Collapseability of Tree Hashes

Aldo Gunsing and Bart Mennink

Digital Security Group, Radboud University, Nijmegen, The Netherlands
aldo.gunsing@ru.nl, b.mennink@cs.ru.nl

Abstract. One oft-endeavored security property for cryptographic hash functions is collision resistance: it should be computationally infeasible to find distinct inputs x, x' such that $H(x) = H(x')$, where H is the hash function. Unruh (EUROCRYPT 2016) proposed collapseability as its quantum equivalent. The Merkle-Damgård and sponge hashing modes have recently been proven to be collapseable under the assumption that the underlying primitive is collapseable. These modes are inherently sequential. In this work, we investigate collapseability of tree hashing. We first consider fixed length tree hashing modes, and derive conditions under which their collapseability can be reduced to the collapseability of the underlying compression function. Then, we extend the result to two methods for achieving variable length hashing: tree hashing with domain separation between message and chaining value, and tree hashing with length encoding at the end of the tree. The proofs are performed using the collapseability composability framework of Fehr (TCC 2018), that allows us to discard of deeply technical quantum details and to focus on proper composition of the tree hashes from their compression function.

Keywords: collapseability · collision resistance · tree hashing · composition

1 Introduction

Hash functions are functions that map arbitrarily long strings, or at least very long strings, to a digest of fixed length. Their introduction dates back to the seminal work of Diffie and Hellman [7] in the context of digital signatures. Nowadays, hash function outgrew their original role: they find thousands of applications in cryptography. These applications all require certain security properties of the hash function. One of these properties is collision resistance: it should be computationally infeasible for an attacker to find two distinct messages with the same hash digest. The notion appeared first in Merkle's PhD thesis [9], and is the leading security property when it comes to breaking hash functions. Well-known hash functions MD5 [11] and SHA-1 [12], and many others, are considered insecure mainly because practical collision attacks were mounted on these (cf. Stevens et al. [15] for MD5 and Stevens et al. [14] for SHA-1). However, finding collisions appears not to be a purely academic exercise: in 2012 the Flame virus exploited collisions in MD5 to act as a properly signed Windows Software Update security patch [13].

When we move to the quantum setting, the classical notion of collision resistance is not strong enough. Unruh [17, Theorem 19] (Theorem 22 in the full version) showed that there is a hash function that is collision resistant and thus can safely be used in a classical commitment scheme, but is not secure when the commitment scheme is used in a quantum setting. We can therefore conclude that, even if a hash function is (classically) collision resistant, it behaves unexpectedly when used in a quantum environment. We need a stronger model in favor of collision resistance.

1.1 Collapseability

Unruh [17] presented collapseability as a quantum equivalent of collision resistance of hash functions. Informally, for cryptographic hash functions, collapseability requires an adversary that outputs a hash value together with a superposition of corresponding preimages is not able to tell if the superposition gets measured or not. We revisit collapseability in Section 3, and particularly outline the idea as to why this would be the quantum equivalent of collision resistance in Section 3.3.

The model of collapseability has gained traction. In a follow-up work, Unruh [16] proved that the Merkle-Damgård hashing mode [10,6] is collapsing if the compression function is. Later, Czajkowski et al. [5] proved collapseability of the sponge construction [1] if the underlying one-way function is (their work combined independent works of Unruh [18] and Czajkowski et al. [4]). These proofs are, however, quite tedious and technically involved. They require the reader to possess a large amount of quantum knowledge.

In [8], Fehr introduced an alternative framework for collapseability. His definition is more algebraic in nature, whereas that of Unruh is more algorithmic. This allowed Fehr to reason about composability of collapseable functions in a neat and compact way. In more detail, Fehr showed that collapseability is closed under certain compositions, all with very concise proofs. These composability properties make it possible to reason about collapseability from a purely classical view, without requiring quantum knowledge. He applied the approach to the Merkle-Damgård and sponge hash construction, proving that they are collapsing if the underlying compression function is. Therewith, he confirmed the correctness of the earlier results in his new framework.

1.2 Our Contribution

We consider collapseability of tree hashing. We will make full use of Fehr's framework [8] in order to argue what conditions a tree hashing mode must meet in order to be collapseable.

First, in Section 4 we consider the basic problem of tree hashing for fixed length messages. For messages of a certain fixed length n, we recursively define a tree hash function TH_n. It is defined based on a split function $split(n) \in \{1, \ldots, n-1\}$ that prescribes how the final digest is derived from two tree hashes

$TH_{split(n)}$ and $TH_{n-split(n)}$ applied to the first $split(n)$ and last $n - split(n)$ message blocks.

Then, in Section 5, we detail how the result can be extended to variable length hashing using domain separation. In this case, it is assumed that processing of message blocks and chaining values is properly domain-separated in the way the mode calls its compression function. One way in doing so is by appending a 0 to message blocks and a 1 to chaining values. Intuitively, this makes it impossible to replace the chaining value of a subtree with a message block with the same value. We prove that the resulting variable length tree hash function is collapsing. This is done by extending trees with 'empty' blocks in such a way that we can reduce collapseability of the variable length mode to that of the fixed length mode.

Finally, in Section 6, we consider a second way to turn the fixed length construction into a variable length hashing mode: length encoding. Here, we allow any tree hashing mode, but the block length of the message will be included by using a final compression function call. This approach makes the final compression functions disjoint for different message lengths, and using previous techniques and the composition results of Fehr, we likewise manage to prove collapseability.

