Recovering the tight security proof of SPHINCS$^+$

Andreas Hülsing and Mikhail Kudinov

Eindhoven University of Technology, Eindhoven, Netherlands
wotstw@huelsing.net

Abstract. In 2020, Kudinov, Kiktenko, and Fedorov pointed out a flaw in the tight security proof of the SPHINCS$^+$ construction. This work gives a new tight security proof for SPHINCS$^+$. The flaw can be traced back to the security proof for the Winternitz one-time signature scheme (WOTS) used within SPHINCS$^+$. In this work, we give a standalone description of the WOTS variant used in SPHINCS$^+$ that we call WOTS-TW. We provide a security proof for WOTS-TW and multi-instance WOTS-TW against non-adaptive chosen message attacks where the adversary only learns the public key after it made its signature query. Afterwards, we show that this is sufficient to give a tight security proof for SPHINCS$^+$. We recover almost the same bound for the security of SPHINCS$^+$, with only a factor w loss compared to the previously claimed bound, where w is the Winternitz parameter that is commonly set to 16. On a more technical level, we introduce new lower bounds on the quantum query complexity for generic attacks against properties of cryptographic hash functions and analyse the constructions of tweakable hash functions used in SPHINCS$^+$ with regard to further security properties.

Keywords: Post-quantum cryptography, hash-based signatures, W-OTS, SPHINCS$^+$, WOTS-TW, hash functions, undetectability, PRF.

1 Introduction

Recently, hash-based signatures have received a lot of attention as they are widely considered the most conservative choice for post-quantum signature schemes. At the time of writing, the stateless hash-based signature scheme SPHINCS$^+$ is a third round alternate candidate in the NIST PQC competition. However, NIST has repeatedly stated the following.

> *"NIST sees SPHINCS+ as an extremely conservative choice for standardization. If NIST's confidence in better performing signature algorithms is shaken by new analysis, SPHINCS+ could provide an immediately available algorithm for standardization at the end of the third round."*
> (Dustin Moody on the pqc-forum mailing list after new attacks on Rainbow and GeMSS were published, January 21, 2021)

One more supporting argument for the security of SPHINCS$^+$ would be a tight security reduction that allows one to derive attack complexities for a given set of parameters. However, the tight proof for SPHINCS$^+$ that was given in [BHK+19] turned out to be flawed [KKF20]. The flaw, pointed out by Kudinov, Kiktenko, and Fedorov is related to the proof of security of the used WOTS scheme. Although the flaw could not be translated into an attack, this resulted in an unsatisfactory situation. While there still exists a non-tight reduction for the security of SPHINCS$^+$, this reduction can not support the claimed security of the used SPHINCS$^+$ parameters.

This work was funded by an NWO VIDI grant (Project No. VI.Vidi.193.066). Part of this work was done while M.K. was still affiliated with the Russian Quantum Center, QApp. Date: March 28, 2022

In this work, we give a new tight security proof for SPHINCS$^+$.

Security of hash-based signatures. Analyzing the security of modern hash-based signature schemes is a multi-stage process. First, the security of the signature scheme is related to the complexity of breaking properties of the used (hash) function families. To support the security of specific parameter sets with proofs, we need an expected complexity for attacks that break the assumed properties. In general, a cryptographic hash function is considered secure if there are no attacks that perform significantly better than generic attacks. Hence, the complexity of generic attacks against these properties is analyzed. In [BHK+19], the abstraction of *tweakable hash functions* (THFs) was introduced to unify the description of schemes that only differ in the inputs that internal hash functions take but follow the same general construction. These THFs are constructed from keyed hash functions (KHFs). When using this abstraction, security of the signature scheme is related to the complexity of breaking the properties of THFs (and possibly further functions, like PRFs, or further KHFs). Security of a THF is then related to the security of the used KHF. Finally, the latter is assessed with regard to generic attacks. In all of these steps, quantum adversaries have to be considered to ensure post-quantum security.

Our contributions. With this work, we contribute to all three levels in the security analysis of SPHINCS$^+$. First, we give a new tight proof for the security of SPHINCS$^+$, assuming the used THFs provide a form of target-collision resistance (TCR), decisional second-preimage resistance (DSPR), preimage resistance (PRE), and undetectability (UD)[1]. As with all previous proofs for SPHINCS$^+$, we require that the KHF used for message compression provides interleaved target-subset resilience (ITSR) and that a secure PRF is available. Note that our new proof closes the gap again without modifying SPHINCS$^+$.

The difference to the previous security proof for SPHINCS$^+$ is in the proof of the used WOTS variant. To make the proof more easily accessible, we first extract this WOTS variant and formally define it, naming it WOTS-TW. WOTS-TW is different from other WOTS variants in that it uses THFs to construct the function chains. We then prove the security of WOTS-TW under non-adaptive chosen message attacks (EU-naCMA) where the adversary receives the public key after it made its signature query. This weaker model allows for a tight security proof for WOTS-TW while also being sufficient for security proofs of schemes like SPHINCS$^+$. A tight proof is possible because a reduction can now generate the WOTS-TW public key based on the signature query instead of guessing the query. This eliminates the loss factor introduced by guessing. At the same time, the notion is sufficient because for SPHINCS$^+$, WOTS-TW is used to sign the roots of hash trees which are generated by the reduction. In short, our new proof combines the work of Dods, Smart, and Stam [DSS05] that uses undetectability to plant preimage challenges, with the second-preimage resistance version of Hülsing [Hül13], and the approach of multi-target mitigation by Hülsing, Rijneveld, and Song [HRS16] and lifts it to the setting of tweakable hash functions. We start with a proof in the single-instance setting for better exposition and move to a proof in a multi-target setting as used in SPHINCS$^+$ afterwards.

As a second contribution, we analyze the security of THFs with respect to undetectability and preimage resistance. The remaining properties were used in the previous SPHINCS$^+$ proof and were hence already analyzed. We obtain results for the two THF-constructions (*simple* and *robust*) used in SPHINCS$^+$ that were considered in [BHK+19]. The *simple* construction simply concatenates all inputs and feeds them into the underlying hash function. This construction was previously analyzed in the quantum-accessible random oracle model (QROM). We give tight bounds for PRE and UD in the QROM (the former is based on a conjecture from [BHK+19]). For the robust construction, we show that PRE and UD can be based on PRE and UD of the used KHF, respectively.

[1] To be precise, we are considering multi-target versions of these notions which we omit in the introduction for the sake of clarity.

As a third contribution, we complete the picture for the hardness of breaking the properties of generic (hash) function families via generic attacks (see Table 1 for an overview). We obtain new results for UD and the security of PRFs for both of which we are only aware of conjectured bounds. Our analysis generally follows the framework of [HRS16], which reduces the problem of distinguishing two distributions over boolean functions to the respective security property. In [HRS16], a distribution over variable weight functions, introduced by Zhandry, is used where every input is mapped to 1 with a fixed probability. In this work, we use distributions over fixed-weight functions where the number of 1's per function is fixed. In this process, we find a useful self-reducibility result for the distinguishing problem with this kind of functions. Moreover, we establish a new bound for PRE, overcoming a previous limitation of the analysis in [HRS16] which only applied to sufficiently compressing functions. Our new approach is a reduction from SPR and DSPR as previously implicitly done in [BH19]. This gives a tight unconditional bound for the single target case. For the multi-target case, we obtain a non-tight unconditional bound and a tight bound based on a previous conjecture made in [BHK$^+$19] regarding the complexity of breaking DSPR in the multi-target case.

Acknowledgments. We want to thank Sydney Antonov for pointing out wrong bounds at the Table 1.

Organization. We introduce necessary definitions and notations as well as describe the EU-naCMA security model in Section 2. Section 3 is devoted to the description of the WOTS-TW scheme. In Section 4 we provide a security reduction for WOTS-TW in the single instance setting and in Section 5 we lift the result to the multi-instance setting with possibly dependent messages. The security proof for SPHINCS$^+$ that uses WOTS-TW as a building block is then given in Section 6. The summary of the state of the art for generic security bounds and analysis of quantum generic security of UD and PRF properties is given in Section 7. In Section 8 we analyze the generic constructions of a tweakable hash function from a keyed hash function.

2 Preliminaries

In this section we introduce the definitions of building blocks, and security notions for hash functions and signatures that we use. We begin with the notion of a tweakable hash function, introduced in the construction of SPHINCS$^+$ [BHK$^+$19], and its security. Beyond the presented notions, we make use of the standard definition for PRFs which for reference can be found in Appendix A. For signatures we consider the common existential unforgeability notion but under non-adaptive message attacks. In this setting the adversary has to select a set of q messages that it will get signed before it receives the public key. For one-time signatures we have $q = 1$. A detailed formal definition can be found in Appendix B.

2.1 Tweakable hash functions.

In this section we recall the definition of tweakable hash functions and related security notions from [BHK$^+$19]. These properties will later be used to prove the security of our WOTS-TW scheme.

Function definition. A *tweakable* hash function takes public parameters P and context information in form of a *tweak* T in addition to the message input. The public parameters might be thought of as a function key or index. The tweak might be interpreted as a nonce.

Definition 1 (Tweakable hash function). *Let $n, m \in \mathbb{N}$, $\mathcal{P}$ the public parameters space and $\mathcal{T}$ the tweak space. A tweakable hash function is an efficient function*

$$\mathbf{Th} : \mathcal{P} \times \mathcal{T} \times \{0,1\}^m \to \{0,1\}^n,\ \mathrm{MD} \leftarrow \mathbf{Th}(P, T, M)$$

mapping an m-bit message M to an n-bit hash value MD *using a function key called public parameter $P \in \mathcal{P}$ and a tweak $T \in \mathcal{T}$.*

We will sometimes denote $\mathbf{Th}(P, T, M)$ as $\mathbf{Th}_{P,T}(M)$. In SPHINCS$^+$, a public value *Seed* is used as public parameter which is part of the SPHINCS$^+$ public key (the name comes from a specific construction of a tweakable hash function that uses the public parameters as seed for a PRG). For the tweak, SPHINCS$^+$ uses a so-called hash function address (**ADRS**) that identifies the position of the hash function call within the virtual structure defined by a SPHINCS$^+$ key pair. We use the same approach for WOTS-TW, i.e., the public parameter is a seed value that becomes part of the public key if WOTS-TW is used standalone. If it is encompassed in a larger structure like SPHINCS$^+$, the public parameter will typically be that used in the encompassing structure and is therefore only part of that structure's public key. In this case, the hash addresses have to be unique within the entire structure. Therefore, the address usually contains a prefix determined by the calling structure.

Security notions. To provide a security proof for WOTS-TW we require that the used tweakable hash functions have certain security properties. Specifically, we require the following properties or some variations of them which will be discussed below:

- *post-quantum single-function, multi-target collision resistance for distinct tweaks* (PQ-SM-DT-TCR);
- *post-quantum single-function, multi-target preimage resistance for distinct tweaks* (PQ-SM-DT-PRE);
- *post-quantum single-function, multi-target undetectability for distinct tweaks* (PQ-SM-DT-UD).

These properties were already considered in previous work. We only slightly adapt them. We introduce a *weak* variant that adds some limitation on when the adversary is allowed to get access to the tweakable hash function and in what way. This week notion has already been used in the [BHK+19] paper for some properties but without naming it. Moreover, in the context of multi-instance constructions like SPHINCS$^+$, we need another generic extension to collections of tweakable hash functions, discussed at the end of the subsection.

If we consider a weak variant of some property Prop we will denote it as W-Prop. Most of the properties that we discuss are defined with a two stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{A}_1$ is used to specify the challenge and $\mathcal{A}_2$ actually outputs the solution for the challenge. In the weak variant of some property the adversary gets the description of the hash function only after he specified the challenge, which is usually done by querying an oracle. In this case $\mathcal{A}_1$ does not have a description of the hash function, only $\mathcal{A}_2$ has. When considered in the QROM, this means that only $\mathcal{A}_2$ is allowed to make queries to the random oracle. Note that we assume that $\mathcal{A}_1$ shares state with $\mathcal{A}_2$. To make this explicit we let $\mathcal{A}_1$ output its state and let $\mathcal{A}_2$ take it as an input. Our reduction proof will show that if the scheme is broken at least one of the required properties of the hash function is broken. One can see that breaking the weak variant of some property is potentially harder than breaking the actual property and clearly not easier. Hence it is less likely that the property will be broken and this leads to a higher security of the scheme.

We generally consider post-quantum security in this work. Therefore, we will omit the PQ prefix from now on and consider it understood that we always consider quantum adversaries. Since we are working in the post-quantum setting, we assume that adversaries have access to a quantum computer but honest parties do not. Hence, any oracles that implement secretly-keyed functions only allow for classical queries. Moreover in all of the properties an adversary can influence the challenges by specifying the tweaks used in challenges. We generally restrict this control in so far as we do not allow more than one challenge for the same tweak (indicated by the DT label). As we have this restriction for all of our properties we omit the DT label in all of the security notions.

Below we will define success probabilities and advantages of the adversaries against different properties of hash functions. Here we define the insecurity of a property Prop for parameter p (which usually denotes the number of targets) of (tweakable) hash function F against time-ξ adversaries as the maximum success probability for finding games or maximum advantage for distinguishing games of any such adversary:

$$\mathrm{InSec}^{\mathsf{Prop}}(F;\xi,p) = \max_{\mathcal{A}}\{\mathrm{Succ/Adv}^{\mathsf{Prop}}_{F,p}(\mathcal{A})\}.$$

Now we will discuss above properties and their variations. We provide additional intuition for those notions in Appendix A.

Definition 2 (W-SM-TCR). *In the following let* $\mathbf{Th}$ *be a tweakable hash function as defined above. We define the success probability of any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against the* W-SM-TCR *security of* $\mathbf{Th}$*. The definition is parameterized by the number of targets* p *for which it must hold that* $p \leq |\mathcal{T}|$*. In the definition,* $\mathcal{A}_1$ *is allowed to make* p *queries to an oracle* $\mathbf{Th}(P,\cdot,\cdot)$*. We denote the set of* $\mathcal{A}_1$*'s queries by* $Q = \{(T_i, M_i)\}_{i=1}^p$ *and define the predicate* $\mathbf{DIST}(\{T_i\}_{i=1}^p) = (\forall i,k \in [1,p], i \neq k) : T_i \neq T_k$*, i.e.,* $\mathbf{DIST}(\{T_i\}_{i=1}^p)$ *outputs 1 iff all tweaks are distinct.*

$$\begin{aligned}&\mathrm{Succ}^{\text{W-SM-TCR}}_{\mathbf{Th},p}(\mathcal{A}) = \Pr[P \leftarrow_{\$} \mathcal{P}; S \leftarrow \mathcal{A}_1^{\mathbf{Th}(P,\cdot,\cdot)}(\ );\\ &(j,M) \leftarrow \mathcal{A}_2(Q,S,P,\mathbf{Th}) : \mathbf{Th}(P,T_j,M_j) = \mathbf{Th}(P,T_j,M)\\ &\wedge M \neq M_j \wedge \mathbf{DIST}(\{T_i\}_{i=1}^p)]\end{aligned}$$

Definition 3 (W-SM-PRE). *In the following let* $\mathbf{Th}$ *be a tweakable hash function as defined above. We define the success probability of any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against the* W-SM-PRE *security of* $\mathbf{Th}$*. The definition is parameterized by the number of targets* p *for which it must hold that* $p \leq |\mathcal{T}|$*. In the definition,* $\mathcal{A}_1$ *is allowed to make* p *queries to an oracle* $\mathbf{Th}(P,\cdot,x_i)$*, where* x_i *is chosen uniformly at random for the query* i *(the value of* x_i *stays hidden from* $\mathcal{A}$*). We denote the set of* $\mathcal{A}_1$*'s queries by* $Q = \{T_i\}_{i=1}^p$ *and define the predicate* $\mathbf{DIST}(\{T_i\}_{i=1}^p)$ *as we did in the definition above.*

$$\begin{aligned}&\mathrm{Succ}^{\text{W-SM-PRE}}_{\mathbf{Th},p}(\mathcal{A}) = \Pr[P \leftarrow_{\$} \mathcal{P}; S \leftarrow \mathcal{A}_1^{\mathbf{Th}(P,\cdot,x_i)}(\ );\\ &(j,M) \leftarrow \mathcal{A}_2(Q,S,P,\mathbf{Th}) : \mathbf{Th}(P,T_j,M) = \mathbf{Th}(P,T_j,x_j) \wedge \mathbf{DIST}(\{T_i\}_{i=1}^p)]\end{aligned}$$

Definition 4 (W-SM-UD). *In the following let* $\mathbf{Th}$ *be a tweakable hash function as defined above. We define the advantage of any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against the* W-SM-UD *security of* $\mathbf{Th}$*. The definition is parameterized by the number of targets* p *for which it must hold that* $p \leq |\mathcal{T}|$*. First the challenger flips a fair coin* b *and chooses a public parameter* $P \leftarrow_{\$} \mathcal{P}$*. Next consider an oracle* $\mathcal{O}_P(\mathcal{T},\{0,1\})$*, which works the following way:* $\mathcal{O}_P(T,0)$ *returns* $\mathbf{Th}(P,T,x_i)$*, where* x_i *is chosen uniformly at random for the query* i*;* $\mathcal{O}_P(T,1)$ *returns* y_i*, where* y_i *is chosen uniformly at random for the query* i*. In the definition,* $\mathcal{A}_1$ *is allowed to make* p *queries to an oracle* $\mathcal{O}_P(\cdot,b)$*. The goal of* $\mathcal{A}$ *is to distinguish whether the oracle is* $\mathcal{O}_P(\mathcal{T},0)$ *or* $\mathcal{O}_P(\mathcal{T},1)$*. We denote the set of* $\mathcal{A}_1$*'s queries by* $Q = \{T_i\}_{i=1}^p$ *and define the predicate* $\mathbf{DIST}(\{T_i\}_{i=1}^p)$ *as we did above.*

$$\begin{aligned}&\mathrm{Adv}^{\text{W-SM-UD}}_{\mathbf{Th},p}(\mathcal{A}) =\\ &\quad|\Pr[P \leftarrow_{\$} \mathcal{P}; S \leftarrow \mathcal{A}_1^{\mathcal{O}_P(\cdot,0)}(\ ); 1 \leftarrow \mathcal{A}_2(Q,S,P,\mathbf{Th}) \wedge \mathbf{DIST}(\{T_i\}_{i=1}^p)] -\\ &\quad\Pr[P \leftarrow_{\$} \mathcal{P}; S \leftarrow \mathcal{A}_1^{\mathcal{O}_P(\cdot,1)}(\ ); 1 \leftarrow \mathcal{A}_2(Q,S,P,\mathbf{Th}) \wedge \mathbf{DIST}(\{T_i\}_{i=1}^p)]|\end{aligned}$$

At this point, we have finished describing the properties that will be needed to construct a reduction proof for WOTS-TW. But for the further analysis of those properties and analysis of SPHINCS$^+$ one would need several more properties.

