# Technical Note: *LeanSig* for Post-Quantum Ethereum

Justin Drake [1] Dmitry Khovratovich [1] Mikhail Kudinov*[2]
Benedikt Wagner [1]

[1] Ethereum Foundation
{justin.drake,dmitry.khovratovich,benedikt.wagner}@ethereum.org
[2] Eindhoven University of Technology
mishel.kudinov@gmail.com

**Abstract**

In this note, we present a new instantiation of the hash-based multi-signature framework introduced by Drake, Khovratovich, Kudinov, and Wagner (CiC Vol 2 Issue 1, eprint 2025/055) for Ethereum's consensus layer. Inspired by a recent work of Khovratovich, Kudinov, and Wagner (Crypto 2025, eprint 2025/889), we instantiate the framework with a novel incomparable encoding that improves the tradeoff between signature size and verification hashing. The purpose of this document is to make explicit how to use the ideas of the latter work within the framework of Drake, Khovratovich, Kudinov, and Wagner.

## Contents

**1 Introduction** **2**

**2 Preliminaries** **2**
2.1 Hypercubes . . . . . 2
2.2 Tweakable Hash Functions . . . . . 3
2.3 Incomparable Encodings . . . . . 5

**3 New Encoding and Message Hash** **6**

**4 Parameter Requirements** **8**

**5 Instantiating the Tweakable Hash Function** **11**

**6 Instances** **12**

**A Bound for the New Notion** **15**

*Mikhail Kudinov was supported by an NWO VIDI grant (Project No. VI.Vidi.193.066).

# 1 Introduction

**Hash-Based Multi-Signatures for Ethereum.** Drake, Khovratovich, Kudinov, and Wagner [DKKW25] recently proposed a family of hash-based (multi-)signature constructions designed as post-quantum replacements for BLS signatures [BLS01] in Ethereum's consensus layer. These constructions are based on variants of the XMSS signature scheme [BDH11, HBG+18], optimized to balance signature size with the cost of verification hashing[1]. The core difference between the variants lies in a component known as an *incomparable encoding*.

**Hypercubes and KKW25.** In a follow-up work, Khovratovich, Kudinov, and Wagner [KKW25] have further explored the tradeoffs among signature size, security, and verification efficiency in hash-based signature schemes using incomparable encodings mapping into hypercubes. Beyond establishing a lower bound, they have introduced new incomparable encodings that achieve improved tradeoffs between size and hashing cost. However, these encodings cannot be directly integrated into the framework of [DKKW25], as the two works use a different language: [KKW25] focuses on a textbook model of Winternitz one-time signatures, while [DKKW25] targets stateful many-time signatures, with a focus on practical efficiency using the modern language of *tweakable* hash functions [BHK+19].

**This Work.** To leverage the new insights from [KKW25] in the framework of [DKKW25], we bridge these two perspectives. Namely, we reinterpret the incomparable encodings from [KKW25] within the language and framework of [DKKW25]. We provide concrete security bounds and discuss practical parameter selection. As such, this note can be viewed as an extension of [DKKW25].

# 2 Preliminaries

We follow the notation used in [DKKW25]. For example, $s \xleftarrow{\$} S$ means that $s$ is sampled uniformly at random from $S$, where $S$ is a finite set, and $x \leftarrow \mathcal{D}$ means that $x$ is sampled from $\mathcal{D}$, where $\mathcal{D}$ is a distribution (e.g., a probabilistic algorithm). For distributions $\mathcal{D}_1, \mathcal{D}_2$ on the same support $\mathcal{X}$, their statistical distance is defined as $\frac{1}{2}\sum_{x\in\mathcal{X}} |\Pr[\mathcal{D}_1 = x] - \Pr[\mathcal{D}_2 = x]|$. For a probabilistic algorithm $\mathcal{A}$, the notation $y := \mathcal{A}(x;\rho)$ denotes that $\mathcal{A}$ outputs $y$ on input $x$ with random coins $\rho$. The notation $y \leftarrow \mathcal{A}(x)$ is used if $\rho$ is sampled uniformly at random from the algorithms randomness space. The notation $y \in \mathcal{A}(x)$ states that there are random coins $\rho$ such that $\mathcal{A}$ outputs $y$ on input $x$ with these coins $\rho$. The running time of an algorithm $\mathcal{A}$ is denoted by $\mathbf{T}(\mathcal{A})$. All algorithms and adversaries have (implicit) access to a set of public system parameters $\mathsf{par}$. The reader may consult [DKKW25] for more discussion on notation.

## 2.1 Hypercubes

Here, we define the hypercube and adjacent notation that will be relevant to us. The definitions are taken from [KKW25]. The only modification is that for this work we will consider hypercubes of the form $\{0,\dots,\mathsf{w}-1\}^v$ instead of $\{1,\dots,\mathsf{w}\}^v$. This is to be consistent with [DKKW25] and to be implementation-friendly.

**Definition 1** (Hypercube)**.** Let $\mathsf{w}$ and $v$ be integers, called the base and the dimension, respectively. The hypercube $\{0,\dots,\mathsf{w}-1\}^v$ is the set of all strings of length $v$ over the alphabet $\{0,\dots,\mathsf{w}-1\}$. It induces a directed graph that has vertex set $\{0,\dots,\mathsf{w}-1\}^v$ and an edge from $x$ to $y$ if and only if there exists an $i \in [v]$ with $y_i = x_i + 1$ and $x_j = y_j$ for all $j \neq i$.

**Definition 2** (Partial Order on the Hypercube)**.** Let $\mathsf{w}$ and $v$ be integers and consider the hypercube $\{0,\dots,\mathsf{w}-1\}^v$. Consider two vertices $x, x' \in \{0,\dots,\mathsf{w}-1\}^v$. We write $x \leq x'$ if and only if there is a directed path from $x$ to $x'$. In particular, this holds if and only if for every $i \in [v]$, we have $x_i \leq x'_i$.

As a directed graph, the hypercube has a single source vertex $(0,\dots,0)$ and a single sink vertex $(\mathsf{w}-1,\dots,\mathsf{w}-1)$. More generally, the vertices can be partitioned into layers via a breadth-first search from the source vertex.

[1]The motivation to minimize verification hashing is that these signatures should later be aggregated using a succinct argument system, and the efficiency of this aggregation process depends on the amount of hashing during signature verification.

**Definition 3** (Hypercube Layers)**.** Let $\mathsf{w}$ and $v$ be integers and consider the hypercube $\{0,\dots,\mathsf{w}-1\}^v$. The layer $\mathcal{L}_d \subseteq \{0,\dots,\mathsf{w}-1\}^v$ is the set of vertices at distance $d$ from the sink $(\mathsf{w}-1,\dots,\mathsf{w}-1)$. Precisely, that is

$$\mathcal{L}_d := \left\{ (x_1, x_2, \dots, x_v) \in \{0,\dots,\mathsf{w}-1\}^v \;\middle|\; v(\mathsf{w}-1) - \sum_{i=1}^{v} x_i = d \right\}.$$

For $x \in \mathcal{L}_d$, we write $\mathrm{layer}(x) = d$. Note that $d$ can range from 0 to $v(\mathsf{w}-1)$. Further, we use the notation $\mathcal{L}_{[d_1:d_2]} := \bigcup_{d=d_1}^{d_2} \mathcal{L}_d$ to denote the union of multiple adjacent layers.

*Remark 1.* For any two distinct vertices $x, x' \in \mathcal{L}_d$, we neither have $x \le x'$ nor $x \ge x'$.

**Definition 4** (Sizes of Hypercube Layers)**.** Let $\mathsf{w}$ and $v$ be integers and consider the hypercube $\{0,\dots,\mathsf{w}-1\}^v$. We denote the size of the $d$th layer of the hypercube as $\ell_d = |\mathcal{L}_d|$. Note that the size $\ell_d$ of the $d$th layer is given by the coefficient of $X^{v(\mathsf{w}-1)-d}$ in the polynomial $(1 + X + X^2 + \cdots + X^{\mathsf{w}-1})^v$. Further, we use the notation $\ell_{[d_1:d_2]} = \sum_{d=d_1}^{d_2} \ell_d$ to denote the size of multiple adjacent layers.

Finally, we also recall the algorithm $\mathsf{MapToVertex}_{\mathsf{w},v,d}(x)$ from [KKW25], which maps an integer to a vertex in a specific layer $d$ of the hypercube. This mapping induces a bijective mapping, which has been shown in [KKW25].