All three collapseability results come with a security bound that expresses the adversarial advantage relative to the collapseability of the underlying compression function, as well as with a complexity analysis of the resulting modes.

1.3 Related Work

One might likewise consider quantum indifferentiability of tree hashing in the framework of Zhandry [19]. We remark, however, that in the classical setting indifferentiability implies collision resistance, but not conversely, and for some applications the weaker property of collision resistance is sufficient. A similar remark applies to the quantum setting, i.e., collapseability is sufficient in many applications such as the simple commitment schemes given by Unruh [17]. In such settings, it is senseless to rely on a stronger property with a more complex security proof and a, likely, weaker security bound.

2 Preliminaries

We will use the following compositions of functions.

- For $g : X \to Y$ and $h : W \to Z$, the *concurrent composition* $g \| h : X \times W \to Y \times Z$ is given by $(x, w) \mapsto (g(x), h(w))$.
- For $g : X \to Y$ and $h : Y \to Z$, the *nested (or sequential) composition* $h \circ g : X \to Z$ is given by $x \mapsto h(g(x))$.
- For $g : X \to Y$ and $h : X \to Z$, the *parallel composition* $(g, h) : X \to Y \times Z$ is given by $x \mapsto (g(x), h(x))$.
- For $g : X \to Y$ and $h : W \to Z$, the *disjoint union* $g \sqcup h : X \cup W \to Y \cup Z$ maps $x \in X$ to $g(x)$ and $w \in W$ to $h(w)$. Furthermore, f and g are required to have disjoint domains and images, so $X \cap W = Y \cap Z = \varnothing$.

Furthermore, we write $S_n = 1 + 2 + \cdots + n = n(n+1)/2$ for the sum of positive integers up to n.

2.1 Qubits and Measurements

Classical computers work with classical bits. These bits can be either a 0 or a 1, but not a combination of both. However, quantum computers work with quantum bits (qubits). These qubits can be just a 0 or 1, but can also be in a state which is a combination of both. A qubit that has a single value is denoted by $|0\rangle$ or $|1\rangle$ (pronounced 'ket 0' and 'ket 1'), for 0 and 1 respectively. However, a qubit can also be a linear combination of both, denoted as $\alpha|0\rangle + \beta|1\rangle$ for some $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. This is called a superposition. A superposition can also be written in a single ket. For example, we could define the state $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$. If a qubit is in a superposition, computations on it are applied to all values at the same time. However, we cannot directly measure the coefficients in the states. We have to do a measurement, which destroys the superposition.

Given a qubit $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, we can measure it in the standard computational basis $\{|0\rangle, |1\rangle\}$. Then one of the following happens:

- $|\phi\rangle$ collapses to the state $|0\rangle$ with probability $|\alpha|^2$. We get the result '0'.
- $|\phi\rangle$ collapses to the state $|1\rangle$ with probability $|\beta|^2$. We get the result '1'.

Since $|\alpha|^2 + |\beta|^2 = 1$, exactly one of these happens with probability 1.

In addition to single qubit states, there are also states of multiple qubits. In general a quantum state of dimension n has the form

$$\sum_{i \in \{0,1\}^n} \alpha_i |i\rangle ,$$

where $\alpha_i \in \mathbb{C}$ and $\sum_i |\alpha_i|^2 = 1$. For example, we can have the 2-dimensional state

$$\frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) .$$

Measurements happen in a similar way to single qubits. When measured in the standard computational basis $\{ \ |i\rangle \ | \ i \in \{0,1\}^n \}$, a state $|\phi\rangle = \sum_i \alpha_i |i\rangle$ collapses to the state $|i\rangle$ with probability $|\alpha_i|^2$. We get the result 'i'.

We can also measure in other bases. This can be any orthonormal basis. The measurements happen in the same way as before, but we have to rewrite our state in the new basis. For example, a popular basis is $\{|+\rangle, |-\rangle\}$ given by

$$|\pm\rangle = \frac{1}{\sqrt{2}} (|0\rangle \pm |1\rangle) .$$

If we want to measure $|0\rangle$ in it, we have to rewrite it as

$$|0\rangle = \frac{1}{2} (|0\rangle + |1\rangle) + \frac{1}{2} (|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}} |+\rangle + \frac{1}{\sqrt{2}} |-\rangle .$$

This means that the measurement results in either $|+\rangle$ or $|-\rangle$, both with probability $1/2$.

3 Collapseability

Unruh [17] introduced *collapseability* as the quantum version of classical collision resistance. His definition uses an 'algorithmic' view on collapseability. Later, Fehr [8] introduced a new formalism of collapseability. His new definition uses a more 'algebraic' view, which allows for simpler proofs of some composition results. We will mostly use Fehr's formalism to classically prove that certain tree hash functions are collapsing.

Although most proofs become easier with the more 'algebraic' definition, we present the more 'algorithmic' one given in [17] as it is easier to understand.

