Decisional second-preimage resistance: When does SPR imply PRE?

Daniel J. Bernstein[1,2] and Andreas Hülsing[3]

[1] Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607–7045, USA
[2] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
djb@cr.yp.to
[3] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
andreas@huelsing.net

Abstract. There is a well-known gap between second-preimage resistance and preimage resistance for length-preserving hash functions. This paper introduces a simple concept that fills this gap. One consequence of this concept is that tight reductions can remove interactivity for multi-target length-preserving preimage problems, such as the problems that appear in analyzing hash-based signature systems. Previous reduction techniques applied to only a negligible fraction of all length-preserving hash functions, presumably excluding all off-the-shelf hash functions.

Keywords: cryptographic hash functions, preimage resistance, second-preimage resistance, provable security, tight reductions, multi-target attacks, hash-based signatures

1 Introduction

Define $S : \{0,1\}^{256} \to \{0,1\}^{256}$ as the SHA-256 hash function restricted to 256-bit inputs. Does second-preimage resistance for S imply preimage resistance for S?

The classic Rogaway–Shrimpton paper "Cryptographic hash-function basics" [15] shows that second-preimage resistance tightly implies preimage resistance for an efficient hash function that maps fixed-length inputs to *much shorter* outputs. The idea of the proof is that one can find a second preimage of a

Author list in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf. This work was supported by the U.S. National Science Foundation under grant 1314919, by the Cisco University Research Program, and by DFG Cluster of Excellence 2092 "CASA: Cyber Security in the Age of Large-Scale Adversaries". "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation" (or other funding agencies). Permanent ID of this document: 36ecc3ad6d0fbbe65ce36226c2e3eb875351f326. Date: 2019.09.12.

random input x with high probability by finding a preimage of the hash of x. But this probability depends on the difference in lengths, and the proof breaks down for length-preserving hash functions such as S.

The same paper also argues that second-preimage resistance cannot imply preimage resistance for length-preserving hash functions. The argument, in a nutshell, is that the identity function from $\{0,1\}^{256}$ to $\{0,1\}^{256}$ provides unconditional second-preimage resistance—second preimages do not exist—even though preimages are trivial to find.

A counterargument is that this identity-function example says nothing about real hash functions such as S. The identity-function example shows that there cannot be a theorem that for *all* length-preserving hash functions proves preimage resistance from second-preimage resistance; but this is only the beginning of the analysis. The example does not rule out the possibility that second-preimage resistance, together with a mild additional assumption, implies preimage resistance.

1.1 Contributions of this paper

We show that preimage resistance (PRE) follows tightly from the conjunction of second-preimage resistance (SPR) and decisional second-preimage resistance (DSPR). **Decisional second-preimage resistance** is a simple concept that we have not found in the literature: it means that the attacker has negligible advantage in deciding, given a random input x, whether x has a second preimage.

There is a subtlety in the definition of advantage here. For almost all length-preserving hash functions, always guessing that x *does* have a second preimage succeeds with probability approximately 63%. (See Section 3.) We define DSPR advantage as an increase in probability compared to this trivial attack.

We provide three forms of evidence that DSPR is a reasonable assumption. First, we show that DSPR holds for random functions even against quantum adversaries that get quantum access to a function. Specifically, a q-query quantum adversary has DSPR advantage at most $32q^2/2^n$ against an oracle for a uniform random hash function from $\{0,1\}^n$ to $\{0,1\}^n$. In [9] the same bound was shown for PRE and SPR together with matching attacks demonstrating the bounds are tight. This means that DSPR is at least as hard to break as PRE or SPR for uniform random hash functions from $\{0,1\}^n$ to $\{0,1\}^n$.

Second, the subtlety mentioned above means that DSPR, when generalized in the most natural way to m-bit-to-n-bit hash functions, becomes unconditionally provable when m is much larger than n. This gives a new proof of PRE from SPR, factoring the original proof by Rogaway and Shrimpton into two steps: first, prove DSPR when m is much larger than n; second, prove PRE from SPR and DSPR.

Third, we have considered ways to attack DSPR for real hash functions such as S, and have found nothing better than taking the time necessary to *reliably* compute preimages. A curious feature of DSPR is that there is no obvious way for a fast attack to achieve *any* advantage. A fast attack that occasionally finds a preimage of $H(x)$ will occasionally find a second preimage, but the baseline is

already guessing that x has a second preimage; to do better than the baseline, one needs to have enough evidence to be reasonably confident that x does *not* have a second preimage. Formally, there exists a fast attack (in the non-uniform model) that achieves a nonzero advantage (by returning 0 if the input matches some no-second-preimage values built into the attack, and returning 1 otherwise), but we do not have a fast way to recognize this attack. See Section 2.3.

1.1.1 Multi-target attacks. We see DSPR as showing how little needs to be assumed beyond SPR to obtain PRE. However, skeptics might object that SPR and DSPR are still two separate assumptions for cryptanalysts to study, that DSPR has received less study than PRE, and that DSPR could be easier to break than PRE, even assuming SPR. Why is assuming both SPR and DSPR, and deducing PRE, better than assuming both SPR and PRE, and ignoring DSPR? We give the following answer.

Consider the following simple interactive game T-openPRE. The attacker is given T targets $H(1, x_1), \ldots, H(T, x_T)$, where $x_1, \ldots, x_T$ are chosen independently and uniformly at random. The attacker is also given access to an "opening" oracle that, given i, returns x_i. The attacker's goal is to output (i, x') where $H(i, x') = H(i, x_i)$ and i was not an oracle query. Games of this type appear in, e.g., analyzing the security of hash-based signatures: legitimate signatures reveal preimages of some hash outputs, and attackers try to find preimages of other hash outputs.

One can try to use an attack against this game to break PRE as follows. Take the PRE challenge, insert it at a random position into a list of $T - 1$ randomly generated targets, and run the attack. Abort if there is an oracle query for the position of the PRE challenge; there is no difficulty answering oracle queries for other positions. The problem here is that a successful attack could query as many as $T - 1$ out of T positions, and then the PRE attack succeeds with probability only $1/T$. What happens if T is large and one wants a tight proof?

If T-openPRE were modified to use targets $H(x_i)$ instead of $H(i, x_i)$ then the attacker could try many guesses for x', checking each $H(x')$ against all of the targets. This generic attack is T times more likely to succeed than a generic attack against PRE using the same number of guesses. However, the inclusion of the prefix i (as in [9]) seems to force attackers to focus on single targets, and opens up the possibility of a security proof that does not quantitatively degrade with T.

One might try to tightly prove security of T-openPRE assuming security of a simpler non-interactive game T-PRE in which the opening oracle is removed: the attacker's goal is simply to find some (i, x') with $H(i, x') = H(i, x_i)$, given T targets $H(1, x_1), \ldots, H(T, x_T)$. This game T-PRE is simple enough that cryptanalysts can reasonably be asked to study it (and have already studied it without the i prefixes). However, the difficulty of answering the oracle queries in T-openPRE seems to be an insurmountable obstacle to a proof of this type.

We show that the security of T-openPRE follows tightly from the conjunction of two simple non-interactive assumptions, T-SPR and T-DSPR. This shows an

important advantage of introducing DSPR, allowing a reduction to remove the interactivity of T-openPRE.

The advantage of SPR (and T-SPR) over PRE (and T-PRE) in answering oracle queries inside reductions was already pointed out in [9]. The remaining issue, the reason that merely assuming T-SPR is not enough, is that there might be an attack breaking PRE (and T-PRE and T-openPRE) only for hash outputs that have unique preimages. Such an attack would never break SPR.

To address this issue, [9] assumes that each hash-function output has at least two preimages. This is a restrictive assumption: it is not satisfied by most length-preserving functions, and presumably it is not satisfied by (e.g.) SHA-256 for 256-bit inputs. Building a hash function that can be reasonably conjectured to satisfy the assumption is not hard—for example, apply SHA-256, truncate the result to 248 bits (see Theorem 11), and apply SHA-256 again to obtain a random-looking 256-bit string—but the intermediate truncation here produces a noticeably smaller security level, and having to do twice as many SHA-256 computations is not attractive.

We instead observe that an attack of this type must somehow be able to recognize hash outputs with unique preimages, and, consequently, must be able to recognize hash inputs without second preimages, breaking DSPR. Instead of assuming that there are always two preimages, we make the weaker assumption that breaking DSPR is difficult. This assumption is reasonable for a much wider range of hash functions.

1.1.2 The strength of SPR. There are some hash functions H where SPR is easy to break, or at least seems easier to break than PRE (and T-PRE and T-openPRE):

- Define $H(x) = 4^x \bmod p$, where p is prime, 4 has order $(p-1)/2$ modulo p, and x is in the range $\{0, 1, \ldots, p-2\}$. Breaking PRE is then solving the discrete-logarithm problem, which seems difficult when p is large, but breaking SPR is a simple matter of adding $(p-1)/2$ modulo $p-1$. (Quantum computers break PRE in this example, but are not known to break PRE for analogous examples based on isogenies.)
- Define $H : \{0,1\}^{2^k n} \to \{0,1\}^n$ by Merkle–Damgård iteration of an n-bit compression function. Then, under reasonable assumptions, breaking SPR for H takes only 2^{n-k} simple operations. See [10]. See also [1] for attacks covering somewhat more general iterated hash functions.

In the first example, proving PRE from SPR+DSPR is useless. In the second example, proving PRE from SPR+DSPR is unsatisfactory, since it seems to underestimate the quantitative security of PRE. This type of underestimate raises the same difficulties as a loose proof: users have to choose larger and slower parameters for the proof to guarantee the desired level of security, or have to take the risk of the "nightmare scenario" that there is a faster attack.

Fortunately, modern "wide-pipe" hash functions and "sponge" hash functions such as SHA-3 are designed to eliminate the internal collisions exploited in attacks such as [10]. Furthermore, input lengths are restricted in applications to

hash-based signatures, and this restriction seems to strengthen SPR even for older hash functions such as SHA-256. The bottom line is that one can easily select hash functions for which SPR and DSPR (and T-SPR and T-DSPR) seem to be as difficult to break as PRE, such as SHA3-256 and SHA-256 restricted to 256-bit inputs.

1.2 Organization of the paper

In Section 2 we define DSPR and show how it can be used to relate SPR and PRE. A consequence of our definition is that a function does not provide DSPR if noticeably more than half the domain elements have no colliding value. In Section 3 we show that the overwhelming majority of length-preserving hash functions have the property that more than half of the domain elements have a colliding value. In Section 4 we extend the analysis to keyed hash functions. We show in Section 5 that DSPR is hard in the quantum-accessible-random-oracle model (QROM). We define T-DSPR in Section 6. We show in Section 7 how to use T-DSPR to eliminate the interactivity of T-openPRE. We close our work with a discussion of the implications for hash-based signatures in Section 8.

2 Decisional second-preimage resistance

In this section we give a formal definition of decisional second-preimage resistance (DSPR) for cryptographic hash functions. We start by defining some notation and recalling some standard notions for completeness before we move on to the actual definition.

2.1 Notation

Fix nonempty finite sets $\mathcal{X}$ and $\mathcal{Y}$ of finite-length bit strings. In this paper, a *hash function* means a function from $\mathcal{X}$ to $\mathcal{Y}$.

As shorthands we write $M = |\mathcal{X}|$; $N = |\mathcal{Y}|$; $m = \log_2 M$; and $n = \log_2 N$. The *compressing* case is that $M > N$, i.e., $|\mathcal{X}| > |\mathcal{Y}|$; the *expanding* case is that $M < N$, i.e., $|\mathcal{X}| < |\mathcal{Y}|$; the *length-preserving* case is that $M = N$, i.e., $|\mathcal{X}| = |\mathcal{Y}|$.

We focus on bit strings so that it is clear what it means for elements of $\mathcal{X}$ or $\mathcal{Y}$ to be algorithm inputs or outputs. Inputs and outputs are required to be bit strings in the most common formal definitions of algorithms. These bit strings are often encodings of more abstract objects, and one could generalize all the definitions in this paper to work with more abstract concepts of algorithms.

2.2 Definitions

We now give several definitions of security concepts for a hash function H. We have not found decisional second-preimage resistance (DSPR) in the literature. We also define a second-preimage-exists predicate (SPexists) and the second-preimage-exists probability (SPprob) as tools to help understand DSPR. The

definitions of preimage resistance (PRE) and second-preimage resistance (SPR) are standard but we repeat them here for completeness.