- $\mathsf{MapToVertex}_{\mathsf{w},v,d}(x) \to a$ for $0 \le x < \ell_d^{\langle v \rangle}$ and $a \in \{0,\dots,\mathsf{w}-1\}^v$
  1. Set $x_1 := x$ and $d_1 := d$.
  2. For $i$ from 1 to $v-1$:
     (a) Find $j_i \in \{\max(0, d_i - (\mathsf{w}-1)(v-i)), \dots, \min(\mathsf{w}-1, d_i)\}$ such that
     $$\sum_{\max(0,d_i-(\mathsf{w}-1)(v-i)) \le j < j_i} \ell_{d_i-j}^{\langle v-i \rangle} \le x_i < \sum_{\max(0,d_i-(\mathsf{w}-1)(v-i)) \le j \le j_i} \ell_{d_i-j}^{\langle v-i \rangle}.$$
     (b) Set $a_i := \mathsf{w} - 1 - j_i \in [\mathsf{w}]$.
     (c) Set $d_{i+1} := d_i - j_i = d_i - (\mathsf{w} - 1 - a_i)$.
     (d) Set $x_{i+1} := x_i - \sum_{\max(0,d_i-(\mathsf{w}-1)(v-i)) \le j < j_i} \ell_{d_i-j}^{\langle v-i \rangle}$.
  3. Set $a_v := \mathsf{w} - 1 - x_v - d_v$.
  4. Output $(a_1, a_2, \dots, a_v)$.

## 2.2 Tweakable Hash Functions

We work with the abstraction of tweakable hash functions introduced in [BHK+19], which has also been used in [DKKW25]. For more discussion, we refer the reader to [DKKW25].

**Definition 5** (Tweakable Hash Function)**.** Consider sets $\mathcal{H}$ (the hash space), $\mathcal{P}$ (the public parameters space), $\mathcal{T}$ (the tweak space), and $\mathcal{M}$ (the message space). A tweakable hash function is an efficiently computable function

$$\mathsf{Th}\colon \mathcal{P} \times \mathcal{T} \times \mathcal{M} \to \mathcal{H}.$$

**Definition 6** (Uniformity)**.** Let $\mathsf{Th}\colon \mathcal{P} \times \mathcal{T} \times \mathcal{M} \to \mathcal{H}$ be a tweakable hash function, where $\mathcal{M} = \mathcal{M}_0 \times \mathcal{R}$. We say that $\mathsf{Th}$ is $\epsilon$-uniform for seed space $\mathcal{R}$ if for all $P \in \mathcal{P}$, all $T \in \mathcal{T}$, and all $\mathsf{m} \in \mathcal{M}_0$, the following two distributions have statistical distance at most $\epsilon$:

$$\{x \mid x \xleftarrow{\$} \mathcal{H}\} \text{ and } \{\mathsf{Th}(P, T, (\mathsf{m}, \rho)) \mid \rho \xleftarrow{\$} \mathcal{R}\}.$$

In [DKKW25], security of the signature scheme follows from multiple standard-model properties of the used tweakable hash function (e.g., variants of target collision-resistance). In addition, the constructions rely on the security of an incomparable encoding (as defined below), which in turn was based on a notion for tweakable hashes called *multi-target collision resistance with random sampling*. We recall this notion now.

**Definition 7** (Multi-Target Collision Resistance with Random Sampling)**.** Let $K \in \mathbb{N}$ be an integer. Let $\mathsf{Th}\colon \mathcal{P} \times \mathcal{T} \times (\mathcal{M} \times \mathcal{R}) \to \mathcal{H}$ be a tweakable hash function, where the input is split into a message part $M \in \mathcal{M}$ and a randomness part $\rho \in \mathcal{R}$. Let $\mathsf{Prop}\colon \mathcal{H} \to \{0,1\}$ be a function that represents some property on the output space. Let $\mathcal{A}$ be a (stateful) algorithm, and $p \in [|\mathcal{T}|]$. Consider the following experiment $\mathbf{SM\text{-}rTCR}^{K}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A})$:

1. Generate a random public parameter $P \xleftarrow{\$} \mathcal{P}$.
2. Run $\mathcal{A}$ with an input $P$ and with (classical) access to an oracle that takes $T \in \mathcal{T}$ and $M \in \mathcal{M}$ and works as follows:
   - If $|Q| \geq p$ or there is a tuple $(T, M', \rho') \in Q$, for some $M', \rho'$, then return $\perp$.
   - Otherwise, set $\mathsf{ctr} := 0$ and $x := \perp$. While $\mathsf{ctr} < K$ and $x = \perp$:
     (a) Sample $\rho \xleftarrow{\$} \mathcal{R}$.
     (b) Set $x := \mathsf{Th}(P, T, M, \rho)$.
     (c) If $\mathsf{Prop}(x) = 1$: Insert $(T, M, \rho)$ into $Q$.
     (d) Else: Set $x := \perp$, $\rho := \perp$.
     (e) Set $\mathsf{ctr} := \mathsf{ctr} + 1$.
   - If $x = \perp$: Insert $(T, M, \perp)$ into $Q$.
   - Output $(x, \rho)$.
3. Obtain from $\mathcal{A}$ an output $(j, M^*, \rho^*)$ with $M \in \mathcal{M}$, $j \in [|Q|]$. Denote the $j$th entry in $Q$ by $(M_j, T_j, \rho_j)$.
4. Output 1 if $\mathsf{Th}(P, T_j, M_j, \rho_j) = \mathsf{Th}(P, T_j, M^*, \rho^*)$ and $(M^*, \rho^*) \neq (M_j, \rho_j)$. Otherwise, output 0.

For any such algorithm $\mathcal{A}$, we define the following advantage:

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR},K}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A}) := \Pr[\mathbf{SM\text{-}rTCR}^{K}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A}) \Rightarrow 1].$$

In this document we only present a novel encoding and leave the rest of the signature construction unchanged. To analyze the security of it based on standard model properties, we need a slight modification of the notion of multi-target collision resistance with random sampling. Intuitively, it is simply multi-target collision resistance with random sampling for a postprocessed hash function, which applies a modular reduction to the output of the hash. A more detailed reasoning why we need this new notion is given in Section 3. In the definition, we highlight the changes compared to the original notion. We give a heuristic bound for the new notion in the quantum and classical random oracle model in Appendix A.

**Definition 8** (Multi-Target Collision Resistance with Random Sampling and Modulo)**.** Let $K \in \mathbb{N}$ be an integer. Let $\eta \in \mathbb{N}$ be an integer (the modulus). Let $\mathsf{Th}\colon \mathcal{P} \times \mathcal{T} \times (\mathcal{M} \times \mathcal{R}) \to \mathcal{H}$ be a tweakable hash function, where the input is split into a message part $M \in \mathcal{M}$ and a randomness part $\rho \in \mathcal{R}$. Assume that $\mathcal{H} = \{0, \ldots, A-1\} \subseteq \mathbb{N}$. Let $\mathsf{Prop}\colon \{0, \ldots, \eta - 1\} \to \{0,1\}$ be a function. Let $\mathcal{A}$ be a (stateful) algorithm, and $p \in [|\mathcal{T}|]$. Consider the following experiment $\mathbf{SM\text{-}rTCR\text{-}mod}^{K,\eta}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A})$:

1. Generate a random public parameter $P \xleftarrow{\$} \mathcal{P}$.
2. Run $\mathcal{A}$ with an input $P$ and with (classical) access to an oracle that takes $T \in \mathcal{T}$ and $M \in \mathcal{M}$ and works as follows:
   - If $|Q| \geq p$ or there is a tuple $(T, M', \rho') \in Q$, for some $M', \rho'$, hen return $\perp$.
   - Otherwise, set $\mathsf{ctr} := 0$ and $x := \perp$. While $\mathsf{ctr} < K$ and $x = \perp$:
     (a) Sample $\rho \xleftarrow{\$} \mathcal{R}$.
     (b) Set $x := \mathsf{Th}(P, T, M, \rho)$ and $x_\eta = x \mod \eta$.
     (c) If $\mathsf{Prop}(x_\eta) = 1$ : Insert $(T, M, \rho)$ into $Q$.
     (d) Else: Set $x := \perp$, $\rho := \perp$.

(e) Set $\mathsf{ctr} := \mathsf{ctr} + 1$.

- If $x = \perp$: Insert $(T, M, \perp)$ into $Q$.
- Output $(x, \rho)$.

3. Obtain from $\mathcal{A}$ an output $(j, M^*, \rho^*)$ with $M \in \mathcal{M}$, $j \in [|Q|]$. Denote the $j$th entry in $Q$ by $(M_j, T_j, \rho_j)$.
4. Output 1 if $\mathsf{Th}(P, T_j, M_j, \rho_j) \equiv \mathsf{Th}(P, T_j, M^*, \rho^*) \pmod{\eta}$, and $(M^*, \rho^*) \neq (M_j, \rho_j)$. Otherwise, output 0.