Definition 1. *Given a (hash) function H, we play the following game. An adversary $\mathcal{A}$ generates a quantum state $|\phi\rangle = \sum_{x\in X} \alpha_x |x\rangle$ such that $H(x) = c$ for all $x \in X$ for some c. Then, one of the following happens:*

1. *The state $|\phi\rangle$ gets measured in the computational basis.*
2. *The state $|\phi\rangle$ is left untouched.*

The adversary $\mathcal{A}$ does not know which one happened, but it tries to determine it. It returns a bit b, indicating which case it thinks has happened. Its advantage is given by

$$\mathrm{cAdv}[H](\mathcal{A}) = \left|\mathbb{P}[b = 1 : \textit{Case (1)}] - \mathbb{P}[b = 1 : \textit{Case (2)}]\right| .$$

In the remaining of this section, we first look at the query complexity of functions in Section 3.1. This allows us to give a definition of collapseability that maximizes over adversaries. Then we look at the composability of collapseability in Section 3.2. These lemmas allow us to reason classically about the collapsing advantage. Finally, in Section 3.3 we explain why collapseability is a stronger notion than collision resistance.

3.1 Complexity

In order to limit the resources of the adversary, we adopt the notion of complexity from Fehr [8]. Given a function f, we assign it a *complexity* $\mathfrak{c}(f)$, which is a non-negative integer. We usually normalize the complexity of the compression function as 1. We also assume that this abstract notion satisfies some natural properties. Simple functions like constants, copying, deleting, the identity, etc., have zero complexity. Furthermore, we assume that the complexity function behaves well under compositions, so that

$$\begin{aligned} \mathfrak{c}(f\|g) &\leqslant \mathfrak{c}(f) + \mathfrak{c}(g) , \\ \mathfrak{c}(g \circ f) &\leqslant \mathfrak{c}(f) + \mathfrak{c}(g) , \\ \mathfrak{c}(f \sqcup g) &\leqslant \mathfrak{c}(f) + \mathfrak{c}(g) . \end{aligned}$$

The final inequality is not mentioned in [8]. However, we will need it in our proofs, and it seems natural to assume. If we were to apply the functions

just classically, it seems enough to bound $\mathfrak{c}(f \sqcup g)$ by $\max(\mathfrak{c}(f), \mathfrak{c}(g))$, as every application computes either f or g, but not both. However, since we work with qubits, we also have to account for superpositions. If a qubit is in a superposition consisting of inputs from both domains, we have to compute both functions. This means that its complexity can be $\mathfrak{c}(f) + \mathfrak{c}(g)$.

Given the notion of complexity, we can define the advantage we get when we limit the resources of the adversaries.

Definition 2. *The* collapsing advantage *of H with complexity q is given by*

$$\mathrm{cAdv}[H](q) = \max_{\mathcal{A} \in \mathcal{A}(q)} \mathrm{cAdv}[H](\mathcal{A}) \,,$$

where $\mathcal{A}(q)$ are all adversaries with query complexity q.

We also say that a function H is ε-collapsing if $\mathrm{cAdv}[H](q) \leqslant \varepsilon(q)$.

3.2 Composability of Collapseability

The main advantage of the formalism introduced by Fehr is that some composability results have very concise proofs. The following lemmas are taken from Fehr [8].

Lemma 1 (Concurrent composition). *The concurrent composition $g \| h$ satisfies*

$$\mathrm{cAdv}[g \| h] \leqslant \mathrm{cAdv}[g] + \mathrm{cAdv}[h] \,.$$

Lemma 2 (Nested composition). *The nested (or sequential) composition $h \circ g$ satisfies*

$$\mathrm{cAdv}[h \circ g](q) \leqslant \mathrm{cAdv}[g](q + \mathfrak{c}(g)) + \mathrm{cAdv}[h](q + \mathfrak{c}(g)) \,.$$

Lemma 3 (Parallel composition). *The parallel composition (g, h) satisfies*

$$\mathrm{cAdv}[(g, h)] \leqslant \min(\mathrm{cAdv}[g], \mathrm{cAdv}[h]) \,.$$

Lemma 4 (Disjoint union). *The disjoint union $g \sqcup h$ satisfies*

$$\mathrm{cAdv}[g \sqcup h] \leqslant \mathrm{cAdv}[g] + \mathrm{cAdv}[h] \,.$$

3.3 Collapseability Implies Collision Resistance

We show that collapseability is a stronger notion than collision resistance. Our reasoning is similar to the one of Unruh [17, Lemma 22] (Lemma 25 in the full version), but simplified to only require the basic knowledge of qubits and measurements given in Section 2.1.

Suppose one can obtain distinct x, x' with $H(x) = H(x')$. We show how such a collision can be used to break the collapseability of H. As the state in the collapsing game, we choose

$$|\phi\rangle = \frac{1}{\sqrt{2}} \left(|x\rangle + |x'\rangle \right) \,.$$

Now $|\phi\rangle$ may or may not get measured in the computational basis. If it gets measured, it either collapses to $|x\rangle$ or to $|x'\rangle$, both with probability 1/2. If it does not get measured, it stays the same.

In order for adversary $\mathcal{A}$ to differentiate between the two cases, we measure $|\phi\rangle$ in a modified basis. Almost all basis elements stay the same, but we replace $|x\rangle$ and $|x'\rangle$ with

$$|+_{x,x'}\rangle = \frac{1}{\sqrt{2}}\left(|x\rangle + |x'\rangle\right) \text{ and } |-_{x,x'}\rangle = \frac{1}{\sqrt{2}}\left(|x\rangle - |x'\rangle\right) .$$

This is still an orthonormal basis. If $|\phi\rangle$ is not measured, it is equal to $|+_{x,x'}\rangle$, hence the result of the measurement is $|+_{x,x'}\rangle$ with probability 1.

