Higher Order Universal One-Way Hash Functions*

Deukjo Hong[1]**, Bart Preneel[2], and Sangjin Lee[1]***

[1] Center for Information Security Technologies(CIST),
Korea University, Seoul, Korea
{hongdj,sangjin}@cist.korea.ac.kr
[2] ESAT/SCD-COSIC, Katholieke Universiteit Leuven, Belgium
bart.preneel@esat.kuleuven.ac.be

Abstract. Universal One-Way Hash Functions (UOWHFs) are families of cryptographic hash functions for which first a target input is chosen and subsequently a key which selects a member from the family. Their main security property is that it should be hard to find a second input that collides with the target input. This paper generalizes the concept of UOWHFs to UOWHFs of order r. We demonstrate that it is possible to build UOWHFs with much shorter keys than existing constructions from fixed-size UOWHFs of order r. UOWHFs of order r can be used both in the linear $(r+1)$-round Merkle-Damgård construction and in a tree construction.

Keywords: Hash Function, Collision Resistant Hash Function (CRHF), Universal One-Way Hash Function (UOWHF), Higher Order Universal One-Way Hash Function

1 Introduction

Since the introduction of the notion of UOWHFs by Naor and Yung in 1989 [5], it is widely believed that UOWHFs form an attractive alternative to CRHFs (Collision Resistant Hash Functions). The main requirement for a UOWHF is that it is hard to find a second preimage. First a challenge input is selected by the opponent, subsequently a key is chosen which selects a member of the class of functions and only after this choice the opponent has to produce a second preimage with the same hash value (for this key) as the challenge input. This should be contrasted to CRHFs, where first a key is selected and subsequently a two colliding inputs need to be found; due to the birthday paradox, a black box approach for a CRHF with an n-bit result takes on average about $2^{n/2}$ queries. Simon [10] has demonstrated that a UOWHF is a strictly weaker concept than a CRHF. UOWHFs can replace CRHFs in many applications; even for digital

* Supported by Korea University Grant in 2003 year.
** A guest researcher at ESAT/SCD-COSIC, K.U.Leuven from 2003 to 2004.
*** Supported (in part) by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

signatures this is feasible, but it should be noted that one becomes vulnerable to attacks by the signers (who can cheat and choose the key before the target message). The concept of UOWHFs has been generalized by Zheng *et al.* [12] and by Mironov [4].

A standard approach to construct hash functions that take input strings of arbitrary length is to start from a compression function that compresses input strings of fixed length. For CRHFs, the Merkle-Damgård construction is a widely used and efficient method [2, 3]. Both authors showed independently that it is sufficient for the hash function to be collision resistant that the compression function is. Damgård also proposed a tree construction. Naor and Yung showed that it is possible in principle to build UOWHFs by composition [5]. However, Bellare and Rogaway showed that even if the compression function is a UOWHF, a 2-round Merkle-Damgård iteration of this function may not be a UOWHF.

Subsequently, provably secure constructions have been developed based on compression functions at the cost of an increase in key length. Bellare and Rogaway [1] propose two types of constructions.

- The first type has a linear structure; two variants of the Merkle-Damgård construction were shown to be secure: the basic linear hash and the XOR linear hash. Later, Shoup improved the XOR linear hash construction. He shows that if one has a fixed size UOWHF which maps n bits to m bits (with $n > m$), one can construct a UOWHF that can hash messages of bit-length $2^t(m-n)+m$ bits to m using a key of bit-length $t \cdot m$ and 2^t applications of the compression function. Mironov has proved that this construction is optimal in terms of key size among linear constructions [4].
- The second type has a tree structure. Here the two constructions with a security proof are the basic tree hash and the XOR tree hash (they extend the work of [5]). XOR tree hash has subsequently been improved further, a.o. by Sarkar [7, 8] and by Lee *et al.* [11], who reduce the key size and extend these structures to higher dimensional parallel constructions.

1.1 Motivation

The special UOWHF made by Bellare and Rogaway [1] loses its universal one-wayness when it is extended to 2-round Merkle-Damgård construction. This example motivated us to study general constructions that work for any UOWHF compression function. It means that the Merkle-Damgård construction cannot be used for extending a universal one-way compression function in general. However, this property does not applied to all UOWHFs. The compression functions of certain UOWHFs may not lose their universal one-wayness until they are extended to 3-round Merkle-Damgård construction. In this case, a 2-round Merkle-Damgård construction based on the compression function can be used as another compression function and so the key size of the whole scheme is reduced by a factor of 2. This lead to promising results, since an important goal of research on constructions extending UOWHFs has been optimalization of the key size. We began with the Merkle-Damgård construction, but we found that the tree construction has the same problem as the Merkle-Damgård construction.