For any such algorithm $\mathcal{A}$, we define the following advantage:

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR\text{-}mod},K,\eta}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A}) := \Pr[\mathbf{SM\text{-}rTCR\text{-}mod}^{K,\eta}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A}) \Rightarrow 1].$$

## 2.3 Incomparable Encodings

In [DKKW25], the authors introduced the abstraction of incomparable encodings to analyze multiple variants of XMSS signatures simultaneously. In our work, we will reuse this analysis but provide a novel encoding. We recall the definition from [DKKW25] almost verbatim. The main change is that we allow encodings to map into hypercubes with general bases $\mathsf{w}$ that are not necessarily powers of two.

**Definition 9** (Incomparable Encoding Scheme)**.** An incomparable encoding (IE) with public parameter space $\mathcal{P}$, randomness space $\mathcal{R}$, lifetime $L$, base $\mathsf{w}$, dimension $v$, and code $\mathcal{C} \subseteq \{0, \ldots, \mathsf{w} - 1\}^v$ is an efficiently computable function

$$\mathsf{IncEnc}\colon \mathcal{P} \times \{0,1\}^{l_{msg}} \times \mathcal{R} \times [L] \to \mathcal{C} \cup \{\perp\},$$

such that for every *distinct* codewords $x = (x_1, \ldots, x_v) \in \mathcal{C}$ and $x' = (x'_1, \ldots, x'_v) \in \mathcal{C}$, we have

$$(\exists i \in [v] : x_i < x'_i) \wedge (\exists i' \in [v] : x'_{i'} < x_{i'}).$$

*Remark 2.* Note that using the language of hypercubes from Section 2.1, the incomparability property states that for any distinct codewords $x \in \mathcal{C}$ and $x' \in \mathcal{C}$ we neither have $x \leq x'$ nor $x' \leq x$, with notation as in Definition 2.

**Definition 10** (Error of IE)**.** Let $\mathsf{IncEnc}\colon \mathcal{P} \times \{0,1\}^{l_{msg}} \times \mathcal{R} \times [L] \to \mathcal{C} \cup \{\perp\}$ be an incomparable encoding scheme. We say that $\mathsf{IncEnc}$ has error at most $\delta$, if for every $P \in \mathcal{P}$, $\mathsf{m} \in \{0,1\}^{l_{msg}}$, and every $\mathsf{ep} \in [L]$, we have

$$\Pr_{\rho \xleftarrow{\$} \mathcal{R}}[\mathsf{IncEnc}(P, \mathsf{m}, \rho, \mathsf{ep}) = \perp] \leq \delta.$$

**Definition 11** (Target Collision Resistance for IE)**.** Let $\mathsf{IncEnc}\colon \mathcal{P} \times \{0,1\}^{l_{msg}} \times \mathcal{R} \times [L] \to \mathcal{C} \cup \{\perp\}$ be an incomparable encoding scheme with code $\mathcal{C} \subseteq \{0, \ldots, \mathsf{w} - 1\}^v$. Let $K \in \mathbb{N}$ be an integer. For any algorithm $\mathcal{A}$, consider the following experiment $\mathbf{T\text{-}COLL\text{-}RES}^K_{\mathsf{IncEnc},p}(\mathcal{A})$:

1. Sample parameters $P \xleftarrow{\$} \mathcal{P}$.
2. Run $\mathcal{A}$ with input $P$ and (classical) access to an oracle that takes as input $\mathsf{m} \in \{0,1\}^{l_{msg}}$, $\mathsf{ep} \in [L]$ and is defined as follows:
   (a) If there exists an entry $(\mathsf{m}', \rho, \mathsf{ep}, x) \in \mathcal{L}$ (with the same $\mathsf{ep}$) or $|\mathcal{L}| \geq p$ , then return $\perp$.
   (b) Set $\mathsf{ctr} := 0$ and $x := \perp$. While $\mathsf{ctr} < K$ and $x = \perp$:
      i. Sample $\rho \xleftarrow{\$} \mathcal{R}$.
      ii. Set $x := \mathsf{IncEnc}(P, \mathsf{m}, \rho, \mathsf{ep})$.
      iii. Set $\mathsf{ctr} := \mathsf{ctr} + 1$.

(c) If $x = \bot$, insert $(\mathsf{m}, \bot, \mathsf{ep}, \bot)$ into $\mathcal{L}$ and return $\bot$.

(d) Else (note: $x \in \mathcal{C}$), insert $(\mathsf{m}, \rho, \mathsf{ep}, x)$ into $\mathcal{L}$ and return $(x, \rho)$.

3. Get from $\mathcal{A}$ a triple $(\mathsf{m}^*, \rho^*, \mathsf{ep}^*) \in \{0,1\}^{l_{msg}} \times \mathcal{R} \times [L]$. Then compute the encoding $x^* := \mathsf{IncEnc}(P, \mathsf{m}^*, \rho^*, \mathsf{ep}^*)$.
4. Output 1 if and only if $x^* \neq \bot$ and there is a pair $(\mathsf{m}, \rho) \neq (\mathsf{m}^*, \rho^*)$ with $(\mathsf{m}, \rho, \mathsf{ep}^*, x^*) \in \mathcal{L}$.

For any such $\mathcal{A}$, we define the following advantage:

$$\mathsf{Adv}^{\mathsf{T\text{-}COLL\text{-}RES},K}_{\mathsf{IncEnc},p}(\mathcal{A}) := \Pr\left[\mathbf{T\text{-}COLL\text{-}RES}^{K}_{\mathsf{IncEnc},p}(\mathcal{A}) \Rightarrow 1\right].$$

# 3 New Encoding and Message Hash

In this section we provide a new incomparable encoding inspired by the encodings presented in [KKW25]. Our construction can then be plugged into the framework of [DKKW25]. We can view this construction as taking Construction 6 in [DKKW25] (Target Sum Winternitz), but with a new message hash with an unusual output domain: the top $D$ layers of the hypercube, which are denoted as $\mathcal{L}_{[0:D]}$. We first specify how this new message hash is implemented.

**Construction 1** (Top Level Message Hash)**.** *Let $v, \mathsf{w}, D, T \in \mathbb{N}$ be integers such that $T \geq v(\mathsf{w}-1) - D$. Let $\mathsf{Th}^{msg}\colon \mathcal{P} \times \mathcal{T} \times (\{0,1\}^{l_{msg}} \times \mathcal{R}) \to \{0, \ldots, A-1\} \subseteq \mathbb{N}$ be a tweakable hash function. We define a new tweakable hash function $\mathsf{TL}^D[\mathsf{Th}^{msg}]\colon \mathcal{P} \times \mathcal{T} \times (\{0,1\}^{l_{msg}} \times \mathcal{R}) \to \mathcal{L}_{[0:D]}$. It is defined as follows for inputs $(P, T, (M, R))$:*

1. *Set $x := \mathsf{Th}^{msg}(P, T, (\mathsf{m}, \rho))$.*
2. *Set $y := x \bmod \ell_{[0:D]}$.*
3. *Map $y$ to a vertex $a \in \mathcal{L}_{[0:D]}$ via the following steps:*
   (a) *Find $d$ such that $\ell_{[0:d-1]} \leq y < \ell_{[0:d]}$.*
   (b) *Set $a := \mathsf{MapToVertex}_{\mathsf{w},v,d}(y - \ell_{[0:d-1]})$*
4. *Output $a$.*

Intuitively, we first hash the input using $\mathsf{Th}^{msg}$, then reduce the hash modulo the output domain size, and then apply the *bijective* mapping MapToVertex to convert the result to a vertex in the hypercube. To use this new message hash $\mathsf{TL}^D[\mathsf{Th}^{msg}]$ in our encoding, we require that it satisfies multi-target collision resistance with random sampling (Definition 7), as introduced in [DKKW25], which is a standard model assumption. As we also want to determine concrete parameter requirements to achieve a given security level, we need a (heuristic) bound on the concrete security level. To determine such a bound, one could naively use the heuristic bounds from [DKKW25]. In particular, this would mean to model $\mathsf{TL}^D[\mathsf{Th}^{msg}]$ as a (quantum or classical) random oracle. We avoid this approach for two reasons: first, note that the modulo operation introduces some statistical distance to a uniform function. Second, we view it as generally more conservative to model the inner function $\mathsf{Th}^{msg}$ as a random oracle, as this function will be implemented by a more established function in practice. The approach we take involves two steps:

1. We argue that multi-target collision resistance with random sampling of $\mathsf{TL}^D[\mathsf{Th}^{msg}]$ follows directly from our new notion multi-target collision resistance with random sampling *and modulo* (Definition 8), in the standard model.
2. To set parameters later, we use the heuristic bound for that notion (see Appendix A). Notably, this bound models the *inner* function $\mathsf{Th}^{msg}$ as a random oracle.

Let us state and prove the lemma for the first step.