Definition 1 (PRE). *The success probability of an algorithm $\mathcal{A}$ against the preimage resistance of a hash function* H *is*

$$\mathrm{Succ}^{\mathrm{PRE}}_{\mathrm{H}}(\mathcal{A}) \stackrel{def}{=} \mathsf{Pr}\left[x \leftarrow_R \mathcal{X}; x' \leftarrow \mathcal{A}(\mathrm{H}(x)) : \mathrm{H}(x) = \mathrm{H}(x')\right].$$

Definition 2 (SPR). *The success probability of an algorithm $\mathcal{A}$ against the second-preimage resistance of a hash function* H *is*

$$\mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{A}) \stackrel{def}{=} \mathsf{Pr}\left[x \leftarrow_R \mathcal{X}; x' \leftarrow \mathcal{A}(x) : \mathrm{H}(x) = \mathrm{H}(x') \wedge x \neq x'\right].$$

Definition 3 (SPexists). *The second-preimage-exists predicate* SPexists(H) *for a hash function* H *is the function $P : \mathcal{X} \to \{0,1\}$ defined as follows:*

$$P(x) \stackrel{def}{=} \begin{cases} 1 & \text{if } |\mathrm{H}^{-1}(\mathrm{H}(x))| \geq 2 \\ 0 & \text{otherwise.} \end{cases}$$

If $P(x) = 0$ then x has no second preimages under H: any $x' \neq x$ has $\mathrm{H}(x') \neq \mathrm{H}(x)$. The only possible successes of an SPR attack are for inputs x where $P(x) = 1$.

Definition 4 (SPprob). *The second-preimage-exists probability* SPprob(H) *for a hash function* H *is* $\mathsf{Pr}\left[x \leftarrow_R \mathcal{X} : P(x) = 1\right]$*, where* $P = \mathrm{SPexists(H)}$.

In other words, $p = \mathrm{SPprob(H)}$ is the maximum of $\mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{A})$ over all algorithms $\mathcal{A}$, without any limits on the cost of $\mathcal{A}$. Later we will see that almost all length-preserving hash functions H have $p > 1/2$. More precisely, $p \approx 1 - e^{-1} \approx 0.63$. For comparison, $p = 0$ for an injective function H, such as the n-bit-to-n-bit identity function; and $p = 1$ for a function where every output has multiple preimages.

Definition 5 (DSPR). *Let $\mathcal{A}$ be an algorithm that always outputs* 0 *or* 1*. The advantage of $\mathcal{A}$ against the decisional second-preimage resistance of a hash function* H *is*

$$\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) \stackrel{def}{=} \max\left\{0, \mathsf{Pr}\left[x \leftarrow_R \mathcal{X}; b \leftarrow \mathcal{A}(x) : P(x) = b\right] - p\right\}$$

where $P = \mathrm{SPexists(H)}$ *and* $p = \mathrm{SPprob(H)}$.

2.3 Examples of DSPR advantages

Here are some examples of computing DSPR advantages. As above, write $P = \mathrm{SPexists(H)}$ and $p = \mathrm{SPprob(H)}$.

If $\mathcal{A}(x) = 1$ for all x, then $\mathsf{Pr}\left[x \leftarrow_R \mathcal{X}; b \leftarrow \mathcal{A}(x) : P(x) = b\right] = p$ by definition, so $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = 0$.

If $\mathcal{A}(x) = 0$ for all x, then $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = \max\{0, 1-2p\}$. In particular, $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = 0$ if $p \geq 1/2$, while $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = 1$ for an injective function H.

More generally, say $\mathcal{A}(x)$ flips a biased coin and returns the result, where the probability of 1 is c, independently of x. Then $\mathcal{A}(x) = P(x)$ with probability $cp + (1-c)(1-p)$, which is between $\min\{1-p, p\}$ and $\max\{1-p, p\}$, so again $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = 0$ if $p \geq 1/2$.

As a more expensive example, say $\mathcal{A}(x)$ searches through all $x' \in \mathcal{X}$ to see whether x' is a second preimage for x, and returns 1 if any second preimage is found, otherwise 0. Then $\mathcal{A}(x) = P(x)$ with probability 1, so $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = 1-p$. This is the maximum possible DSPR advantage.

More generally, say $\mathcal{A}(x)$ runs a second-preimage attack $\mathcal{B}$ against H, and returns 1 if $\mathcal{B}$ is successful (i.e., the output x' from $\mathcal{B}$ satisfies $x' \neq x$ and $\mathrm{H}(x') = \mathrm{H}(x)$), otherwise 0. By definition $\mathcal{A}(x) = 1$ with probability $\mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{B})$, and if $\mathcal{A}(x) = 1$ then also $P(x) = 1$, so $\mathcal{A}(x) = 1 = P(x)$ with probability $\mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{B})$. Also $P(x) = 0$ with probability $1-p$ and if $P(x) = 0$ also $\mathcal{A}(x) = 0$ as there simply does not exist any second-preimage for $\mathcal{B}$ to find. Hence, $\mathcal{A}(x) = 0 = P(x)$ with probability $1-p$. Overall $\mathcal{A}(x) = P(x)$ with probability $1-p+\mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{B})$, so

$$\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = \max\left\{0, 1-2p+\mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{B})\right\}.$$

This advantage is 0 whenever $0 \leq \mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{B}) \leq 2p-1$: even if $\mathcal{B}$ breaks second-preimage resistance with probability as high as $2p-1$ (which is approximately 26% for almost all length-preserving H), $\mathcal{A}$ breaks DSPR with advantage 0. If $\mathcal{B}$ breaks second-preimage resistance with probability p, the maximum possible, then $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = 1-p$, the maximum possible advantage.

As a final example, say $x_1 \in \mathcal{X}$ has no second preimage, and say $\mathcal{A}(x)$ returns 0 if $x = x_1$, otherwise 1. Then $\mathcal{A}(x) = P(x)$ with probability $p + 1/2^m$, so $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = 1/2^m$. This example shows that an efficient algorithm can achieve a (very small) nonzero DSPR advantage. We can efficiently generate an algorithm $\mathcal{A}$ of this type with probability $1-p$ by choosing $x_1 \in \mathcal{X}$ at random (in the normal case that $\mathcal{X} = \{0,1\}^m$), but for typical hash functions H we do not have an efficient way to recognize whether $\mathcal{A}$ is in fact of this type, i.e., whether x_1 in fact has no second preimage: recognizing this is exactly the problem of breaking DSPR!

2.4 Why DSPR advantage is defined this way

Many security definitions require the attacker to distinguish two possibilities, each of which naturally occurs with probability 1/2. Any sort of blind guess is correct with probability 1/2. Define a as the probability of a correct output minus 1/2; a value of a noticeably larger than 0 means that the algorithm is noticeably more likely than a blind guess to be correct.

If an algorithm is noticeably *less* likely than a blind guess to be correct then one can do better by (1) replacing it with a blind guess or (2) inverting its output. The first option replaces a with $\max\{0, a\}$; the second option replaces a with $|a|$; both options have the virtue of eliminating negative values of a. Advantage

is most commonly defined as $|a|$, or alternatively as $2|a|$, the distance between the probability of a correct output and the probability of an incorrect output. These formulas are simpler than $\max\{0, a\}$.

For DSPR, the two possibilities are not naturally balanced. A second preimage exists with probability p, and almost all length-preserving (or compressing) hash functions have $p > 1/2$. Guessing 1 is correct with probability p; guessing 0 is correct with probability $1-p$; random guesses can trivially achieve any desired intermediate probability. What is interesting—and what is naturally considered in our proofs—is an algorithm $\mathcal{A}$ that guesses correctly with probability larger than p. We thus define the advantage as $\max\{0, \mathrm{Succ}(\mathcal{A}) - p\}$, where $\mathrm{Succ}(\mathcal{A})$ is the probability of $\mathcal{A}$ generating a correct output.

An algorithm $\mathcal{A}$ that guesses correctly with probability smaller than $1-p$ is also useful. We could define advantage as $\max\{0, \mathrm{Succ}(\mathcal{A}) - p, (1 - \mathrm{Succ}(\mathcal{A})) - p\}$ to take this into account, rather than leaving it to the attack developer to invert the output. However, this formula is more complicated than $\max\{0, \mathrm{Succ}(\mathcal{A}) - p\}$.

If $p < 1/2$ then, with our definitions, guessing 0 has advantage $1 - 2p > 0$. In particular, if $p = 0$ then guessing 0 has advantage 1: our definitions state that injective functions are trivially vulnerable to DSPR attacks. It might seem intuitive to define DSPR advantage as beating the best blind guess, i.e., as probability minus $\max\{p, 1-p\}$ rather than probability minus p. This, however, would break the proof that SPR $\wedge$ DSPR implies PRE: the identity function would have both SPR and DSPR but not PRE. We could add an assumption that $p \geq 1/2$, but the approach we have taken is simpler.

2.5 DSPR plus SPR implies PRE

We now present the main application of DSPR in the simplest case: We show that a second-preimage-resistant and decisional-second-preimage-resistant hash function is preimage resistant.

We first define the two reductions we use, SPfromP and DSPfromP, and then give a theorem statement analyzing success probabilities. The algorithm $\mathrm{SPfromP}(\mathrm{H}, \mathcal{A})$ is the standard algorithm that tries to break SPR using an algorithm $\mathcal{A}$ that tries to break PRE. The algorithm $\mathrm{DSPfromP}(\mathrm{H}, \mathcal{A})$ is a variant that tries to break DSPR. Each algorithm uses one computation of H, one call to $\mathcal{A}$, and (for DSPfromP) one string comparison, so each algorithm has essentially the same cost as $\mathcal{A}$ if H is efficient.

Definition 6 (SPfromP). *Let* H *be a hash function. Let* $\mathcal{A}$ *be an algorithm. Then* $\mathrm{SPfromP}(\mathrm{H}, \mathcal{A})$ *is the algorithm that, given* $x \in \mathcal{X}$*, outputs* $\mathcal{A}(\mathrm{H}(x))$.

Definition 7 (DSPfromP). *Let* H *be a hash function. Let* $\mathcal{A}$ *be an algorithm. Then* $\mathrm{DSPfromP}(\mathrm{H}, \mathcal{A})$ *is the algorithm that, given* $x \in \mathcal{X}$*, outputs* $[x \neq \mathcal{A}(\mathrm{H}(x))]$.

This output is 0 if $\mathcal{A}(\mathrm{H}(x))$ returns the preimage x that was already known for $\mathrm{H}(x)$, and 1 otherwise. Note that the 0 case provides some reason to believe that there is only one preimage. If there are $i > 1$ preimages then x, which is

not known to $\mathcal{A}$ except via $\mathrm{H}(x)$, is information-theoretically hidden in a set of size i, so $\mathcal{A}$ cannot return x with probability larger than $1/i$.

Theorem 8 (DSPR $\wedge$ SPR $\Rightarrow$ PRE). *Let* H *be a hash function. Let* $\mathcal{A}$ *be an algorithm. Then*

$$\mathrm{Succ}^{\mathrm{PRE}}_{\mathrm{H}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{B}) + 3 \cdot \mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{C})$$

where $\mathcal{B} = \mathrm{DSPfromP}(\mathrm{H}, \mathcal{A})$ *and* $\mathcal{C} = \mathrm{SPfromP}(\mathrm{H}, \mathcal{A})$.

Proof. This is a special case of Theorem 25 below, modulo a change of syntax. To apply Theorem 25 we set $\mathcal{K}$ to be $\{()\}$, where $()$ is the empty string. The change of syntax views a keyed hash function with an empty key as an unkeyed hash function. □

3 The second-preimage-exists probability

This section mathematically analyzes $\mathrm{SPprob}(\mathrm{H})$, the probability that a uniform random input to H has a second preimage. The DSPR advantage of any attacker is information-theoretically bounded by $1 - \mathrm{SPprob}(\mathrm{H})$.

3.1 Simple cases

In retrospect, the heart of the Rogaway–Shrimpton SPR-PRE reduction [15, Theorem 7] is the observation that $\mathrm{SPprob}(\mathrm{H})$ is very close to 1 for all highly compressing hash functions H. See Theorem 9. We show that $\mathrm{SPprob}(\mathrm{H})$ is actually *equal* to 1 for almost all hash functions H that compress more than a few bits; see Theorem 11.

Theorem 9 (lower bound on SPprob in the compressing case). *If* H *is a hash function and* $M > N$ *then* $\mathrm{SPprob}(\mathrm{H}) \geq 1 - (N-1)/M$.