However, if $|\phi\rangle$ is measured, it is either $|x\rangle$ or $|x'\rangle$. We rewrite these in the new basis as

$$|x\rangle = \frac{1}{2}\left(|x\rangle + |x'\rangle\right) + \frac{1}{2}\left(|x\rangle - |x'\rangle\right) = \frac{1}{\sqrt{2}}|+_{x,x'}\rangle + \frac{1}{\sqrt{2}}|-_{x,x'}\rangle ,$$
$$|x'\rangle = \frac{1}{2}\left(|x\rangle + |x'\rangle\right) - \frac{1}{2}\left(|x\rangle - |x'\rangle\right) = \frac{1}{\sqrt{2}}|+_{x,x'}\rangle - \frac{1}{\sqrt{2}}|-_{x,x'}\rangle .$$

This means that both $|x\rangle$ and $|x'\rangle$ get measured as either $|+_{x,x'}\rangle$ or $|-_{x,x'}\rangle$, both with probability 1/2.

The adversary thus operates as follows. If it finds $|+_{x,x'}\rangle$, it concludes that $|\phi\rangle$ was not measured, and if it finds $|-_{x,x'}\rangle$, it concludes that $|\phi\rangle$ was measured. Since it gets the case without measurement always right, and it basically guesses in the case with measurement, its advantage is 1/2. Hence H is not collapsing.

4 Fixed Length Tree Hashing

Fehr [8] showed that some simple hash constructions like Merkle-Damgård can be proven to be collapsing by only using the composition lemmas of Section 3.2. This means that the reasoning in the proofs is just classical, as the hash constructions are broken down as the composition of smaller functions for which the composition lemmas apply. The main technicality moved from actually proving collapseability to describing hash functions as clever compositions of collapseable functions. In this work we investigate tree hashes.

We start with tree hashes of fixed length that can only take a fixed number of blocks and for which the structure is the same for all inputs.

Let $f : \{0,1\}^c \times \{0,1\}^c \to \{0,1\}^c$ be a compression function. We define a tree hashing mode by its splitting points, that is a function $\mathit{split} : \mathbb{N}_{\geqslant 2} \to \mathbb{N}$. Given a message $x_1, \ldots, x_n$ the parts $x_1, \ldots, x_{\mathit{split}(n)}$ and $x_{\mathit{split}(n)+1}, \ldots, x_n$ are hashed separately and compressed with f. No part can be empty, so $1 \leqslant \mathit{split}(n) \leqslant n-1$ for all n. An example is displayed in Figure 1 with a splitting function such that

$$\begin{aligned} \mathit{split}(5) &= 3 , \\ \mathit{split}(3) &= 2 , \\ \mathit{split}(2) &= 1 . \end{aligned} \tag{1}$$

Note that it is irrelevant for this particular tree what value $split(n)$ takes for $n \notin \{2, 3, 5\}$.

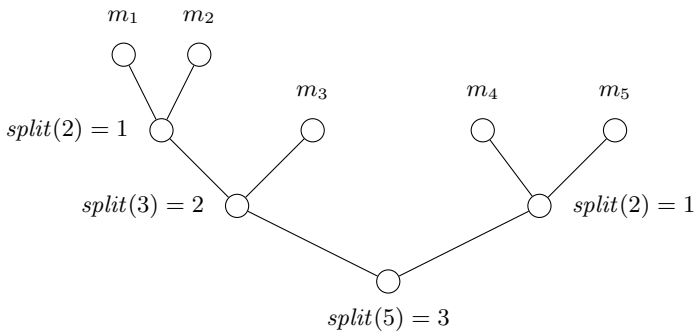


Fig. 1: Tree hash with the split function of (1). The circles with messages m_i represent the message blocks, while the other circles represent calls to the compression function f and their resulting chaining value. The final digest is the result of the compression call of the root.

Likewise, we can express the Merkle-Damgård mode as a tree hash with $split(n) = n - 1$. The example mode of Sakura [2, Section 5.4], which balances the tree very well, can be expressed with $split(n)$ the largest power of 2 smaller than n.

We now define the general tree hashing mode that we will consider in this work.

Definition 3. *Given a compression function $f : \{0,1\}^c \times \{0,1\}^c \to \{0,1\}^c$ and a function $split : \mathbb{N}_{\geqslant 2} \to \mathbb{N}$, we recursively define the tree hashes $TH_n : (\{0,1\}^c)^n \to \{0,1\}^c$ for $n \in \mathbb{N}_{\geqslant 1}$ as*

$$\begin{aligned} TH_1(x_1) &= x_1\,, \\ TH_n(x_1, \ldots, x_n) &= f(\,TH_k(x_1, \ldots, x_k),\, TH_{n-k}(x_{k+1}, \ldots, x_n))\,, \end{aligned}$$

where $k = split(n)$. Note that we can express this definition equivalently using the composition functions defined in Section 3.2 as follows:

$$\begin{aligned} TH_1 &= \mathrm{id}\,, \\ TH_n &= f \circ (\,TH_k \| \,TH_{n-k})\,. \end{aligned}$$

The function TH_n calls the compression function $n - 1$ times. As we normalize the complexity of f as $\mathfrak{c}(f) = 1$, we would expect the complexity of TH_n to be $n - 1$. This is indeed the case.