**Lemma 1.** *For any algorithm $\mathcal{A}$, there is an algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ and*

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR},K}_{\mathsf{TL}^D[\mathsf{Th}^{msg}],p,\mathsf{Prop}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{SM\text{-}rTCR\text{-}mod},K,\ell_{[0:D]}}_{\mathsf{Th}^{msg},p,\mathsf{Prop}}(\mathcal{B}).$$

*Proof.* This follows by definition and the fact that MapToVertex is a bijection, as shown in [KKW25]. □

We will now present the new encoding. Intuitively, it is obtained by using $\mathsf{TL}^D[\mathsf{Th}^{msg}]$ as a message hash in Construction 6 in [DKKW25].

**Construction 2** (IE for Top Layer Target Sum Winternitz)**.** *Let $v, \mathsf{w}, D, T \in \mathbb{N}$ be integers such that $T \geq v(\mathsf{w}-1) - D$. Let $\mathsf{Th}^{msg} \colon \mathcal{P} \times \mathcal{T} \times (\{0,1\}^{l_{msg}} \times \mathcal{R}) \to \{0,\dots,A-1\} \subseteq \mathbb{N}$ be a tweakable hash function. Define the code*

$$\mathcal{C} := \left\{ (x_1,\dots,x_v) \in \{0,\dots,\mathsf{w}-1\}^v \;\middle|\; \sum_{i=1}^{v} x_i = T \right\} \subseteq \{0,\dots,\mathsf{w}-1\}^v.$$

*Note that $\mathcal{C} = \mathcal{L}_{v(\mathsf{w}-1)-T}$ in the notation from Section 2.1. We define the encoding function*

$$\mathsf{IncEnc}_{\mathsf{TLTSW}}[\mathsf{Th}^{msg}, D, T] \colon \mathcal{P} \times \{0,1\}^{l_{msg}} \times \mathcal{R} \times [L] \to \mathcal{C} \cup \{\perp\}.$$

*It is given by the following instructions on input $P \in \mathcal{P}, \mathsf{m} \in \{0,1\}^{l_{msg}}, \rho \in \mathcal{R}, \mathsf{ep} \in [L]$:*

1. *$a := \mathsf{TL}^D[\mathsf{Th}^{msg}](P, \mathsf{tweakm}(\mathsf{ep}), (\mathsf{m}, \rho))$, where $\mathsf{TL}^D[\mathsf{Th}^{msg}]$ is as in Construction 1.*
2. *If $a \notin \mathcal{C}$, return $\perp$. Else, return $a$.*

*Here, we assume $\mathsf{tweakm} \colon [L] \to \mathcal{T}$ is a fixed publicly known injective mapping.*

**Lemma 2** (Correctness and Error of Top Layer Target Sum Winternitz)**.** *Consider the encoding function $\mathsf{IncEnc}_{\mathsf{TLTSW}}[\mathsf{Th}^{msg}, D, T]$ as defined in Construction 2, and assume that $\mathsf{TL}^D[\mathsf{Th}^{msg}]$ is $\epsilon$-uniform for seed space $\mathcal{R}$ (see Definition 6). Then, $\mathsf{IncEnc}_{\mathsf{TLTSW}}[\mathsf{Th}^{msg}, T]$ is an incomparable encoding scheme and has error*

$$\delta = \epsilon + \left(1 - \frac{\ell_{v(\mathsf{w}-1)-T}}{\ell_{[0:D]}}\right).$$

*Proof.* The proof of this lemma is almost identical to the proof of Lemma 7 in [DKKW25]. □

**Lemma 3** (Target Collision-Resistance of Top Layer Target Sum Winternitz)**.** *Consider the function $\mathsf{IncEnc}_{\mathsf{TLTSW}}[\mathsf{Th}^{msg}, D, T]$ as defined in Construction 2, and any $K, p \in \mathbb{N}$. Let $\mathsf{Prop} \colon \{0,1\}^* \to \{0,1\}$ be the predicate that outputs $1$ if and only if its input is in $\mathcal{C}$. Then, for every algorithm $\mathcal{A}$, there is an algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ and*

$$\mathsf{Adv}^{\mathsf{T\text{-}COLL\text{-}RES},K}_{\mathsf{IncEnc}_{\mathsf{W}}[\mathsf{Th}^{msg}],p}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{SM\text{-}rTCR},K}_{\mathsf{TL}^D[\mathsf{Th}^{msg}],p,\mathsf{Prop}}(\mathcal{B}).$$

*Proof.* The reduction works exactly as in the proof of Lemma 8 in [DKKW25]. □

Combining Lemmata 1 to 3 with the results in [DKKW25] (concretely, Theorem 1), we obtain the following corollary. It states that – in the standard model – security of the resulting signature scheme relies on SM-TCR, SM-UD, and SM-PRE of the tweakable hash function Th (as defined in [DKKW25]), and SM-rTCR-mod of $\mathsf{Th}^{msg}$.

**Corollary 1** (Top Layer Target Sum Winternitz Instantiation)**.** *Let $\mathsf{Th}^{msg} \colon \mathcal{P} \times \mathcal{T} \times (\{0,1\}^{l_{msg}} \times \mathcal{R}) \to \{0,\dots,A-1\} \subseteq \mathbb{N}$ be a tweakable hash function and assume that $\mathsf{TL}^D[\mathsf{Th}^{msg}]$ is $\epsilon$-uniform for seed space $\mathcal{R}$ (see Definition 6). Let $v, \mathsf{w}, D, T \in \mathbb{N}$ be integers such that $T \geq v(\mathsf{w}-1) - D$. Let $\mathsf{Th} \colon \mathcal{P} \times \mathcal{T} \times \mathcal{M} \to \mathcal{H}$ be a tweakable hash function, such that $\mathcal{H} \subseteq \mathcal{M}$, $\mathcal{H}^2 \subseteq \mathcal{M}$, and $\mathcal{H}^v \subseteq \mathcal{M}$. Fix integers $T \in \mathbb{N}$ and $K \in \mathbb{N}$. Let $\mathsf{Prop} \colon \{0,1\}^* \to$*

$\{0,1\}$ *be the predicate as in Lemma 3. Consider the scheme* $\mathsf{SIG} := \mathsf{SIG}[\mathsf{IncEnc}_{\mathsf{TLTSW}}[\mathsf{Th}^{msg}, T], \mathsf{Th}, K]$ *obtained from combining Construction 3 from [DKKW25] and Construction 2.*

*Then, this scheme has correctness error at most*

$$\left(\epsilon + \left(1 - \frac{\ell_{v(\mathsf{w}-1)-T}}{\ell_{[0:D]}}\right)\right)^K.$$

*Furthermore, for every algorithm* $\mathcal{A}$*, there are algorithms* $\mathcal{B}_i$ *with* $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B}_i)$ *for all* $i$ *and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SY\text{-}UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{SM\text{-}TCR}}_{\mathsf{Th},2\cdot L\cdot v\cdot \mathsf{w}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{SM\text{-}rTCR\text{-}mod},K,\ell_{[0:D]}}_{\mathsf{Th}^{msg},p,\mathsf{Prop}}(\mathcal{B}_2) + 2\cdot \mathsf{Adv}^{\mathsf{SM\text{-}TCR}}_{\mathsf{Th},L\cdot v\cdot \mathsf{w}}(\mathcal{B}_3)\\ + L\cdot v\cdot \mathsf{w}\left(\mathsf{w}\cdot \mathsf{Adv}^{\mathsf{SM\text{-}UD}}_{\mathsf{Th},\mathcal{H},1}(\mathcal{B}_5) + \mathsf{Adv}^{\mathsf{SM\text{-}PRE}}_{\mathsf{Th},\mathcal{H},1}(\mathcal{B}_6)\right),\end{aligned}$$

*where* $q_s$ *is the number of signing queries that* $\mathcal{A}$ *makes.*

# 4 Parameter Requirements

In this section, we discuss how to set parameters of the new scheme. Following the approach to set parameters from Section 6 of [DKKW25], we assume a target security level $k_C$ of classical and $k_Q$ of quantum security. Our overall goal is then to set parameters such that for any adversary $\mathcal{A}$, we have

$$\mathsf{Adv}^{\mathsf{SY\text{-}UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})/\mathbf{T}(\mathcal{A}) \le 2^{-k},$$

where $k = k_C$ or $k = k_Q$ depending on whether $\mathcal{A}$ is classical or quantum. We use our security bound from Corollary 1 to do that. Namely, we want to ensure that each of the five additive terms in that bound (including their coefficients, and when divided by $\mathbf{T}(\mathcal{A})$) is at most $2^{-(k+\log 5)}$. Now, we need to find parameters such that the hash function satisfies these requirements. To do that, we rely on the heuristic bounds obtained from modeling the hash function as a random oracle.

In the following, we give the details of this approach. Notably, we have observed that some of the bounds in [DKKW25] have been overly conservative, and we can improve them by making the *global* assumption that the adversary runs in time $\mathbf{T}(\mathcal{A}) < 2^k$. Indeed, if $\mathbf{T}(\mathcal{A}) \ge 2^k$, then we are done trivially. In contrast, [DKKW25] made a similar assumption but *locally* for each of the individual notions, which for some parameters resulted in larger lower bounds.