Intuitively, a UOWHF which does not lose its universal one-wayness until it is extended to 3-round Merkle-Damgård construction is a slightly stronger primitive than Bellare-Rogaway's special UOWHF. More generally, a UOWHF which does not lose its universal one-wayness until it is extended to more round Merkle-Damgård construction is stronger. So, we need new security notions to classify UOWHFs.

1.2 Our Contribution

We define the order of a UOWHF. We can classify UOWHFs according to the order. The classes of UOWHFs of same order form a chain between CRHF and UOWHF classes.

We show in Theorem 1 that if a UOWHF has a higher order, a Merkle-Damgård construction with more rounds based on it becomes a UOWHF. Theorem 3 states that if a UOWHF has a higher order, a tree construction with more levels becomes a UOWHFs. Theorems 1 and 3 are our main results. They consider collisions of the same length only, since we want to use our Merkle-Damgård and tree constructions only as a compression function which plays the role of a building block in the known constructions. Theorems 2 and 4 are generalizations of Theorem 1 and 3 which are mainly of theoretical interest.

1.3 Organization of this Paper

This paper is organized as follows. Section 2 introduces our notation and definitions and presents the counterexample of Bellare and Rogaway. In Sect. 3, our new definition of higher order UOWHFs is introduced. Section 4 and 5 present respectively the Merkle-Damgård and the tree construction based on higher order UOWHFs. Some concluding remarks are made in Sect. 6.

2 Preliminaries

We will follow the notation and computation models in [1].

2.1 Notation

We denote the concatenation of strings x and x' by $x||x'$ or xx'. Σ^n is the set of all strings of bit-length n. We use the notation Σ_n^m instead of Σ^{nm} when we want to stress that each string consists of m blocks of bit-length n. The set of all strings whose lengths are multiple of n is denoted by Σ_n^+.

A hash function family is a function $H : \Sigma^k \times \Sigma^m \to \Sigma^c$, where Σ^k is the key space, Σ^m is the message space, and Σ^c is the set of hash values. We often need to change Σ^m to describe different hash function families.

We write $x \xleftarrow{R} \Sigma^n$ for choosing a string of n-bit length uniformly at random. For a string x, $|x|$ is its bit-length. When A is an algorithm, program or adversary, $A(x) \to y$ means that A gets an information of x to output y. When we want to

address that A has no information to outputs y, we write $A(null) \to y$ with the null string $null$.

We take the RAM (Random Access Machine) model of computation which is also used in [1], and measure the running time of a program with respect to that model. If $H : \Sigma^k \times \Sigma^m \to \Sigma^c$ is a hash function family, we let T_H indicate the worst-case time to compute $H(K, x)$, in the underlying model of computation, when $K \in \Sigma^k$ and $x \in \Sigma^m$.

2.2 Definitions of CRHF and UOWHF

Recently, Rogaway and Shrimpton suggested seven simple and nice definitions of hash functions including CRHF and UOWHF [6], but we prefer to use some games to define our objects and to describe our work.

Definition 1 (CRHF). *A hash function family $H : \Sigma^k \times \Sigma^m \to \Sigma^c, m \geq c$, is (t, ε)-CRHF if no adversary A wins in the following game with the probability ε and within the time t:*

Game(CRHF,A,H)
$K \xleftarrow{R} \Sigma^k$
$A(K) \to (x, x')$

A wins if $x \neq x'$ and $H(K, x) = H(K, x')$.

In the game of Definition 1, the adversary gets the key K of H. This implies that the adversary knows everything about $H(K, \cdot)$ and so it can try any experiments until it produces its output within the time t. However, the behavior of the adversary is more restricted in Definition 2.

Definition 2 (UOWHF). *A hash function family $H : \Sigma^k \times \Sigma^m \to \Sigma^c, m \geq c$, is (t, ε)-UOWHF if no adversary $A = (A_1, A_2)$ wins in the following game with the probability ε and within the time t:*

Game(UOWHF,A,H)
$A_1(null) \to (x, State)$
$K \xleftarrow{R} \Sigma^k$
$A_2(K, x, State) \to x'$

$A = (A_1, A_2)$ wins if $x \neq x'$ and $H(K, x) = H(K, x')$.

In Definition 2, algorithm A_1 outputs the target message x. The only information which the adversary has before producing the target message is H. *State*, the other output of A_1, is some extra state information which helps A_2 to find a collision. Algorithm A_2 outputs the sibling message x' on input $(x, State)$.