Lemma 5. *If f has complexity $\mathfrak{c}(f) = 1$, then $\mathfrak{c}(TH_n) \leqslant n - 1$ for all n.*

Proof. We use strong induction to n. Since we assume the identity has 0 complexity, we get

$$\mathfrak{c}(TH_1) = \mathfrak{c}(\mathrm{id}) = 0\,.$$

For $n > 1$ we have $TH_n = f \circ (TH_k \| TH_{n-k})$, which means, using the composition properties of the complexity function (See Section 3.1), that

$$\begin{aligned}\mathfrak{c}(TH_n) &= \mathfrak{c}(f \circ (TH_k \| TH_{n-k})) \\ &\leqslant \mathfrak{c}(f) + \mathfrak{c}(TH_k) + \mathfrak{c}(TH_{n-k}) \\ &\leqslant 1 + (k-1) + (n-k-1) = n-1\,.\end{aligned}$$

□

We now look at the collapseability of TH_n. As it only accepts a fixed number of blocks, we find that it is collapsing regardless of the *split* function used.

Proposition 1. *If f has complexity $\mathfrak{c}(f) = 1$ and is ε-collapsing, then TH_n is collapsing for all n with advantage*

$$\mathrm{cAdv}[TH_n](q) \leqslant (n-1) \cdot \varepsilon(q + S_{n-2})\,.$$

Proof. We use strong induction to n. Since the identity has an advantage of 0, we get

$$\mathrm{cAdv}[TH_1](q) = 0\,.$$

For $n > 1$ we have $TH_n = f \circ (TH_k \| TH_{n-k})$, which means, using nested composition and the fact $\mathfrak{c}(TH_m) = m - 1$ (Lemma 5), that

$$\begin{aligned}\mathrm{cAdv}[TH_n](q) &\leqslant \mathrm{cAdv}[f]\left(q + \mathfrak{c}\left(TH_k \| TH_{n-k}\right)\right) \\ &\quad + \mathrm{cAdv}[TH_k \| TH_{n-k}]\left(q + \mathfrak{c}\left(TH_k \| TH_{n-k}\right)\right) \\ &\leqslant \varepsilon(q+n-2) + \mathrm{cAdv}[TH_k \| TH_{n-k}](q+n-2)\,.\end{aligned}$$

Using concurrent composition and the induction hypothesis, we find that

$$\begin{aligned}\mathrm{cAdv}[TH_k \| TH_{n-k}](q+n-2) &\leqslant \mathrm{cAdv}[TH_k](q+n-2) \\ &\quad + \mathrm{cAdv}[TH_{n-k}](q+n-2) \\ &\leqslant (k-1) \cdot \varepsilon(q+n-2+S_{k-2}) \\ &\quad + (n-k-1) \cdot \varepsilon(q+n-2+S_{n-k-2})\,.\end{aligned}$$

Since k and $n-k$ are strictly smaller than n, we have that $n-2+S_{k-2}$ and $n-2+S_{n-k-2}$ are less or equal to S_{n-2}, which means that

$$\begin{aligned}\mathrm{cAdv}[TH_k \| TH_{n-k}](q+n-2) &\leqslant (k-1) \cdot \varepsilon(q+S_{n-2}) \\ &\quad + (n-k-1) \cdot \varepsilon(q+S_{n-2}) \\ &= (n-2) \cdot \varepsilon(q+S_{n-2})\,.\end{aligned}$$

Putting these together, and using that $n-2 \leqslant S_{n-2}$, we get that

$$\begin{aligned}\mathrm{cAdv}[TH_n](q) &\leqslant \varepsilon(q+n-2) + (n-2) \cdot \varepsilon(q+S_{n-2}) \\ &\leqslant (n-1) \cdot \varepsilon(q+S_{n-2})\,.\end{aligned}$$

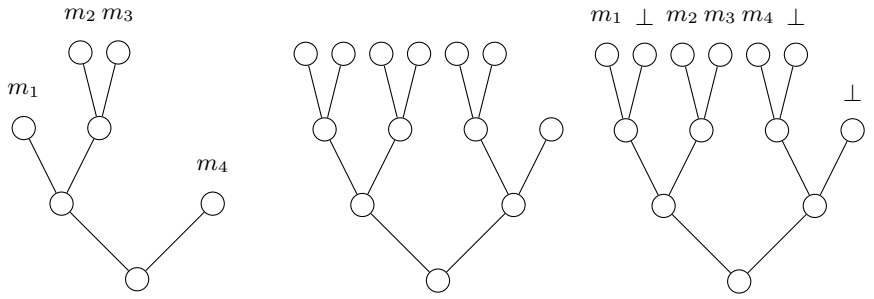
□

5 Variable Length Using Domain Separation

We have seen that any binary tree hash of fixed length is collapsing. However, a hash function has to be able to hash inputs of varying lengths. We cannot just use the tree hash corresponding to the input we got, as this leads to collisions. For example, suppose that we have a big tree hash. We can replace two leaves, that get compressed together to a chaining value CV, by just a single leaf with CV as the message. This is a collision, as the resulting digests of both trees are the same.

One solution to this problem is using domain separation between the message blocks and the chaining values. For example, all message blocks can end with a 0, while the chaining values all end with a 1. We see that this requirement arises naturally when we apply a similar idea as Fehr [8]. He proves Merkle-Damgård secure by limiting the block size of the input to some L. Every smaller message is expanded to a message of size L by prepending 'empty' blocks $\perp$. Then the compression function is modified to also take these 'empty' blocks as input and ignore them. We use a similar strategy where we embed a smaller tree in a larger tree and fill the extra leaves with these 'empty' blocks.