**Security Levels for Hash Function Properties.** Our goal is that for any adversary $\mathcal{A}$ running in time $\mathbf{T}(\mathcal{A})$, the fraction $\mathsf{Adv}^{\mathsf{SY\text{-}UF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A})/\mathbf{T}(\mathcal{A})$ is at most $2^{-k}$, where $k = k_C$ or $k = k_Q$ depending on whether $\mathcal{A}$ is quantum. As mentioned already, we can restrict ourselves to adversaries that run in time at most $2^k$. Looking at Corollary 1, we see that the advantage is the sum of five terms. Consequently, we want that each of these terms, divided by the running time, is at most $2^{-k-\log 5}$. This means we need to ensure the following hardness bounds, for any algorithm $\mathcal{A}$ (running in time at most $2^k$):

$$\mathsf{Adv}^{\mathsf{SM\text{-}TCR}}_{\mathsf{Th},2\cdot L\cdot v\cdot \mathsf{w}}(\mathcal{A})/\mathbf{T}(\mathcal{A}) \le 2^{-(k+\log 5)}. \tag{1}$$

$$\mathsf{Adv}^{\mathsf{SM\text{-}TCR}}_{\mathsf{Th},L\cdot v\cdot \mathsf{w}}(\mathcal{A})/\mathbf{T}(\mathcal{A}) \le 2^{-(k+\log 5+1)}. \tag{2}$$

$$\mathsf{Adv}^{\mathsf{SM\text{-}UD}}_{\mathsf{Th},\mathcal{H},1}(\mathcal{A})/\mathbf{T}(\mathcal{A}) \le 2^{-(k+\log 5+2\log \mathsf{w}+\log L+\log v)}. \tag{3}$$

$$\mathsf{Adv}^{\mathsf{SM\text{-}PRE}}_{\mathsf{Th},\mathcal{H},1}(\mathcal{A})/\mathbf{T}(\mathcal{A}) \le 2^{-(k+\log 5+\log L+\log \mathsf{w}+\log v)}. \tag{4}$$

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR\text{-}mod},K,\ell_{[0:D]}}_{\mathsf{Th}^{msg},p,\mathsf{Prop}}(\mathcal{A})/\mathbf{T}(\mathcal{A}) \le 2^{-(k+\log 5)}. \tag{5}$$

Under the standard model assumption that $\mathsf{Th}$ and $\mathsf{Th}^{msg}$ satisfy these bounds, our scheme has $k$ bits of security, as shown by Corollary 1. If we want to determine concrete parameters and instantiations for hash functions, we need to use heuristics, namely, the heuristics from [DKKW25], Table 1, and our new bound from Lemma 5. This is what we will do in the following.

**Message Hash and Randomness – Top Layer Target Sum Winternitz.** We start with the parameters specific to our new Top Layer Target Sum Winternitz instantiation (Construction 2). Specifically, we assume that $v, \mathsf{w}$, $K$, and $q_s$ are given[2] and we want to determine requirements on the randomness

[2]One can always upper bound $q_s$ with $q_s \le L$.

domain $|\mathcal{R}|$, the size of the hash function output domain before and after the modular reduction, i.e., $A$ and $|\mathcal{L}_{[0:D]}| = \ell_{[0:D]}$ (see Construction 1). What we need to satisfy is Equation (5). We use our new bound in Lemma 5 for that and note that $p = q_s$ and therefore $q' = q + q_s K$. We also note that $\ell_{[0:D]}$ takes the role of $\eta$. We will use that the running time of the adversary must be at least $q' + 1$, and that it is at most $2^k$. We start with the classical setting, and plug in the bound from Lemma 5: we want that

$$\left(2q \cdot \frac{\ell_{[0:D]}}{A} + \frac{q'+1}{\ell_{[0:D]}} + \frac{q' q_s K}{|\mathcal{R}|}\right) \Big/ \mathbf{T}(\mathcal{A}) \leq 2^{-(k_C + \log 5)}.$$

Lower bounding the running time by $q$, $q' + 1$, and $q'$, respectively, this becomes

$$2 \cdot \frac{\ell_{[0:D]}}{A} + \frac{1}{\ell_{[0:D]}} + \frac{q_s K}{|\mathcal{R}|} \leq 2^{-(k_C + \log 5)}.$$

It is sufficient to require that each of these bounds is at most $2^{-(k_C + \log 5 + \log 3)}$. This yields the requirements for the classical setting.

Now, we turn to the quantum setting. Again, we use Lemma 5. The desired inequality now becomes

$$\left(8q^2 \cdot \sqrt{\frac{\ell_{[0:D]}}{A}} + \frac{8(q'+1)^2}{\ell_{[0:D]}} + \frac{3 \cdot q_s K}{2} \cdot \sqrt{\frac{q'}{|\mathcal{R}|}}\right) \Big/ \mathbf{T}(\mathcal{A}) \leq 2^{-(k_Q + \log 5)}.$$

We focus on each of the three additive terms individually, and want that each of them is at most $2^{-(k_Q + \log 5 + \log 3)}$. For the first term, lower bound $\mathbf{T}(\mathcal{A})$ by $q$, and get the requirement

$$8q \cdot \sqrt{\frac{\ell_{[0:D]}}{A}} \leq 2^{-(k_Q + \log 5 + \log 3)}.$$

Now, we upper bound $q$ by $2^{k_Q}$ using our global assumption that $\mathcal{A}$'s running time is at most $2^{k_Q}$. Then, the requirement becomes

$$\sqrt{\frac{\ell_{[0:D]}}{A}} \leq 2^{-(2k_Q + \log 5 + \log 3 + 3)},$$

or

$$\log A \geq 4k_Q + 2\log 5 + 2\log 3 + 6 + \log \ell_{[0:D]}.$$

We continue with the second term. Here, we proceed in a similar way, but use $q' + 1$ as a lower bound for the running time. We get the requirement

$$\log \ell_{[0:D]} \geq 2k_Q + \log 5 + \log 3 + 3.$$

Continue with the third additive term. We divide by the running time, lower bounded by $q'$, and the left hand-side becomes (using $q_s \leq q'$)

$$\frac{3}{2} q_s K \cdot \sqrt{\frac{q'}{|\mathcal{R}|}} \cdot q'^{-1} = \frac{3}{2} K \cdot \sqrt{\frac{q_s^2 q'}{|\mathcal{R}| q'^2}} = \frac{3}{2} K \cdot \sqrt{\frac{q_s^2}{|\mathcal{R}| q'}} \leq \frac{3}{2} K \cdot \sqrt{\frac{q_s}{|\mathcal{R}|}},$$

We want that this is at most $2^{-(k_Q + \log 5 + \log 3)}$. For that, it is sufficient to set

$$\log |\mathcal{R}| \geq 2(k_Q + \log 5 + 2\log 3 + \log K - 1) + \log q_s.$$

With that, we get the following list of requirements.

**Parameter Requirement 1** (Parameters for Top Layer Target Sum Winternitz)**.** *Let* $\mathsf{w}$, $v$, $q_s$, $K$ *be given, and assume we use Construction 2. Then, if we want (approximately)* $k_C$ *bits of classical security and* $k_Q$ *bits of quantum security, we need to satisfy the following:*

$$\log \ell_{[0:D]} \geq \max\{k_C + \log 5 + \log 3, 2k_Q + \log 5 + \log 3 + 3\}, \tag{6}$$

$$\log |\mathcal{R}| \geq \max\{k_C + \log 5 + \log q_s + \log K + \log 3, 2(k_Q + \log 5 + 2\log 3 + \log K - 1) + \log q_s\}, \tag{7}$$

$$\log A \geq \max\{k_C + \log \ell_{[0:D]} + \log 5 + \log 3 + 1, 4k_Q + \log \ell_{[0:D]} + 2\log 5 + 2\log 3 + 6\}. \tag{8}$$

**Hash and Parameter Length.** Assume now that $v, \mathsf{w}, D$ and the lifetime $L$ are given. We now want to determine lower bounds on the parameters $|\mathcal{H}|$ or $|\mathcal{P}|$ related to $\mathsf{Th}$, by using Equations (1) to (4) and the bounds in [DKKW25, Table 1]. Assuming $\mathsf{w} \geq 2$ and $L, v \geq 2$, we can use the bounds derived in [DKKW25] for the classical setting:

$$\log|\mathcal{H}| \geq k_C + \log 5 + 2\log \mathsf{w} + \log L + \log v, \;\; \log|\mathcal{P}| \geq \max k_C + \log 5 + 3.$$

The reader can verify that this satisfies Equations (1) to (4) using the classical bounds in [DKKW25, Table 1]. For that, the reader can lower bound the running time with $q+1$ and upper bound $2q+1$ with $2\mathbf{T}(\mathcal{A})$, where $q$ denotes the number of random oracle queries.