The maximum possible DSPR advantage in this case is $(N-1)/M$. For example, if $M > 1$ and $N = 1$ then $\mathrm{SPprob}(\mathrm{H}) = 1$ and the DSPR advantage is always 0. As another example, a 320-bit-to-256-bit hash function H has $\mathrm{SPprob}(\mathrm{H}) \geq 1 - (2^{256} - 1)/2^{320}$, and the DSPR advantage is at most $(2^{256} - 1)/2^{320} < 1/2^{64}$.

Proof. Define I as the set of elements of $\mathcal{X}$ that have no second preimages; i.e., the set of $x \in \mathcal{X}$ such that $|\mathrm{H}^{-1}(\mathrm{H}(x))| = 1$.

The image set $\mathrm{H}(I) \subseteq \mathcal{Y}$ has size $|I|$, so $|I| \leq |\mathcal{Y}| = N < M = |\mathcal{X}|$. The complement $\mathcal{X} - I$ is thus nonempty, so the image set $\mathrm{H}(\mathcal{X} - I)$ is also nonempty. This image set cannot overlap $\mathrm{H}(I)$: if $\mathrm{H}(x') = \mathrm{H}(x)$ with $x' \in \mathcal{X} - I$ and $x \in I$ then x', x are distinct elements of $\mathrm{H}^{-1}(\mathrm{H}(x))$, but $|\mathrm{H}^{-1}(\mathrm{H}(x))| = 1$ by definition of I. Hence $|I| \leq N - 1$.

By definition $\mathrm{SPprob}(\mathrm{H})$ is the probability that $|\mathrm{H}^{-1}(\mathrm{H}(x))| \geq 2$ where x is a uniform random element of $\mathcal{X}$, i.e., the probability that x is not in I. This is at least $1 - (N-1)/M$. □

Theorem 10 (average of SPprob). *The average of* SPprob(H) *over all hash functions* H *is* $1-(1-1/N)^{M-1}$.

For example, the average is $1-(1-1/2^{256})^{2^{256}-1} \approx 1-1/e \approx 0.63212$ if $M = 2^{256}$ and $N = 2^{256}$; see also Theorem 12. The average converges rapidly to 1 as N/M drops: for example, the average is approximately $1-2^{-369.33}$ if $M = 2^{256}$ and $N = 2^{248}$, and is approximately $1-2^{-94548}$ if $M = 2^{256}$ and $N = 2^{240}$, while the lower bounds from Theorem 9 are approximately $1-2^{-16}$ and approximately $1-2^{-32}$ respectively.

The average converges to 0 as N/M increases. The average crosses below $1/2$, making DSPR trivially breakable for the average function, as N/M increases past about $1/\log 2 \approx 1.4427$.

Proof. For each $x \in \mathcal{X}$, there are exactly $N(N-1)^{M-1}$ hash functions H for which x has no second preimages. Indeed, there are N choices of $\mathrm{H}(x)$, and then for each $i \in \mathcal{X} - \{x\}$ there are $N-1$ choices of $\mathrm{H}(i) \in \mathcal{Y} - \{\mathrm{H}(x)\}$.

Hence there are exactly $M(N^M - N(N-1)^{M-1})$ pairs (H, x) where x has a second preimage under H; i.e., the total of SPprob(H) over all N^M hash functions H is $N^M - N(N-1)^{M-1}$; i.e., the average of SPprob(H) over all hash functions H is $1 - N(N-1)^{M-1}/N^M = 1-(1-1/N)^{M-1}$. □

Theorem 11 (how often SPprob is 1). *If* H *is a uniform random hash function then* SPprob(H) $= 1$ *with probability at least* $1-M(1-1/N)^{M-1}$.

This is content-free in the length-preserving case but becomes more useful as N/M drops. For example, if $M = 2^{256}$ and $N = 2^{248}$, then the chance of SPprob(H) < 1 is at most $2^{256}(1-1/2^{248})^{2^{256}-1} \approx 2^{-113.33}$. Hence almost all 256-bit-to-248-bit hash functions have second preimages for all inputs, and therefore have perfect DSPR (DSPR advantage 0) against all attacks.

Proof. Write q for the probability that SPprob(H) $= 1$. Then SPprob(H) $\leq 1-1/M$ with probability $1-q$. The point here is that SPprob(H) is a probability over M inputs, and is thus a multiple of $1/M$.

The average of SPprob(H) is at most $q+(1-q)(1-1/M) = 1-(1-q)/M$. By Theorem 10, this average is exactly $1-(1-1/N)^{M-1}$. Hence $1-(1-1/N)^{M-1} \leq 1-(1-q)/M$; i.e., $q \geq 1-M(1-1/N)^{M-1}$. □

Theorem 12 (average of SPprob vs. $1-1/e$ in the length-preserving case). *If* $M = N > 1$ *then the average* a *of* SPprob(H) *over all hash functions* H *has* $1-(1/e)N/(N-1) < a < 1-1/e$.

The big picture is that almost all length-preserving hash functions H have SPprob(H) close to $1-1/e$. This theorem states part of the picture: the average of SPprob(H) is extremely close to $1-1/e$ if N is large. Subsequent theorems fill in the rest of the picture.

Proof. The point is that $(N/(N-1))^{N-1} < e < (N/(N-1))^N$ for $N \geq 2$. See, e.g., [4]. In other words, $e(N-1)/N < (N/(N-1))^{N-1} < e$. Invert to see that $1/e < (1-1/N)^{N-1} < (1/e)N/(N-1)$. Finally, the average a of SPprob(H) is $1-(1-1/N)^{N-1}$ by Theorem 10. □

3.2 How SPprob varies

This subsection analyzes the distribution of SPprob(H) as H varies. Theorem 14 amounts to an algorithm that computes the probability of each possible value of SPprob(H) in time polynomial in $M+N$. Theorem 16, used in Section 3.3, gives a simple upper bound on each term in the probability.

Theorem 13. *Let a, b be nonnegative integers. Define $c(a,b)$ as the coefficient of x^b in the power series $b!(e^x-1-x)^a/a!$. Then $a!c(a,b)$ is the number of functions from $\{1,\ldots,b\}$ to $\{1,\ldots,a\}$ for which each of $\{1,\ldots,a\}$ has at least two preimages.*

This is a standard example of "generatingfunctionology". See, e.g., [16, sequence A000478, "E.g.f."] for $a=3$ and [16, sequence A058844, "E.g.f."] for $a=4$.

Note that $c(a,b)=0$ for $b<2a$, and that $c(0,b)=0$ for $b>0$.

Proof. Choose integers $i_1,\ldots,i_a \geq 2$ with $i_1+\cdots+i_a=b$, and consider any function f built as follows. Let π be a permutation of $\{1,\ldots,b\}$. Define $f(\pi(1)) = f(\pi(2)) = \ldots = f(\pi(i_1)) = 1$; note that 1 has $i_1 \geq 2$ preimages. Define $f(\pi(i_1+1)) = f(\pi(i_1+2)) = \ldots = f(\pi(i_1+i_2)) = 2$; note that 2 has $i_2 \geq 2$ preimages. Et cetera.

There are exactly $b!$ choices of π, producing exactly $b!/i_1!\cdots i_a!$ choices of f. This covers all functions f for which 1 has exactly i_1 preimages, 2 has exactly i_2 preimages, etc.

The total number of functions being counted is thus the sum of $b!/i_1!\cdots i_a!$ over all $i_1,\ldots,i_a \geq 2$ with $i_1+\cdots+i_a=b$.

For comparison, the power series e^x-1-x is $\sum_{i\geq 2} x^i/i!$, so

$$(e^x-1-x)^a = \sum_{i_1,\ldots,i_a \geq 2} x^{i_1+\cdots+i_a}/i_1!\cdots i_a!.$$

The coefficient of x^b is the sum of $1/i_1!\cdots i_a!$ over all $i_1,\ldots,i_a \geq 2$ with $i_1+\cdots+i_a=b$. By definition $a!c(a,b)/b!$ is this coefficient, so $a!c(a,b)$ is the sum of $b!/i_1!\cdots i_a!$ over all $i_1,\ldots,i_a \geq 2$ with $i_1+\cdots+i_a=b$. □

Theorem 14 (exact distribution of SPprob). *There are exactly*

$$\binom{M}{j} \sum_{j \leq k \leq N} c(k-j, M-j) \frac{N!}{(N-k)!}$$

hash functions H *with* SPprob(H) $= 1-j/M$.

Fig. 1. Cumulative distribution of SPprob(H) for $M = N = 1$; $M = N = 2$; $M = N = 4$; $M = N = 8$; $M = N = 16$; $M = N = 32$; $M = N = 64$. The probabilities that SPprob(H) ≤ 0.5 are, respectively, 1; 0.5; 0.65625; $\approx$0.417366; $\approx$0.233331; $\approx$0.100313; and $\approx$0.023805. As $N \to \infty$ with $M = N$, the distribution converges to a vertical line at $1 - 1/e$.

The summand is 0 if $k > (M + j)/2$, i.e., if $M - j < 2(k - j)$, since then $c(k - j, M - j) = 0$. The summand is also 0 if $k = j$ and $M > j$, since then $c(0, M - j) = 0$.

In particular, if $j > N$ then SPprob(H) $= 1 - j/M$ with probability 0; and if $j = N < M$ then SPprob(H) $= 1 - j/M$ with probability 0. This calculation shows that Theorem 14 includes Theorem 9.

The distribution of $M - j$ here, for a uniform random hash function H, is equal to the distribution of "K_1" in [3, formula (2.21)], but the formulas are different. The sum in [3, formula (2.21)] is an alternating sum with cancellation between large terms. The sum in Theorem 14 is a sum of nonnegative terms; this is important for our asymptotic analysis.

Figure 1 shows the cumulative distribution of SPprob(H) when $M = N \in \{1, 2, 4, 8, 16, 32, 64\}$. Each graph ranges from 0 through 1 horizontally, and from 0 through 1 vertically. At horizontal position p, the (maximum) vertical position is the probability that SPprob(H) $\leq p$. We computed these probabilities using Theorem 14.

Proof. We count the hash functions that (1) have exactly $k \geq j$ outputs and (2) have exactly j inputs with no second preimages.

Choose the j inputs. There are $\binom{M}{j}$ ways to do this.

Choose a partition of the N outputs into

- j outputs that will be used (without second preimages) by the j inputs;
- $k - j$ outputs that will be used (with second preimages) by the other $M - j$ inputs; and
- $N - k$ outputs that will not be used.

There are $N!/j!(k - j)!(N - k)!$ ways to do this.

Choose an injective function from the j inputs to the j outputs. There are $j!$ ways to do this.

Choose a function from the other $M - j$ inputs to the other $k - j$ outputs for which each of these $k - j$ outputs has at least two preimages. By Theorem 13, there are $(k - j)!c(k - j, M - j)$ ways to do this.

This produces a hash function that, as desired, has exactly k outputs and has exactly j inputs with no second preimages. Each such function is produced exactly once. Hence there are $\binom{M}{j}c(k - j, M - j)N!/(N - k)!$ such functions.

Finally, sum over k to see that there are

$$\binom{M}{j} \sum_{j \le k \le N} c(k-j, M-j) \frac{N!}{(N-k)!}$$

hash functions H that have exactly j inputs with no second preimages, i.e., hash functions H that have SPprob(H) $= 1 - j/M$. □

Theorem 15. *Let a, b be positive integers. Let ζ be a positive real number. Assume that $b/a = \zeta + \zeta^2/(e^\zeta - 1 - \zeta)$. Then $c(a,b) \le (e^\zeta - 1 - \zeta)^a \zeta^{-b} b!/a!$.*

Our proof applies [5, Proposition VIII.7], which is an example of the "saddle-point method" in analytic combinatorics. With more work one can use the saddle-point method to improve bounds by a polynomial factor, but our main concern here is exponential factors.

Proof. Define $B(z) = \sum_{i \ge 2} z^{i-2}/i! = 1/2 + z/6 + z^2/24 + \cdots$. Note that $z^2 B(z) = e^z - 1 - z$, and that $zB'(z) = \sum_{i \ge 3} (i-2) z^{i-2}/i! = (z-2)B(z) + 1$. Also define $A(z) = 1$; $R = \infty$; $T = \infty$; $N = b - 2a$; $n = a$; and $\lambda = b/a - 2$.

Check the hypotheses of [5, Proposition VIII.7]: A and B are analytic functions of the complex variable z, with all coefficients nonnegative; $B(0) = 1/2 \neq 0$; the coefficient of z in B is nonzero; the radius of convergence of B is ∞; the radius of convergence of A is also ∞; the limit of $xB'(x)/B(x)$ as $x \to \infty$ is ∞; λ is a positive real number; $N = \lambda n$; and $\zeta B'(\zeta)/B(\zeta) = \zeta - 2 + 1/B(\zeta) = b/a - 2 = \lambda$.