Decisional Second Preimage Resistance (DSPR) and its variants were introduced and motivated in [BH19]. Here we present a multi-target version of DSPR which is denoted as W-SM-DSPR. To do so, we need a second-preimage exists predicate for THFs.

Definition 5 ($\mathsf{SP}_{P,T}$). *A second preimage exists predicate of tweakable hash function* $\mathbf{Th} : \mathcal{P} \times \mathcal{T} \times \{0,1\}^m \to \{0,1\}^n$ *with a fixed* $P \in \mathcal{P}$, $T \in \mathcal{T}$ *is the function* $\mathsf{SP}_{P,T} : \{0,1\}^m \to \{0,1\}$ *defined as follows:*

$$\mathsf{SP}_{P,T}(x) \stackrel{def}{=} \begin{cases} 1 \text{ if } |\mathbf{Th}_{P,T}^{-1}(\mathbf{Th}_{P,T}(x))| \geq 2 \\ 0 \text{ otherwise} \end{cases},$$

where $\mathbf{Th}_{P,T}^{-1}$ *refers to the inverse of the tweakable hash function with fixed public parameter and a tweak.*

Now we present the definition of W-SM-DSPR from [BHK$^+$19] for a tweakable hash function.

Definition 6 (W-SM-DSPR). *Let* $\mathbf{Th}$ *be a tweakable hash function. Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a two stage adversary. The number of targets is denoted with* p*, where the following inequality must hold:* $p \leq |\mathcal{T}|$. $\mathcal{A}_1$ *is allowed to make* p *queries to an oracle* $\mathbf{Th}(P,\cdot,\cdot)$. *We denote the query set* Q *and predicate* $\mathsf{DIST}(\{T_i\}_1^p)$ *as in previous definitions.*

$$\mathrm{Adv}^{\text{W-SM-DSPR}}_{\mathbf{Th},p}(\mathcal{A}) = \max\{0, \mathsf{succ} - \mathsf{triv}\},$$

where

$$\begin{aligned} \mathsf{succ} = \Pr[P \leftarrow_{\$} \mathcal{P}; S \leftarrow \mathcal{A}_1^{\mathbf{Th}(P,\cdot,\cdot)}(); (j,b) \leftarrow \mathcal{A}_2(Q,S,P,\mathbf{Th}) : \\ \mathsf{SP}_{P,T_j}(M_j) = b \wedge \mathsf{DIST}(\{T_i\}_1^p)]. \\ \mathsf{triv} = \Pr[P \leftarrow_{\$} \mathcal{P}; S \leftarrow \mathcal{A}_1^{\mathbf{Th}(P,\cdot,\cdot)}(); (j,b) \leftarrow \mathcal{A}_2(Q,S,P,\mathbf{Th}) : \\ \mathsf{SP}_{P,T_j}(M_j) = 1 \wedge \mathsf{DIST}(\{T_i\}_1^p)]. \end{aligned}$$

Security for a collection of tweakable hash functions. In more complex constructions like SPHINCS$^+$, we make use of a collection of tweakable hash functions which we call $\mathbf{Th}_\lambda$. In this case $\mathbf{Th}_\lambda$ consists of a set of tweakable hash functions $\mathbf{Th}_{m_i}$ for different m_i, the length of messages they process. This notion of a collection of tweakable hash functions is necessary as we use the same public parameters for all functions in the collection. Especially, it is necessary to make the security notions above usable in the context of SPHINCS$^+$. The problem is that when used in constructions like SPHINCS$^+$ or XMSS, queries to the challenge oracle may depend on the outputs of other functions in the collection, or even the same function but with different tweaks. This is incompatible with above definitions as the public parameters are only given to the adversary after all challenge queries are made.

We solve this issue by extending all the above *stand-alone* security properties to the case of collections. The definitions for functions that are part of a collection only differ from the above in a single spot. We give the first part of the adversary $\mathcal{A}_1$, that makes the challenge queries, access to another oracle $\mathbf{Th}_\lambda(P,\cdot,\cdot)$, initialized with P. The oracle takes an input M and a tweak T and, depending on the length $m = |M|$ of M returns $\mathbf{Th}_m(P,M,T)$. The only limitation is that $\mathcal{A}$ is not allowed to use the same tweak in queries to both oracles, the challenge oracle and the collection oracle. In general, $\mathcal{A}$ is allowed to query the challenge oracle as well as $\mathbf{Th}_\lambda$ with a message of length x, as long as the used tweak is never used in a query to the challenge oracle.

Definition 7 (W-SM-TCR, W-SM-PRE, W-SM-UD, W-SM-DSPR for members of a collection). *Let* $\mathbf{Th}_m$ *be a* THF *as defined above with message length* m*. Moreover, let* $\mathbf{Th}_m$ *be an element of a collection* $\mathbf{Th}_\lambda$ *of* THFs *as described above. Consider an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against the* W-SM-TCR *(,* W-SM-PRE*,* W-SM-UD*,* W-SM-DSPR*) security of* $\mathbf{Th}_m$ *as part of collection* $\mathbf{Th}_\lambda$ *(which we denote as* $\mathbf{Th}_m \in \mathbf{Th}_\lambda$*). Let* $\mathbf{Th}_\lambda(P, \cdot, \cdot)$ *denote an oracle for* $\mathbf{Th}_\lambda$ *as described above and denote by* $\{T_i^\lambda\}_1^{p_\lambda}$ *the tweaks used in the queries made by* $\mathcal{A}$*. We define the success probability of* $\mathcal{A}$ *against* W-SM-TCR *(,* W-SM-PRE*,* W-SM-UD*,* W-SM-DSPR*) security of* $\mathbf{Th}_m$ *as part of collection* $\mathbf{Th}_\lambda$ *as the success probability of* $\mathcal{A}$ *against standalone* W-SM-TCR *(,* W-SM-PRE*,* W-SM-UD*,* W-SM-DSPR*) security of* $\mathbf{Th}_m$ *defined above, when* $\mathcal{A}_1$ *is additionally given oracle access to* $\mathbf{Th}_\lambda(P, \cdot, \cdot)$ *with the condition that* $\{T_i\}_1^p \cap \{T_i^\lambda\}_1^{p_\lambda} = \emptyset$*.*

In the case of W-SM-TCR, we will abuse notation when it comes to the security of SPHINCS$^+$ and consider the joined security of several members of a collection of tweakable hash functions.

3 WOTS-TW

SPHINCS$^+$ [BHK+19] developed its own variant of the Winternitz OTS. However, the authors never explicitly defined that variant. As the flaw in the SPHINCS$^+$ security proof was in the proof for their WOTS scheme, we give a separate description of the scheme in this section. As the distinguishing feature of this variant is the use of tweakable hash functions, we call it WOTS-TW.

3.1 Parameters

WOTS-TW uses several parameters. The main security parameter is $n \in \mathbb{N}$. The length of messages that are signed is denoted as m. In the case of SPHINCS$^+$, $m = n$. The Winternitz parameter $w \in \mathbb{N}$ determines a base of the representation that is used in the scheme and determines the parameter l:

$$l_1 = \left\lceil \frac{m}{\log(w)} \right\rceil, \quad l_2 = \left\lfloor \frac{\log(l_1(w-1))}{\log(w)} \right\rfloor + 1, \quad l = l_1 + l_2.$$

The tweak space $\mathcal{T}$ must be at least of size lw. The size of the tweak space should be bigger if we use several instances of WOTS-TW in a bigger construction such as SPHINCS$^+$ so we can use a different tweak for each hash function call. We also need a pseudorandom function $\mathbf{PRF} : \{0,1\}^n \times \mathcal{T} \to \{0,1\}^n$, and a tweakable hash function $\mathbf{Th} : \{0,1\}^n \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$.

3.2 Addressing scheme

For the tweakable hash functions to guarantee security, they have to be called with different tweaks. This is achieved using what was called an addressing scheme in SPHINCS$^+$. Such an addressing scheme assigns a unique address to every tweakable hash function call in the scheme and the address space is part of the tweak space such that addresses can be used as tweaks. We do not specify a concrete addressing scheme in this work (see the SPHINCS$^+$ specification [ABB+20] for an example). Abstractly, we achieve unique addresses the following way. A Winternitz key pair defines a structure of l hash chains, each of which makes $w-1$ calls to the tweakable hash function. For a unique addressing scheme, one may use any injective function that takes as input $i \in [0, l-1]$, $j \in [0, w-2]$, and possibly a prefix, and maps into the address space. The prefix is necessary to ensure uniqueness if many instances of WOTS-TW are used in a single construction. We will use **ADRS** to denote that prefix. The tweak associated with the j-th function call in the i-th chain is then defined as the output of this function on input i, j (and a possible prefix) and denoted as $T_{i,j}$.

3.3 WOTS-TW scheme

The main difference between WOTS variants is in the way they do hashing. Previously, the distinction was made in the definition of the so-called chaining function that describes how the hash chains are computed. For WOTS-TW this distinction is further shifted into the construction of the tweakable hash function **Th**. The chaining function then looks as follows:

Chaining function $c^{j,k}(x, i, \text{Seed})$: The chaining function takes as inputs a message $x \in \{0,1\}^n$, iteration counter $k \in \mathbb{N}$, start index $j \in \mathbb{N}$, chain index i, and public parameters Seed. The chaining function then works the following way. In case $k \leq 0$, c returns x, i.e., $c^{j,0}(x, i, \text{Seed}) = x$. For $k > 0$ we define c recursively as

$$c^{j,k}(x, i, \text{Seed}) = \mathbf{Th}(\text{Seed}, T_{i,j+k-1}, c^{j,k-1}(x, i, \text{Seed}))\,.$$

If we consider several instances of WOTS-TW then we will use $c^{j,k}_{\mathbf{ADRS}}(x, i, \text{Seed})$ to denote that tweaks that are used to construct the chain have **ADRS** as a prefix. With this chaining function, we describe the algorithms of WOTS-TW.

Key Generation Algorithm $(\mathsf{SK}, \mathsf{PK}) \leftarrow$ WOTS-TW.kg$(\mathcal{C}; \mathcal{S})$: The key generation algorithm optionally takes as input context information $\mathcal{C} = (\text{Seed}, \mathbf{ADRS})$, consisting of a public seed Seed $\in \{0,1\}^n$ and a global address **ADRS**, as well as randomness $\mathcal{S} \in \{0,1\}^n$ which we call the secret seed. These inputs are meant for the use in more complex protocols. If they are not provided, key generation randomly samples the seeds and sets **ADRS** to 0. The key generation algorithm then computes the internal secret key $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_l)$ as $\mathsf{sk}_i \leftarrow \mathbf{PRF}(\mathcal{S}, T_{i,0}))$, i.e., the $l \cdot n$ bit secret key elements are derived form the secret seed using addresses. The element of the public key pk is computed as

$$\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l) = (c^{0,w-1}(\mathsf{sk}_1, 1, \text{Seed}), \ldots, c^{0,w-1}(\mathsf{sk}_l, l, \text{Seed})).$$

The key generation algorithm returns $\mathsf{SK} = (\mathcal{S}, \mathcal{C})$ and $\mathsf{PK} = (\mathsf{pk}, \mathcal{C})$ Note that we can compute sk and pk from SK.

Signature Algorithm $\sigma \leftarrow$ WOTS-TW.sign(M, SK): On input of an m-bit message M, and the secret key $\mathsf{SK} = (\mathcal{S}, \mathcal{C})$, the signature algorithm first computes a base w representation of $M : M = (M_1, \ldots, M_{l_1})$, $M_i \in \{0, \ldots, w-1\}$. That is, M is treated as the binary representation of a natural number x and then the w-ary representation of x is computed. Next it computes the checksum $C = \sum_{i=1}^{l_1}(w - 1 - M_i)$ and its base w representation $C = (C_1, \ldots, C_{l_2})$. We set $B = (b_1, \ldots, b_l) = M||C$, the concatenation of the base w representations of M and C. Then the internal secret key is regenerated using $\mathsf{sk}_i \leftarrow \mathbf{PRF}(\mathcal{S}, T_{i,0})$ the same way as during key generation. The signature is computed as

$$\sigma = (\sigma_1, \ldots, \sigma_l) = (c^{0,b_1}(\mathsf{sk}_1, 1, \text{Seed}), \ldots, c^{0,b_l}(\mathsf{sk}_l, l, \text{Seed})).$$

Verification Algorithm $(\{0,1\} \leftarrow$ WOTS-TW.vf$(M, \sigma, \mathsf{PK}))$: On input of m-bit message M, a signature σ, and public key $\mathsf{PK} = (\mathsf{pk}, \mathcal{C})$, the verification algorithm computes the $b_i, 1 \leq i \leq l$ as described above and checks if

$$\mathsf{pk} \stackrel{?}{=} \mathsf{pk}' = (\mathsf{pk}'_1, \ldots, \mathsf{pk}'_l) = (c^{b_1, w-1-b_1}(\sigma_1, 1, \text{Seed}), \ldots, c^{b_l, w-1-b_l}(\sigma_l, l, \text{Seed}))\,.$$

In case of equality the algorithm outputs true and false otherwise.

4 Security of WOTS-TW

Now we will reduce the security of WOTS-TW to the security properties of the tweakable hash function **Th** and the pseudorandom function family PRF. To do so we will give a standard game-hopping proof. Intuitively the proof goes through the following steps.

- First, we replace the inner secret key elements that are usually generated using PRF by uniformly random values. The two cases must be computationally indistinguishable if PRF is indeed pseudorandom.
- Next we replace the blocks in the chains that become part of the signature by the hash of random values. We need this so that we can later place preimage challenges at these positions of the chain. Here it is important to note that preimage challenges are exactly such hashes of random domain elements and not random co-domain elements. To argue that these two cases are indistinguishable, we need a hybrid argument since for most chains we replace the outcome of several iterations of hashing with a random value.
- Lastly we show that breaking the EU-naCMA property of our scheme in this final case will either allow us to extract a target-collision or a preimage for a given challenge.

Theorem 1. *Let n, $w \in \mathbb{N}$ and $w = poly(n)$. Let $\mathbf{Th} : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a* W-SM-TCR*,* W-SM-PRE*, and* W-SM-UD *function. Let $\mathbf{PRF} : \mathcal{S} \times \mathcal{T} \to \{0,1\}^n$ be a pseudorandom function. Then the insecurity of the WOTS-TW scheme against EU-naCMA attack is bounded by*

$$\begin{aligned}\mathrm{InSec}^{EU\text{-}naCMA}(\text{WOTS-TW}; t, 1) \leq \\ \mathrm{InSec}^{\text{PRF}}(\mathbf{PRF}; \widetilde{t}, l) + \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{Th}; \widetilde{t}, lw) + \\ \mathrm{InSec}^{\text{W-SM-PRE}}(\mathbf{Th}; \widetilde{t}, l) + w \cdot \mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{Th}; \widetilde{t}, l)\end{aligned}$$

with $\widetilde{t} = t + lw$, where time is given in number of **Th** *evaluations.*

Proof. First consider the following two games: GAME.1 is the original EU-naCMA game and GAME.2 is the same as GAME.1 but all outputs of **PRF** are replaced by random values. We claim that the difference in the success probability of $\mathcal{A}$ playing these games must be bound by $\mathrm{InSec}^{\text{PRF}}(\mathbf{PRF}; \widetilde{t}, l)$.

Next we consider GAME.3 which is the same as GAME.2 but to answer the message signing request we build the signature from nodes that are computed applying **Th** only once instead of b_i times (except if $b_i = 0$, then we return a random value as in the previous game). The public key is constructed from that signature by finishing the chain according to the usual algorithm. We will detail the process in the proof below. We claim that the difference in the success probability of $\mathcal{A}$ playing these games must be bounded by $w \cdot \mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{Th}; \widetilde{t}, l)$.

Afterwards, we consider GAME.4, which differs from GAME.3 in that we are considering the game lost if an adversary outputs a valid forgery (M', σ') where there exists an i such that $b'_i < b_i$ and $c^{(b'_i, b_i - b'_i)}(\sigma'_i, i, \text{Seed}) \neq \sigma_i$. We claim that the difference in the success probability of $\mathcal{A}$ playing these games must be bound by $\mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{Th}; \widetilde{t}, lw)$.

If we now consider how $\mathcal{A}$ can win in GAME.4 there is just one viable case left. By the properties of the checksum, there has to be at least one i with $b'_i < b_i$. For any such i the values that get computed from the forgery during verification fully agree with those values that are computed during the verification of the signature by the last game hop. This means that we can use an $\mathcal{A}$ that wins in GAME.4 to find a preimage. We claim that the success probability of the adversary $\mathcal{A}$ in GAME.4 must be bounded by $\mathrm{InSec}^{\text{W-SM-PRE}}(\mathbf{Th}; \widetilde{t}, l)$.

In summary, we get the following claims:

Claim 1. $|\text{Succ}^{\text{GAME.1}}(\mathcal{A}) - \text{Succ}^{\text{GAME.2}}(\mathcal{A})| \leq \text{InSec}^{\text{PRF}}(\mathbf{PRF}; \widetilde{t}, l)$.