For the quantum setting, we revisit the analysis to obtain slightly improved bounds for $|\mathcal{P}|$. To this end, we first plug the quantum bounds in [DKKW25, Table 1] into the desired inequalities Equations (1) to (4):

$$\begin{aligned}
\left(\frac{32(q+1)^2}{|\mathcal{H}|} + \frac{32q^2}{|\mathcal{P}|}\right) \Big/ \mathbf{T}(\mathcal{A}) &\leq 2^{-(k_Q + \log 5)}. \\
\left(\frac{32(q+1)^2}{|\mathcal{H}|} + \frac{32q^2}{|\mathcal{P}|}\right) \Big/ \mathbf{T}(\mathcal{A}) &\leq 2^{-(k_Q + \log 5 + 1)}. \\
\frac{12q}{\sqrt{|\mathcal{H}|}} \Big/ \mathbf{T}(\mathcal{A}) &\leq 2^{-(k_Q + \log 5 + 2\log \mathsf{w} + \log L + \log v)}. \\
\left(\frac{8(q+1)^2}{|\mathcal{H}|} + \frac{12(q+1)}{\sqrt{|\mathcal{H}|}}\right) \Big/ \mathbf{T}(\mathcal{A}) &\leq 2^{-(k_Q + \log 5 + \log L + \log \mathsf{w} + \log v)}.
\end{aligned}$$

Clearly, the first requirement is implied by the second one and hence can be ignored from now on. Next, we lower bound $\mathbf{T}(\mathcal{A})$ with $q$ (resp. $q+1$). With this, we get that satisfying the following bounds is sufficient:

$$\begin{aligned}
\frac{32(q+1)}{|\mathcal{H}|} + \frac{32q}{|\mathcal{P}|} &\leq 2^{-(k_Q + \log 5 + 1)}. \\
\frac{12}{\sqrt{|\mathcal{H}|}} &\leq 2^{-(k_Q + \log 5 + 2\log \mathsf{w} + \log L + \log v)}. \\
\frac{8(q+1)}{|\mathcal{H}|} + \frac{12}{\sqrt{|\mathcal{H}|}} &\leq 2^{-(k_Q + \log 5 + \log L + \log \mathsf{w} + \log v)}.
\end{aligned}$$

In the next step, we use the assumption that $\mathcal{A}$ runs in time at most $2^{k_Q}$, which also means that we can upper bound $q$ and $q+1$ with $2^{k_Q}$. This again simplifies our bounds and eliminates $q$. Further, we split up the first and third bound by requiring that the two respective terms contribute at most half to the respective sum. With these two steps, we get that satisfying the following bounds is sufficient:

$$\begin{aligned}
\frac{32 \cdot 2^{k_Q}}{|\mathcal{H}|} &\leq 2^{-(k_Q + \log 5 + 2)}. \\
\frac{32 \cdot 2^{k_Q}}{|\mathcal{P}|} &\leq 2^{-(k_Q + \log 5 + 2)}. \\
\frac{12}{\sqrt{|\mathcal{H}|}} &\leq 2^{-(k_Q + \log 5 + 2\log \mathsf{w} + \log L + \log v)}. \\
\frac{8 \cdot 2^{k_Q}}{|\mathcal{H}|} &\leq 2^{-(k_Q + \log 5 + \log L + \log \mathsf{w} + \log v + 1)}. \\
\frac{12}{\sqrt{|\mathcal{H}|}} &\leq 2^{-(k_Q + \log 5 + \log L + \log \mathsf{w} + \log v + 1)}.
\end{aligned}$$

Again, we eliminate redundant bounds: the fifth one is implied by the third one, and the first one is implied by the fourth one, as $\log \mathsf{w} \geq 1$ and $L, v \geq 2$. Rearranging, we get

$$\begin{aligned}
\log|\mathcal{P}| &\geq 2k_Q + \log 5 + 7. \\
\log\sqrt{|\mathcal{H}|} &\geq k_Q + \log 5 + 2\log \mathsf{w} + \log L + \log v + \log 12. \\
\log|\mathcal{H}| &\geq 2k_Q + \log 5 + \log L + \log \mathsf{w} + \log v + 4.
\end{aligned}$$

Clearly, the third bound is also implied by the second one. This gives us our final parameter requirements.

**Parameter Requirement 2** (Hash and Parameter Length)**.** *Let $L$, $\mathsf{w}$, $v$ be given, and assume that $\mathsf{w} \geq 2$, $L, v \geq 2$. Then, if we want (approximately) $k_C$ bits of classical security and $k_Q$ bits of quantum security, we need to satisfy the following:*

$$\log |\mathcal{H}| \geq \max\{k_C + \log 5 + 2\log \mathsf{w} + \log L + \log v, 2(k_Q + \log 5 + 2\log \mathsf{w} + \log L + \log v + \log 12)\}, \quad (9)$$
$$\log |\mathcal{P}| \geq \max\{k_C + \log 5 + 3, 2k_Q + \log 5 + 7\}. \quad (10)$$

# 5 Instantiating the Tweakable Hash Function

As we tailor our construction to the specific use case of signature aggregation via succinct arguments, we instantiate our tweakable hash function with a construction friendly to such an argument system. At the time of writing, the most efficient plausibly post-quantum secure proof systems are defined over medium-size prime fields. We suggest using the Poseidon2 hash function [GKS23] defined over a 31-bit prime, e.g., the KoalaBear prime $p = 2^{31} - 2^{24} + 1$ or the BabyBear prime $p = 2^{31} - 2^{27} + 1$. For more details on our instantiation, we refer to [DKKW25, Section 7.3].

**Poseidon2 Variants and Their security.** The Poseidon2 hash over 31-bit prime fields is constructed via a Poseidon2 permutation $P$, which operates over either 16- or 24-element state, i.e., it maps $\mathbb{F}^{16} \to \mathbb{F}^{16}$ or $\mathbb{F}^{24} \to \mathbb{F}^{24}$. Given such a $t$-element permutation $P$ and an integer $u < t$, a $t$-to-$u$ element hash function is constructed as

$$x \longmapsto \mathrm{Truncate}_u(P(x) + x),$$

where $+$ is a componentwise addition, and $\mathrm{Truncate}_u$ denotes the function that truncates its input to $u$ elements. This scheme is considered to securely behave like a random oracle for all practical purposes with $128$ bits of classical and $64$ bits of quantum security as long as $\max(u, t-u) \cdot \log p \geq 128$ [GKS23, Unr21]. Thus, as long as we satisfy this constraint, we can use the parameter bounds from the previous section to set parameters. We denote this function as $\mathsf{PoseidonCompress}_{t,u}$ and use it for all hash invocations in the signature scheme, except for the hashing of chain ends, for which we employ the sponge mode, see below.

In the sponge mode the permutation $P$ is iterated on the $t$-bit state, and after each iteration the state outputs (as the hash output) or absorbs (as the hash input) up to $r$ elements. The initial state takes up to $r$ elements as well, with the rest $t - r$ elements can be initialized to a domain separator [KBM23]. This scheme is also considered to securely instantiate a random oracle with 128 bits of classical and 64 bits of quantum security as long as $(t - r) \cdot \log p \geq 256$ [Unr21]. Due to computational overhead, this scheme is used only to hash inputs that do not fit into the 24-element state of PoseidonCompress. Further details on the sponge mode can be found in [DKKW25, Section 7.3].

**Usage of Poseidon2 in the Signature Scheme.** Here we outline which Poseidon2 variant should be used in each part of the signature scheme:

- **Message hashing.** Here, we hash $(P, T, M, R) \in \mathcal{P} \times \mathcal{T} \times \mathcal{M} \times \mathcal{R}$ and produce an output from $\{0, \ldots, A-1\}$. The total length of $P, T, R$ is $14$ field elements in all instances from Section 6, the message $M$ is supposed to be 256 bit long (i.e., $9$ field elements), and $A$ at most $15$ elements long. Thus, we use one call to $\mathsf{PoseidonCompress}_{24,15}$ for message hashing.
- **Chain hashing.** Here, we hash $(P, T, H)$ where $H \in \mathcal{H}$ and produce an output from $\mathcal{H}$. The combined inputs occupy $15$ elements (Section 6), so we use $\mathsf{PoseidonCompress}_{16,8}$ for chain hashing.
- **Tree hashing.** Here, we hash $(P, T, H_1, H_2)$ where $H_1, H_2 \in \mathcal{H}$ and produce an output from $\mathcal{H}$. The combined inputs occupy $23$ elements (Section 6), so we use $\mathsf{PoseidonCompress}_{24,8}$ for chain hashing.
- **Leaf hashing.** Here, we hash $(H_1, H_2, \ldots, H_v)$ where $H_i \in \mathcal{H}$ and produce an output from $\mathcal{H}$. We use the Poseidon2 permutation with width-24 in the sponge mode.