To do so, we define the mapping Extend$[U](T)$ which takes an unlabeled binary tree U and a labeled binary tree T, such that T is a subtree of U. This means that every node in T has to be a node in U as well, and not a leaf. Then Extend$[U](T)$ outputs a labeled binary tree with the same structure as U, but with labels based on T. Every leaf of T with value m is mapped to its part in U, with the label m on the leftmost leaf, and the label $\perp$ on all the other leaves. An example is displayed in Figure 2.



(a) Labeled tree T (b) Unlabeled tree U (c) Extend$[U](T)$

Fig. 2: Extending a tree. The tree T has to be a subtree of U.

More formally, assume that the tree hashing mode uses domain separation: input blocks can be differentiated from the chaining values. This means that we can identify two disjoint sets $M \subseteq \{0,1\}^c$ and $C = \{0,1\}^c \setminus M$ that cover all message block values and chaining values, respectively. Next, we define three

different types of tree hashing modes, with different types of output: a message block in M, a chaining value in C or either one in $\{0,1\}^c$. These functions are defined as $TH_n^M : M_n \to M$, $TH_n^C : C_n \to C$, and $TH_n^X : X_n \to \{0,1\}^c$, where the domains are recursively defined as

$$\begin{aligned} M_n &= M \times \{\bot\}^{n-1}\,, \\ C_1 &= \varnothing\,, \\ C_n &= X_k \times X_{n-k}\,, \\ X_n &= M_n \sqcup C_n\,, \end{aligned}$$

where $k = \mathit{split}(n)$. Given these domains, we now define the three types of tree hashing modes themselves.

Definition 4. *Given a compression function $f : \{0,1\}^c \times \{0,1\}^c \to C$ and a function split $: \mathbb{N}_{\geqslant 2} \to \mathbb{N}$, we recursively define the tree hashes $TH_n^M : M_n \to M$, $TH_n^C : C_n \to C$ and $TH_n^X : X_n \to \{0,1\}^c$ as*

$$\begin{aligned} TH_n^M(m, \bot, \ldots, \bot) &= m\,, \\ TH_n^C(x_1, \ldots, x_n) &= f(TH_k^X(x_1, \ldots, x_k), TH_{n-k}^X(x_{k+1}, \ldots, x_n))\,, \\ TH_n^X &= TH_n^M \sqcup TH_n^C\,, \end{aligned}$$

where $k = \mathit{split}(n)$.

Using these functions, we finally define the variable input length function $TH_{\mathrm{dom}}^{\leqslant L} : M^{\leqslant L} \to \{0,1\}^c$ which hashes variable length trees up to length L. Let U, with a size of K blocks, be the smallest tree of which all trees up to length L are a subtree of. $TH_{\mathrm{dom}}^{\leqslant L}$ maps an input $m_1, \ldots, m_n$, to $TH_K^X(x_1, \ldots, x_K)$, where $x_1, \ldots, x_K$ is the tree Extend$[U](T_n)$, where T_n is the tree of size n with labels of $m_1, \ldots, m_n$. This means that x_i is equal to either some m_j or $\bot$. We can apply this mapping non-ambiguously as every tree up to length L is a subtree of U.

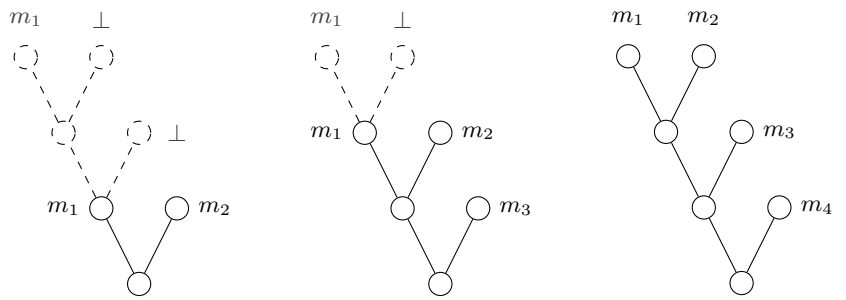
For example, in the Merkle-Damgård construction every tree of size n is a subtree of the tree of size n' if $n \leqslant n'$. This means that U is just the normal tree of size L, hence $K = L$. This is displayed in Figure 3 for $L = 4$. Note that this is not the case in general. For example in the construction in Sakura, if L is equal to $2^\ell + 1$, then the right branch only contains one leaf, which means that it does not contain the tree for 2^ℓ, which is a full binary tree with a right branch of size $2^{\ell-1}$. However, we can still choose the next power of two as a tree that contains the necessary trees, hence $K \leqslant 2L$. This is displayed in Figure 4 for $L = 5$.

We are ready to prove the collapseability of $TH_{\mathrm{dom}}^{\leqslant L}$.

Theorem 1. *If f has complexity $\mathfrak{c}(f) = 1$ and is ε-collapsing, then $TH_{\mathrm{dom}}^{\leqslant L}$ is collapsing for any L with advantage*

$$\mathrm{cAdv}[TH_{\mathrm{dom}}^{\leqslant L}](q) \leqslant (K-1) \cdot \varepsilon(q + S_{K-2})\,,$$

where K is the size of the smallest tree of which all the trees up to length L are a subtree of. Furthermore, we have that $L \leqslant K \leqslant S_L$.