Claim 2. $|\text{Succ}^{\text{GAME.2}}(\mathcal{A}) - \text{Succ}^{\text{GAME.3}}(\mathcal{A})| \leq w \cdot \text{InSec}^{\text{W-SM-UD}}(\mathbf{Th}; \widetilde{t}, l)$.

Claim 3. $|\text{Succ}^{\text{GAME.3}}(\mathcal{A}) - \text{Succ}^{\text{GAME.4}}(\mathcal{A})| \leq \text{InSec}^{\text{W-SM-TCR}}(\mathbf{Th}; \widetilde{t}, lw)$.

Claim 4. $\text{Succ}^{\text{GAME.4}}(\mathcal{A}) \leq \text{InSec}^{\text{W-SM-PRE}}(\mathbf{Th}; \widetilde{t}, l)$.

The remainder of the proof consists of proving these claims. We then combine the bounds from the claims to obtain the bound of the theorem.

Proof of Claim 1.

Claim 1. $|\text{Succ}^{\text{GAME.1}}(\mathcal{A}) - \text{Succ}^{\text{GAME.2}}(\mathcal{A})| \leq \text{InSec}^{\text{PRF}}(\mathbf{PRF}; \widetilde{t}, l)$.

Proof. We replace **PRF** in GAME.1 by the oracle provided by the PRF game and output 1 whenever $\mathcal{A}$ succeeds. If the oracle is the real **PRF** function keyed with a random secret key, the view of $\mathcal{A}$ is identical to that in GAME.1. If the oracle is the truly random function the argument is a bit more involved. In this case, it is important to note that $\mathcal{A}$ never gets direct access to the oracle but only receives outputs of the oracle. The inputs on which the oracle is queried to obtain these outputs are all unique. Hence, the outputs are uniformly random values. Therefore, the view of $\mathcal{A}$ in this case is exactly that of GAME.2. Consequently, the difference of the probabilities that the reduction outputs 1 in either of the two cases (which is the PRF distinguishing advantage) is exactly the difference of the success probabilities of $\mathcal{A}$ in the two games.

Proof of Claim 2. We first give a more detailed description of GAME.3. In the EU-naCMA game the adversary $\mathcal{A}$ asks to sign a message M without knowing the public key. This message M gets encoded as $B = b_1, \ldots, b_l$. In GAME.3, to answer the query we will perform the following operations. First we generate l values uniformly at random: $u_i \leftarrow_{\$} \{0,1\}^n,\ i \in \{1, \ldots, l\}$. Next we answer the signing query with a signature $\sigma = (\sigma_1, \ldots, \sigma_l)$, where $\sigma_i = \mathbf{Th}(\text{Seed}, T_{i,b_i-1}, u_i)$ if $b_i > 0$ and $\sigma_i = u_i$ if $b_i = 0$. Then the public key is constructed as

$$\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l) = (c^{b_1, w-1-b_1}(\sigma_1, 1, \text{Seed}), \ldots, c^{b_l, w-1-b_l}(\sigma_l, l, \text{Seed}))\,, \tag{1}$$

and public key and signature are returned to the adversary. The reason we consider this game is that to bound the final success probability in GAME.4 we will have a reduction replace the u_i with W-SM-PRE challenges. The resulting signatures have exactly the same distribution as the ones we get here. To show that this cannot change the adversary's success probability in a significant way, we now prove the following claim.

Claim 2. $|\text{Succ}^{\text{GAME.2}}(\mathcal{A}) - \text{Succ}^{\text{GAME.3}}(\mathcal{A})| \leq w \cdot \text{InSec}^{\text{W-SM-UD}}(\mathbf{Th}; \widetilde{t}, l)$.

Proof. Consider the following scenario. Let the adversary's query be M. During the signing algorithm M is encoded as $B = \{b_1, \ldots, b_l\}$. Consider two distributions $D_0 = \{\xi_1, \ldots, \xi_l\}$, where $\xi_i \leftarrow_{\$} \{0,1\}^n,\ i \in [1, l]$ and $D_{Kg} = \{y_1, \ldots, y_l\}$, where $y_i = c^{0, b_i - 1}(\xi_i, i, \text{Seed}),\ \xi_i \leftarrow_{\$} \{0,1\}^n,\ i \in [1, l]$. Samples from the first distribution are just random values, and the samples from D_{Kg} are distributed the same way as the $(b_i - 1)$-th values of valid WOTS-TW chains. Assume we play a game where we get access to an oracle $\mathcal{O}_\phi$ that on input B returns $\phi = \{\phi_1, \ldots, \phi_l\}$, either initialized with a sample from D_0 or with a sample from D_{Kg}. Each case occurs with probability 1/2. Then we can construct an algorithm $\mathcal{M}^{\mathcal{A}}_{2-3}$ as in Algorithm 1 that can distinguish these two cases using a forger $\mathcal{A}$.

Algorithm 1: $\mathcal{M}^{\mathcal{A}}_{2-3}$

Input : Access to a distribution oracle $\mathcal{O}_\phi$ and forger $\mathcal{A}$
Output: 0 or 1.

Start $\mathcal{A}$ to obtain query with a message M.
Encode M as $B = b_1, \ldots, b_l$ as in signature algorithm.
Call $\mathcal{O}_\phi(B)$ to obtain sample ϕ
Construct the signature σ doing one chain step on each sample where $b > 0$ and compute the public key from the signature:
for $1 \leq i \leq l$ **do**
  if $b_i > 0$ **then**
    $\sigma_i = c^{b_i-1,1}(\phi_i, i, \text{Seed})$
  $\mathsf{pk}_i = c^{b_i,w-1-b_i}(\sigma_i, i, \text{Seed})$
Send $\mathsf{PK} = (\mathsf{pk}, \text{Seed})$ and σ to $\mathcal{A}$.
if $\mathcal{A}$ *returns a valid forgery* (M', σ') **then**
  return *1*
else
  return *0*

Let us consider the behavior of $\mathcal{M}^{\mathcal{A}}_{2-3}$ when $\mathcal{O}_\phi$ samples from $\mathcal{D}_{Kg}$. In this case all the elements in the chains are distributed the same as in GAME.2. The probability that $\mathcal{M}^{\mathcal{A}}_{2-3}$ outputs 1 is the same as the success probability of the adversary in GAME.2. If ϕ instead is from $\mathcal{D}_0$, then the distribution of the elements in the chains is the same as in GAME.3. Hence the probability that $\mathcal{M}^{\mathcal{A}}_{\mathcal{UD}}$ outputs 1 is the same as the success probability of the adversary in GAME.3. By definition, the advantage of $\mathcal{M}^{\mathcal{A}}_{2-3}$ in distinguishing $\mathcal{D}_0$ from $\mathcal{D}_{Kg}$ is hence given by

$$\text{Adv}_{\mathcal{D}_0,\mathcal{D}_{Kg}}(\mathcal{M}^{\mathcal{A}}_{2-3}) = |\text{Succ}^{\text{GAME.2}}(\mathcal{A}) - \text{Succ}^{\text{GAME.3}}(\mathcal{A})| \tag{2}$$

The remaining step is to derive an upper bound for $\text{Adv}_{\mathcal{D}_0,\mathcal{D}_{Kg}}(\mathcal{M}^{A}_{\text{UD}})$ using the insecurity of the W-SM-UD property. For this purpose we use a hybrid argument.

Let $b_{\max} = \max\{b_1, \ldots, b_l\}$ be the maximum of the values in the message encoding of M. Let H_k be the distribution obtained by computing the values in ϕ as $\phi_i = c^{k,b_i-1-k}(\xi_i, i, Seed)$, $\xi_i \leftarrow_\$ \{0,1\}^n$. Then $H_0 = D_{Kg}$ and $H_{b_{\max}-1} = D_0$ (Note that the chaining function returns the identity when asked to do a negative amount of steps). As $\mathcal{M}^{\mathcal{A}}_{2-3}$ distinguishes the extreme cases, by a hybrid argument there are two consecutive hybrids H_j and H_{j+1} that can be distinguished with probability $\geq \text{Adv}_{\mathcal{D}_0,\mathcal{D}_{Kg}}(\mathcal{M}^{\mathcal{A}}_{2-3})/(b_{\max} - 1)$.

To bound the success probability of an adversary in distinguishing two such consecutive hybrids, we build a second reduction $\mathcal{M}^{\mathcal{B}}_{\text{UD}}$ that uses $\mathcal{B} = \mathcal{M}^{\mathcal{A}}_{2-3}$ to break W-SM-UD. For this purpose, $\mathcal{M}^{\mathcal{B}}_{\text{UD}}$ simulates $\mathcal{O}_\phi$. To answer a query for $B = b_1, \ldots, b_l$, $\mathcal{M}^{\mathcal{B}}_{\text{UD}}$ plays in the W-SM-UD game, interacting with the W-SM-UD oracle $\mathcal{O}_{\text{UD}}(\cdot, b)$ to construct hybrid H_{j+b}, depending on the secret bit b of the oracle. To do so $\mathcal{M}^{\mathcal{B}}_{\text{UD}}$ makes queries to $\mathcal{O}_{\text{UD}}$ with tweaks $\{T_{1,j}, \ldots, T_{l,j}\}$. Then, depending on b, the responses ψ of $\mathcal{O}_{\text{UD}}$ are either l random values or $\psi = (c^{j,1}(\xi_1, 1, Seed), \ldots, c^{j,1}(\xi_l, l, Seed)$, $\xi_i \leftarrow_\$ \{0,1\}^n$, $i \in [1, l]$). After that $\mathcal{M}^{\mathcal{B}}_{\text{UD}}$ requests Seed from the W-SM-UD challenger. Next, $\mathcal{M}^{\mathcal{B}}_{\text{UD}}$ applies the hash chain to the oracle responses ψ to compute samples

$$\phi_i = \begin{cases} c^{j+1,b_i-1-(j+1)}(\psi_i, i, Seed), \text{ if } j < b_i - 1 \\ \xi_i \leftarrow_\$ \{0,1\}^n, \text{ otherwise}, \end{cases}$$

and returns it to $\mathcal{M}^{\mathcal{A}}_{2-3}$. $\mathcal{M}^{\mathcal{B}}_{\text{UD}}$ returns whatever $\mathcal{M}^{\mathcal{A}}_{2-3}$ returns. If ψ consisted of random values the distribution was H_{j+1}, otherwise H_j. Consequently, the advantage of distinguishing any two hybrids

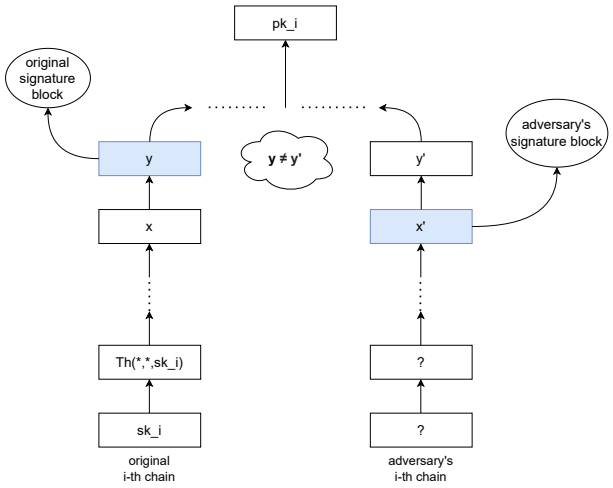

Fig. 1: Example of a case in claim 3

must be bound by $\mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{Th}; \xi, l)$. Putting things together, we see that $b_{\max} \leq w$ for any message M. Hence we get

$$|\mathrm{Succ}^{\mathrm{GAME.2}}(\mathcal{A}) - \mathrm{Succ}^{\mathrm{GAME.3}}(\mathcal{A})| = \mathrm{Adv}_{\mathcal{D}_0, \mathcal{D}_{Kg}}(\mathcal{M}^{\mathcal{A}}_{2-3}) \\ \leq w \cdot \mathrm{Adv}^{\text{W-SM-UD}}_{\mathbf{Th},l}(\mathcal{M}^{\mathcal{B}}_{\mathrm{UD}}) \leq w \cdot \mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{Th}; \xi, l)$$

which concludes the proof of the claim.

Proof of Claim 3. Recall that GAME.4 differs from GAME.3 in that we are considering the game lost if an adversary outputs a valid forgery (M', σ') where there exists i such that $b'_i < b_i$ and $c^{(b'_i, b_i - b'_i)}(\sigma'_i, i, \mathrm{Seed}) \neq \sigma_i$. So the difference in success probability is exactly the probability that $\mathcal{A}$ outputs a valid forgery and there exists an i such that $b'_i < b_i$ and $c^{(b'_i, b_i - b'_i)}(\sigma'_i, i, \mathrm{Seed}) \neq \sigma_i$. We will now prove Claim 3 which claims the following bound on this probability:

Claim 3. $|\mathrm{Succ}^{\mathrm{GAME.3}}(\mathcal{A}) - \mathrm{Succ}^{\mathrm{GAME.4}}(\mathcal{A})| \leq \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{Th}; \widetilde{t}, lw)$.

Proof. To prove the claim we construct an algorithm $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ that reduces W-SM-TCR of **Th** to the task of forging a signature that fulfills the above condition. The algorithm is based on the following idea. $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ simulates GAME.4. In GAME.4 the adversary sends a query to sign a message M. To answer this query and compute the public key, $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ interacts with the W-SM-TCR oracle. This way, $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ obtains target-collision challenges corresponding to the nodes in the signature and all intermediate values of the chain computations made to compute the public key. Then $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ requests the public parameters P from the W-SM-TCR challenger. We set the public seed Seed of WOTS-TW equal to P and return the constructed signature and public key to $\mathcal{A}$. When $\mathcal{A}$ returns a forgery (M', σ'), there exists i such that $b'_i < b_i$ and $c^{(b'_i, b_i - b'_i)}(\sigma'_i, i, \mathrm{Seed}) \neq \sigma_i$ per assumption. By a pigeon hole argument there must be a collision on the way to the public key element. $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ extracts this collision and returns it. Algorithm 2 gives a detailed description of $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ in pseudocode. For the visual representation of the idea described in the mentioned algorithm see Figure 1. The algorithm is broken into two logically separated parts: Challenge placement and obtaining the result.

Here we detail which W-SM-TCR challenges we create per chain in line 11 of Algorithm 2. Assume we have σ_i at position b_i. Then the first query will be (T_{i,b_i}, σ_i). Lets denote the answer for

Algorithm 2: $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$

Input : Security parameter n, oracle access to W-SM-TCR challenger C and EU-naCMA forger $\mathcal{A}$.
Output: A pair (j, M) or fail.

1 **begin** Challenge placement
2 Start $\mathcal{A}$ to obtain query with a message M.
3 Encode M as $B = b_1, \ldots, b_l$ as in signature algorithm.
4 **for** $i \in \{1, \ldots, l\}$ **do**
5 **if** $b_i = 0$ **then**
6 Set $\sigma_i \leftarrow_{\$} \{0,1\}^n$.
7 **else**
8 Sample $\xi_i \leftarrow_{\$} \{0,1\}^n$,
9 Query C for W-SM-TCR challenge with inputs ξ_i, T_{1,b_i-1}.
10 Store answer as σ_i. `// i.e.,` $\sigma_i = \mathbf{Th}(P, T_{i,b_i-1}, \xi_i)$
11 Compute public key element $\mathsf{pk}_i = c^{b_i, w-1-b_i}(\sigma_i, i, \cdot)$ as in the verification algorithm but using the W-SM-TCR challenge oracle provided by C in place of **Th**. `// That is why no` Seed `is needed`
12 Get public parameters P from the challenger and set Seed $= P$.
13 Set signature $\sigma = (\sigma_1, \ldots, \sigma_l)$ and $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_l)$.
14 **begin** Obtaining the result
15 Return σ and $\mathsf{PK} = (\mathsf{pk}, \text{Seed})$ to the adversary $\mathcal{A}$.
16 **if** *The adversary returns a valid forgery* (M', σ') **then**
17 Encode M' as $B' = (b'_1, \ldots, b'_l)$ according to sign.
18 **if** $\exists$ *i such that* $b'_i < b_i$ *and* $c^{(b'_i, b_i - b'_i)}(\sigma'_i, i, \text{Seed}) \neq \sigma_i$ **then**
19 Let j be the smallest integer such that the chains collide: $c^{b_i,j}(y_i, i, \text{Seed}) = c^{b'_i,j}(\sigma'_i, i, \text{Seed}))$.
20 **return** W-SM-TCR *solotion* $(i, c^{b'_i,(j-1)}(\sigma'_i, i, \text{Seed}))$
21 **else**
22 **return** *fail*
23 **else**
24 **return** *fail*

that query as c_1. The next query will be (T_{i,b_i+1}, c_1). We denote the answer for that query as c_2. In general we will make queries of the form (T_{i,b_i+k}, c_k). And we denote the answers for those queries as c_{k+1}. We make queries until we get c_{w-1-b_i}. We set pk_i to be c_{w-1-b_i}.

As we are set to bound the probability of those cases where the adversary outputs a valid forgery and there exists i such that $b'_i < b_i$ and $c^{(b'_i, b_i - b'_i)}(\sigma'_i, i, \text{Seed}) \neq \sigma_i$, $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ never runs into the fail cases in lines 22 and 24. Moreover, the distribution of inputs to $\mathcal{A}$ when run by $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ is identical to that in GAME.4. Therefore, $\mathcal{M}^{\mathcal{A}}_{\mathrm{TCR}}$ returns a target-collision with probability $|\mathrm{Succ}^{\mathrm{GAME.3}}(\mathcal{A}) - \mathrm{Succ}^{\mathrm{GAME.4}}(\mathcal{A})|$ which concludes the proof of the claim.

