required only once in a conservative model. The best known classical SVP solver runs in time $\approx 2^{0.292\times b}$ and the best known quantum SVP solver runs in time $\approx 2^{0.265\times b}$. Therefore, we decide to adopt the BKZ cost model of $0.292b$ for the classical model and the BKZ cost model of $0.265b$ for the quantum model where $b$ is the BKZ block size. Table 2 shows the concrete security level of GCKSign, according to each parameter set. To estimate the hardness of the parametrized Module- {SIS, LWE} problems, we make use of the SIS and LWE estimators in [10] that reflect the above-mentioned BKZ algorithm.

## 4 Discussion

### 4.1 Comparison

Table 3 presents the comparison between the two GCK-based signatures, [13] and GCKSign. In terms of security, [13] is based on the collision-resistance of GCK function, whereas GCKSign is based on the TMO problem of Module-GCK function. Because of the collision resistance in [13], their GCK function must be set to guarantee the WI property, which requires to satisfy the condition (at least) $(2\eta+1)^{n\ell} \geq q^{nk} \times 2^{128}$ for the coefficient bound $\eta$ of a secret key polynomial. [13] sets $\eta = 2047$, which turns out to increase the sizes of public/secret keys and signatures. On the other hand, GCKSign removes the WI property by proving their security under the TMO problem of GCK function, and importantly is able to relax to the condition (at least) $(2\beta+1)^{n\ell} \geq q^{nk} \times 2^{128}$ for $\beta = 2(B-L_s)$. Because of this relaxation, GCKSign can set much smaller values of $q$ and $\eta$, and obtain a significant increase in efficiency compared to [13]. For instance, Table 3 shows that at the (almost) same 132-bit security level, the signature size of GCKSign is about 3.4 ($\approx$ 14875/4384) times shorter and the public-key size of GCKSign is about 2.4 ($\approx$ 6125/2528) times shorter, compared to [13].

Table 3: **Comparison between GCK-based signatures**

| | $n$ | $(k,\ell)$ | $q$ | $\eta$ | $sig$ (bytes) | $pk$ (bytes) | $pk+sig$ (bytes) | Classical security | Hardness problem |
|---|---|---|---|---|---|---|---|---|---|
| [13] | 512 | $(1,5)$ | $\approx 2^{60}$ | 2047 | $9,000$ | $3,875$ | $12,875$ | 71 | GCK-CR |
| | 512 | $(1,8)$ | $\approx 2^{96}$ | 2047 | $14,875$ | $6,125$ | $21,000$ | 132 | |
| | $1,024$ | $(1,8)$ | $\approx 2^{96}$ | 2047 | $30,750$ | $12,250$ | $43,000$ | 283 | |
| [10] | 256 | $(4,4)$ | $\approx 2^{23}$ | 2 | $2,420$ | $1,312$ | $3,732$ | 123 | Module-SIS, Module-LWE |
| | 256 | $(6,5)$ | $\approx 2^{23}$ | 4 | $3,293$ | $1,952$ | $5,245$ | 182 | |
| | 256 | $(8,7)$ | $\approx 2^{23}$ | 2 | $4,595$ | $2,592$ | $7,187$ | 252 | |
| Ours | 256 | $(2,5)$ | $\approx 2^{25}$ | 1 | $2,592$ | $1,632$ | $4,224$ | 71 | Module-SIS, Module GCK-TMO |
| | 256 | $(3,8)$ | $\approx 2^{26}$ | 1 | $4,384$ | $2,528$ | $6,912$ | 134 | |
| | 256 | $(7,17)$ | $\approx 2^{27}$ | 1 | $10,368$ | $6,080$ | $16,448$ | 291 | |

However, the sizes of public key and signature of GCKSign are about 1.8 ($\approx 2528/1312$) times and 1.9 ($\approx 4384/2420$) times larger, compared to [10] at the (almost) same 132-bit security level. This is caused by the key recovery attack. An adversary may try to recover the secret key $\boldsymbol{s}$ from the public key $pk = (\boldsymbol{A}, \boldsymbol{t})$, where $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$ in case of GCKSign. Because $\boldsymbol{s}$ is sampled from $R^{\ell\times 1}_{[-\eta,\eta]}$ and $\eta = 1$, this amounts to solving the OW problem of Module-GCK function, which is reduced to the Module-LWE problem relative to $\mathsf{Adv}^{\text{M-LWE}}_{n,k,(\ell-k),q,1}$, and the dimension is reduced to $\ell - k$ instead of $\ell$ itself. Therefore, We need to increase the dimension for our scheme by $k$ to maintain the security level which the LWE-based signature sets like Dilithium [10], and this leads to increase of the sizes of public key and signature at the end.