Now [5, Proposition VIII.7] states that the coefficient of z^N in $A(z)B(z)^n$ is at most $A(\zeta)B(\zeta)^n \zeta^{-N}$; i.e., the coefficient of z^{b-2a} in $((e^z - 1 - z)/z^2)^a$ is at most $B(\zeta)^a \zeta^{2a-b}$; i.e., the coefficient of z^b in $(e^z - 1 - z)^a$ is at most $B(\zeta)^a \zeta^{2a-b}$. Hence $c(a,b) \le B(\zeta)^a \zeta^{2a-b} b!/a! = (e^\zeta - 1 - \zeta)^a \zeta^{-b} b!/a!$. □

Theorem 16 (exponential convergence of SPprob). *Let j be an integer with $0 < j < M$. Let k be an integer with $j < k < N$. Define $\mu = M/N$, $\alpha = j/N$, and $\kappa = k/N$. Let ζ be a positive real number. Assume that $(\mu - \alpha)/(\kappa - \alpha) = \zeta + \zeta^2/(e^\zeta - 1 - \zeta)$. Then*

$$\binom{M}{j} c(k-j, M-j) \frac{N!}{(N-k)!} \le \frac{M! N! e^N \tau^N}{N^N}$$

where $\tau = (e^\zeta - 1 - \zeta)^{\kappa - \alpha} / \zeta^{\mu - \alpha} \alpha^\alpha (\kappa - \alpha)^{\kappa - \alpha} (1 - \kappa)^{1 - \kappa}$.

The proof combines Theorem 15 with the weak Stirling bound $N! \ge (N/e)^N$. See [14] for a proof that $(N/e)^N \sqrt{2\pi N} e^{1/(12N+1)} \le N! \le (N/e)^N \sqrt{2\pi N} e^{1/12N}$.

Proof. Define $a = k - j$ and $b = M - j$. Then a and b are positive integers, and $b/a = (\mu - \alpha)/(\kappa - \alpha) = \zeta + \zeta^2/(e^\zeta - 1 - \zeta)$, so

$$c(k-j, M-j) = c(a,b) \le \frac{(e^\zeta - 1 - \zeta)^a b!}{\zeta^b a!}$$

by Theorem 15, so

$$\binom{M}{j} c(k-j, M-j) \frac{N!}{(N-k)!} \le \frac{M!N!(e^\zeta - 1 - \zeta)^a}{j!\zeta^b a!(N-k)!}$$
$$\le \frac{M!N!(e^\zeta - 1 - \zeta)^a}{(j/e)^j \zeta^b (a/e)^a ((N-k)/e)^{N-k}}$$

by the weak Stirling bound. Now substitute $j = \alpha N$, $k = \kappa N$, $a = (\kappa - \alpha)N$, and $b = (\mu - \alpha)N$:

$$\binom{M}{j} c(k-j, M-j) \frac{N!}{(N-k)!}$$
$$\le \frac{M!N!(e^\zeta - 1 - \zeta)^{(\kappa-\alpha)N}}{(\alpha N/e)^{\alpha N} \zeta^{(\mu-\alpha)N} ((\kappa-\alpha)N/e)^{(\kappa-\alpha)N} ((N-\kappa N)/e)^{N-\kappa N}}$$
$$= \frac{M!N!(e^\zeta - 1 - \zeta)^{(\kappa-\alpha)N}}{(N/e)^N \alpha^{\alpha N} \zeta^{(\mu-\alpha)N} (\kappa-\alpha)^{(\kappa-\alpha)N} (1-\kappa)^{N-\kappa N}} = \frac{M!N!\tau^N}{(N/e)^N}$$

as claimed. □

3.3 Maximization

This subsection formalizes and proves our claim that SPprob(H) is close to $1 - 1/e$ for almost all length-preserving hash functions H: as N increases (with $M = N$), the distributions plotted in Figure 1 converge to a vertical line.

The basic idea here is that τ in Theorem 16 is noticeably below e when j/N is noticeably below or above $1/e$. One can quickly see this by numerically plotting τ as a function of α and ζ: note that any choice of α and ζ (along with $\mu = 1$) determines $\kappa = \alpha + (\mu - \alpha)/(\zeta + \zeta^2/(e^\zeta - 1 - \zeta))$ and thus determines τ. The plot suggests that $\zeta = 1$ maximizes τ for each α, and that moving α towards $1/e$ from either side increases τ up to its maximum value e. One could use interval arithmetic to show, e.g., that $\tau/e < 0.998$ for $j/N > 0.4$, but the required number of subintervals would rapidly grow as j/N approaches $1/e$. Our proof also handles some corner cases that are not visible in the plot.

Theorem 17. *Let $\mu, \alpha, \kappa, \zeta$ be positive real numbers with $\alpha < \mu$; $\alpha < \kappa < 1$; and $(\mu-\alpha)/(\kappa-\alpha) = \zeta + \zeta^2/(e^\zeta - 1 - \zeta)$. First, there is a unique positive real number Z such that $Z(e^Z - 1)/(e^Z - Z) = (\mu-\alpha)/(1-\alpha)$. Second, there is a unique real number K such that $\alpha < K < 1$ and $(\mu - \alpha)/(K - \alpha) = Z + Z^2/(e^Z - 1 - Z)$. Third,*

$$\frac{(e^\zeta - 1 - \zeta)^{\kappa-\alpha}}{\zeta^{\mu-\alpha} \alpha^\alpha (\kappa-\alpha)^{\kappa-\alpha} (1-\kappa)^{1-\kappa}} \le \frac{(e^Z - 1 - Z)^{K-\alpha}}{Z^{\mu-\alpha} \alpha^\alpha (K-\alpha)^{K-\alpha} (1-K)^{1-K}}.$$

Fourth, if $\mu = 1$ then

$$\frac{(e^Z - 1 - Z)^{K-\alpha}}{Z^{\mu-\alpha} \alpha^\alpha (K-\alpha)^{K-\alpha} (1-K)^{1-K}} = \frac{(e-1)^{1-\alpha}}{\alpha^\alpha (1-\alpha)^{1-\alpha}}.$$

Proof. See full version of this paper online. □

Theorem 18. *Let α, κ, ζ, A be positive real numbers. Assume that $\alpha < \kappa < 1$; that $(1-\alpha)/(\kappa-\alpha) = \zeta + \zeta^2/(e^\zeta - 1 - \zeta)$; and that $1/e \le A \le \alpha$ or $\alpha \le A \le 1/e$. Then*

$$\frac{(e^\zeta - 1 - \zeta)^{\kappa-\alpha}}{\zeta^{1-\alpha}\alpha^\alpha(\kappa-\alpha)^{\kappa-\alpha}(1-\kappa)^{1-\kappa}} \le \frac{(e-1)^{1-A}}{A^A(1-A)^{1-A}}.$$

Proof. See full version of this paper online. □

Theorem 19. *Assume that $M = N$. Let A be a real number with $0 < A < 1$. Let* H *be a uniform random hash function. If $A > 1/e$, define E as the event that* $\mathrm{SPprob}(\mathrm{H}) \le 1 - A$. *If $A \le 1/e$, define E as the event that* $\mathrm{SPprob}(\mathrm{H}) \ge 1 - A$. *Then E occurs with probability at most $(T/e)^N 2\pi N^2(N+1)e^{1/6N}$ where*

$$T = \max\{1 + \sqrt{2}, (e-1)^{1-A}/A^A(1-A)^{1-A}\}.$$

Any $A \neq 1/e$ has $T/e < 1$, and then the important factor in the probability for large N is $(T/e)^N$. For example, if $A = 0.4$ then $T/e < 0.99780899$, so $(T/e)^N$ is below $1/2^{2^{247}}$ for $N = 2^{256}$. As another example, if $A = 0.37$ then $T/e < 0.99999034$, so $(T/e)^N$ is below $1/2^{2^{239}}$ for $N = 2^{256}$.

Proof. See full version of this paper online. □

4 DSPR for keyed hash functions

In this section we lift the discussion to the setting of keyed hash functions. We model keyed hash functions as functions $\mathrm{H} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that take a dedicated key as additional input argument. One might also view a keyed hash function as a family of hash functions where elements of the family H are obtained by fixing the first input argument which we call the function key. We write $\mathrm{H}_k \stackrel{\text{def}}{=} \mathrm{H}(k, \cdot)$ for the function that is obtained from H by fixing the first input as $k \in \mathcal{K}$.

We assume that $\mathcal{K}$, like $\mathcal{X}$ and $\mathcal{Y}$, is a nonempty finite set of finite-length bit strings. We define the *compressing*, *expanding*, and *length-preserving* cases as the cases $|\mathcal{X}| > |\mathcal{Y}|$, $|\mathcal{X}| < |\mathcal{Y}|$, and $|\mathcal{X}| = |\mathcal{Y}|$ respectively, ignoring the size of $\mathcal{K}$.

We recall the definitions of preimage and second-preimage resistance for keyed hash functions for completeness:

Definition 20 (PRE for keyed hash functions). *The success probability of adversary $\mathcal{A}$ against the preimage resistance of a keyed hash function* H *is*

$$\mathrm{Succ}^{\mathrm{PRE}}_{\mathrm{H}}(\mathcal{A}) \stackrel{def}{=} \mathsf{Pr}\left[x \leftarrow_R \mathcal{X}; k \leftarrow_R \mathcal{K}; x' \leftarrow \mathcal{A}(\mathrm{H}_k(x), k) : \mathrm{H}_k(x) = \mathrm{H}_k(x')\right].$$

Definition 21 (SPR for keyed hash functions). *The success probability of adversary $\mathcal{A}$ against the second-preimage resistance of a keyed hash function* H *is*

$$\mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{A}) \stackrel{def}{=} \mathsf{Pr}\left[x \leftarrow_R \mathcal{X}; k \leftarrow_R \mathcal{K}; x' \leftarrow \mathcal{A}(x, k) : \mathrm{H}_k(x) = \mathrm{H}_k(x') \wedge x \neq x'\right].$$

Our definition of DSPR for a keyed hash function H relies on the second-preimage-exists predicate SPexists and the second-preimage-exists probability SPprob for the functions H_k. If H is chosen uniformly at random then, for large N and any reasonable size of $\mathcal{K}$, it is very likely that *all* of the functions H_k have $\mathrm{SPprob}(\mathrm{H}_k)$ close to $1 - 1/e$; see Theorem 19.

Definition 22 (DSPR for keyed hash functions). *Let $\mathcal{A}$ be an algorithm that always outputs 0 or 1. The advantage of $\mathcal{A}$ against the decisional second-preimage resistance of a keyed hash function* H *is*

$$\mathrm{Adv}^{\mathrm{DSPR}}_{\mathcal{H}}(\mathcal{A}) \stackrel{def}{=} \max\{0, \mathsf{Pr}[x \leftarrow_R \mathcal{X}, k \leftarrow_R \mathcal{K}, b \leftarrow \mathcal{A}(x,k) : P_k(x) = b] - p\}$$

where $P_k = \mathrm{SPexists}(\mathrm{H}_k)$ and p is the average of $\mathrm{SPprob}(\mathrm{H}_k)$ over all k.

As an example, consider the keyed hash function H with $\mathcal{X} = \mathcal{Y} = \{0,1\}^{256}$, $\mathcal{K} = \{0,1\}$, $\mathrm{H}_0(x) = x$, and $\mathrm{H}_1(x) = (x_1, x_2, \ldots, x_{255}, 0)$ where the x_i denote the bits of x. Then $P_k(x) = k$, $\mathrm{SPprob}(\mathrm{H}_k) = k$, and $p = 1/2$. A trivial adversary that outputs k has success probability 1 and thus DSPR advantage $1/2$, the maximum possible DSPR advantage: this function does not have decisional second-preimage resistance.

It might seem natural to define SPprob(H) as the average mentioned in the theorem. However, we will see later in the multi-target context that p is naturally replaced by a more complicated quantity influenced by the algorithm.

4.1 DSPR plus SPR implies PRE

Before we show that DSPR is hard in the QROM (see Section 5), we give a generalization of Theorem 8 for keyed hash functions. This theorem states that second-preimage and decisional second-preimage resistance together imply preimage resistance.