Proof of Claim 4. It remains to prove the last claim. Consider a forgery σ' and the positions b'_i of the σ' elements. There must exist a j such that $b'_j < b_j$ by the properties of the checksum. Remember that in GAME.4, the case where $c^{(b'_j, b_j - b'_j)}(\sigma'_j, j, \text{Seed}) \neq \sigma_j$ is excluded for all such j. Hence, it must hold for these j that $c^{(b'_j, b_j - b'_j)}(\sigma'_j, j, \text{Seed}) = \sigma_j$. Therefore, we can use $\mathcal{A}$ to compute a preimage of σ_j. We use this to prove Claim 4.

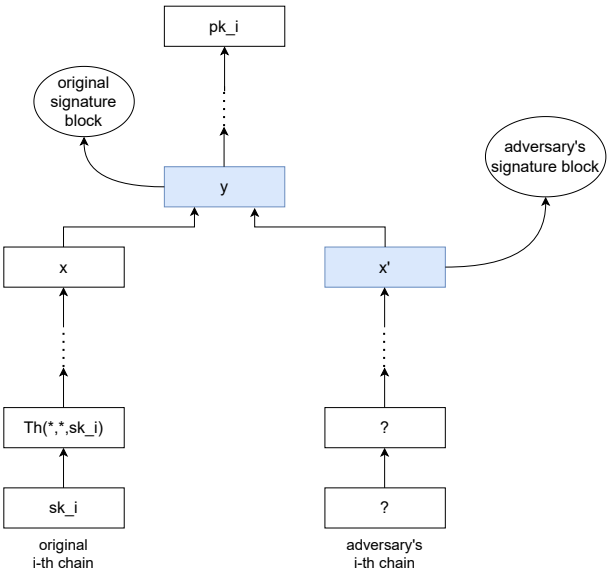

Fig. 2: Example of a case in Claim 4.

Claim 4. $\mathrm{Succ}^{\mathrm{GAME.4}}(\mathcal{A}) \leq \mathrm{InSec}^{\text{W-SM-PRE}}(\mathbf{Th}; \widetilde{t}, l)$.

Proof. As for the previous claim, we construct an algorithm $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ that uses a forger in GAME.4 to solve a W-SM-PRE challenge. In the beginning, $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ receives a query to sign a message M from the adversary $\mathcal{A}$ and encodes it into b_i's. To answer the query $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ interacts with the W-SM-PRE challenger to receive preimage challenges y_i for tweaks that make the challenges fit into positions b_i. That way, $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ can use the challenges as signature values $\sigma_i = y_i$. Then $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ asks the W-SM-PRE challenger to return public parameters P. Given P, $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ can construct the public key using the recomputation method used in the signature verification algorithm. $\mathcal{M}^{\mathcal{A}}$ sets the public seed Seed of WOTS-TW to be P and returns the constructed signature and public key to $\mathcal{A}$. When $\mathcal{A}$ returns a valid forgery, this forgery must contain a signature value σ_j with index j such that $b'_j < b_j$ and $c^{b'_j,(b_j-b'_j)}(\sigma'_j, j, \mathrm{Seed}) = \sigma_j$ per definition of the game. $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ returns preimage $(j, c^{b'_j,(b_j-b'_j-1)}(\sigma'_j, j, \mathrm{Seed}))$. A pseudocode version of $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ is given as Algorithm 3. For a visual representation of the ideas described in the Algorithm 3 see Figure 2. The algorithm is broken into two logically separated parts: Challenge placement and obtaining the result.

Due to the properties of GAME.4, $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ succeeds whenever $\mathcal{A}$ succeeds, as the failure case in line 19 never occurs when $\mathcal{A}$ succeeds. Moreover, the distribution of the inputs to $\mathcal{A}$ when run by $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ is identical to that in GAME.4 (this was ensured in the game hop to GAME.3). Therefore, $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$ returns preimages with probability $\mathrm{Succ}^{\mathrm{GAME.4}}(\mathcal{A})$ which proves the claim.

5 Extension to multiple instances with same public seed

One-time signatures are often used in more complex constructions. Indeed, WOTS-TW was developed as part of SPHINCS$^+$. The distinguishing feature of this setting is that many WOTS-TW instances are used within one instance of the complex construction. In this section we will show that we can base the security of multiple WOTS-TW instances on the same multi-target security properties used for a single instance. While, obviously, the number of targets increases, we argue

Algorithm 3: $\mathcal{M}^{\mathcal{A}}_{\mathrm{PRE}}$

Input : Security parameter n, access to W-SM-PRE challenger C and forger $\mathcal{A}$.
Output: A pair (j, M) or fail.

begin Challenge placement
  Run $\mathcal{A}$ to receive initial query for a signature on message M.
  Encode M as $B = b_1, \ldots, b_l$ following the steps in the signature algorithm.
  for $1 \leq i \leq l$ **do**
    if $b_i > 0$ **then**
      Query C for preimage challenge y_i with tweak T_{1,b_i-1}. // $y_i = \mathbf{Th}(P, T_{i,b_i-1}, \xi_i)$
    else
      $y_i \leftarrow_{\$} \{0,1\}^n$.
    Set $\sigma_i = y_i$.
  Get the seed P from C and set Seed $= P$.
  Compute public key $\mathsf{pk} = (\mathsf{pk}_1, \ldots \mathsf{pk}_l)$, as $\mathsf{pk}_i = c^{w-1-b_i}(y_i, i, \mathrm{Seed})$.
begin Obtaining the result
  Return σ and $\mathsf{PK} = (\mathsf{pk}, P)$ to $\mathcal{A}$.
  if $\mathcal{A}$ *returns a valid forgery* (M', σ') **then**
    Compute $B' = (b'_1, \ldots, b'_l)$ encoding M'
    if $\exists 1 \leq j \leq l$ *such that* $b'_j < b_j$ *and* $c^{(b'_j, b_j - b'_j)}(\sigma'_j, j, \mathrm{Seed}) = \sigma_j$ **then**
      return W-SM-PRE *solution* $(j, c^{b'_j, (b_j - b'_j - 1)}(\sigma'_j, j, \mathrm{Seed}))$
    else
      return *fail*
  else
    return *fail*

in Section 7 that the complexity of generic attacks is not influenced by the number of targets for these notions. Hence, there is no decrease in security to be expected when using multiple instances. We will show that this even works when the same public seed is used for all instances, as long as different prefixes are used for the tweaks.

In SPHINCS-like constructions WOTS-TW is used to sign the roots of trees which are not controlled by an adversary against the construction but by the signer. More generally, this is the case in many such constructions. Hence we use an extension of the EU-naCMA model from last section to d instances. Below, we define EU-naCMA security for d instances of WOTS-TW with respect to a collection of THFs. Our definition is non-generic but tailored to WOTS-TW and the way it is used within SPHINCS$^+$ and other constructions. The reason is that in these settings the THF used in WOTS-TW is a member of the collection of THFs used in the construction and uses the same public parameters. We could have introduced a generic model for this but this would have required the introduction of further abstractions that would unnecessarily complicate the presentation.

The security of multiple WOTS-TW instances is analyzed using the following experiment. In this experiment a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, A_2)$ is allowed to make signing queries to a signing oracle WOTS-TW.sign$(\cdot, (Seed, \cdot, \mathcal{S}))$ and THF oracle $\mathbf{Th}_\lambda$. The signing oracle takes as inputs a message M and address **ADRS**. First it runs $(\mathsf{SK}, \mathsf{PK}) \leftarrow$ WOTS-TW.kg$(\mathcal{C} = (Seed, \mathbf{ADRS}); \mathcal{S})$. Then it computes $\sigma \leftarrow$ WOTS-TW.sign$(M; \mathsf{SK})$. By PK' we denote PK without $Seed$, i.e. $\mathsf{PK}' = (\mathsf{pk}, \mathbf{ADRS})$. The signing oracle returns $(\sigma, M, \mathsf{PK}')$ to the adversary. We also restrict $\mathcal{A}$ from querying $\mathbf{Th}_\lambda$ with tweaks for **ADRSs** that are used in signature queries. We define a function $\mathsf{adrs}(\cdot)$ that takes a tweak as an input and returns **ADRS** of that tweak. We denote the set of queries to signing oracle

as $Q = \{(M_i, \mathbf{ADRS}_i)\}_{i=1}^{d}$ and the set of tweaks that are used to query $\mathbf{Th}_\lambda$ is $T = \{T_i\}_{i=1}^{p}$. We are concerned with one-time signatures, so the number of allowed signing queries for each **ADRS** is restricted to 1.

Experiment $\mathsf{Exp}^{\mathsf{d\text{-}EU\text{-}naCMA}}_{\mathsf{WOTS\text{-}TW}}(\mathcal{A})$

- $Seed \leftarrow_{\$} \{0,1\}^n$
- $\mathcal{S} \leftarrow_{\$} \{0,1\}^n$
- $state \leftarrow \mathcal{A}_1^{\text{WOTS-TW.sign}(\cdot,(Seed,\cdot,\mathcal{S})),\mathbf{Th}_\lambda(Seed,\cdot,\cdot)}(\,)$
- $(M^\star, \sigma^\star, j) \leftarrow \mathcal{A}_2(state, Seed)$
- Return 1 iff $j \in [1,d] \wedge [\text{Vf}(\mathsf{PK}_j, \sigma^\star, M^\star) = 1] \wedge [M^\star \neq M_j] \wedge [\mathbf{DIST}(\{\mathbf{ADRS}_i\}_{i=1}^{d})]$ $\wedge[\forall \mathbf{ADRS}_i \in Q, \mathbf{ADRS}_i \notin T' = \{\mathsf{adrs}(\mathsf{T_i})\}_{i=1}^{p}]$.

We define the success probability of an adversary $\mathcal{A}$ in the described experiment with d instances as

$$\text{Succ}^{\mathsf{d\text{-}EU\text{-}naCMA}}_{\mathsf{WOTS\text{-}TW},d}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr\left[\mathsf{Exp}^{\mathsf{d\text{-}EU\text{-}naCMA}}_{\mathsf{WOTS\text{-}TW}}(\mathcal{A}) = 1\right].$$

The following theorem can be proven by generalization of the proof of Theorem 1. The main idea behind the proof is the following. First of all we use different tweaks in different instances of WOTS-TW as we use different **ADRSs** for each instance. Next point is that we obtain d times more challenges and we separate them in d sets. Each set will be used for one instance of WOTS-TW. Then the proof follows the same path as in Theorem 1.

Theorem 2. *Let n, $w \in \mathbb{N}$ and $w = poly(n)$. Let $\mathbf{F} := \mathbf{Th}_1 : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a* W-SM-TCR, W-SM-PRE, W-SM-UD THF *as a member of a collection. Let $\mathbf{PRF} : \mathcal{S} \times \mathcal{T} \to \{0,1\}^n$ be a* KHF. *Then the following inequality holds:*

$$\begin{aligned}\text{InSec}^{\mathsf{d\text{-}EU\text{-}naCMA}}(\mathsf{WOTS\text{-}TW}; t, d) < \\ \text{InSec}^{\text{PRF}}(\mathbf{PRF}; \widetilde{t}, d \cdot l) + \text{InSec}^{\text{W-SM-TCR}}(\mathbf{F} \in \mathbf{Th}; \widetilde{t}, d \cdot lw) + \\ \text{InSec}^{\text{W-SM-PRE}}(\mathbf{F} \in \mathbf{Th}; \widetilde{t}, d \cdot l) + w \cdot \text{InSec}^{\text{W-SM-UD}}(\mathbf{F} \in \mathbf{Th}; \widetilde{t}, d \cdot l) \quad (3)\end{aligned}$$

with $\widetilde{t} = t + d \cdot lw$, where time is given in number of **Th** *and* **PRF** *evaluations.*

Proof sketch. Let us give a brief description how the proof for the multi-instance case is obtained. We have the same game hopping as in Theorem 1.

GAME.1 is the original d-EU-naCMA game and GAME.2 is the same as GAME.1 but the pseudorandom outputs from **PRF** are replaced by truly random values. We claim that $|\text{Succ}^{\text{GAME.1}}(\mathcal{A}) - \text{Succ}^{\text{GAME.2}}(\mathcal{A})| \leq \text{InSec}^{\text{PRF}}(\mathbf{PRF}; \widetilde{t}, d \cdot l)$. The reasoning here is the same as in Claim 1 in Theorem 1. Note that all inputs on which the oracle in the PRF game is queried are unique due to the unique **ADRSs** for each instance. Hence, the outputs are uniformly random values as desired.

GAME.3 is different from GAME.2 in that for each signing query we answer with a hash of a random value rather than building it with a chaining function. In Claim 2 of Theorem 1 we reduced it to the W-SM-UD property by using a hybrid argument. Here we need to apply the same reasoning. To obtain the needed hybrids in case of d instances we will do the following. We use an additional index to denote the B-values associated with the i-th message M_i. So M_i is transferred into $b_{i,1}, \ldots, b_{i,l}$. We now consider the d-fold distributions $D_{d\text{-}Kg} = \{y_{1,1}, \ldots, y_{1,l}, \ldots, y_{d,1}, \ldots, y_{d,l}\}$, where $y_{i,j} = c^{0,b_j-1}_{\mathbf{ADRS}_i}(\xi_{i,j}, j, \text{Seed})$ and $D_{d\text{-}0} = \{\xi_{1,1}, \ldots, \xi_{1,l}, \ldots, \xi_{d,1}, \ldots, \xi_{d,l}\}$, where $\xi_{i,j} \leftarrow_{\$} \{0,1\}^n$, $i \in [1,d]$, $j \in [1,l]$. The distinguishing advantage of an adversary against those two distributions is exactly the difference of these two games. To limit this distinguishing advantage we need to build hybrids. We do this in the same manner as in the proof of Theorem 1. Let $b_{\max} = \max\{b_{1,1}, \ldots, b_{1,l} \ldots, b_{d,1}, \ldots, b_{d,l}\}$ be the maximum of the values in the message encoding of all M_i. Let H_k be the distribution obtained

by computing the values as $c^{k,b_{i,j}-1-k}_{\mathbf{ADRS}_i}(\xi_{i,j}, j, Seed)$, $\xi_{i,j} \leftarrow_\$ \{0,1\}^n$. One can notice that $H_0 = D_{Kg}$ and $H_{b_{\max}-1} = D_0$. There must be two consecutive hybrids H_γ and $H_{\gamma+1}$ that we can distinguish with probability close to the distinguishing advantage. By playing W-SM-UD and interacting with the oracle $\mathcal{O}(\cdot, b)$ we can construct hybrid $H_{\gamma+b}$. This is done in just the same way as in Claim 2 of Theorem 1. Hence we obtain the following bound:

$$|\mathrm{Succ}^{\mathrm{GAME.2}}(\mathcal{A}) - \mathrm{Succ}^{\mathrm{GAME.3}}(\mathcal{A})| \leq w \cdot \mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{F} \in \mathbf{Th}; \widetilde{t}, d \cdot l)\,.$$

Notice that in case of one instance we obtained Seed from the W-SM-UD challenger that we used to construct the hybrids and obtain the WOTS-TW public key. Here instead of using Seed we need to interact with the $\mathbf{Th}_\lambda(\mathrm{Seed}.\cdot, \cdot)$ oracle. Only after all of the signing queries are made we will obtain the Seed.

GAME.4 is different from GAME.3 in that we are considering the game lost if an adversary outputs a valid forgery $(M^\star, \sigma^\star, j)$ where there exist such i that $b^\star_{i,j} < b_{i,j}$ and $c^{(b^\star_{i,j}, b_{i,j}-b^\star_{i,j})}_{\mathbf{ADRS}_i}(\sigma^\star_j, j, \mathrm{Seed}) \neq \sigma_j$. To show the bound we can build a reduction that works as follows. To answer the signature queries and compute the public key, the reduction interacts with the W-SM-TCR oracle. The difference in case of d instances from one instance is that we will need d times more interactions with the W-SM-TCR oracle. Per assumption, the must exist at least one chain such that the chain that we built and the chain obtained from the forged signature are different but lead to the same public key. Hence by a pigeon hole argument there must be a collision on the way to the public key element. This collision is a solution for the W-SM-TCR challenge. So we proved that

$$|\mathrm{Succ}^{\mathrm{GAME.3}}(\mathcal{A}) - \mathrm{Succ}^{\mathrm{GAME.4}}(\mathcal{A})| \leq \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{H} \in \mathbf{Th}; \widetilde{t}, d \cdot lw)\,.$$

To give a bound on the success probability for GAME.4 we use the W-SM-PRE property. To answer signing queries we will interact with the W-SM-PRE oracle and place challenges obtained from that oracle in place of signatures. To construct public keys of WOTS-TW instances we will behave in the same way as in the undetectability case. By interacting with $\mathbf{Th}_\lambda(\mathrm{Seed}, \cdot, \cdot)$ we can build the chains of WOTS-TW structures. Again there must exist a j such that $b^\star_{i,j} < b_{i,j}$ by the properties of the checksum. And since we excluded the case where $c^{(b^\star_{i,j}, b_{i,j}-b^\star_{i,j})}_{\mathbf{ADRS}_i}(\sigma^\star_j, j, \mathrm{Seed}) \neq \sigma_j$ we can obtain a preimage by computing $c^{(b^\star_{i,j}-1, b_{i,j}-b^\star_{i,j})}_{\mathbf{ADRS}_i}(\sigma^\star_j, j, \mathrm{Seed})$. So we obtain

$$|\mathrm{Succ}^{\mathrm{GAME.4}}(\mathcal{A})| \leq \mathrm{InSec}^{\text{W-SM-PRE}}(\mathbf{F} \in \mathbf{Th}; \widetilde{t}, d \cdot l).$$

This concludes the sketch of the proof.

6 SPHINCS$^+$

In this section we will recap the SPHINCS$^+$ structure and afterwards give fixes to the original SPHINCS$^+$ proof. To obtain a fixed proof we will utilize the results from Theorem 2. In SPHINCS$^+$ a special function to compute message digest is introduced. An expected property of that function is interleaved target subset-resilience. The formal definition of this property is given in Appendix A. The part of the proof where we use this property is the same as in the SPHINCS$^+$ paper [BHK+19]. Hence we will not discuss it in details.