## 4.2 Performance Analysis

In Table 2, we evaluate the performance of our implementations on a 3.7GHz Intel Core i7-8700k running Ubuntu 20.04 LTS. The table shows the key generation, signing, and verification algorithms of our scheme. Our implementation codes are publicly available at https://github.com/KU-Cryptographic-Protocol-Lab/GCKS.

A polynomial multiplication is the most costly operation in implementing a cryptosystem. We choose the modulus $q$ satisfying the condition $q \equiv 2\rho + 1 \mod 4\rho$ for some $\rho | n$ to satisfy the security requirement. As a result, we can not use the "fully-splitting" NTT algorithm for the multiplication operation. Instead, we choose to follow the approach of [17], using "partially-splitting" NTT algorithm with a Karatsuba multiplication [5] and Toom-Cook polynomial multiplication method [8] to efficiently multiply in a partially-splitting ring.

## 5 Conclusion

In this paper, we have addressed the challenge of proving the security of GCK-based signature schemes without relying on the WI property. By introducing the TMO problem and showing its reduction to the one-wayness and collision-resistance problems of the GCK function, we have provided a solution to overcome the limitations imposed by the WI property in GCK-based signatures. Additionally, we have presented a more efficient GCK-based signature scheme, GCKSign, which extends the original scheme [13] by incorporating the Module-GCK function. Through the analysis of the Module GCK-TMO problem and its reduction to Module-$\{$SIS, LWE$\}$ problems, we prove the security of GCKSign in the random oracle model without relying on the WI property.

One notable result of our work is the significant reduction in signature size and public key size compared to Lyubashevsky's signature scheme [13]. GCKSign achieves approximately 3.4 times shorter signature size and 2.4 times shorter public key size at the same security level. This efficiency improvement directly stems from the elimination of the WI property.

Even though our scheme still has larger sizes than Dilithium [10], our findings not only contribute to the advancement of GCK-based signature scheme [13] but also provide insights into the design and analysis of cryptographic primitives based on structured lattice problems. The TMO problem introduces a novel perspective on approximate one-wayness, and its connections to existing hardness assumptions offer new avenues for future research. In future work, it would be interesting to explore the applicability of the TMO problem in other cryptographic protocols and settings.

**Acknowledgements**

This work was supported as part of Military Crypto Research Center (UD210027XD) funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD).

## References

[1] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[2] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.

[3] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *Topics in Cryptology - CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 28–47. Springer, 2014.

[4] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM Conference on Computer and Communications Security - CCS 2006*, pages 390–399. ACM, 2006.

[5] Marco Bodrato and Alberto Zanoni. Integer and polynomial multiplication: towards optimal toom-cook matrices. In *Symbolic and Algebraic Computation, International Symposium, ISSAC 2007*, pages 17–24. ACM, 2007.

[6] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.

[7] Jung Hee Cheon, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Marc Möller, Damien Stehlé, and MinJune Yi. Haetae: Shorter lattice-based fiat-shamir signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(3):25–75, 2024.

[8] Stephen A Cook and Aanderaa O. Stål. On the minimum computation time of functions. *Transactions of the American Mathematical Society*, 142:291–314, 1969.

[9] Julien Devevey, Alain Passelègue, and Damien Stehlé. G+ g: a fiat-shamir lattice signature based on convolved gaussians. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 37–64. Springer, 2023.

[10] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[11] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *Cryptographic Hardware and Embedded Systems - CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2012.

[13] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

[14] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, 2012.

[15] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *International Colloquium on Automata, Languages, and Programming - ICALP 2006*, pages 144–155. Springer, 2006.

[16] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 1–23. Springer, 2010.

[17] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *Advances in Cryptology - EUROCRYPT 2018*, pages 204–224. Springer, 2018.

[18] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *computational complexity*, 16(4):365–411, 2007.

[19] Chris Peikert. Lattice cryptography for the internet. In *International workshop on post-quantum cryptography*, pages 197–219. Springer, 2014.

[20] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography Conference - TCC 2006*, pages 145–166. Springer, 2006.