As in Theorem 8, we first define the two reductions we use, and then give a theorem statement analyzing success probabilities. The special case that $\mathcal{K} = \{()\}$, where () means the empty string, is the same as Theorem 8, modulo syntactic replacements such as replacing the pair $((), x)$ with x.

Definition 23 (SPfromP for keyed hash functions). *Let* H *be a keyed hash function. Let $\mathcal{A}$ be an algorithm. Then* $\mathrm{SPfromP}(\mathrm{H}, \mathcal{A})$ *is the algorithm that, given $(k, x) \in \mathcal{K} \times \mathcal{X}$, outputs $\mathcal{A}(\mathrm{H}_k(x), k)$.*

Definition 24 (DSPfromP for keyed hash functions). *Let* H *be a keyed hash function. Let $\mathcal{A}$ be an algorithm. Then* $\mathrm{DSPfromP}(\mathrm{H}, \mathcal{A})$ *is the algorithm that, given $(k, x) \in \mathcal{K} \times \mathcal{X}$, outputs $[x \neq \mathcal{A}(\mathrm{H}_k(x), k)]$.*

Theorem 25 (DSPR $\wedge$ SPR $\Rightarrow$ PRE for keyed hash functions). *Let* H *be a keyed hash function. Let $\mathcal{A}$ be an algorithm. Then*

$$\mathrm{Succ}^{\mathrm{PRE}}_{\mathrm{H}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{B}) + 3 \cdot \mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{C})$$

where $\mathcal{B} = \mathrm{DSPfromP}(\mathrm{H}, \mathcal{A})$ and $\mathcal{C} = \mathrm{SPfromP}(\mathrm{H}, \mathcal{A})$.

Proof. To analyze the success probabilities, we split the universe of possible events into mutually exclusive events across two dimensions: the number of preimages of $\mathrm{H}_k(x)$, and whether $\mathcal{A}$ succeeds or fails in finding a preimage. Specifically, define

$$S_i \stackrel{\text{def}}{=} \left[\left|\mathrm{H}_k^{-1}(\mathrm{H}_k(x))\right| = i \wedge \mathrm{H}_k(\mathcal{A}(\mathrm{H}_k(x), k)) = \mathrm{H}_k(x)\right]$$

as the event that there are exactly i preimages and that $\mathcal{A}$ succeeds, and define

$$F_i \stackrel{\text{def}}{=} \left[\left|\mathrm{H}_k^{-1}(\mathrm{H}_k(x))\right| = i \wedge \mathrm{H}_k(\mathcal{A}(\mathrm{H}_k(x), k)) \neq \mathrm{H}_k(x)\right]$$

as the event that there are exactly i preimages and that $\mathcal{A}$ fails.

Note that there are only finitely many i for which the events S_i and F_i can occur, namely $i \in \{1, 2, \ldots, M\}$. All sums below are thus finite sums.

Define s_i and f_i as the probabilities of S_i and F_i respectively. The probability space here includes the random choices of x and k, and any random choices made inside $\mathcal{A}$. The conditional probabilities mentioned below are conditional probabilities given S_i.

PRE success probability. By definition, $\mathrm{Succ}_{\mathrm{H}}^{\mathrm{PRE}}(\mathcal{A})$ is the probability of the event that $\mathrm{H}_k(x) = \mathrm{H}_k(\mathcal{A}(\mathrm{H}_k(x), k))$. This event is the union of S_i, so $\mathrm{Succ}_{\mathrm{H}}^{\mathrm{PRE}}(\mathcal{A}) = \sum_i s_i$.

DSPR success probability. Define $P_k = \mathrm{SPexists}(\mathrm{H}_k)$. For the $i = 1$ cases, we have $P_k(x) = 0$ by definition of SPexists, so $\mathcal{B}$ is correct if and only if $\mathcal{A}$ succeeds. For the $i > 1$ cases, we have $P_k(x) = 1$, so $\mathcal{B}$ is correct as long as $\mathcal{A}$ does not output x. There are two disjoint ways for this to occur:

- $\mathcal{A}$ succeeds (case S_i). Then $\mathcal{A}$ outputs x with conditional probability exactly $\frac{1}{i}$, since x is information-theoretically hidden in a set of size i; so there is conditional probability exactly $\frac{i-1}{i}$ that $\mathcal{A}$ does not output x.
- $\mathcal{A}$ fails (case F_i). Then $\mathcal{A}$ does not output x.

Together we get

$$\Pr[\mathcal{B}(x, k) = P_k(x)] = s_1 + \sum_{i>1} \frac{i-1}{i} s_i + \sum_{i>1} f_i.$$

DSPR advantage. By definition $\mathrm{Adv}_{\mathrm{H}}^{\mathrm{DSPR}}(\mathcal{B}) = \max\{0, \Pr[\mathcal{B}(x, k) = P_k(x)] - p\}$ where p is the average of $\mathrm{SPprob}(\mathrm{H}_k)$ over all k.

By definition $\mathrm{SPprob}(\mathrm{H}_k)$ is the probability over all choices of x that x has a second preimage under H_k. Hence p is the same probability over all choices of x and k; i.e., $p = \sum_{i>1} s_i + \sum_{i>1} f_i$. Now subtract:

$$\begin{aligned}
\mathrm{Adv}_{\mathrm{H}}^{\mathrm{DSPR}}(\mathcal{B}) &= \max\{0, \Pr[\mathcal{B}(x, k) = P_k(x)] - p\} \\
&\geq \Pr[\mathcal{B}(x, k) = P_k(x)] - p \\
&= s_1 + \sum_{i>1} \frac{i-1}{i} s_i + \sum_{i>1} f_i - \sum_{i>1} s_i - \sum_{i>1} f_i \\
&= s_1 - \sum_{i>1} \frac{1}{i} s_i.
\end{aligned}$$

SPR success probability. For the $i = 1$ cases, $\mathcal{C}$ never succeeds. For the $i > 1$ cases, $\mathcal{C}$ succeeds if and only if $\mathcal{A}$ succeeds and returns a value different from x. This happens with conditional probability $\frac{i-1}{i}$ for the same reason as above. Hence

$$\mathrm{Succ}^{\mathrm{SPR}}_{\mathrm{H}}(\mathcal{C}) = \sum_{i>1} \frac{i-1}{i} s_i.$$

Combining the probabilities. We have

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

as claimed.

The formal structure of the proof is concluded at this point, but we close with some informal comments on how to interpret this proof. What happens is the following. The cases where the plain reduction from SPR ($\mathcal{C}$ in the above) fails are the S_1 cases, i.e., $\mathcal{A}$ succeeds when there is only one preimage. If the probability that they occur (s_1) gets close to $\mathcal{A}$'s total success probability, the success probability of $\mathcal{C}$ goes towards zero. However, s_1 translates almost directly to the DSPR advantage of $\mathcal{B}$. This is also intuitively what we want. For a brute-force attack, one would expect s_1 to be less than a $1 - p$ fraction of $\mathcal{A}$'s success probability. If it is higher, this allows to distinguish. On the extreme: If $s_1 = s$, then $\mathcal{B}$'s DSPR advantage is exactly $\mathcal{A}$'s success probability and the reduction is tight. If $s_1 = 0$, $\mathcal{B}$ has no advantage over guessing, but $\mathcal{C}$ wins with at least half the success probability of $\mathcal{A}$ (in this case our generic $1/3$ bound can be tightened). As mentioned above, in general one would expect s_1 to be a recognizable fraction of s but clearly smaller than s. In these cases, both reductions succeed. □

5 DSPR is hard in the QROM

So far we have highlighted relations between DSPR and other hash function properties. However, all this is useful only if DSPR is a hard problem for the hash functions we are interested in. In the following we show that DSPR is hard for a quantum adversary as long as the hash function behaves like a random function. We do this presenting a lower bound on the quantum query complexity for DSPR.

To make previous results reusable, we first need a result that relates the success probability of an adversary in a biased distinguishing game like the DSPR game to its success probability in the balanced version of the game.

Theorem 26. *Let B_λ denote the Bernoulli distribution that assigns probability λ to 1, $\mathcal{X}_b$ for $b \in \{0,1\}$ a non-empty set,*

$$\mathrm{Succ}_\lambda(\mathcal{A}) \stackrel{def}{=} \mathsf{Pr}[b \leftarrow_R B_\lambda; x \leftarrow_R \mathcal{X}_b; g \leftarrow \mathcal{A}(x) : g = b],$$

and

$$\mathrm{Adv}_\lambda(\mathcal{A}) \stackrel{def}{=} \max\{0, \mathrm{Succ}_\lambda(\mathcal{A}) - \lambda\}$$

Then for $p \geq 1/2$ we have

$$\mathrm{Adv}_p(\mathcal{A}) \leq p\,|\mathsf{Pr}[x \leftarrow_R \mathcal{X}_1 : 1 \leftarrow \mathcal{A}(x)] - \mathsf{Pr}[x \leftarrow_R \mathcal{X}_0 : 1 \leftarrow \mathcal{A}(x)]|.$$

More specifically

$$\mathrm{Succ}_{\frac{1}{2}}(\mathcal{A}) \geq \frac{1}{2p}\mathrm{Succ}_p(\mathcal{A}), \quad \mathrm{Adv}_{\frac{1}{2}}(\mathcal{A}) \geq \frac{1}{2p}\mathrm{Adv}_p(\mathcal{A}),$$

and

$$\frac{1}{2}\,|\mathsf{Pr}[x \leftarrow_R \mathcal{X}_1 : 1 \leftarrow \mathcal{A}(x)] - \mathsf{Pr}[x \leftarrow_R \mathcal{X}_0 : 1 \leftarrow \mathcal{A}(x)]| \geq \mathrm{Adv}_{\frac{1}{2}}(\mathcal{A})$$

Proof. Let $s_0 = \mathsf{Pr}[b = 0 \wedge g = 0] = \mathsf{Pr}[b = g \mid b = 0]\,\mathsf{Pr}[b = 0]$ and $s'_0 = \mathsf{Pr}[b = g \mid b = 0]$. Define $s_1 = \mathsf{Pr}[b = 1 \wedge g = 1] = \mathsf{Pr}[b = g \mid b = 1]\,\mathsf{Pr}[b = 1]$ and $s'_1 = \mathsf{Pr}[b = g \mid b = 1]$, accordingly. Then

$$\begin{aligned}\frac{1}{2p}\mathrm{Succ}_p(\mathcal{A}) &= \frac{1}{2p}((1-p)s'_0 + ps'_1)\\ &= \frac{1-p}{p}\cdot\frac{1}{2}s'_0 + \frac{1}{2}s'_1 \leq \frac{1}{2}s'_0 + \frac{1}{2}s'_1 = \mathrm{Succ}_{\frac{1}{2}}(\mathcal{A}),\end{aligned}$$

where we used $p \geq 1/2$. Now, for a zero advantage in the biased game the second sub-claim is trivially true. For a non-zero advantage $\mathrm{Adv}_p(\mathcal{A})$ we get

$$\begin{aligned}\mathrm{Adv}_p(\mathcal{A}) &= \max\{0, \mathrm{Succ}_p(\mathcal{A}) - p\}\\ \mathrm{Adv}_p(\mathcal{A}) + p &= \mathrm{Succ}_p(\mathcal{A})\\ \frac{1}{2p}(\mathrm{Adv}_p(\mathcal{A}) + p) &\leq \mathrm{Succ}_{\frac{1}{2}}(\mathcal{A})\\ \frac{1}{2p}\mathrm{Adv}_p(\mathcal{A}) + \frac{1}{2} &\leq \mathrm{Succ}_{\frac{1}{2}}(\mathcal{A})\\ \frac{1}{2p}\mathrm{Adv}_p(\mathcal{A}) &\leq \mathrm{Succ}_{\frac{1}{2}}(\mathcal{A}) - \frac{1}{2} = \mathrm{Adv}_{\frac{1}{2}}(\mathcal{A}).\end{aligned}$$

The last sub-claim follows from

$$\begin{aligned}\mathrm{Adv}_{\frac{1}{2}}(\mathcal{A}) &= \max\left\{0, \mathrm{Succ}_{\frac{1}{2}}(\mathcal{A}) - \frac{1}{2}\right\} \leq \left|\mathrm{Succ}_{\frac{1}{2}}(\mathcal{A}) - \frac{1}{2}\right|\\ &= \left|\frac{1}{2}\left(\mathsf{Pr}[x \leftarrow_R \mathcal{X}_1 : 1 \leftarrow \mathcal{A}(x)] + \mathsf{Pr}[x \leftarrow_R \mathcal{X}_0 : 0 \leftarrow \mathcal{A}(x)]\right) - \frac{1}{2}\right|\\ &= \left|\frac{1}{2}\left(\mathsf{Pr}[x \leftarrow_R \mathcal{X}_1 : 1 \leftarrow \mathcal{A}(x)] + 1 - \mathsf{Pr}[x \leftarrow_R \mathcal{X}_0 : 1 \leftarrow \mathcal{A}(x)]\right) - \frac{1}{2}\right|\\ &= \frac{1}{2}\,|\mathsf{Pr}[x \leftarrow_R \mathcal{X}_1 : 1 \leftarrow \mathcal{A}(x)] - \mathsf{Pr}[x \leftarrow_R \mathcal{X}_0 : 1 \leftarrow \mathcal{A}(x)]|\end{aligned}$$

The main statement follows from plugging the last two sub-claims together. □

Our approach to show that DSPR is hard is giving a reduction from an average-case distinguishing problem that was used in the full version of [9]. The problem makes use of the following distribution D_λ over boolean functions.