6.1 Brief description

First we give a brief description of the SPHINCS$^+$ signature scheme. An example of the SPHINCS$^+$ structure is shown in Figure 3. A detailed description can be found in [BHK+19]. The public key

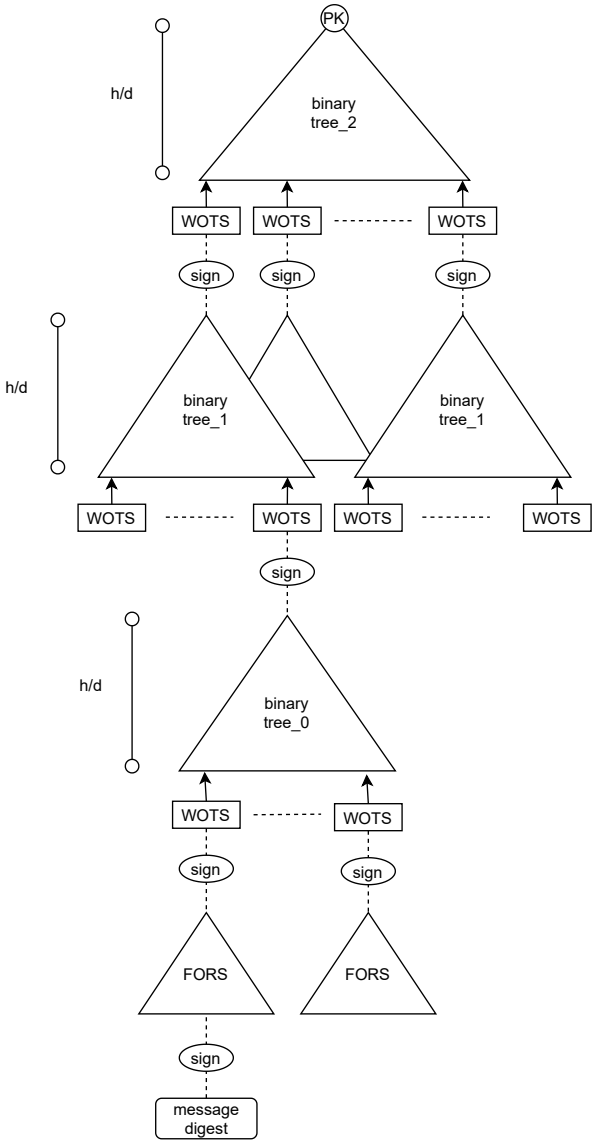

Fig. 3: Example of a SPHINCS$^+$ structure

consists of two n-bit values: a random public seed **PK**.seed and the root of the top tree in the hypertree structure. **PK**.seed is used as a first argument for all of the tweakable hash functions calls. The private key contains two more n-bit values **SK**.seed and **SK**.prf. We discuss the main parts of SPHINCS$^+$. First we describe the addressing scheme. As SPHINCS$^+$ uses THFs, different tweaks are required for all calls to THFs. The tweaks are instantiate by the addresses. The address is a 32 byte value. Address coding can be done in any convenient way. Each address has a prefix that denotes to which part of the SPHINCS$^+$ structure it belongs. We denoted this prefix as **ADRS** in previous sections.

Then we need to discuss binary trees. In the SPHINCS$^+$ algorithm, binary trees of height γ always have 2^γ leaves. Each leaf $L_i,\ i \in [0, 2^\gamma - 1]$ is a bit string of length n. Each node of the tree

$N_{i,j},\ 0 < j \leq \gamma, 0 \leq i < 2^{\gamma - j}$ is also a bit string of length n. The values of the internal nodes of the tree are calculated from the children of that node using a THF. A leaf of a binary tree is the output of a THF that takes the elements of a WOTS-TW public key as input.

Binary trees and WOTS-TW signature schemes are used to construct a hypertree structure. WOTS-TW instances are used to sign the roots of binary trees on lower levels. WOTS-TW instances on the lowest level are used to sign the public key of a FORS few-time signature scheme instance. FORS is defined with the following parameters: $k \in \mathbb{N}$, $t = 2^a$. This algorithm can sign message digests of length ka-bits.

FORS key pair. The private key of FORS consists of kt pseudorandomly generated n-bit values grouped into k sets of t elements each. To get the public key, k binary hash trees are constructed. The leaves in these trees are k sets (one for each tree) which consist of t values, each. Thus we get k trees of height a. As roots of k binary trees are calculated they are compressed using a THF. The resulting value will be the FORS public key.

FORS Signature. A message of ka bits is divided into k lines of a bits. Each of these lines is interpreted as a leaf index corresponding to one of the k trees. The signature consists of these leaves and their authentication paths. An authentication path for a leaf is the set of siblings of the nodes on the path from this leaf to the root. The verifier reconstructs the tree roots, compresses them, and verifies them against the public key. If there is a match, it is said that the signature was verified. Otherwise, it is declared invalid.

The last thing to discuss is the way the message digest is calculated. First, a pseudorandom value $\mathbf{R}$ is prepared as $\mathbf{R} = \mathbf{PRF}_{msg}(\mathbf{SK}_{\mathsf{prf}}, \mathsf{OptRand}, M)$ using a dedicated secret key element $\mathbf{SK}.\mathsf{prf}$ and the message. This function can be made non-deterministic initializing the value $\mathsf{OptRand}$ with randomness. The $\mathbf{R}$ value is part of the signature. Using $\mathbf{R}$, we calculate the index of the FORS key pair with which the message will be signed and the message digest itself: $(\mathsf{MD}||\mathsf{idx}) = \mathbf{H_{msg}}(\mathbf{R}, \mathbf{PK}.\mathsf{seed}, \mathbf{PK}.\mathsf{root}, M)$.

The signature consists of the randomness $\mathbf{R}$, the FORS signature (under idx from $\mathbf{H_{msg}}$) of the message digest, the WOTS-TW signature of the corresponding FORS public key, and a set of authentication paths and WOTS-TW signatures of tree roots. To test this chain, the verifier iteratively reconstructs the public keys and tree roots until it gets the root of the top tree. If this matches the root given in the SPHINCS$^+$ public key, the signature is accepted.

6.2 SPHINCS$^+$ proof

In this part we fix the proof of security of the SPHINCS$^+$ framework. The security had several issues which are described in [KKF20, ABB+20]. The SPHINCS$^+$ construction uses the following functions:

$\mathbf{F} := \mathbf{Th}_1 : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$; $\quad \mathbf{H} := \mathbf{Th}_2 : \mathcal{P} \times \mathcal{T} \times \{0,1\}^{2n} \to \{0,1\}^n$;
$\mathbf{Th}_l : \mathcal{P} \times \mathcal{T} \times \{0,1\}^{ln} \to \{0,1\}^n$; $\quad \mathbf{Th}_k : \mathcal{P} \times \mathcal{T} \times \{0,1\}^{kn} \to \{0,1\}^n$;
$\mathbf{PRF} : \{0,1\}^n \times \{0,1\}^{256} \to \{0,1\}^n$; $\quad \mathbf{PRF_{msg}} : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$;
$\mathbf{H_{msg}} : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$.

In this section, we prove the following Theorem about the standard EU-CMA-security (for a definition see Appendix B) of SPHINCS$^+$. Note that $\mathbf{F}$, $\mathbf{H}$, $\mathbf{Th}_l$, and $\mathbf{Th}_k$ are members of a collection $\mathbf{Th}$ of tweakable hash functions with different message lengths.

Theorem 3. *For parameters n, w, h, d, m, t, k as described in [BHK+19] and l be the number of chains in WOTS-TW instances the following bound can be obtained:*

$$\begin{aligned}
&\mathrm{InSec}^{\mathrm{EU-CMA}}(\mathrm{SPHINCS}^+; \xi, q_s) \leq \\
&\mathrm{InSec}^{\mathrm{PRF}}(\mathbf{PRF}, \xi, q_1) + \mathrm{InSec}^{\mathrm{PRF}}(\mathbf{PRF_{msg}}, \xi, q_s) + \\
&\mathrm{InSec}^{\mathrm{ITSR}}(\mathbf{H_{msg}}, \xi, q_s) + w \cdot \mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{F} \in \mathbf{Th}; \xi, q_2) + \\
&\mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{F} \in \mathbf{Th}; \xi, q_3 + q_7) + \mathrm{InSec}^{\text{W-SM-PRE}}(\mathbf{F} \in \mathbf{Th}; \xi, q_2) + \\
&\mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{H} \in \mathbf{Th}; \xi, q_4) + \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{Th}_k \in \mathbf{Th}; \xi, q_5) + \\
&\mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{Th}_l \in \mathbf{Th}; \xi, q_6) + \\
&3 \cdot \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{F} \in \mathbf{Th}; \xi, q_8) + \mathrm{InSec}^{\text{W-SM-DSPR}}(\mathbf{F} \in \mathbf{Th}; \xi, q_8)\,,
\end{aligned}$$

where $q_1 < 2^{h+1}(kt + l)$, $q_2 < 2^{h+1} \cdot l$, $q_3 < 2^{h+1} \cdot l \cdot w$, $q_4 < 2^{h+1}k \cdot 2t$, $q_5 < 2^h$, $q_6 < 2^{h+1}$, $q_7 < 2^{h+1}kt$, $q_8 < 2^h \cdot kt$ and q_s denotes the number of signing queries made by $\mathcal{A}$.

Proof. We want to bound the success probability of a (quantum) adversary $\mathcal{A}$ against the EU-CMA security of SPHINCS$^+$. Towards this end we use the following series of games. We start with GAME.0 which is the EU-CMA experiment for SPHINCS$^+$. Now consider a GAME.1 which is essentially GAME.0 but the experiment makes use of a SPHINCS$^+$ version where all the outputs of PRF, i.e., the WOTS-TW and FORS secret-key elements, get replaced by truly random values.

Next, consider a game GAME.2, which is the same as GAME.1 but in the signing oracle $\mathbf{PRF_{msg}}(\mathrm{SK.prf}, \cdot)$ is replaced by a truly random function.

Afterwards, we consider GAME.3, which differs from GAME.2 in that we are considering the game lost if an adversary outputs a valid forgery $(\mathrm{M}, \mathrm{SIG})$ where the FORS signature part of SIG contains only secret values which were contained in previous signatures with that FORS key pair obtained by $\mathcal{A}$ via the signing oracle.

Now consider what are the possibilities of the adversary to win the game. The FORS signature in a forgery must include the preimage of a FORS leaf node that was not previously revealed to it. There are two separate cases for that leaf:

1. The FORS leaf is different to the leaf that we would generate for that place.
2. The FORS leaf is the same to the leaf that we would generate for that place;

Lets consider GAME.4 which differs from GAME.3 in that we are considering that the game is lost in the first "leaf case" scenario.

Now lets analyze those games.

GAME.0 - GAME.3 The hops between GAME.0 and GAME.3 are fully presented in the SHINCS+ paper [BHK+19]. The bound for these games are

$$|\mathrm{Succ}_{\mathcal{A}}^{\mathrm{GAME.0}} - \mathrm{Succ}_{\mathcal{A}}^{\mathrm{GAME.1}}| \leq \mathrm{InSec}^{\mathrm{PRF}}(\mathbf{PRF}, \xi, q_1), \tag{4}$$

$$|\mathrm{Succ}_{\mathcal{A}}^{\mathrm{GAME.1}} - \mathrm{Succ}_{\mathcal{A}}^{\mathrm{GAME.2}}| \leq \mathrm{InSec}^{\mathrm{PRF}}(\mathbf{PRF_{msg}}, \xi, q_s), \tag{5}$$

$$|\mathrm{Succ}_{\mathcal{A}}^{GAME.2} - \mathrm{Succ}_{\mathcal{A}}^{GAME.3}| \leq \mathrm{InSec}^{\mathrm{ITSR}}(\mathbf{H_{msg}}, \xi, q_s), \tag{6}$$

where $q_1 < 2^{h+1}(kt + l)$ and q_s denotes the number of signing queries made by $\mathcal{A}$.

GAME.3 - GAME.4 Lets break the hop between GAME.3 and GAME.4 into several steps. Since the FORS leaf is different to the leaf that we would generate for that place there are two possible outcomes. First case is that the forged signature contains a second preimage for some input of a THF. This can occur in the FORS or WOTS-TW instances, the compression of FORS or WOTS-TW public keys, and in the binary trees. And second case that a WOTS-TW forgery occurs.

Consider GAME.3.1 in which the game is lost if there is a second preimage contained in the forged signature for an input of $\mathbf{H}$ in a binary tree. The difference for this case can be bounded by building a W-SM-TCR adversary for $\mathbf{H}$ as a member of a collection. We construct the SPHINCS$^+$ structure using W-SM-TCR challenger for every input to $\mathbf{H}$ and the oracle $\mathbf{Th}_\lambda$ for the rest. Here we also consider binary trees of FORS as part of the challenge. Hence we obtain

$$|\mathrm{Succ}_{\mathcal{A}}^{GAME.3} - \mathrm{Succ}_{\mathcal{A}}^{GAME.3.1}| \leq \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{H} \in \mathbf{Th}; \xi, q_4), \tag{7}$$

where $q_4 < 2^{h+1} \cdot k \cdot 2t$.

Now we introduce GAME.3.2 which is different from GAME.3.1 in that we are considering the game lost if a second preimage for $\mathbf{Th}_k$ is contained in the FORS tree nodes computed while verifying the forged signature. As in the previous case this can be bounded by

$$|\mathrm{Succ}_{\mathcal{A}}^{GAME.3.1} - \mathrm{Succ}_{\mathcal{A}}^{GAME.3.2}| \leq \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{Th}_k \in \mathbf{Th}; \xi, q_5), \tag{8}$$

where $q_5 < 2^h$.

Next the GAME.3.3 is considered lost if a second preimage for $\mathbf{Th}_l$ is contained in the WOTS-TW public keys computed from the forged signature. Following the same ideas as above we obtain

$$|\mathrm{Succ}_{\mathcal{A}}^{GAME.3.2} - \mathrm{Succ}_{\mathcal{A}}^{GAME.3.3}| \leq \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{Th}_l \in \mathbf{Th}; \xi, q_6), \tag{9}$$

where $q_6 < 2^{h+1}$.

The GAME.3.4 is lost if there is a second preimage in the forged signature for some input for $\mathbf{F}$ outside the WOTS-TW instances, i.e., in as a FORS signature value. The bound for this case is

$$|\mathrm{Succ}_{\mathcal{A}}^{GAME.3.3} - \mathrm{Succ}_{\mathcal{A}}^{GAME.3.4}| \leq \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{F} \in \mathbf{Th}; \xi, q_7), \tag{10}$$

where $q_7 < 2^h \cdot k \cdot t$.

So the only case left to hop to GAME.4 is a WOTS-TW forgery for one out of $d < 2^{h+1}$ instances. Using the bound in Theorem 2 we obtain

$$\begin{aligned} |\mathrm{Succ}_{\mathcal{A}}^{GAME.3.4} - \mathrm{Succ}_{\mathcal{A}}^{GAME.4}| \leq\ & \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{F} \in \mathbf{Th}; \xi, q_3) + \\ & \mathrm{InSec}^{\text{W-SM-PRE}}(\mathbf{F} \in \mathbf{Th}; \xi, q_2 + w \cdot \mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{F} \in \mathbf{Th}; \xi, q_2), \end{aligned} \tag{11}$$

where $q_2 < 2^{h+1} \cdot l$, $q_3 < 2^{h+1} \cdot lw$.

GAME.4 The analysis of GAME.4 can be found in the SPHINCS$^+$ paper [BHK+19](Claim 23). Here we note that we cannot use the W-SM-PRE bound as the reduction is an instance of a T-openPRE game as introduced in [BH19], i.e., the reduction needs to know some preimages. The only difference is that we have already excluded the WOTS-TW preimage case. Hence we obtain the following bound:

$$\mathrm{Succ}_{\mathcal{A}}^{GAME.4} \leq 3 \cdot \mathrm{InSec}^{\text{W-SM-TCR}}(\mathbf{F}; \xi, q_8) + \mathrm{InSec}^{\text{W-SM-DSPR}}(\mathbf{F}; \xi, q_8), \tag{12}$$

where $q_8 < 2^h \cdot kt$.

Combining the inequalities we obtain the bound from the theorem.

Table 1: Success probability of generic attacks – In the "Success probability" column we give the bound for a quantum adversary $\mathcal{A}$ that makes q quantum queries to the function and p classical queries to the challenge oracle. The security parameter n is the output length of **Th**. We use $X = \sum_\gamma \left(1 - \left(1 - \frac{1}{t}\right)^\gamma\right)^k \binom{p}{\gamma} \left(1 - \frac{1}{2^h}\right)^{p-\gamma} \frac{1}{2^{h\gamma}}$.

Property	Success probability	Status
W-SM-TCR	$\Theta((q+1)^2/2^n)$	proven ([BHK+19, HRS16])
W-SM-DSPR	$\Theta((q+1)^2/2^n)$	conjectured ([BHK+19])
W-SM-PRE	$\Theta((q+1)^2/2^n)$	based on conjecture ([BH19, BHK+19])
PRF	$\Theta(12q/\sqrt{2^n})$	proven (this work)
W-SM-UD	$\Theta(12q/\sqrt{2^n})$	proven (this work)
ITSR	$\Theta((q+1)^2 \cdot X)$	conjectured ([BHK+19])

7 Analyzing Quantum Generic Security

In this section we collect bounds on the complexity of generic attacks against the properties discussed so far for THFs and KHFs. For definitions of the properties for KHFs see Appendix C. A hash function **Th** is commonly considered a good function if there are no attacks known for any security property that perform better against **Th** than a generic attack against a random function. First we discuss the current situation which is summarized in Table 1. Then we give new proofs for the W-SM-UD and PRF properties. To do so we follow the approach of [HRS16] where different instances of average-case distinguishing problems over boolean functions are reduced to breaking the different hash function security properties. The advantage of this approach is that we know lower bounds for these decision problems, even for quantum algorithms. This allows us to derive lower bounds on the complexity of quantum attacks against our security properties.