# 6 Instances

We provide Python scripts to select parameters for our instantiation, following the requirements from Section 4, in

`https://github.com/b-wagn/hypercube-hashsig-parameters`.

Further, we have integrated a prototype implementation of our new encoding into the implementation from [DKKW25], which can be found at

`https://github.com/b-wagn/hash-sig`.

We present three example parameter settings for security levels $k_C = 128$ and $k_Q = 64$, as in [DKKW25]. We use a 31-bit field $\mathbb{F}$, i.e., $p = |\mathbb{F}| \approx 2^{31}$. All parameters are computed using the scripts mentioned above, which respect the bounds from Section 4. The target sum $T$ is chosen maximally (i.e., minimal verification hashing) subject to the following constraint: the expected number of trials $\ell_{[0:D]}/\ell_{v(\mathsf{w}-1)-T}$ for the signer to find a valid randomness $\rho$ needs to be at most 30. We pick the upper bound for the number of trials before giving up as $K = 2^{12}$ by default. Here are the parameters, all for lifetime $L = 2^{32}$:

- **Hashing-Optimized Instance.** Hypercube with $v = 64$, $\mathsf{w} = 8$:
  - Randomness for message hash: $\mathcal{R} = \mathbb{F}^7$;
  - Message hash output before modulo: $A \approx 2^{31 \cdot 15}$ (i.e., 15 field elements);
  - Final layer of hypercube part into which we hash: $D = 77$;
  - Target sum: $T = 375$;
  - Hash function parameter: $\mathcal{P} = \mathbb{F}^5$;
  - Field elements to encode a tweak: 2
  - Hash function output: $\mathcal{H} = \mathbb{F}^8$;
  - Capacity: $c = 9$;
  - Signature size: 3.03 KiB;
  - Verification hashing: $69\times$ Poseidon2 with width 24, $73\times$ Poseidon2 with width 16;
- **Tradeoff Instance.** Hypercube with $v = 48$, $\mathsf{w} = 10$:
  - Randomness for message hash: $\mathcal{R} = \mathbb{F}^7$;
  - Message hash output before modulo: $A \approx 2^{31 \cdot 15}$ (i.e., 15 field elements);
  - Final layer of hypercube part into which we hash: $D = 112$;
  - Target sum: $T = 326$;
  - Hash function parameter: $\mathcal{P} = \mathbb{F}^5$;
  - Field elements to encode a tweak: 2
  - Hash function output: $\mathcal{H} = \mathbb{F}^8$;
  - Capacity: $c = 9$;
  - Signature size: 2.53 KiB;
  - Verification hashing: $61\times$ Poseidon2 with width 24, $106\times$ Poseidon2 with width 16;
- **Size-Optimized Instance.** Hypercube with $v = 32$, $\mathsf{w} = 26$:
  - Randomness for message hash: $\mathcal{R} = \mathbb{F}^7$;
  - Message hash output before modulo: $A \approx 2^{31 \cdot 15}$ (i.e., 15 field elements);
  - Final layer of hypercube part into which we hash: $D = 231$;
  - Target sum: $T = 579$;
  - Hash function parameter: $\mathcal{P} = \mathbb{F}^5$;

- Field elements to encode a tweak: $2$
- Hash function output: $\mathcal{H} = \mathbb{F}^8$;
- Capacity: $c = 9$;
- Signature size: $2.03$ KiB;
- Verification hashing: $52\times$ Poseidon2 with width $24$, $221\times$ Poseidon2 with width $16$;

# References


[BDF+10] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. Cryptology ePrint Archive, Report 2010/428, 2010.

[BDH11] Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 117–129. Springer, Berlin, Heidelberg, November / December 2011.

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS$^+$ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2129–2146. ACM Press, November 2019.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.

[DKKW25] Justin Drake, Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum ethereum. Cryptology ePrint Archive, Paper 2025/055, 2025.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A Faster Version of the Poseidon Hash Function. In *AFRICACRYPT*, Lecture Notes in Computer Science. Springer Nature Switzerland, 2023.

[HBG+18] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: eXtended Merkle Signature Scheme. RFC 8391, May 2018.

[KBM23] Dmitry Khovratovich, Mario Marhuenda Beltrán, and Bart Mennink. Generic security of the SAFE API and its applications. In *ASIACRYPT (8)*, volume 14445 of *Lecture Notes in Computer Science*, pages 301–327. Springer, 2023.

[KKW25] Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. At the top of the hypercube – better size-time tradeoffs for hash-based signatures. Cryptology ePrint Archive, Paper 2025/889, 2025.

[Unr21] Dominique Unruh. Compressed permutation oracles (and the collision-resistance of sponge/SHA3). Cryptology ePrint Archive, Paper 2021/062, 2021.

[Zha12] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 758–775. Springer, Berlin, Heidelberg, August 2012.

# A Bound for the New Notion

We will now give a bound for multi-target collision resistance with random sampling and modulo (Definition 8) in the classical and quantum random oracle model. For the analysis in the quantum random oracle model, we need the following lemma.

**Lemma 4** (Lemma 3 in [BDF+10])**.** *Let $\mathcal{A}$ be an algorithm that makes at most $q$ quantum queries to an oracle $O$ that outputs values in $\mathcal{H}$. The oracle $O$ follows one of two distributions. The first is the random oracle distribution, denoted by $O_0$. The second is the distribution of oracles where the value of the oracle at each input is identically and independently distributed by some distribution $\mathcal{D}$ over $\mathcal{H}$, denoted by $O_1$. Then, we have*

$$\left|\Pr\left[\mathcal{A}^{O_0}(1^n)=1\right]-\Pr\left[\mathcal{A}^{O_1}(1^n)=1\right]\right|\leq 4q^2\sqrt{\epsilon},$$

*where $\epsilon$ is the variational distance to uniform, defined as*

$$\epsilon := \sum_{y\in\mathcal{H}}\left|\Pr\left[\mathcal{D}=y\right]-\frac{1}{|\mathcal{H}|}\right|.$$

**Lemma 5.** *Let $K,\eta\in\mathbb{N}$ be integers. Let $\mathsf{Th}\colon \mathcal{P}\times\mathcal{T}\times(\{0,1\}^{l_{msg}}\times\mathcal{R})\to\mathcal{H}\subseteq\mathbb{N}$ be a tweakable hash function modeled as a (classical or quantum) random oracle, where $\mathcal{H}=\{0,\ldots,A-1\}\subseteq\mathbb{N}$. Let $\mathsf{Prop}\colon\{0,\ldots,\eta-1\}\to\{0,1\}$ be any property. Let $\mathcal{A}$ be any (classical or quantum) adversary against multi-target collision resistance with random sampling and modulo (Definition 8) of $\mathsf{Th}$ that makes at most $q$ (classical or quantum) queries to the random oracle $\mathsf{Th}$ and $p$ classical queries to its challenge oracle (denote $q'=q+pK$). Then, we have the following bounds:*

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR\text{-}mod},K,\eta}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A})\leq 2q\cdot\frac{\eta}{A}+\frac{q'+1}{\eta}+\frac{q'pK}{|\mathcal{R}|} \qquad \textit{for a classical adversary.}$$

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR\text{-}mod},K,\eta}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A})\leq 8q^2\cdot\sqrt{\frac{\eta}{A}}+\frac{8(q'+1)^2}{\eta}+\frac{3\cdot pK}{2}\cdot\sqrt{\frac{q'}{|\mathcal{R}|}} \qquad \textit{for a quantum adversary.}$$

*Proof.* We give a sequence of games **Game.**$i$ to prove the claim, and denote the probability that the $i$th game outputs 1 by $\mathsf{Adv}^{\mathbf{Game.}i}(\mathcal{A})$.

**Game.0:** Our initial game is the original game for multi-target collision resistance with random sampling and modulo, see Definition 8. To recall, the adversary gets as input a random public parameter $P\xleftarrow{\$}\mathcal{P}$ and it gets classical access to a challenge oracle that takes as input some message $M\in\mathcal{M}$ and a tweak $T\in\mathcal{T}$. The tweak must be fresh (not used in the previous queries). The oracle then randomly samples a seed $\rho\xleftarrow{\$}\mathcal{R}$ until the digest $x:=\mathsf{Th}(P,T,M,\rho)$ such that $\mathsf{Prop}(x_\eta)=1$, where $x_\eta=x \mod \eta$. If the oracle finds such a seed then it returns $(x,\rho)$. If the oracle does not manage to find a seed after $K$ tries, it returns $\perp$. The task of the adversary is to find a message and a seed that collides with one of the returned digests under the same tweak and modular reduction. By definition, we have

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR\text{-}mod},K,\eta}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A})=\mathsf{Adv}^{\mathbf{Game.0}}(\mathcal{A}).$$

**Game.1:** In this game, we substitute the function $\mathsf{Th}$ with $\hat{\mathsf{Th}}$, that first computes a mapping to $\mathbb{Z}_\eta$ and then lifts it to $\mathcal{H}$. More precisely, write $A=b\eta+r$ and consider the following internal random oracles, not exposed to the adversary:

- Random oracle $RO_\eta\colon\mathcal{P}\times\mathcal{T}\times(\{0,1\}^{l_{msg}}\times\mathcal{R})\to\mathbb{Z}_\eta$,
- Random oracle $RO_b\colon\mathcal{P}\times\mathcal{T}\times(\{0,1\}^{l_{msg}}\times\mathcal{R})\to\mathbb{Z}_b$,
- Random oracle $RO_{b+1}\colon\mathcal{P}\times\mathcal{T}\times(\{0,1\}^{l_{msg}}\times\mathcal{R})\to\mathbb{Z}_{b+1}$.