Definition 27 [9]. *Let* $\mathcal{F} \stackrel{def}{=} \{f : \{0,1\}^m \to \{0,1\}\}$ *be the collection of all boolean functions on* $\{0,1\}^m$. *Let* $\lambda \in [0,1]$ *and* $\varepsilon > 0$. *Define a family of distributions* D_λ *on* $\mathcal{F}$ *such that* $f \leftarrow_R D_\lambda$ *satisfies*

$$f : x \mapsto \begin{cases} 1 & \text{with prob. } \lambda, \\ 0 & \text{with prob. } 1-\lambda \end{cases}$$

for any $x \in \{0,1\}^m$.

In [9] the following bound on the distinguishing advantage of any q-query quantum adversary was shown.

Theorem 28 [9]. *Let* D_λ *be defined as in Definition 27, and* $\mathcal{A}$ *be any quantum algorithm making at most* q *quantum queries to its oracle. Then*

$$\mathrm{Adv}_{D_0,D_\lambda}(\mathcal{A}) \stackrel{def}{=} \left| \Pr_{f \leftarrow D_0}[\mathcal{A}^f(\cdot) = 1] - \Pr_{f \leftarrow D_\lambda}[\mathcal{A}^f(\cdot) = 1] \right| \leq 8\lambda q^2 .$$

We still have to briefly discuss how DSPR is defined in the (quantum-accessible) random oracle model. Instead of giving a description of the hash function H as implicitly done in Definition 5, we provide $\mathcal{A}$ with an oracle $\mathcal{O}$ that implements a function $\mathrm{F} : \mathcal{X} \to \mathcal{Y}$. As for most other notions that can be defined for unkeyed hash functions, DSPR in the (Q)ROM becomes the same for keyed and non-keyed hash functions. For keyed functions, instead of giving a description of the keyed hash function H and a key k to the adversary $\mathcal{A}$, we provide $\mathcal{A}$ with an oracle that implements a function $\mathrm{F} : \mathcal{X} \to \mathcal{Y}$ which now models H for a fixed key k. Hence, the following result applies to both cases. This can be seen as the key space might contain just a single key.

Now we got all tooling we need to show that DSPR is a hard problem.

Theorem 29. *Let* $n \in \mathbb{N}$, $N = 2^n$, $\mathrm{H} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ *as defined above be a random, length-preserving keyed hash function. Any quantum adversary* $\mathcal{A}$ *that solves DSPR making* q *quantum queries to* H *can be used to construct a quantum adversary* $\mathcal{B}$ *that makes* $2q$ *queries to its oracle and distinguishes* D_0 *from* $D_{1/N}$ *with success probability*

$$\mathrm{Adv}_{D_0,D_{1/N}}(\mathcal{B}) \geq \mathrm{Adv}_{\mathrm{H}}^{\mathrm{DSPR}}(\mathcal{A}).$$

Proof. By construction. The algorithm $\mathcal{B}$ generates an DSPR instance as in Figure 2 and runs $\mathcal{A}$ on it. It outputs whatever $\mathcal{A}$ outputs. To answer an H query $\mathcal{B}$ needs two f queries as it also has to uncompute the result of the f query after

Given: Oracle access to $f : \mathcal{X} \to \{0,1\}$.

1. Sample $x' \leftarrow \mathcal{X}$ and $y' \leftarrow \mathcal{Y}$ independently and uniformly at random.
2. Let $g : \mathcal{X} \to \mathcal{Y}\backslash\{y'\}$ be a random function. We construct $\mathrm{H} : \mathcal{X} \to \mathcal{Y}$ as follows: for any $x \in \mathcal{X}$

$$x \mapsto \begin{cases} y' & \text{if } x = x' \\ y' & \text{if } x \neq x' \wedge f(x) = 1 \\ g(x) & \text{otherwise.} \end{cases}$$

Output: DSPR instance (H, x'). Namely an adversary is given x' and oracle access to H, and the goal is to decide if x' has a second preimage under H.

Fig. 2. Reducing distinguishing D_0 from $D_{1/N}$ to DSPR.

it was used. The random function g can be efficiently simulated using $2q$-wise independent hash functions as discussed in [9].

Now, if $f \leftarrow_R D_0$, (H, x') is a random DSPR challenge from the set of all DSPR challenges with $P_{\mathrm{H}}(x') = 0$ (slightly abusing notation as we do not know a key for our random function). Similarly, if $f \leftarrow_R D_{1/N}$, (H, x') is a random DSPR challenge from the set of all DSPR challenges.

$$\begin{aligned}
\mathrm{Adv}_{D_0,D_\lambda}(\mathcal{B}) &= \left| \Pr_{f \leftarrow D_0}[\mathcal{B}^f(\cdot) = 1] - \Pr_{f \leftarrow D_\lambda}[\mathcal{B}^f(\cdot) = 1] \right| \\
&= \left| \Pr_{f \leftarrow D_0}[\mathcal{A}^{\mathrm{H}}(x') = 1] - \Pr_{f \leftarrow D_\lambda}[\mathcal{A}^{\mathrm{H}}(x') = 1] \right| \\
&= \big| \Pr[\mathcal{A}^{\mathrm{H}}(x') = 1 \mid P_{\mathrm{H}}(x') = 0] - \big(p \cdot \Pr[\mathcal{A}^{\mathrm{H}}(x') = 1 \mid P_{\mathrm{H}}(x') = 1] \\
&\quad + (1-p) \cdot \Pr[\mathcal{A}^{\mathrm{H}}(x') = 1 \mid P_{\mathrm{H}}(x') = 0]\big) \big| \\
&= p \cdot \big| \Pr[\mathcal{A}^{\mathrm{H}}(x') = 1 \mid P_{\mathrm{H}}(x') = 1] - \Pr[\mathcal{A}^{\mathrm{H}}(x') = 1 \mid P_{\mathrm{H}}(x') = 0] \big| \\
&\geq \mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}),
\end{aligned}$$

where the last inequality follows from Theorem 26. □

Theorem 30. *Let $n \in \mathbb{N}$, $N = 2^n$, $\mathrm{H} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ as defined above be a random, length-preserving keyed hash function. Any quantum adversary $\mathcal{A}$ that makes no more than q quantum queries to its oracle can only solve the decisional second-preimage problem with advantage*

$$\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) \leq 32q^2/N.$$

Proof. Use Theorem 29 to construct an adversary $\mathcal{B}$ that makes $2q$ queries and that has advantage at least $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A})$ of distinguishing D_0 from $D_{1/N}$. This advantage is at most $8(1/N)(2q)^2 = 32q^2/N$ by Theorem 28. □

6 DSPR for multiple targets

Multi-target security considers an adversary that is given T independent targets and is asked to solve a problem for one out of the T targets. This section defines T-DSPR, a multi-target version of DSPR.

We draw attention to an unusual feature of this definition: the advantage of an adversary $\mathcal{A}$ is defined as the improvement from p to q, where p and q are two probabilities that can *both* be influenced by $\mathcal{A}$. The second probability q is $\mathcal{A}$'s chance of correctly predicting whether the input selected by $\mathcal{A}$ has a second preimage. The first probability p is the chance that the input selected by $\mathcal{A}$ does have a second preimage.

This deviates from the usual view of advantage as how much $\mathcal{A}$ improves upon success probability compared to some trivial baseline attack. What we are doing, for multi-target attacks, is asking how much $\mathcal{A}$ improves upon success probability compared to the baseline attack *against the same target that $\mathcal{A}$ selected*. In most of the contexts considered in the literature, the success probability of the baseline attack is independent of the target, so this matches the usual view. DSPR is different, because the success probability of the baseline attack depends on the target.

One can object that this allows the baseline attack to be affected (positively or negatively) by $\mathcal{A}$'s competence in target selection. We give two responses to this objection. First, our definition enables a proof (Theorem 33) that T-DSPR is at most T times easier to break than DSPR. Second, our definition enables an interactive multi-target generalization (Theorem 38) of our proof that DSPR and SPR together imply PRE.

Definition 31 (T-DSPR). *Let T be a positive integer. Let $\mathcal{A}$ be an algorithm with output in $\{1,\ldots,T\} \times \{0,1\}$. The advantage of $\mathcal{A}$ against the T-target decisional second-preimage resistance of a keyed hash function* H *is*

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

where

$$\begin{aligned} q = \mathsf{Pr}\big[&(x_1,k_1,\ldots,x_T,k_T) \leftarrow_R (\mathcal{X}\times\mathcal{K})^T; \\ &(j,b) \leftarrow \mathcal{A}(x_1,k_1,\ldots,x_T,k_T) : P_{k_j}(x_j) = b\big]; \\ p = \mathsf{Pr}\big[&(x_1,k_1,\ldots,x_T,k_T) \leftarrow_R (\mathcal{X}\times\mathcal{K})^T; \\ &(j,b) \leftarrow \mathcal{A}(x_1,k_1,\ldots,x_T,k_T) : P_{k_j}(x_j) = 1\big]; \end{aligned}$$

and $P_{k_j} = \mathrm{SPexists}(\mathrm{H}_{k_j})$.

The only difference between the formulas for q and p is that q compares $P_{k_j}(x_j)$ to b while p compares it to 1. If $T > 1$ then an algorithm might be able to influence p up or down, compared to any particular $\mathrm{SPprob}(\mathrm{H}_{k_i})$, through the choice of j. Obtaining a significant T-DSPR advantage then means obtaining q significantly larger than p, i.e., making a prediction of $P_{k_j}(x_j)$ significantly better than always predicting that it is 1.

As an extreme case, consider the following slow algorithm. Compute each $P_{k_j}(x_j)$ by brute force; choose j where $P_{k_j}(x_j) = 0$ if such a j exists, else $j = 1$; and output $P_{k_j}(x_j)$. This algorithm has $q = 1$ and thus T-DSPR advantage $1-p$. The probability p for this algorithm is the probability that *all* of $x_1, \ldots, x_T$ have second preimages. For most length-preserving functions, this probability is approximately $(1 - 1/e)^T$, which rapidly converges to 0 as T increases, so the T-DSPR advantage rapidly converges to 1.

Definition 32. *Let $\mathcal{A}$ be an algorithm, and let T be a positive integer. Then* $\mathrm{Plant}_T(A)$ *is the following algorithm:*

- *Input* $(x, k) \in \mathcal{X} \times \mathcal{K}$.
- *Generate* $i \leftarrow_R \{1, \ldots, T\}$.
- *Generate* $(x_1, k_1, \ldots, x_T, k_T) \leftarrow_R (\mathcal{X} \times \mathcal{K})^T$.
- *Overwrite* $(x_i, k_i) \leftarrow (x, k)$.
- *Compute* $(j, b) \leftarrow \mathcal{A}(x_1, k_1, \ldots, x_T, k_T)$.
- *Output* b *if* $j = i$, *or* 1 *if* $j \neq i$.

This uses the standard technique of planting a single-target challenge at a random position in a multi-target challenge. With probability $1/T$, the multi-target attack chooses the challenge position; in the other cases, this reduction outputs 1. The point of Theorem 33 is that this reduction interacts nicely with the subtraction of probabilities in the DSPR and T-DSPR definitions.

The cost of $\mathrm{Plant}_T(A)$ is the cost of generating a random number i between 1 and T, generating $T - 1$ elements of $\mathcal{X} \times \mathcal{K}$, running $\mathcal{A}$, and comparing j to i. The algorithm has essentially the same cost as $\mathcal{A}$ if $\mathcal{X}$ and $\mathcal{K}$ can be efficiently sampled.