7.1 Estimated security

The success probability of generic attacks against W-SM-TCR and a reduction to an average-case search problem was given in [BHK+19]. A generic attack using Grover search against plain TCR is given in [HRS16], which is applicable against W-SM-TCR – as it runs a second preimage search when all information is available – and has a success probability matching the proven bound.

With regard to W-SM-DSPR, two bounds are proven in [BH19]. On the one hand, the bound $O((q+1)^2/2^n)$ is proven for single-target DSPR of a KHF, which is tight. This proof perfectly transfers to the W-SM-DSPR notion of a THF by specifying the tweak we analyze $\mathbf{Th}(P, T, \cdot)$ which can be viewed as a KHF with a fixed key. For a T-target version a factor-T lose bound is obtained via a standard plug'n'pray argument, placing the challenge instance at a random position, hoping that that will be the one that gets distinguished by the adversary. In [BHK+19], the authors conjecture that the actual multi-target bound should be the same as the single-target bound. A supporting argument for this conjecture is that the best attack against multi-target DSPR for now is still a second-preimage search which has the same complexity in both cases.

For PRE of a KHF h, a bound of $\mathrm{Succ}^{\mathrm{PRE}}_{h,p}(\mathcal{A}) = \Theta((q+1)^2/2^n)$ is given in [HRS16] that also holds in a multi-function, multi-target setting. The bound is proven for h that are random and compressing by at least a factor 2 in the message length. It is conjectured that it also applies for length preserving hash functions, i.e., functions that map n-bit messages to n-bit outputs, possibly

taking additional inputs like function keys or tweaks. A bound for W-SM-PRE can be proven using W-SM-TCR and W-SM-DSPR ($\mathrm{Succ}^{\text{W-SM-PRE}}_{\mathbf{Th},p}(\mathcal{A}) \leq 3{\cdot}\mathrm{Succ}^{\text{W-SM-TCR}}_{\mathbf{Th},p}(\mathcal{A}) + \mathrm{Adv}^{\text{W-SM-DSPR}}_{\mathbf{Th},p}(\mathcal{A})$) as shown in [BHK+19, BH19]. With this we derive the same bound of $\mathrm{Succ}^{\text{W-SM-PRE}}_{\mathbf{Th},p}(\mathcal{A}) = \tilde{\Theta}((q+1)^2/2^n)$. For the case of multiple targets, the tight bound needs above conjecture for W-SM-DSPR. A factor-T-loose, unconditional bound follows from the loose bound for W-SM-DSPR.

The success probability of generic attacks against PRF is analyzed in this work. Assuming that $F : S \times \mathcal{T} \to \{0,1\}^n$ behaves like a random function a reduction to a distinguishing problem between a boolean function of weight 0 and a random boolean function of weight 1 is given. From this we obtain a bound for quantum adversaries. Note that this bound is not trivial if $|S| < (2^n)^{|\mathcal{T}|}$.

The notion of undetectability was introduced in [DSS05]. In that work, the authors give a bound for single-target undetectability considering classical adversaries as $\mathcal{O}(q/2^n)$. Below, we give a bound for multi-target undetectability of random **Th** considering quantum adversaries.

For all notions we conjecture that the bounds are exactly the same for the case of collections. The reason is that for a random tweakable function, every tweak is related to an independent random function. Hence, giving access to those does not give any information about the targets to the adversary. This is also reflected in the reductions that we know so far. In these, the function for a tweak that is not used for a challenge is simulated by an independent random function and we can give access to this function in parallel to the challenge oracle as we do not touch it in the reduction.

In Section 8 we discuss properties of KHFs which are similar to ones discussed above. Specifically DM-SPR, DM-UD, DM-PRE, DM-DSPR. In that section we show that it is possible to obtain exactly the same table of success probabilities by replacing W-SM-TCR with DM-SPR, W-SM-DSPR with DM-DSPR, W-SM-PRE with DM-PRE, and W-SM-UD with DM-UD. In the following subsections we give proofs for the PRF and W-SM-UD properties.

7.2 Decision Problem

Here we define a distinguishing problem over boolean functions for which an optimal query complexity bound is known. In our reductions to show lower bounds, we assume we have access to some random functions G and g. Hence, we need to simulate G and g efficiently so that any algorithm with q queries cannot notice a difference. According to [Zha12] this can be simulated using 2q-wise independent hash functions or QPRFs.

Assume we have a family $\mathcal{F}$ of all n-bit boolean functions. We will call the weight of a boolean function f the result of the following function: $wt(f) = |\{x : f(x) = 1\}|$. Lets denote $S_i = \{f \in \mathcal{F} | wt(f) = i\}$. We define the distinguishing advantage for two sets S_i and S_j as

Definition 8 (Dist-i,j). *Let S_i be as defined above. We define the distinguishing advantage between* $\mathrm{Adv}^{S_i,S_j}_q(\mathcal{A}) \overset{def}{=} \left| \Pr_{f \leftarrow_\$ S_i}\left[\mathcal{A}^f(\cdot) = 1\right] - \Pr_{f \leftarrow_\$ S_j}\left[\mathcal{A}^f(\cdot) = 1\right] \right|.$

According to the results from Theorem 9.3.2 [KLM06] one can derive the following lemma.

Lemma 1 ([KLM06]). *Let S_i be as defined above. The advantage of any q query quantum algorithm in distinguishing S_0 from S_1 is bounded by* $\mathrm{Adv}^{S_0,S_1}_q(\mathcal{A}) \leq 6q/\sqrt{2^n}$.

7.3 PRF

First we analyze the PRF property. We assume that our function $F : S \times \mathcal{T} \to \{0,1\}^n$ behaves like a random function. The PRF property assumes that we can substitute $F(s,\cdot)$ for $s \leftarrow_\$ S$ with a random function. It is important to see that this substitution in theory can still be noticed, since the adversary has access to unchanged function $F(\cdot,\cdot)$. So we need to show that detecting the change

Algorithm 4: Dist-0,1 to PRF

Input : f, PRF adversary $\mathcal{A}$
Output: $b' \in \{0,1\}^n$

1 Construct a random function $F : S \times \mathcal{T} \to \{0,1\}^n$ using random function $g : \mathcal{T} \to \{0,1\}^n$ and a random function $G : S \times \mathcal{T} \to \{0,1\}^n$ the following way:

2
$$s \times t \to \begin{cases} g(t) & \text{if } f(s) = 1 \\ G(s,t) & \text{otherwise} \end{cases}$$

3 **return** $\mathcal{A}^{g,F}()$

has a negligible chance. To do so we will construct an algorithm that, given a boolean function from some distribution, constructs a function which is distributed the same as a random function. Breaking the PRF property of the function will lead to solving the decision problem of the boolean function. Hence we obtain an upper bound on breaking the PRF property for a random function.

Lemma 2. *Let $n \in \mathbb{N}$, $F : S \times \mathcal{T} \to \{0,1\}^n$ - a random function. Any quantum adversary $\mathcal{A}$ that solves* PRF *making q quantum queries to F can be used to construct a quantum adversary $\mathcal{B}$ that makes $2q$ queries to its oracle and distinguishes S_0 from S_1 with advantage*

$$\mathrm{Adv}^{\mathrm{PRF}}_{F,q}(\mathcal{A}) \leq 12q/\sqrt{2^n}\,.$$

Proof. To prove the bound we show that $\mathrm{Adv}^{\mathrm{PRF}}_{F,q}(\mathcal{A}) = \mathrm{Adv}^{S_0,S_1}_{2q}(\mathcal{B})$. The bound in the lemma follows then from Lemma 1. Assume we want to distinguish between S_0 and S_1. So we obtain a boolean function $f : S \to \{0,1\}$ chosen from either S_0 or S_1 with probability $1/2$, each. We give a reduction of the distinguishing problem to the PRF property in Algorithm 4.

Let's analyze the work of the reduction. Assume that $f \leftarrow S_1$ then g is an element of F. Hence, $\mathcal{A}$'s view is the same as in the case of choosing g by picking a random s and setting $g := F(s,\cdot)$. In case $f \leftarrow D_0$, g is a random function, independent of F. This is identical to the case where $\mathcal{A}$ is given access to a random function instead of an element of F. Referring to Algorithm 4 as quantum adversary $\mathcal{B}$ we obtain the following bound:

$$\begin{aligned}\mathrm{Adv}^{S_0,S_1}_{2q}(\mathcal{B}) &= |\Pr_{f\leftarrow_\$ S_0}[\mathcal{B}^f() = 1] - \Pr_{f\leftarrow_\$ S_1}[\mathcal{B}^f() = 1]| \\ &= |\Pr_{f\leftarrow_\$ S_0}[\mathcal{A}^{g,F}() = 1] - \Pr_{f\leftarrow_\$ S_1}[\mathcal{A}^{g,F}() = 1]| = \mathrm{Adv}^{\mathrm{PRF}}_{F,q}(\mathcal{A})\,.\end{aligned}$$

The reason that the number of queries doubles is that $\mathcal{B}$ has to possibly uncompute garbage to run in acceptable space. We conclude that $\mathrm{Adv}^{\mathrm{PRF}}_{F,q}(\mathcal{A}) = \mathrm{Adv}^{S_0,S_1}_{2q}(\mathcal{A}) \leq 12q/\sqrt{2^n}$.

7.4 W-SM-UD

In this section we analyze the W-SM-UD property. The analysis for a single-target case can be found in Appendix E. We follow the same approach for the multi-target case. In our reduction we need sets S^l_0 and S^l_1. S^l_i will contain all functions $f : [1,l] \times \{0,1\}^n \to \{0,1\}^n$. Where $f(j,\cdot),\ j \in [1,l]$ is a random function from S_i. We will now show that distinguishing $f \leftarrow_\$ S^l_1$ from $f \leftarrow_\$ S^l_0$ is as hard as distinguishing $f \leftarrow_\$ S_1$ from $f \leftarrow_\$ S_0$.

Algorithm 5: Dist-1,0 to W-SM-UD

Input : f, W-SM-UD adversary $\mathcal{A}$
Output: $b' \in \{0,1\}^n$

Choose random values $y_1, \dots y_l \leftarrow_\$ \{0,1\}^n$ and a public parameter $P \leftarrow_\$ \mathcal{P}$
Get tweaks $T_1, \dots, T_l$ from adversary $\mathcal{A}_1$
Answer the queries with $y_1, \dots y_l$
Construct a random tweakable hash function $H : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ using random function $F : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ the following way:

$$H(p,t,x) : \begin{cases} \text{if } (p = P, t = T_i, f(i,x) = 1) : \text{ Return } y_i \\ \text{Return } F(p,t,x) \end{cases}$$

Give oracle access to H and public parameter P to adversary $\mathcal{A}_2$
return Output of $\mathcal{A}_2$

Lemma 3. *Consider sets* $S_0,\ S_1,\ S_0^l, S_1^l$ *as defined above. Then* $\mathrm{Adv}_q^{S_0,S_1}(\mathcal{A}) = \mathrm{Adv}_q^{S_0^l,S_1^l}(\mathcal{A})$.

Proof. Assume we can distinguish $f \leftarrow_\$ S_1$ from $f \leftarrow_\$ S_0$ with some algorithm $\mathcal{A}$. Then to distinguish $f \leftarrow_\$ S_1^l$ from $f \leftarrow_\$ S_0^l$ we run $\mathcal{A}$ on $f(1,\cdot)$. Hence $\mathrm{Adv}_q^{S_0,S_1}(\mathcal{A}) \le \mathrm{Adv}_q^{S_0^l,S_1^l}(\mathcal{A})$.

To show equality we now give the reduction in the opposite direction. Assume we can distinguish $f \leftarrow_\$ S_1^l$ from $f \leftarrow_\$ S_0^l$. Our task is to distinguish $f' \leftarrow_\$ S_1$ from $f' \leftarrow_\$ S_0$. To build f from f' we can sample $z_i \leftarrow_\$ \{0,1\}^n$, $i \in [1,l]$, and set $f(i,x) \stackrel{\text{def}}{=} f'(x \oplus z_i)$. One can see that if f' was a constant zero function then f is a collection of constant zero functions, so $f \in S_0^l$. If $f' \in S_1$ then $f(i,\cdot)$ outputs 1 for one random value since z_i were chosen uniformly at random, so $f \in S_1^l$. Hence $\mathrm{Adv}_q^{S_0,S_1}(\mathcal{A}) \ge \mathrm{Adv}_q^{S_0^l,S_1^l}(\mathcal{A})$ which concludes the proof.

Along the lines of the single-target case we prove the following theorem:

Lemma 4. *Let* $n \in \mathbb{N}$, $H : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *- a random hash function. Any quantum adversary* $\mathcal{A}$ *that solves* W-SM-UD *for* p *targets making* q *queries to* H *can be used to construct a quantum adversary* $\mathcal{B}$ *that makes* $2q$ *queries to its oracle and distinguishes* S_0 *from* S_1 *with an advantage*

$$\mathrm{Adv}_{H,p}^{\text{W-SM-UD}}(\mathcal{A}) \le 12q/\sqrt{2^n}\,.$$

Proof. We give a reduction that distinguishes S_0^p from S_1^p. The lemma follows then by applying Lemmas 1 and 3. Assume we obtain a function f either from S_0^p or from S_1^p. We build the reduction shown in Algorithm 5. As in the single-target case we can see that for any f we construct a truly random tweakable hash function. If $f \in S_0^p$ we answer the adversary with random values. If $f \in S_1^p$ we answer the queries with outputs of the hash function on randomly chosen inputs. Referring to Algorithm 5 as quantum adversary $\mathcal{B}$ we get

$$\begin{aligned} \mathrm{Adv}_{2q}^{S_0^p,S_1^p}(\mathcal{B}) &= \Big|\Pr_{f\leftarrow_\$ S_0^p}[\mathcal{B}^f() = 1] - \Pr_{f\leftarrow_\$ S_1^p}[\mathcal{B}^f() = 1]\Big| \\ &= \mathrm{Adv}_{H,p}^{\text{W-SM-UD}}(\mathcal{A})\,. \end{aligned}$$

Combining this with Lemmas 1 and 3 we obtain the final bound:

$$\mathrm{Adv}_{\mathbf{Th},p}^{\text{W-SM-UD}}(\mathcal{A}) \le \mathrm{Adv}_{2q}^{S_0^p,S_1^p}(\mathcal{B}) = \mathrm{Adv}_{2q}^{S_0,S_1}(\mathcal{B}) \le 12q/\sqrt{2^n},$$

where q denotes the number of queries to **Th**.

8 Generic constructions

In the last section we saw bounds for the security of random THFs for the different security properties. In this section, we discuss how to construct THFs from typical hash functions. In this context we recall two constructions from [BHK+19]. One construction uses a KHF $H : \mathcal{K} \times \{0,1\}^\alpha \to \{0,1\}^n$ to build a THF. The other starts from a plain, key-less hash function. Since the properties we require go beyond those required in [BHK+19] we need to analyze those constructions again with respect to the newly added properties. We focus on the following two constructions:

Construction 1 ([BHK+19]) *Given two hash functions* $H_1 : \{0,1\}^{2n} \times \{0,1\}^\alpha \to \{0,1\}^n$ *with* $2n$*-bit keys, and* $H_2 : \{0,1\}^{2n} \to \{0,1\}^\alpha$ *we construct* **Th** *with* $\mathcal{P} = \mathcal{T} = \{0,1\}^n$*, as*

$$\mathbf{Th}(P,T,M) = H_1(P||T, M^\oplus), \text{ with } M^\oplus = M \oplus H_2(P||T)$$

Construction 2 ([BHK+19]) *Given a hash function* $H : \{0,1\}^{2n+\alpha} \to \{0,1\}^n$*, we construct* **Th** *with* $\mathcal{P} = \mathcal{T} = \{0,1\}^n$*, as*

$$\mathbf{Th}(P,T,M) = H(P||T||M)$$

Security of Construction 2 can only be shown in the (Q)ROM. Assuming that H behaves like a random function, we can simply apply the bounds discussed in the last section. When analyzing Construction 1, security can be based on the security of the used KHF in the QROM. In [BHK+19] three constructions were analyzed, the third one missing here is a variant of Construction 1 that is secure in the standard model at the price of huge public parameters. This third construction was there to demonstrate that the only reason the QROM is needed in the analysis of Construction 1 is to prove the parameter compression secure. Below, we determine the required properties of KHFs to obtain the desired properties of the THF constructed via Construction 1.

8.1 Construction 1

First we recall the results from [BHK+19] that show under which conditions this construction is W-SM-TCR and W-SM-DSPR. Afterwards we give bounds for W-SM-PRE and W-SM-UD. Below, we refer to a property as "with tweak advice" if the adversary informs the oracle about all p keys or tweaks it will use ahead of its queries.

W-SM-TCR security. To prove W-SM-TCR we use the DM-SPR property of the KHF. This property is similar to W-SM-TCR. It is a TCR notion for KHFs where the adversary specifies the keys for which he will obtain the challenges. For a formal definition of the property see Appendix C. The following result has been shown in [BHK+19]:

Theorem 4. *Let* H_1 *and* H_2 *be hash functions as in Construction 1 and* **Th** *the* THF *constructed by Construction 1. Then the success probability of any* q*-query time-*ξ *(quantum) adversary* $\mathcal{A}$ *against* W-SM-TCR *of* **Th** *with tweak advice is bounded by*

$$\mathrm{Succ}^{\text{W-SM-TCR}}_{\mathbf{Th},p}(\mathcal{A}) \leq \mathrm{InSec}^{\text{DM-SPR}}(H_1; \xi, p),$$

when modeling H_2 *as quantum-accessible random oracle and not giving* $\mathcal{A}_1$ *access to this oracle.*

This result requires tweak advice for technical reasons. But this is sufficient for SPHINCS$^+$ as all the tweaks that are needed to construct the challenge are known ahead. A bound for DM-SPR for a random function H was given in [BHK+19]: $\mathrm{Succ}^{\text{DM-SPR}}_{H,p}(\mathcal{A}) \leq \Theta(\frac{(q+1)^2}{2^n})$.