Now, in **Game.1**, $\mathcal{A}$ gets (classical or quantum) access to $\hat{\mathsf{Th}}$ that is constructed the following way:

1. On input $x \in \mathcal{P} \times \mathcal{T} \times (\{0,1\}^{l_{msg}} \times \mathcal{R})$ compute $\gamma = RO_\eta(x)$. Note that $\gamma \in \mathbb{Z}_\eta$.
2. If $\gamma > r$. compute $c = RO_b(x)$ and return $c\eta + \gamma$. Note that this is in $\{0, \ldots, b\eta - 1\}$.
3. If $\gamma \leq r$. compute $c = RO_{b+1}(x)$ and return $c\eta + \gamma$. Note that this is in $\{0, \ldots, b\eta + r - 1\}$.

To analyze the game hop, we will use Lemma 4. To this end, we first compute the variational distance (see Lemma 4) for a fixed value $x$ that is queried to the random oracle. To this end, fix an input $x$ and a value $y \in \mathcal{H}$ and write it as $y = c_y\eta + r_y$, where $r_y \in \mathbb{Z}_\eta$ and $c_y \in \mathbb{Z}_b$ if $r_y \geq r$ and $c_y \in \mathbb{Z}_{b+1}$ if $r_y < r$. We distinguish the cases when $r_y \geq r$ and when $r_y < r$, and use the notation $\gamma = RO_\eta(x)$ and $c = RO_b(x)$ ($c = RO_{b+1}(x)$ resp.) as in the definition of $\hat{\mathsf{T}}\mathsf{h}$. If $r_y \geq r$ then

$$\begin{aligned}\Pr\left[\hat{\mathsf{T}}\mathsf{h}(x) = y\right] &= \Pr\left[\gamma = r_y \wedge c = c_y\right] = \Pr\left[\gamma = r_y\right] \cdot \Pr\left[c = c_y \mid \gamma = r_y\right] \\ &= \Pr\left[RO_\eta(x) = r_y\right] \cdot \Pr\left[RO_b(x) = c_y\right] = \frac{1}{\eta b}.\end{aligned}$$

If $r_y < r$ then

$$\begin{aligned}\Pr\left[\hat{\mathsf{T}}\mathsf{h}(x) = y\right] &= \Pr\left[\gamma = r_y \wedge c = c_y\right] = \Pr\left[\gamma = r_y\right] \cdot \Pr\left[c = c_y \mid \gamma = r_y\right] \\ &= \Pr\left[RO_\eta(x) = r_y\right] \cdot \Pr\left[RO_{b+1}(x) = c_y\right] = \frac{1}{\eta(b+1)}.\end{aligned}$$

With this at hand, we can now compute the variational distance $\epsilon$ from uniform, as defined in Lemma 4:

$$\begin{aligned}\epsilon &= \sum_{y \in \mathcal{H}} \left| \Pr\left[\hat{\mathsf{T}}\mathsf{h}(x) = y\right] - \frac{1}{A} \right| \\ &= \sum_{y \in \mathcal{H}, r_y < r} \left( \frac{1}{A} - \frac{1}{\eta(b+1)} \right) + \sum_{y \in \mathcal{H}, r_y \geq r} \left( \frac{1}{\eta b} - \frac{1}{A} \right).\end{aligned}$$

The first sum contains $r(b+1)$ terms, and the second sum contains $b(\eta - r)$ terms. So, we get

$$\begin{aligned}\epsilon &= r(b+1)\left( \frac{1}{A} - \frac{1}{\eta(b+1)} \right) + b(\eta - r)\left( \frac{1}{\eta b} - \frac{1}{A} \right) \\ &= \frac{r(b+1)}{A} - \frac{r}{\eta} + \frac{\eta - r}{\eta} - \frac{b(\eta - r)}{A} \\ &= \frac{r(b+1) - b(\eta - r)}{A} + \frac{\eta - 2r}{\eta}.\end{aligned}$$

We go to a common denominator $A\eta$ and use $A = b\eta + r$:

$$\begin{aligned}\epsilon &= \frac{r(b+1) - b(\eta - r)}{A} + \frac{\eta - 2r}{\eta} \\ &= \frac{\eta(rb + r - b\eta + br) + A(\eta - 2r)}{A\eta} \\ &= \frac{\eta rb + \eta r - b\eta^2 + \eta br + b\eta^2 + r\eta - 2b\eta r - 2r^2}{A\eta} \\ &= \frac{2\eta r - 2r^2}{A\eta}.\end{aligned}$$

Now, we upper bound, using $r \leq \eta$:

$$\epsilon = \frac{2\eta r - 2r^2}{A\eta} \leq \frac{2\eta r}{A\eta} = \frac{2r}{A} \leq \frac{2\eta}{A}.$$

Now that we have a bound for the variational distance, we can use Lemma 4 for quantum adversaries, and a simple hybrid argument over all queries for classical adversaries. We get

$$|\mathsf{Adv}^{\mathbf{Game}.0}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A}) - \mathsf{Adv}^{\mathbf{Game}.1}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A})| \leq \begin{cases} 2q\eta/A & \text{for classical adversary } \mathcal{A}, \\ 8q^2\sqrt{\eta/A} & \text{for quantum adversary } \mathcal{A} \end{cases},$$

where we have upper bounded $2\eta$ with $4\eta$ for the quantum case.

**Final Reduction:** To bound the success probability in **Game.1**, we will rely on multi-target collision resistance with random sampling of the internal random oracle $RO_\eta$. That is, given an adversary $\mathcal{A}$ against **Game.1**, we build a reduction $\mathcal{B}$ that breaks multi-target collision resistance with random sampling of $RO_\eta$. The reduction gets as input parameters $P$ and forwards it to the adversary. To provide oracle $\hat{\mathsf{Th}}$ to $\mathcal{A}$, it uses its random oracle $RO_\eta$ and internal random functions[3] $RO_b$ and $RO_{b+1}$. When $\mathcal{A}$ queries its challenge oracle with input a message and a tweak, $\mathcal{B}$ simply forwards this query to its own game, and obtains back a randomness $\rho$ and a hash value (output by $RO_\eta$.). It then uses $RO_b$ and $RO_{b+1}$ to lift the value $x$ to a corresponding output for $\hat{\mathbf{Th}}$ what to return to $\mathcal{A}$. As $\mathsf{Prop}$ is evaluated on the hash value after the modular reduction, one can see that this provides a perfect simulation of the challenge oracle. Finally, any solution for **Game.1** that $\mathcal{A}$ outputs immediately yields a solution for multi-target collision resistance with random sampling against $RO_\eta$, which $\mathcal{B}$ can then output. Hence, we obtain:

$$\mathsf{Adv}^{\mathbf{Game}.1}_{\mathsf{Th},p,\mathsf{Prop}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{SM\text{-}rTCR},K}_{RO_\eta,p,\mathsf{Prop}}(\mathcal{B}).$$

Using the bounds from [DKKW25, Section C],

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR},K}_{RO_\eta,p,\mathsf{Prop}}(\mathcal{B}) \leq \frac{q'+1}{|\eta|} + \frac{q'pK}{|\mathcal{R}|} \qquad \text{for a classical adversary.}$$

$$\mathsf{Adv}^{\mathsf{SM\text{-}rTCR},K}_{RO_\eta,p,\mathsf{Prop}}(\mathcal{B}) \leq \frac{8(q'+1)^2}{|\eta|} + \frac{3\cdot pK}{2}\cdot\sqrt{\frac{q'}{|\mathcal{R}|}} \qquad \text{for a quantum adversary.}$$

Combining all the bounds, we obtain the lemma statement. □

[3]Strictly speaking, the random functions $RO_b$ and $RO_{b+1}$ can either be simulated using a $2q$-wise independent function [Zha12] or we can assume that they are also provided by the game to the reduction. It is easy to see that the bound from [DKKW25] does not change in this case.