Theorem 33 (T-loose implication DSPR $\Rightarrow$ T-DSPR). *Let* H *be a keyed hash function. Let T be a positive integer. Let $\mathcal{A}$ be an algorithm with output in* $\{1, \ldots, T\} \times \{0, 1\}$. *Then*

$$\mathrm{Adv}_{\mathrm{H}}^{T\text{-DSPR}}(\mathcal{A}) = T \cdot \mathrm{Adv}_{\mathrm{H}}^{\mathrm{DSPR}}(\mathcal{B})$$

where $\mathcal{B} = \mathrm{Plant}_T(\mathcal{A})$.

Proof. By definition $\mathrm{Adv}_{\mathrm{H}}^{T\text{-DSPR}}(\mathcal{A})$ runs $\mathcal{A}$ with T independent uniform random targets $(x_1, k_1, \ldots, x_T, k_T)$. Write (j, b) for the output of $\mathcal{A}(x_1, k_1, \ldots, x_T, k_T)$. Then $\mathrm{Adv}_{\mathrm{H}}^{T\text{-DSPR}}(\mathcal{A}) = \max\{0, q-p\}$, where q is the probability that $P_{k_j}(x_j) = b$, and p is the probability that $P_{k_j}(x_j) = 1$.

To analyze q and p, we split the universe of possible events into four mutually exclusive events:

$$E_{00} \stackrel{\text{def}}{=} [b = 0 \wedge P_{k_j}(x_j) = 0];$$
$$E_{01} \stackrel{\text{def}}{=} [b = 0 \wedge P_{k_j}(x_j) = 1];$$
$$E_{10} \stackrel{\text{def}}{=} [b = 1 \wedge P_{k_j}(x_j) = 0];$$
$$E_{11} \stackrel{\text{def}}{=} [b = 1 \wedge P_{k_j}(x_j) = 1].$$

Then $q = \Pr E_{00} + \Pr E_{11}$ and $p = \Pr E_{01} + \Pr E_{11}$, so $q - p = \Pr E_{00} - \Pr E_{01}$.

For comparison, $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{B})$ runs $\mathcal{B}$, which in turn runs $\mathcal{A}$ with T independent uniform random targets $(x_1, k_1, \ldots, x_T, k_T)$. One of these targets (x_i, k_i) is the uniform random target (x, k) provided to $\mathcal{B}$ as a challenge; $\mathcal{B}$ randomly selects i and the remaining targets. The output b' of $\mathcal{B}(x, k)$ is b if $j = i$, and 1 if $j \neq i$.

The choice of i is not visible to $\mathcal{A}$, so the event that $i = j$ has probability $1/T$. Furthermore, this event is independent of $E_{00}, E_{01}, E_{10}, E_{11}$: i.e., $i = j$ has conditional probability $1/T$ given E_{00}, conditional probability $1/T$ given E_{01}, etc.

Write q' for the chance that $P_k(x) = b'$, and p' for the chance that $P_k(x) = 1$. Then $\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{B}) = \max\{0, q' - p'\}$. To analyze q' and p', we split into mutually exclusive events as follows:

- E_{00} occurs and $i = j$. This has probability $(\Pr E_{00})/T$. Then $(x_j, k_j) = (x_i, k_i) = (x, k)$ so $P_k(x) = P_{k_j}(x_j) = 0 = b = b'$. This contributes to q' and not to p'.
- E_{01} occurs and $i = j$. This has probability $(\Pr E_{01})/T$. Then $(x_j, k_j) = (x, k)$ so $P_k(x) = 1$, while $b' = b = 0$. This contributes to p' and not to q'.
- All other cases: $b' = 1$ (since $b' = 0$ can happen only if $b = 0$ and $i = j$). We further split this into two cases:
  - $P_k(x) = 1$. This contributes to q' and to p'.
  - $P_k(x) = 0$. This contributes to neither q' nor p'.

To summarize, $q' - p' = (\Pr E_{00})/T - (\Pr E_{01})/T = (q - p)/T$. Hence

$$\max\{0, q - p\} = \max\{0, T(q' - p')\} = T \max\{0, q' - p'\};$$

i.e., $\mathrm{Adv}^{T\text{-}\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) = T \cdot \mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{B})$. □

7 Removing interactivity

The real importance of DSPR for security proofs is that it allows interactive versions of preimage resistance to be replaced by non-interactive assumptions without penalty. Interactive versions of preimage resistance naturally arise in, e.g., the context of hash-based signatures; see Section 8.

The example discussed in this section is the T-openPRE notion already informally introduced in Section 1.1.1. We first review T-SPR, a multi-target version of second-preimage resistance. Then we formally define the interactive notion T-openPRE and show that its security tightly relates to T-SPR and T-DSPR.

T-SPR is what is called multi-function, multi-target second-preimage resistance in [9]. It was shown in [9] that a generic attack against T-SPR has the same complexity as a generic attack against SPR.

Definition 34 [9] (T-SPR). *The success probability of an algorithm $\mathcal{A}$ against the T-target second-preimage resistance of a keyed hash function* H *is*

$$\begin{aligned}\mathrm{Succ}^{T\text{-}\mathrm{SPR}}_{\mathrm{H}}(\mathcal{A}) \stackrel{def}{=} \mathsf{Pr}\big[&(x_1, k_1, \ldots, x_T, k_T) \leftarrow_R (\mathcal{X} \times \mathcal{K})^T; \\ &(j, x) \leftarrow \mathcal{A}(x_1, k_1, \ldots, x_T, k_T) : \\ &\mathrm{H}_{k_j}(x) = \mathrm{H}_{k_j}(x_j) \wedge x \neq x_j\big].\end{aligned}$$

T-openPRE is essentially what would be T-PRE (which we did not define) but with the additional tweak that the adversary gets access to an opening oracle. The adversary is allowed to query the oracle for the preimages of all but one of the targets and has to output a preimage for the remaining one.

Definition 35 (T-openPRE). *Let* H *be a keyed hash function. The success probability of an algorithm $\mathcal{A}$ against the T-target opening-preimage resistance of* H *is defined as*

$$\mathrm{Succ}_{\mathrm{H}}^{T\text{-}\mathsf{openPRE}}(\mathcal{A}) \stackrel{def}{=} \mathsf{Pr}\left[(x_1, k_1, \ldots, x_T, k_T) \leftarrow_R (\mathcal{X} \times \mathcal{K})^T; \right.$$
$$(j, x') \leftarrow \mathcal{A}^{\mathsf{Open}}(\mathrm{H}_{k_1}(x_1), k_1, \ldots, \mathrm{H}_{k_T}(x_T), k_T) :$$
$$\left. \mathrm{H}_{k_j}(x') = \mathrm{H}_{k_j}(x_j) \wedge j \text{ was no query of } \mathcal{A}\right]$$

where $\mathsf{Open}(i) = x_i$.

Now, it is of course possible to reduce PRE to T-openPRE. However, such a reduction has to guess the index j for which $\mathcal{A}$ will output a preimage (and hence does not make a query) correctly. Otherwise, if the reduction embeds its challenge image in any of the other positions, it cannot answer $\mathcal{A}$'s query for that index. As $\mathcal{A}$ does not lose anything by querying all indices but j, we can assume that it actually does so. Hence, such a reduction from PRE must incur a loss in tightness of a factor T. For some applications discussed below, T can reach the order of $\sqrt[4]{N}$. This implies a quarter loss in the security level.

Theorem 38 shows that T-openPRE is tightly related to the non-interactive assumptions T-DSPR and T-SPR: if H is T-target decisional-second-preimage resistant and T-target second-preimage resistant then it is T-target opening-preimage-resistant. As before, we first define the reductions and then state a theorem regarding probabilities.

Definition 36 (T-target SPfromP). *Let* H *be a keyed hash function. Let $\mathcal{A}$ be an algorithm using an oracle. Let T be a positive integer. Then* $\mathrm{SPfromP}_T(\mathrm{H}, \mathcal{A})$ *is the following algorithm:*

- *Input* $(x_1, k_1, \ldots, x_T, k_T) \in (\mathcal{X} \times \mathcal{K})^T$.
- *Output* $\mathcal{A}^{\mathsf{Open}}(\mathrm{H}_{k_1}(x_1), k_1, \ldots, \mathrm{H}_{k_T}(x_T), k_T)$, *where* $\mathsf{Open}(i) = x_i$.

This generalizes the standard SPfromP reduction: it handles multiple targets in the obvious way, and it easily answers oracle queries with no failures since it knows all the x_i inputs. The algorithm $\mathrm{SPfromP}_T(\mathrm{H}, \mathcal{A})$ uses T calls to H (which can be deferred until their outputs are used) and one call to $\mathcal{A}$.

Definition 37 (T-target DSPfromP). *Let* H *be a keyed hash function. Let $\mathcal{A}$ be an algorithm. Then* $\mathrm{DSPfromP}_T(\mathrm{H}, \mathcal{A})$ *is the following algorithm:*

- *Input* $(x_1, k_1, \ldots, x_T, k_T) \in (\mathcal{X} \times \mathcal{K})^T$.
- *Compute* $(j, x') \leftarrow \mathcal{A}^{\mathsf{Open}}(\mathrm{H}_{k_1}(x_1), k_1, \ldots, \mathrm{H}_{k_T}(x_T), k_T)$, *where* $\mathsf{Open}(i) = x_i$.
- *Compute* $b \leftarrow ((x' \neq x_j) \vee j$ *was a query of* $\mathcal{A})$.

– *Output* (j, b).

This is an analogous adaptation of our DSPfromP reduction to the interactive multi-target context. Again oracle queries are trivial to answer. Note that the case that $\mathcal{A}$ cheats, returning an index j that it used for an Open query, is a failure case for $\mathcal{A}$ by definition; the algorithm $\mathrm{DSPfromP}_T(\mathrm{H}, \mathcal{A})$ outputs 1 in this case, exactly as if $\mathcal{A}$ had failed to find a preimage. In other words, this algorithm returns 0 whenever $\mathcal{A}$ returns a solution that contains the preimage that was already known by the reduction (but *not* given to $\mathcal{A}$ via Open), and 1 otherwise.

Theorem 38 (T-DSPR $\wedge$ T-SPR $\Rightarrow$ T-openPRE). *Let* H *be a keyed hash function. Let* T *be a positive integer. Let* $\mathcal{A}$ *be an algorithm. Then*

$$\mathrm{Succ}_{\mathrm{H}}^{T\text{-}\mathsf{openPRE}}(\mathcal{A}) \leq \mathrm{Adv}_{\mathrm{H}}^{T\text{-}\mathrm{DSPR}}(\mathcal{B}) + 3 \cdot \mathrm{Succ}_{\mathrm{H}}^{T\text{-}\mathrm{SPR}}(\mathcal{C})$$

where $\mathcal{B} = \mathrm{DSPfromP}_T(\mathrm{H}, \mathcal{A})$ *and* $\mathcal{C} = \mathrm{SPfromP}_T(\mathrm{H}, \mathcal{A})$.

The core proof idea is the following. As noted above, the reductions attacking T-SPR and T-DSPR can perfectly answer all of $\mathcal{A}$'s oracle queries as they know preimages. However, for the index for which $\mathcal{A}$ outputs a preimage (without cheating), it did not learn the preimage known to the reduction. Hence, from there on we can apply a similar argument as in the proof of Theorem 25. We include a complete proof below to aid in verification.

Proof. Write (j, x') for the output of $\mathcal{A}^{\mathsf{Open}}(\mathrm{H}_{k_1}(x_1), k_1, \ldots, \mathrm{H}_{k_T}(x_T), k_T)$. As in the proof of Theorem 25, we split the universe of possible events into mutually exclusive events across two dimensions: the number of preimages of $\mathrm{H}_{k_j}(x_j)$, and whether $\mathcal{A}$ succeeds or fails in finding a preimage. Specifically, define

$$S_i \stackrel{\text{def}}{=} \left[\left|\mathrm{H}_{k_j}^{-1}(\mathrm{H}_{k_j}(x_j))\right| = i \wedge \mathrm{H}_{k_j}(x') = \mathrm{H}_{k_j}(x_j) \wedge j \text{ was no query of } \mathcal{A}\right],$$

as the event that there are exactly i preimages and that $\mathcal{A}$ succeeds, and define

$$F_i \stackrel{\text{def}}{=} \left[\left|\mathrm{H}_{k_j}^{-1}(\mathrm{H}_{k_j}(x_j))\right| = i \wedge \left(\mathrm{H}_{k_j}(x') \neq \mathrm{H}_{k_j}(x_j) \vee j \text{ was a query of } \mathcal{A}\right)\right]$$

as the event that there are exactly i preimages and that $\mathcal{A}$ fails. Note that there are only finitely many i for which the events S_i and F_i can occur.