SM-DSPR security. SM-DSPR gets related to distinct function, multi-target decisional second-preimage resistance (DM-DSPR). This is a DSPR property for KHFs where the adversary can define the keys used for the challenges. See Appendix C for more details.

Algorithm 6: Reducing DM-PRE to W-SM-PRE

Input : W-SM-PRE adversary $\mathcal{A} = (A_1, A_2)$, DM-PRE challenger C, **Th**, H_1, H_2
Output: $M^* \in \{0,1\}^\alpha$

Generate $P \leftarrow_\$ \{0,1\}^n$.
For each T_i obtained from $\mathcal{A}_1$ query C with $P||T_i$.
For each query $P||T_i$ obtain y_i from C
Return y_i to $\mathcal{A}$ as an answer for query T_i
After all queries return P to $\mathcal{A}$
Obtain the result (j, M') from $\mathcal{A}$
return $M^* = M' \oplus H_2(P||T_j)$

Theorem 5 ([BHK$^+$19]). *Let H_1 and H_2 be hash functions as in Construction 1 and* **Th** *the* THF *constructed by Construction 1. Then the advantage of any q-query time-ξ (quantum) adversary $\mathcal{A}$ against* W-SM-DSPR *of* **Th** *with tweak advice is bounded by*

$$\mathrm{Adv}^{\text{W-SM-DSPR}}_{\mathbf{Th},p}(\mathcal{A}) \leq \mathrm{InSec}^{\text{DM-SPR}}(H, \xi, p),$$

when modeling H_2 as quantum-accessible random oracle and not giving $\mathcal{A}_1$ access to this oracle.

In [BHK$^+$19] the bound for DM-DSPR of a random function is conjectured to be $\mathrm{Succ}^{\text{DM-SPR}}_{H,p}(\mathcal{A}) \leq \Theta(\frac{(q+1)^2}{2^n})$.

W-SM-PRE security. Since in the new proof of SPHINCS$^+$ we also need the W-SM-PRE property we have to analyze under which conditions Construction 1 will provide this property. To do so we will need distinct function, multi-target preimage resistance (DM-PRE), a formal definition of the property is given in Appendix C.

Theorem 6. *Let H_1 and H_2 be hash functions as in Construction 1 and* **Th** *the* THF *constructed by Construction 1. Then the success probability of any time-ξ (quantum) adversary $\mathcal{A}$ against* W-SM-PRE *of* **Th** *with tweak advice is bounded by*

$$\mathrm{Succ}^{\text{W-SM-PRE}}_{\mathbf{Th},p}(\mathcal{A}) \leq \mathrm{InSec}^{\text{DM-PRE}}(H_1; \xi, p)\,.$$

Proof. Assume we are given access to an adversary $\mathcal{A}$ against W-SM-PRE of **Th** with tweak advice. We show how to construct an oracle machine $M^{\mathcal{A}}$ that breaks DM-PRE of H_1. This procedure is presented in the Algorithm 6.

First we sample a random $P \leftarrow_\$ \{0,1\}^n$. For each tweak T that we receive from the adversary we construct a key for H_1: $K = P||T$. Then we query the DM-PRE challenger with that key and obtain $y = H_1(P||T, M')$ for a random M'. $\mathcal{A}$ expects $y = H_1(P||T, M^{\oplus})$ where $M^{\oplus} = M \oplus H_2(P||T)$ for a uniformly random M. One can see that $H_2(P||T)$ is independent from M and $M = M^{\oplus} \oplus H_2(P||T)$. Since $M' = M^{\oplus}$ in this case and M' is a uniformly random message, M is uniformly distributed. One can conclude that answering $\mathcal{A}$ with y does not change $\mathcal{A}$'s behavior as y follows the same distribution as in the original game.

Since we consider W-SM-PRE with tweak advice we can collect all tweaks from $\mathcal{A}_1$, generate keys $K_1, \ldots K_p$, query DM-PRE challenger with those keys, get the answer $Y = \{y_i\}_{i=1}^p$ for that query and return Y to $\mathcal{A}$. In response, $\mathcal{A}$ produces an answer (j, M'). If this is a preimage for **Th**, we can obtain the preimage for H_1 by calculating $M^* = M' \oplus H_2(P||T)$. So we obtain the bound $\mathrm{Succ}^{\text{W-SM-PRE}}_{\mathbf{Th},p}(\mathcal{A}) \leq \mathrm{InSec}^{\text{DM-PRE}}(H_1; \xi, p)$.

The bound for DM-PRE of H modeled as a random function can be obtained by showing that $\mathrm{Succ}^{\text{DM-PRE}}_{\mathbf{Th},p}(\mathcal{A}) \leq \mathrm{Succ}^{\text{DM-SPR}}_{\mathbf{Th},p}(\mathcal{B}) + 3 \cdot \mathrm{Succ}^{\text{DM-SPR}}_{\mathbf{Th},p}(\mathcal{C})$ for some algorithms $\mathcal{B}$ and $\mathcal{C}$. A proof that this holds can be found in Appendix D. If we believe the Conjecture in [BHK+19] regarding the tight bound for DM-DSPR then we obtain $\mathrm{Succ}^{\text{DM-PRE}}_{\mathbf{Th},p}(\mathcal{A}) \leq \Theta(\frac{(q+1)^2}{2^n})$.

W-SM-UD security. Another property to finalize the analysis of **Th** constructions for SPHINCS$^+$ is W-SM-UD. For this part we will utilize the distinct function, multi-target undetectability property (DM-UD). For a formal definition see Appendix C.

Theorem 7. *Let H_1 and H_2 be hash functions as in Construction 1 and* **Th** *the* THF *constructed by Construction 1. Then the following equality holds:*

$$\mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{Th}; \xi, p) = \mathrm{InSec}^{\text{DM-UD}}(H_1; \xi, p).$$

Proof. Lets analyze the distribution of $y \leftarrow H_1(P||T, x)$, where $x \leftarrow_\$ \{0,1\}^\alpha$ and $y' \leftarrow \mathbf{Th}(P, T, x')$, where $x' \leftarrow_\$ \{0,1\}^\alpha$ (for now we fix P and T).

We can show a bijection between $\{y = H_1(P||T, x), x\}$ and $\{y = \mathbf{Th}(P, T, x'), x'\}$. The bijection is $x \to x \oplus H_2(P||T)$. Note that

$$\mathbf{Th}(P, T, x \oplus H_2(P||T)) = H_1(P||T, x \oplus H_2(P||T) \oplus H_2(P||T)) = H_1(P||T, x)\,.$$

If x is a uniformly distributed random variable than $x \oplus H_2(P||T)$ is also a uniformly distributed random variable. From this bijection we can conclude that for fixed P and T and randomly sampled x we have the same distribution of outputs for **Th** and H_1. Hence

$$\mathrm{Adv}_{\mathbf{Th}(P,T,\cdot),H_1(P||T,\cdot)}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathbf{Th}_{P,T}}() = 1] - \Pr[\mathcal{A}^{H_{1_{P||T}}}() = 1]| = 0$$

The same argument applies if we have multiple pairs $\{P, T_i\}_{i=1}^p$

Assume we have the following distributions for some fixed set $Q = \{P, T_i\}_{i=1}^p$:

- $X_0 = \{\mathbf{Th}(P, T_i, x_i)\}_{i=1}^p$, where $x_i \leftarrow_\$ \{0,1\}^\alpha$;
- $X_1 = \{H_1(P||T_i, x_i)\}_{i=1}^p$, where $x_i \leftarrow_\$ \{0,1\}^\alpha$;
- $X_2 = \{y_i\}_{i=1}^p$, where $y_i \leftarrow_\$ \{0,1\}^n$.

The adversary will work with exactly these distributions in the W-SM-UD or DM-UD game if he queried the set Q during the first stage. First by the argument above

$$\mathrm{Adv}_{X_0,X_1}(\mathcal{A}) = \mathrm{Adv}_{X_1,X_0}(\mathcal{A}) = 0.$$

Next by the triangle inequality one can see that

$$\mathrm{Adv}_{X_0,X_2}(\mathcal{A}) \leq \mathrm{Adv}_{X_0,X_1}(\mathcal{A}) + \mathrm{Adv}_{X_1,X_2}(\mathcal{A}).$$

Since the $\mathrm{Adv}_{X_0,X_1}(\mathcal{A}) = 0$ we conclude that $\mathrm{Adv}_{X_0,X_2}(\mathcal{A}) \leq \mathrm{Adv}_{X_1,X_2}(\mathcal{A})$. Applying the same argument note that

$$\mathrm{Adv}_{X_1,X_2}(\mathcal{A}) \leq \mathrm{Adv}_{X_1,X_0}(\mathcal{A}) + \mathrm{Adv}_{X_0,X_2}(\mathcal{A}) = \mathrm{Adv}_{X_0,X_2}(\mathcal{A}).$$

From the last two inequalities we conclude that $\mathrm{Adv}_{X_0,X_2}(\mathcal{A}) = \mathrm{Adv}_{X_1,X_2}(\mathcal{A})$. Now it is important to observe that $\mathrm{Adv}_{X_0,X_2}(\mathcal{A}) = \mathrm{Adv}^{\text{W-SM-UD}}_{\mathbf{Th},p}(\mathcal{A})$ while $\mathrm{Adv}_{X_1,X_2}(\mathcal{A}) = \mathrm{Adv}^{\text{DM-UD}}_{H_1,p}(\mathcal{A})$. The above equality holds for any set Q that defines the distribution of X_0, X_1, X_2. The adversary playing the UD game will specify the set Q by performing the queries to its oracle. Hence we have shown that the distinguishing advantage of the adversary playing the DM-UD and W-SM-UD games is identical.

We can conclude that

$$\mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{Th}; \xi, p) = \mathrm{InSec}^{\text{DM-UD}}(H_1; \xi, p).$$

It remains to analyze the DM-UD property for H when modeled as a random function. Since we have analyzed the W-SM-UD property for random $\mathbf{Th}$ and have that $\mathrm{InSec}^{\text{W-SM-UD}}(\mathbf{Th};\xi,p) = \mathrm{InSec}^{\text{DM-UD}}(H;\xi,p)$, we can conclude that $\mathrm{Succ}^{\text{DM-UD}}_{H,p}(\mathcal{A}) \leq 12q/\sqrt{2^n}$ where q denotes the number of $\mathcal{A}$'s queries to H.

References

ABB+20. Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. SPHINCS$^+$. Submission to NIST's post-quantum crypto standardization project, v.3, 2020. http://sphincs.org/data/sphincs+-round3-specification.pdf.

BH19. Daniel J. Bernstein and Andreas Hülsing. Decisional second-preimage resistance: When does SPR imply PRE? In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 33–62, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

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: 26th Conference on Computer and Communications Security*, pages 2129–2146. ACM Press, November 11–15, 2019.

DSS05. C. Dods, Nigel P. Smart, and Martijn Stam. Hash based digital signature schemes. In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 96–115, Cirencester, UK, December 19–21, 2005. Springer, Heidelberg, Germany.

GMR88. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

HRS16. Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 387–416, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

Hül13. Andreas Hülsing. W-OTS+ - shorter signatures for hash-based signature schemes. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *AFRICACRYPT 13: 6th International Conference on Cryptology in Africa*, volume 7918 of *Lecture Notes in Computer Science*, pages 173–188, Cairo, Egypt, June 22–24, 2013. Springer, Heidelberg, Germany.

KKF20. Mikhail Kudinov, Evgeniy Kiktenko, and Aleksey Fedorov. [pqc-forum] round 3 official comment: Sphincs+. https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/official-comments/Sphincs-Plus-round3-official-comment.pdf, 2020. Accessed: 2022-2-1.

KLM06. Phillip Kaye, Raymond Laflamme, and Michele Mosca. An introduction to quantum computing. Oxford University Press, 2006.

Zha12. Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 758–775, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

A Hash functions properties for the SPHINCS$^+$ security proof

In this section we provide some informal description of SM-TCR, SM-PRE, SM-UD properties of THFs, extending on the formal definitions in Section 2. Moreover, we give formal definitions of the PRF and ITSR properties for KHFs.

SM-TCR. One can view SM-TCR as a variant of target-collision resistance. Consider an adversary $\mathcal{A}$ which consists of two parts $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}$ will play a two-stage game. In the first stage, $\mathcal{A}_1$ is allowed to adaptively specify p targets (multi-target). The target specification is implemented via access to an oracle implementing the function with a fixed public parameter (single-function as the same public parameter is used for all targets). The query consists of a tweak and a message. Every query to this oracle defines a target. It is important that $\mathcal{A}$ is not allowed to query the oracle with the same tweak more than once. The challenge is to find a collision for one of the suggested messages under the corresponding tweak. This is the task of $\mathcal{A}_2$ which will obtain all the information from $\mathcal{A}_1$ as well as the public parameters.

SM-PRE. As for W-SM-TCR, SM-PRE is a two-stage game and can be seen as a variant of preimage resistance. Adversary $\mathcal{A}_1$ is allowed to specify p targets during the first stage. The speciation is again done querying an oracle with tweaks. The oracle implements a tweakable hash function with a fixed public parameter. Each query is answered with a hash value of the public parameter, the queried tweak and a random input. Again, the adversary is restricted in that it may only make one query per tweak. The second stage $\mathcal{A}_2$ receives all the information from $\mathcal{A}_1$ as well as the public parameters and has to output a preimage for one of the targets.

SM-UD. Also SM-UD is a variant of an established notion, in this case undetectability [DSS05], that makes use of a two stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. During the first stage $\mathcal{A}_1$ specifies p targets by querying an oracle with distinct tweaks. The oracle is implemented in one of two ways. The first instantiation initially samples a public parameter. Given a tweak as query, the oracle samples a uniformly random message and returns the result of applying the THF using the public parameter and the given tweak. The second instantiation simply returns a uniformly random element from the THF's co-domain. Each query is answered with the result of processing the oracle on the queried tweak and a random message. The advantage of the adversary is its ability to distinguish between these two possible oracle instantiations.

ITSR. During the SPHINCS$^+$ signature algorithm a message digest is computed. This message digest is interpreted as leaves of one of the FORS structures. Interleaved target subset resilience (ITSR) is the property of a hash function that computes the message digest. Assume that several messages were mapped to some set of leaves of the FORS structures. We do not want the adversary to be able to find a different message that maps to a subset of used leaves. The formal definition of the property is given bellow.

Definition 9 (ITSR [BHK+19]). *Let* $\mathrm{H} : \mathcal{K} \times \{0,1\}^\alpha \to \{0,1\}^m$ *be a keyed hash function. Also consider a mapping function* $\mathsf{MAP}_{h,k,t} : \{0,1\}^m \to \{0,1\}^h \times [0, t-1]^k$ *which maps an* m*-bit string to a set of* k *indexes. We denote those indexes as* $((I,1,J_1),\ldots,(I,k,J_k))$*, where* I *is chosen from* $[0, 2^h - 1]$ *and each* J_i *is chosen from* $[0, t-1]$*.*

The success probability of an adversary $\mathcal{A}$ *against* ITSR *of* H *is defined as follows. Let* $G = \mathsf{MAP}_{h,k,t} \circ \mathrm{H}$*. Let* $O(\cdot)$ *be an oracle which on input of an* α*-bit message* M_i *samples a key* $K_i \leftarrow_\$ \mathcal{K}$ *and returns* $G(K_i, M_i)$*. The adversary* $\mathcal{A}$ *is allowed to query the oracle with messages of its choice. Denote the number of queries with* q*. Then,*

$$\mathrm{Succ}^{\mathrm{ITSR}}_{\mathrm{H},q}(\mathcal{A}) = \Pr[(R,M) \leftarrow \mathcal{A}^{O(\cdot)}(1^n) \\ s.t.\ G(K,M) \subseteq \bigcup_{j=1}^{q} G(K_j, M_j) \wedge (K,M) \notin \{(K_j,M_j)\}_{j=1}^{q}],$$

where $\{(K_j, M_j)\}_{j=1}^{q}$ *represent the responses of the oracle* $O(\cdot)$*.*

PRF. Assume we have a KHF and we sampled a random key. If interactions with such subfunction is computationally indistinguishable from interacting with a random function we call such KHF a pseudorandom function. The formal definition of the property is given bellow.

Definition 10 (Keyed hash function). *Let $\mathcal{K}$ be the key space, $\mathcal{M}$ the message space, and $\mathcal{N}$ the output space. A keyed hash function is an efficient function*

$$F : \mathcal{K} \times \mathcal{M} \to \mathcal{N}$$

generating an n-bit value out of a key and a message.

In the following we give the definition for PRF security of a KHF $F : \mathcal{K} \times \mathcal{M} \to \mathcal{N}$. In the definition of the PRF distinguishing advantage, the adversary $\mathcal{A}$ gets (classical) oracle access to either $F(S, \cdot)$ for a uniformly random secret key $S \in \mathcal{K}$ or to a function G drawn from the uniform distribution over the set $\mathcal{G}(\mathcal{M}, \mathcal{N})$ of all functions with domain $\mathcal{M}$ and range $\mathcal{N}$. The goal of $\mathcal{A}$ is to distinguish both cases.

Definition 11 (PRF). *Let F be defined as above. We define the* PRF *distinguishing advantage of an adversary $\mathcal{A}$ making q queries to its oracle as*

$$\mathrm{Adv}^{\mathrm{PRF}}_{F,q}(\mathcal{A}) = \left| \Pr_{S \leftarrow_\$ \mathcal{K}}[\mathcal{A}^{F(S,\cdot)} = 1] - \Pr_{G \leftarrow_\$ \mathcal{G}(\mathcal{M},\mathcal{N})}[\mathcal{A}^{G(\cdot)} = 1] \right|.$$

B Security model

Here we define what a digital signature scheme is and the EU-CMA and EU-naCMA security model.