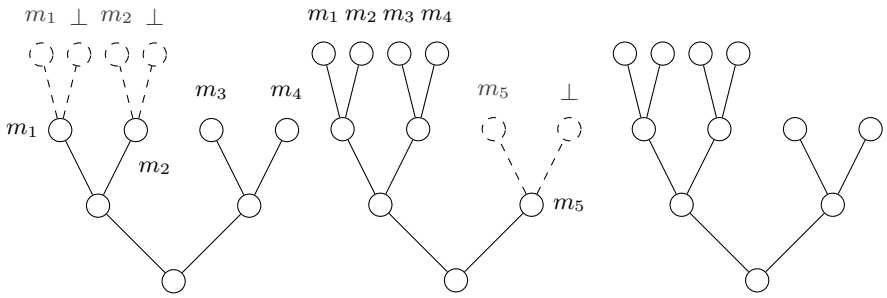


(a) Tree T_2 (solid) and Extend[U](T_2) (dashed).

(b) Tree T_3 (solid) and Extend[U](T_3) (dashed).

(c) Tree T_4, equal to U.

Fig. 3: Hashing variable sized hashes with the Merkle-Damgård mode. U, displayed in Figure 3c, is the smallest tree structure of which all trees up to length 4 are a subtree of. This tree happens to be the same as the normal Merkle-Damgård construction with $n = 4$.



(a) Tree T_4 (solid) and Extend[U](T_4) (dashed).

(b) Tree T_5 (solid) and Extend[U](T_5) (dashed).

(c) Tree U, not equal to any previous tree.

Fig. 4: Hashing variable sized hashes with the example mode of Sakura. The trees up to length 3 are not shown. Tree U, displayed in Figure 4c, is the smallest tree structure of which all trees up to length 5 are a subtree of. This tree is different from all the constructions up to length 5.

Proof. As the first step, an input $m_1, \ldots, m_n$ is mapped to the extended input $x_1, \ldots, x_K$, on which the tree hash TH^X_K is applied. This mapping is injective, hence 0-collapsing, hence we have to show that

$$\mathrm{cAdv}[TH^X_K](q) \leqslant (K-1) \cdot \varepsilon(q + S_{K-2})\,.$$

As TH^M_n is injective, it is 0-collapsing, hence we see that $\mathrm{cAdv}[TH^X_n] = \mathrm{cAdv}[TH^C_n]$ by disjoint union. Furthermore, TH^C_n is defined in almost the same way as TH_n in Definition 3. The only difference is that the recursive call is to TH^X_k instead

of itself. However, as the advantage of TH_k^X is the same as that of TH_k^C we can still apply the same proof as in Proposition 1, which gives the desired result.

Furthermore, we look at the value of K, which is the size of the smallest tree U of which all the trees up to length L are a subtree of. First, the tree of size L has to be a subtree of U, which means that its size is at least L, hence $L \leqslant K$. Second, every tree of size n adds at most n leaves to U for all n up to L. This means that its size is at most $1 + 2 + \cdots + L = S_L$, hence $K \leqslant S_L$. □

6 Variable Length Using Length Encoding

We have seen that we can make a variable length tree hash collapsing by using domain separation between the message blocks and the chaining values. However, this method adds some overhead and might not work well with the alignment of the message blocks, as at least a bit has to be added to every block. Another way to allow a variable length input is by using length encoding. Here, the length of the message is used in a final compression call. We define the hash with length encoding $TH_{\text{len}}^{\leqslant L} : (\{0,1\}^c)^{\leqslant L} \to \{0,1\}^c$, which hashes variable length trees up to length L, as

$$TH_{\text{len}}^{\leqslant L}(x_1, \ldots, x_n) = f(\underline{n}, TH_n(x_1, \ldots, x_n)) \,,$$

where $\underline{n}$ is the number n encoded as a binary number in $\{0,1\}^c$, which limits L to be at most 2^c.

This method requires less overhead than domain separation, as the length of the message is added just once. We find that any tree hash with length encoding in a final compression call is collapseable, by applying the composition lemmas on some smaller functions.

Theorem 2. *If f has complexity $\mathfrak{c}(f) = 1$ and is ε-collapsing, then $TH_{\text{len}}^{\leqslant L}$ is collapsing for any L with advantage*

$$\text{cAdv}[TH_{\text{len}}^{\leqslant L}](q) \leqslant (S_{L-1} + 1) \cdot \varepsilon(q + (L-1)^2) \,.$$

Proof. Instead of looking at $TH_{\text{len}}^{\leqslant L}$ directly, we build it as the composition of smaller functions. First we define for every $n \in \mathbb{N}$ the function $LTH_n : (\{0,1\}^c)^n \to \{0,1\}^c \times \{0,1\}^c$ as

$$LTH_n(x_1, \ldots, x_n) = (\underline{n}, TH_n(x_1, \ldots, x_n)) \,.$$

Note that we can also write this as $LTH_n = (c_{\underline{n}}, TH_n)$, where $c_{\underline{n}} = x \mapsto \underline{n}$ is the constant function with value $\underline{n}$. By parallel composition we get that

$$\text{cAdv}[LTH_n](q) = \text{cAdv}[TH_n](q) \leqslant (n-1) \cdot \varepsilon(q + S_{n-2}) \,.$$