Define s_i and f_i as the probabilities of S_i and F_i respectively. The probability space here includes the random choices of $(x_1, k_1, \ldots, x_T, k_T)$, and any random choices made inside $\mathcal{A}$.

***T*-openPRE success probability.** By definition, $\mathrm{Succ}_{\mathrm{H}}^{T\text{-}\mathsf{openPRE}}(\mathcal{A})$ is the probability that x' is a non-cheating preimage of $\mathrm{H}_{k_j}(x_j)$; i.e., that $\mathrm{H}_{k_j}(x') = \mathrm{H}_{k_j}(x_j)$ and j was not a query to the oracle. This event is the union of the events S_i, so $\mathrm{Succ}_{\mathrm{H}}^{T\text{-}\mathsf{openPRE}}(\mathcal{A}) = \sum_i s_i$.

***T*-DSPR success probability.** By definition $\mathcal{B}$ outputs the pair (j, b), where $b = ((x' \neq x_j) \vee j$ was a query of $\mathcal{A})$.

Define $P_{k_j} = \text{SPexists}(\text{H}_{k_j})$, and define q as in the definition of $\text{Adv}^{T\text{-DSPR}}_{\text{H}}(\mathcal{B})$. Then q is the probability that $\mathcal{B}$ is correct, i.e., that $b = P_{k_j}(x_j)$. There are four cases:

- If the event S_1 occurs, then there is exactly 1 preimage of $\text{H}_{k_j}(x_j)$, so $P_{k_j}(x_j) = 0$ by definition of SPexists. Also, $\mathcal{A}$ succeeds: i.e., j was not a query, and x' is a preimage of $\text{H}_{k_j}(x_j)$, forcing $x' = x_j$. Hence $b = 0 = P_{k_j}(x_j)$.
- If the event F_1 occurs, then again $P_{k_j}(x_j) = 0$, but now $\mathcal{A}$ fails: i.e., j was a query, or x' is not a preimage of $\text{H}_{k_j}(x_j)$. Either way $b = 1 \neq P_{k_j}(x_j)$. (We could skip this case in the proof, since we need only a lower bound on q rather than an exact formula for q.)
- If the event S_i occurs for $i > 1$, then $P_{k_j}(x_j) = 1$ and $\mathcal{A}$ succeeds. Hence j was not a query, and x' is a preimage of $\text{H}_{k_j}(x_j)$, so $x' = x_j$ with conditional probability exactly $\frac{1}{i}$. Hence $b = 1 = P_{k_j}(x_j)$ with conditional probability exactly $\frac{i-1}{i}$.
- If the event F_i occurs for $i > 1$, then $P_{k_j}(x_j) = 1$ and $\mathcal{A}$ fails. Failure means that x' is not a preimage, so in particular $x' \neq x_j$, or that j was a query. Either way $b = 1 = P_{k_j}(x_j)$.

To summarize, $q = s_1 + \sum_{i>1} \frac{i-1}{i} s_i + \sum_{i>1} f_i$.

***T*-DSPR advantage.** Define p as in the definition of $\text{Adv}^{T\text{-DSPR}}_{\text{H}}(\mathcal{B})$. Then $\text{Adv}^{T\text{-DSPR}}_{\text{H}}(\mathcal{B}) = \max\{0, q - p\}$.

The analysis of p is the same as the analysis of q above, except that we compare $P_{k_j}(x_j)$ to 1 instead of comparing it to b. We have $1 = P_{k_j}(x_j)$ exactly for the events S_i and F_i with $i > 1$. Hence $p = \sum_{i>1} s_i + \sum_{i>1} f_i$. Subtract to see that

$$\text{Adv}^{T\text{-DSPR}}_{\text{H}}(\mathcal{B}) = \max\{0, q - p\} \geq q - p = s_1 - \sum_{i>1} \frac{1}{i} s_i.$$

***T*-SPR success probability.** By definition $\mathcal{C}$ outputs (j, x'). The T-SPR success probability $\text{Succ}^{T\text{-SPR}}_{\text{H}}(\mathcal{C})$ is the probability that x' is a second preimage of x_j under H_{k_j}, i.e., that $\text{H}_{k_j}(x') = \text{H}_{k_j}(x_j)$ while $x' \neq x_j$.

It is possible for $\mathcal{C}$ to succeed while $\mathcal{A}$ fails: perhaps $\mathcal{A}$ learns $x_j = \mathsf{Open}(j)$ and then computes a second preimage for x_j, which does not qualify as an T-openPRE success for $\mathcal{A}$ but does qualify as a T-SPR success for $\mathcal{C}$. We ignore these cases, so we obtain only a lower bound on $\text{Succ}^{T\text{-SPR}}_{\text{H}}(\mathcal{C})$; this is adequate for the proof.

Assume that event S_i occurs with $i > 1$. Then x' is a preimage of $\text{H}_{k_j}(x_j)$. Furthermore, $\mathcal{A}$ did not query j, so x_j is not known to $\mathcal{A}$ except via $\text{H}_{k_j}(x_j)$. There are i preimages, so $x' = x_j$ with conditional probability exactly $\frac{1}{i}$. Hence $\mathcal{C}$ succeeds with conditional probability $\frac{i-1}{i}$.

To summarize, $\text{Succ}^{T\text{-SPR}}_{\text{H}}(\mathcal{C}) \geq \sum_{i>1} \frac{i-1}{i} s_i$.

Combining the probabilities. We conclude as in the proof of Theorem 25:

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

□

8 Applications to hash-based signatures

The interactive notion of T-openPRE with a huge number of targets naturally arises in the context of hash-based signatures. This was already observed and extensively discussed in [9]. One conclusion of the discussion there is to use keyed hash functions with new (pseudo)random keys for each hash-function call made in a hash-based signature scheme.

When applying this idea to Lamport one-time signatures (L-OTS) [11], the standard security notion for OTS of existential unforgeability under one chosen message attacks (EU-CMA) becomes T-openPRE where $\mathcal{A}$ is allowed to make $T/2$ queries. Using L-OTS in a many-time signature scheme such as the Merkle Signature Scheme [13] and variants like [12,2,8] can easily amplify the difference in tightness between a reduction that uses (T-)PRE and a reduction from T-SPR and T-DSPR to 2^{70}.

Indeed, the general idea of using T-SPR instead of (T-)PRE in security reductions for hash-based signatures already occurs in [9]. However, there the authors make use of the assumption that for the used hash function every input has a colliding value for all keys, i.e., $\mathrm{SPprob}(\mathrm{H}) = 1$ in our notation. This is unlikely to hold for common length-preserving keyed hash functions as Section 3 shows $\mathrm{SPprob}(\mathrm{H}) \approx 1 - 1/e$ for random H. However, as shown above, it is also not necessary to require $\mathrm{SPprob}(\mathrm{H}) = 1$. Instead, it suffices to require (T-)DSPR.

For modern hash-based signatures like XMSS [7] L-OTS is replaced by variants [6] of the Winternitz OTS (W-OTS) [13]. For W-OTS the notion of EU-CMA security does not directly translate to T-openPRE. Indeed, the security reduction gets far more involved as W-OTS uses hash chains. However, as shown in [9] one can replace (T-)PRE in this context by T-SPR and the assumption that $\mathrm{SPprob}(\mathrm{H}) = 1$. Along the lines of the above approach we can then replace the assumption that $\mathrm{SPprob}(\mathrm{H}) = 1$ by T-DSPR.

References

1. Elena Andreeva, Charles Bouillaguet, Orr Dunkelman, Pierre-Alain Fouque, Jonathan J. Hoch, John Kelsey, Adi Shamir, and Sébastien Zimmer. New second-preimage attacks on hash functions. *J. Cryptology*, 29(4):657–696, 2016. https://www.di.ens.fr/~fouque/pub/joc11.pdf.

2. Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. Merkle signatures with virtually unlimited signature capacity. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security*, volume 4521 of *LNCS*, pages 31–45. Springer, 2007. https://link.springer.com/chapter/10.1007/978-3-540-72738-5_3.
3. Charalambos A. Charalambides. Distributions of random partitions and their applications. *Methodology and Computing in Applied Probability*, 9(2):163–193, 2007.
4. Heinrich Dörrie. *100 great problems of elementary mathematics*. Courier Corporation, 2013.
5. Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, 2009. http://ac.cs.princeton.edu/home/AC.pdf.
6. Andreas Hülsing. W-OTS+ – shorter signatures for hash-based signature schemes. In Amr Youssef, Abderrahmane Nitaj, and Aboul-Ella Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, volume 7918 of *LNCS*, pages 173–188. Springer, 2013. https://eprint.iacr.org/2017/965.
7. Andreas Hülsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: eXtended Merkle Signature Scheme. RFC 8391, May 2018. https://rfc-editor.org/rfc/rfc8391.txt.
8. Andreas Hülsing, Lea Rausch, and Johannes Buchmann. Optimal parameters for XMSS^{MT}. In Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar Weippl, and Lida Xu, editors, *Security Engineering and Intelligence Informatics*, volume 8128 of *LNCS*, pages 194–208. Springer, 2013. https://eprint.iacr.org/2017/966.
9. 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*, volume 9614 of *LNCS*, pages 387–416. Springer, Berlin, Heidelberg, 2016. https://eprint.iacr.org/2015/1256.
10. John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer, 2005. https://eprint.iacr.org/2004/304.pdf.
11. Leslie Lamport. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979. https://lamport.azurewebsites.net/pubs/dig-sig.pdf.
12. Tal Malkin, Daniele Micciancio, and Sara Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In Lars Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 400–417. Springer Berlin / Heidelberg, 2002. https://cseweb.ucsd.edu/~daniele/papers/MMM.html.
13. Ralph Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *LNCS*, pages 218–238. Springer, 1990. https://merkle.com/papers/Certified1979.pdf.
14. Herbert Robbins. A remark on Stirling's formula. *The American Mathematical Monthly*, 62(1):26–29, 1955.
15. Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors,

Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004. https://eprint.iacr.org/2004/035.

16. Neil J.A. Sloane. The on-line encyclopedia of integer sequences, 2019. https://oeis.org.

A Some single-variable functions

This appendix proves features of some functions used in the proofs of theorems in Section 3. The proofs in this appendix are split into small lemmas to support verification, and proofs of the lemmas appear in the full version online. The notation $\mathbf{R}_{>0}$ means the set of positive real numbers.

Lemma 39. *If $x \neq 0$ then $e^x > 1 + x$.*

Lemma 40. *Any $x \in \mathbf{R}$ has $e^x - 2x \geq 2 - 2\log 2 > 0$.*

Lemma 41. *If $x > 0$ then $e^x - 1 + x - x^2 > 0$.*

Lemma 42. *Define $\varphi_1(x) = x(e^x - 1)/(e^x - x)$. Then φ_1 is increasing, and maps $\mathbf{R}_{>0}$ bijectively to $\mathbf{R}_{>0}$.*

Lemma 43. *If $x \neq 0$ then $e^x + e^{-x} > 2$.*

Lemma 44. *If $x > 0$ then $e^x - e^{-x} - 2x > 0$.*

Lemma 45. *If $x > 0$ then $e^x + e^{-x} - 2 - x^2 > 0$.*

Lemma 46. *Define $\varphi_2(x) = x(e^x - 1)/(e^x - 1 - x)$ for $x > 0$. Then φ_2 is increasing, and maps $\mathbf{R}_{>0}$ bijectively to $\mathbf{R}_{>2}$.*

Lemma 47. *The ratio $(e-1)^{1-x}/x^x(1-x)^{1-x}$ for $0 < x < 1$ increases for $0 < x < 1/e$, has maximum value e at $x = 1/e$, and decreases for $1/e < x < 1$.*

Lemma 48. *The maximum value of $1/(2x-1)^{2x-1}(1-x)^{2(1-x)}2^{1-x}$ for $1/2 < x < 1$ is $1 + \sqrt{2}$.*

Lemma 49. *Define $\varphi_5(x) = xe^x - e^x + 1$. Then φ_5 decreases for $x < 0$, has minimum value 0 at $x = 0$, and increases for $x > 0$.*

Lemma 50. *Let x be a positive real number. Define $y = e^x - 1 - x$ and $z = 1/(x + x^2/y)$; then $0 < z < 1/2$. Define $\gamma = y^z/xz^z(1-z)^{1-z}$; then $\gamma \leq e - 1$.*