Definition 12 (Digital signature schemes). *Let $\mathcal{M}$ be a message space. A digital signature scheme* $\mathsf{Dss} = (\mathsf{Kg}, Sign, Vf)$ *is a triple of probabilistic polynomial time algorithms:*

- $\mathsf{Kg}(1^n)$ *on input of a security parameter 1^n outputs a private key* sk *and a public key* pk*;*
- $\mathsf{Sign}(\mathsf{sk}, M)$ *outputs a signature σ under secret key* sk *for message $M \in \mathcal{M}$;*
- $\mathsf{Vf}(\mathsf{pk}, \sigma, M)$ *outputs 1 iff σ is a valid signature on M under* pk*;*

such that $\forall(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kg}(1^n), \forall(M \in \mathcal{M}) : \mathsf{Vf}(\mathsf{pk}, \mathsf{Sign}(\mathsf{sk}, M), M) = 1$.

The standard security notion for digital signature schemes is existential unforgeability under adaptive chosen-message attacks (EU-CMA) [GMR88]. The notion is defined using the following experiment for signature scheme Dss. In the experiment, the adversary $\mathcal{A}$ is given access to a signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ which is initialized with the target secret key. The q queries to $\mathsf{Sign}(\mathsf{sk}, \cdot)$ are denoted $\{M_i\}_1^{q_s}$. Following the reasoning above, even quantum adversaries are limited to classical queries to this oracle as it simulates an honest and hence classical user.

Experiment $\mathsf{Exp}^{\mathsf{EU-CMA}}_{\mathsf{Dss}(1^n)}(\mathcal{A})$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^n)$
$(M^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
Return 1 iff $\mathsf{Vf}(\mathsf{pk}, M^\star, \sigma^\star) = 1$ and $M^\star \notin \{M_i\}_1^{q_s}$.

Definition 13 (EU-CMA). *Let* Dss *be a digital signature scheme. We define the success probability of an adversary $\mathcal{A}$ against the EU-CMA security of* Dss *as the probability that the above experiment outputs 1:*

$$\mathrm{Succ}^{\mathsf{EU-CMA}}_{\mathsf{Dss}(1^n),q}(\mathcal{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{EU-CMA}}_{\mathsf{Dss}(1^n)}(\mathcal{A}) = 1\right],$$

where q denotes the number of queries that $\mathcal{A}$ makes to its oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$.

We also define existential unforgeability under *non-adaptive* chosen message attack (EU-naCMA). It is defined using the following experiment where S makes the shared state of $\mathcal{A}_1$ and $\mathcal{A}_2$ explicit.

Experiment $\mathsf{Exp}^{\mathsf{EU-naCMA}}_{\mathsf{Dss}(1^n)}(\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2))$**:**

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(1^n)$.
$(\{M_1, \ldots, M_q\}, S) \leftarrow \mathcal{A}_1()$.
Compute $\{(M_i, \sigma_i)\}_{i=1}^q$ using $\mathsf{Sign}(\mathsf{sk}, \cdot)$.
$(M^\star, \sigma^\star) \leftarrow \mathcal{A}_2(S, \{(M_i, \sigma_i)\}_{i=1}^q, pk)$
Return 1 iff $\mathrm{Vf}(\mathsf{pk}, \sigma^\star, M^\star) = 1$ and $M^\star \notin \{M_i\}_{i=1}^q$.

Definition 14 (EU-naCMA). *Let* Dss *be a digital signature scheme. We define the success probability of an adversary* $\mathcal{A}$ *against the* EU-naCMA *security of* Dss *as the probability that the above experiment outputs 1:*

$$\mathrm{Succ}^{\mathsf{EU-naCMA}}_{\mathsf{Dss}(1^n),q}(\mathcal{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{EU-naCMA}}_{\mathsf{Dss}(1^n)}(\mathcal{A}) = 1\right],$$

where q *denotes the number of messages that* $\mathcal{A}_1$ *asks the game to sign.*

C Keyed Hash function properties

In this section we provide formal definitions of security properties for KHFs used in Section 8. We define distinct-function, multi-target versions of second-preimage resistance, decisional second-preimage resistance, preimage resistance, and undetectability.

Definition 15 (DM-SPR [BHK+19]). *Let* $H : \mathcal{K} \times \{0,1\}^\alpha \to \{0,1\}^n$ *be a keyed hash function. We define the advantage of any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against distinct-function, multi-target second-preimage resistance* (DM-SPR)*. This definition is parameterized by the number of targets* p*.*

$$\begin{aligned}\mathrm{Succ}^{\text{DM-SPR}}_{H,p}(\mathcal{A}) = \Pr\,[&\{K_i\}_{i=1}^p \leftarrow \mathcal{A}_1(\,), \{M_i\}_1^p \leftarrow_\$ (\{0,1\}^\alpha)^p;\\ &(j, M') \leftarrow_\$ \mathcal{A}_2(\{(K_i, M_i)\}_{i=1}^p) : M' \neq M_j\\ &\wedge H(K_j, M_j) = H(K_j, M') \wedge \mathbf{DIST}(\{K_i\}_{i=1}^p)]\,.\end{aligned}$$

where we assume that $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *share state and* $\mathbf{DIST}(\{K_i\}_1^p)$ *is as in* W-SM-TCR.

Towards defining decisional second preimage resistance we need the notion of a second-preimage exists predicate:

Definition 16 (SPexists for keyed hash functions [BHK+19]). *The second-preimage-exists predicate* $\mathrm{SPexists}(H)$ *for a keyed hash function* H *is the function* $\mathbf{SP} : \mathcal{K} \times \{0,1\}^\alpha \to \{0,1\}$ *defined as follows:*

$$\mathbf{SP}_K(M) = \begin{cases} 1 & \text{if } |H_K^{-1}(H_K(M))| \geq 2\\ 0 & \text{otherwise,}\end{cases}$$

Definition 17 (DM-DSPR [BHK+19]). *In the following let* H *be a keyed hash function as defined above. We define the advantage of any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against* DM-DSPR *of* H*. The definition is parameterized by the number of targets* p*.*

$$\mathrm{Adv}^{\text{DM-DSPR}}_{H,p}(\mathcal{A}) \stackrel{def}{=} \max\{0, \mathrm{succ} - \mathrm{triv}\},$$

where

$$\begin{aligned}\text{succ} = &\Pr[\{K_i\}_1^p \leftarrow \mathcal{A}_1(\,); \{M_i\}_1^p \leftarrow_{\$} (\{0,1\}^\alpha)^p;\\ &(j,b) \leftarrow \mathcal{A}_2(\{(K_i,M_i)\}_1^p) : \mathbf{SP}_{K_j}(M_j) = b \wedge \mathbf{DIST}(\{K_i\}_1^p)];\\ \text{triv} = &\Pr[\{K_i\}_1^p \leftarrow \mathcal{A}_1(\,); \{M_i\}_1^p \leftarrow_{\$} (\{0,1\}^\alpha)^p;\\ &(j,b) \leftarrow \mathcal{A}_2(\{(K_i,M_i)\}_1^p) : \mathbf{SP}_{K_j}(M_j) = 1 \wedge \mathbf{DIST}(\{K_i\}_1^p)];\end{aligned}$$

and where $\mathbf{DIST}(\{K_i\}_1^p)$ *is defined as in* W-SM-TCR.

Definition 18 (DM-PRE). *Let* $H : \mathcal{K} \times \{0,1\}^\alpha \to \{0,1\}^n$ *be a keyed hash function. We define the advantage of any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against distinct-function, multi-target preimage resistance* (DM-PRE)*. This definition is parameterized by the number of targets* p*. We denote* $\mathcal{A}_1$*'s output by* $Q = \{K_i\}_{i=1}^p$ *and use the same predicate DIST as in previous definitions.*

$$\begin{aligned}\text{Succ}_{H,p}^{\text{DM-PRE}}(\mathcal{A}) = \Pr[&\{K_i\}_{i=1}^p \leftarrow \mathcal{A}_1(\,), \{M_i\}_{i=1}^p \leftarrow_{\$} (\{0,1\}^\alpha)^p;\\ &(j, M') \leftarrow_{\$} \mathcal{A}_2(\{K_i, H(K_i,M_i)\}_{i=1}^p) :\\ &H(K_j, M') = H(K_j, M_j) \wedge \mathbf{DIST}(Q)]\,.\end{aligned}$$

Definition 19 (DM-UD). *Let* $H : \mathcal{K} \times \{0,1\}^\alpha \to \{0,1\}^n$ *be a keyed hash function. We define the advantage of any adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *against distinct-function, multi-target undetectability* (DM-UD)*. This definition is parameterized by the number of targets* p*.*

Consider an oracle $\mathcal{O}(\mathcal{K}, \{0,1\})$*, which works as follows:* $\mathcal{O}(K,0)$ *returns* $H(K,M_i)$*, where* $M_i \leftarrow_{\$} \{0,1\}^\alpha$ *for each query* i*;* $\mathcal{O}(K,1)$ *returns* Y_i*, where* $Y_i \leftarrow_{\$} \{0,1\}^n$ *for each query* i*. Algorithm* $\mathcal{A}_1$ *is allowed to make* p *queries to the oracle* $\mathcal{O}(\cdot,b)$*. The set of* $\mathcal{A}_1$*'s queries is denoted by* $Q = \{K_i\}_{i=1}^p$*.*

$$\begin{aligned}&\text{Adv}_{H,p}^{\text{DM-UD}}(\mathcal{A}) =\\ &\quad|\Pr[S \leftarrow \mathcal{A}_1^{\mathcal{O}(\cdot,0)}(\,); 1 \leftarrow \mathcal{A}_2(Q,S) \wedge \mathbf{DIST}(Q)] -\\ &\quad\Pr[S \leftarrow \mathcal{A}_1^{\mathcal{O}(\cdot,1)}(\,); 1 \leftarrow \mathcal{A}_2(Q,S) \wedge \mathbf{DIST}(Q)]|\end{aligned}$$

D DM-PRE generic security

In this section we show the relation between DM-SPR, DM-DSPR and DM-PRE, for completeness. The whole section closely follows [BH19]. We start with the following two reductions:

Definition 20 (p-target DM-SPR from DM-PRE (DM-SP$\textsc{from}$P$_p$)). *Let* H *be a keyed hash function. Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be an algorithm that plays the* DM-PRE *game. Let* p *be a positive integer. Let* C *be a* DM-SPR *challenger. Then* DM-SP$\textsc{from}$P$_p(H, \mathcal{A})$ *is the following algorithm:*

- *Run* $\mathcal{A}_1$ *to obtain* p *keys* $\{K_i\}_{i=1}^p$ *and send them to* C*;*
- *On input* $\{K_i, M_i\}_{i=1}^p$ *from* C *run* $\mathcal{A}_2$ *on input* $\{K_i, H(K_i,M_i)\}_{i=1}^p$*;*
- *Output the* (M', j) *that* $\mathcal{A}_2$ *outputs.*

Definition 21 (p-target DM-DSPR from DM-PRE (DM-DSP$\textsc{from}$P$_p$)). *Let* H *be a keyed hash function. Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be an algorithm that plays the* DM-PRE *game. Let* p *be a positive integer. Let* C *be a* DM-DSPR *challenger. Then* DM-DSP$\textsc{from}$P$_p(H, \mathcal{A})$ *is the following algorithm:*

- *Run* $\mathcal{A}_1$ *to obtain* p *keys* $\{K_i\}_{i=1}^p$ *and send them to* C*;*

- *On input* $\{K_i, M_i\}_{i=1}^p$ *from* C *run* $\mathcal{A}_2$ *on input* $\{K_i, H(K_i, M_i)\}_{i=1}^p$*;*
- *Obtain* (M', j) *from* $\mathcal{A}_2$*;*
- *Compute* $b \leftarrow (M' \neq M_j)$*;*
- *Output* (j, b)*.*

The following theorem closely follows the ideas from [BH19].

Theorem 8 (DM-DSPR $\wedge$ DM-SPR $\Rightarrow$ DM-PRE). *Let* H *be a keyed hash function. Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be an algorithm that plays the* DM-PRE *game. Let* p *be a positive integer. Then*

$$\mathrm{Succ}_{H,p}^{\text{DM-PRE}}(\mathcal{A}) \leq \mathrm{Succ}_{H,p}^{\text{DM-DSPR}}(\mathcal{B}) + 3 \cdot \mathrm{Succ}_{H,p}^{\text{DM-SPR}}(\mathcal{C}),$$

where $\mathcal{B} = \text{DM-DSPfromP}_p(H, \mathcal{A})$ *and* $\mathcal{C} = \text{DM-SPfromP}_p(H, \mathcal{A})$*.*

Proof. In this case we only consider $\mathcal{A}$ that perform correct queries, i.e. $\mathbf{DIST}\{K_i\}_{i=1}^p = 1$. As in [BH19] we split the universe of possible events into mutually exclusive events across two dimensions: the number of preimages of $H(K_j, M_j)$, and whether $\mathcal{A}$ succeeds or fails in finding a preimage. We define

$$S_i \stackrel{\text{def}}{=} [|H^{-1}(K_j, H(K_j, M_j))| = i \wedge H(K_j, M') = H(K_j, M_j)]$$

and

$$F_i \stackrel{\text{def}}{=} [|H^{-1}(K_j, H(K_j, M_j))| = i \wedge H(K_j, M') \neq H(K_j, M_j)]$$

We denote with s_i and f_i the probabilities of S_i and F_i respectively.

DM-PRE success probability. By definition the DM-PRE success probability of algorithm $\mathcal{A}$ is $\sum_i s_i$.

DM-SPR success probability. The DM-SPR success probability of $\mathcal{C}$ can be calculated as $\sum_{i>1} \frac{i-1}{i} s_i$.

DM-DSPR success probability. The success probability of $\mathcal{B}$ against DM-SPR can be calculated as $s_1 + \sum_{i>1} \frac{i-1}{i} s_i + \sum_{i>1} f_i$. The trivial function probability is $\sum_{i>1} s_i + \sum_{i>1} f_i$. Hence the advantage of $\mathcal{B}$ against DM-DSPR is $s_1 - \sum_{i>1} \frac{s_i}{i}$.

Combining the probabilities.

$$\begin{aligned}
\mathrm{Adv}_{H,p}^{\text{DM-DSPR}}(\mathcal{B}) + 3 \cdot \mathrm{Succ}_{H,p}^{\text{DM-SPR}}(\mathcal{C}) &\geq s_1 - \sum_{i>1} \frac{1}{i} s_i + 3 \cdot \sum_{i>1} \frac{i-1}{i} s_i \\
&= s_1 + \sum_{i>1} \frac{3i-4}{i} s_i \\
&\geq s_1 + \sum_{i>1} s_i \geq \mathrm{Succ}_{H,p}^{\text{DM-PRE}}(\mathcal{A})
\end{aligned}$$

E Single target W-SM-UD

Let us recall the undetectability notion for one target. We analyze this notion for a random THF. First the adversary chooses a tweak T and the challenger responds in one of two ways. The challenger either returns a random value y or an output of a THF on a random input (with that tweak and secret public parameters). We now prove the following lemma.

Algorithm 7: Dist-1,0 to UD

Input : f, W-SM-UD adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$
Output: $b' \in \{0,1\}^n$

Choose a random value $y \leftarrow_\$ \{0,1\}^n$ and a public parameter $P \leftarrow_\$ \mathcal{P}$
Get a tweak T and state S from $\mathcal{A}_1$
Construct a random THF $H : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ using random THF $G : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ the following way:

$$H(p,t,x) : \begin{cases} \text{if } (p = P, t = T, f(x) = 1) : \text{ Return } y \\ \text{Return } G(p,t,x) \end{cases}$$

Run $\mathcal{A}_2$ with input S, P, y, and oracle access to H
return Output of $\mathcal{A}$

Lemma 5. *Let $n \in \mathbb{N}$, $H : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ - a random THF. Any quantum adversary $\mathcal{A}$ that solves* W-SM-UD *for one target making q queries to H can be used to construct a quantum adversary $\mathcal{B}$ that makes $2q$ queries to its oracle and distinguishes S_0 from S_1 with advantage*

$$\mathrm{Adv}_{2q}^{S_0,S_1}(\mathcal{B}) \leq 12q/\sqrt{2^n}$$

Proof. To prove the lemma, we will prove that $\mathrm{Adv}_{2q}^{S_0,S_1}(\mathcal{B}) = \mathrm{Adv}_{H,1}^{\text{W-SM-UD}}(\mathcal{A})$. The statement then immediately follows from Lemma 1. Towards this end, we build a THF $H : \mathcal{P} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ given a boolean function $f : \{0,1\}^n \to \{0,1\}$ either from S_0 or from S_1. We first choose another random THF G and sample a public parameter $P \leftarrow_\$ \mathcal{P}$ as well as a random value y. Given a tweak T we build H as:

$$H(p,t,x) : \begin{cases} \text{if } (p = P, t = T, f(x) = 1) : \text{ Return } y \\ \text{Return } G(p,t,x) \end{cases}$$

Let's analyze this construction. Assume we got f from S_0 then we have a random THF and an independent, randomly chosen value y. If we have f from S_1 we still have a random THF. Moreover since we have $f(x) = 1$ for some random x we may assume that we chose that x for the undetectability challenge and we returned the output of our hash function for that input. We did not change any distributions compared to the oracle that outputs the image of a random input. Hence the undetectability challenge is fully correlated with the task of distinguishing S_0 and S_1. We formalize these statements in Algorithm 7.

Denoting Algorithm 7 as quantum adversary $\mathcal{B}$, a similar argument to the above gives

$$\begin{aligned} \mathrm{Adv}_{2q}^{S_0,S_1}(\mathcal{B}) &= |\Pr_{f \leftarrow_\$ S_0}[\mathcal{B}^f() = 1] - \Pr_{f \leftarrow_\$ S_1}[\mathcal{B}^f() = 1]| \\ &= \mathrm{Adv}_{H,1}^{\text{W-SM-UD}}(\mathcal{A}) \end{aligned}$$

Summarizing and applying Lemma 1 gives $\mathrm{Adv}_{H,1}^{\text{W-SM-UD}}(\mathcal{A}) = \mathrm{Adv}_{\mathcal{B}}^{2q}(S_0, S_1) \leq 12q/\sqrt{2^n}$.