Let $LTH^{\leqslant L} : (\{0,1\}^c)^{\leqslant L} \to \{0,1\}^c \times \{0,1\}^c$ be

$$LTH^{\leqslant L} = \bigsqcup_{n=1}^{L} LTH_n \,.$$

Note that the images are disjoint as LTH_n stores $\underline{n}$ in the output, which is different for every n, as long as $L < 2^c$. By disjoint union we get that

$$\begin{aligned}\mathrm{cAdv}[LTH^{\leqslant L}](q) &\leqslant \sum_{n=1}^{L} \mathrm{cAdv}[LTH_n](q) \\ &\leqslant \sum_{n=1}^{L} (n-1) \cdot \varepsilon(q + S_{n-2}) \\ &\leqslant S_{L-1} \cdot \varepsilon(q + S_{L-2})\,.\end{aligned}$$

For its complexity we have that

$$\begin{aligned}\mathfrak{c}(LTH^{\leqslant L}) &\leqslant \sum_{n=1}^{L} \mathfrak{c}(LTH_n) \\ &= \sum_{n=1}^{L} \mathfrak{c}(\,TH_n) \\ &\leqslant \sum_{n=1}^{L} (n-1) \\ &= S_{L-1}\,.\end{aligned}$$

Finally we have that $TH^{\leqslant L}_{\mathrm{len}} = f \circ LTH^{\leqslant L}$. Using the facts that $S_{L-1} \leqslant (L-1)^2$ and $S_{L-1} + S_{L-2} = (L-1)^2$ we get

$$\begin{aligned}\mathrm{cAdv}[TH^{\leqslant L}_{\mathrm{len}}](q) &\leqslant \mathrm{cAdv}[f](q + \mathfrak{c}(LTH^{\leqslant L})) + \mathrm{cAdv}[LTH^{\leqslant L}](q + \mathfrak{c}(LTH^{\leqslant L})) \\ &\leqslant \varepsilon(q + S_{L-1}) + S_{L-1} \cdot \varepsilon(q + S_{L-1} + S_{L-2}) \\ &\leqslant \varepsilon(q + (L-1)^2) + S_{L-1} \cdot \varepsilon(q + (L-1)^2) \\ &= (S_{L-1} + 1) \cdot \varepsilon(q + (L-1)^2)\,.\end{aligned}$$

□

Acknowledgments. The authors would like to thank Joan Daemen and the anonymous reviewers of PQCrypto for their valuable feedback. Aldo Gunsing is supported by the Netherlands Organisation for Scientific Research (NWO) under TOP grant TOP1.18.002 SCALAR.

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. Ecrypt Hash Workshop 2007 (May 2007)
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sakura: A flexible coding for tree hashing. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8479, pp. 217–234. Springer (2014), https://doi.org/10.1007/978-3-319-07536-5_14
3. Brassard, G. (ed.): Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings, Lecture Notes in Computer Science, vol. 435. Springer (1990), https://doi.org/10.1007/0-387-34805-0
4. Czajkowski, J., Groot Bruinderink, L., Hülsing, A., Schaffner, C.: Quantum preimage, 2nd-preimage, and collision resistance of SHA3. Cryptology ePrint Archive, Report 2017/302 (2017)
5. Czajkowski, J., Groot Bruinderink, L., Hülsing, A., Schaffner, C., Unruh, D.: Post-quantum security of the sponge construction. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10786, pp. 185–204. Springer (2018), https://doi.org/10.1007/978-3-319-79063-3_9
6. Damgård, I.: On the existence of bit commitment schemes and zero-knowledge proofs. In: Brassard [3], pp. 17–27, https://doi.org/10.1007/0-387-34805-0_3
7. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Information Theory 22(6), 644–654 (1976), https://doi.org/10.1109/TIT.1976.1055638
8. Fehr, S.: Classical proofs for the quantum collapsing property of classical hash functions. In: Beimel, A., Dziembowski, S. (eds.) Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11240, pp. 315–338. Springer (2018), https://doi.org/10.1007/978-3-030-03810-6_12
9. Merkle, R.: Secrecy, Authentication, and Public Key Systems. UMI Research Press (1979)
10. Merkle, R.C.: One way hash functions and DES. In: Brassard [3], pp. 428–446, https://doi.org/10.1007/0-387-34805-0_40
11. Rivest, R.: The MD5 message-digest algorithm. Request for Comments (RFC) 1321 (April 1992), http://tools.ietf.org/html/rfc1321
12. National Institute of Standards and Technology. FIPS 180-4: Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-4 (August 2015)
13. Stevens, M.: Counter-cryptanalysis. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 129–146. Springer (2013), https://doi.org/10.1007/978-3-642-40041-4_8
14. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 570–596. Springer (2017), https://doi.org/10.1007/978-3-319-63688-7_19

15. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A.K., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5677, pp. 55–69. Springer (2009), https://doi.org/10.1007/978-3-642-03356-8_4
16. Unruh, D.: Collapse-binding quantum commitments without random oracles. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10032, pp. 166–195 (2016), https://doi.org/10.1007/978-3-662-53890-6_6
17. Unruh, D.: Computationally binding quantum commitments. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 497–527. Springer (2016), https://doi.org/10.1007/978-3-662-49896-5_18
18. Unruh, D.: Collapsing sponges: Post-quantum security of the sponge construction. Cryptology ePrint Archive, Report 2017/282 (2017)
19. Zhandry, M.: How to record quantum queries, and applications to quantum indifferentiability. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11693, pp. 239–268. Springer (2019), https://doi.org/10.1007/978-3-030-26951-7_9