Strictly speaking, when we are given $H : \Sigma^k \times \Sigma^m \to \Sigma^c$, we should call it a *CRHF family* or a *UOWHF family*, but for simplicity we often just call it *CRHF* or *UOWHF*.

3 Higher Order UOWHFs

Let us revisit Definition 2. No access to any oracles is given to A_1. A_1 outputs $(x, State)$ with no information. So, random selection of K from $\mathcal{K}$ is independent of A_1's behavior. Consequently, changing the order of steps 1 and 2 in the game doesn't effect the success probability of the adversary. So, the following game is essentially equivalent to the game in Definition 2.

Definition 3. *A hash function family $H : \Sigma^k \times \Sigma^m \to \Sigma^c, m \geq c$, is (t, ε)-UOWHF′ if no adversary $A = (A_1, A_2)$ wins in the following game with the probability ε and within the time t:*

Game(UOWHF′, A, H)
$K \xleftarrow{R} \Sigma^k$
$A_1(null) \to (x, State)$
$A_2(K, x, State) \to x'$

$A = (A_1, A_2)$ wins if $x \neq x'$ and $H(K, x) = H(K, x')$.

However, unlike the game in Definition 2, we can add an oracle $\mathcal{O}^{H(K,\cdot)}$ to the game in Definition 3, which gets a query x and returns $y = H(K, x)$. We can then allow the adversary to access the oracle before he chooses the target message. Now we give the following definition. Let Q be a set of adaptive query-answer pairs associated with the oracle $\mathcal{O}^{H(K,\cdot)}$ which is initialized to the empty set $\emptyset$ in the game.

Definition 4 (r-th order UOWHF). *A hash function family $H : \Sigma^k \times \Sigma^m \to \Sigma^c, m \geq c$, is (t, ε)-UOWHF(r) if no adversary $A = (A_1, A_2)$ wins in the following game with the probability ε and within the time t:*

Game(UOWHF(r), A)
$K \xleftarrow{R} \Sigma^k$; $Q \leftarrow \emptyset$
if $r > 0$ **do**:
  for $i = 1, ..., r$ **do**:
    $A_1(Q) \to x_i$
    $y_i \leftarrow \mathcal{O}^{H(K,x_i)}$
    $Q \leftarrow \{(x_i, y_i)\} \cup Q$
$A_1(Q) \to (x, State)$
$A_2(K, x, State) \to x'$

$A = (A_1, A_2)$ wins if $x \neq x'$ and $H(K, x) = H(K, x')$.

Indeed, the hash function families which satisfy Definition 3 can be regarded as UOWHF(0) families. The relationships among Definitions 1, 2, 3 and 4 can be summarized as follows.

Proposition 1. *Let $H : \Sigma^k \times \Sigma^m \to \Sigma^c, m \geq c$, be a hash function family. Then,*

1. *H is a (t,ε)-UOWHF $\Leftrightarrow$ H is a (t,ε)-UOWHF(0).*
2. *For any $r \geq 0$, H is a (t',ε)-UOWHF$(r+1) \Rightarrow$ H is a (t,ε)-UOWHF(r), where $t = t' - \Theta(T_H + m + c)$.*
3. *For any $r \geq 0$, H is a (t',ε)-CRHF $\Rightarrow$ H is a (t,ε)-UOWHF(r), where $t = t' - \Theta(r)(T_H + m + c)$.*

Proof. The proofs of 1 and 2 are trivial. So, we only prove 3. Suppose that $A = (A_1, A_2)$ is an adversary for H in the UOWHF(r) sense. We use it to make the adversary B who works in **Game**(CRHF,B, H) as follows.

Game(CRHF,B, H)
$K \xleftarrow{R} \Sigma^k$
$B(K)$ **do**:
  $Q \leftarrow \emptyset$
  if $r > 0$ **do**:
    for $i = 1, ..., r$ **do**:
      $A_1(Q) \rightarrow x_i$
      $y_i \leftarrow H(K, x_i)$
      $Q \leftarrow \{(x_i, y_i)\} \cup Q$
  $A_1(Q) \rightarrow (x, State)$
  $A_2(K, x, State) \rightarrow x'$
  output (x, x')

In the above game, B simulates $\mathcal{O}^{H(K,\cdot)}$ for A_1. Since B just outputs a collision which A found, the probability that B wins the game is same as A. The running time of B is at most $t + \Theta(r)(T_H + m + c)$. □

We claim that H is not a UOWHF(1) in Bellare and Rogaway's example [1]. If the adversary asks any query x and get the answer $y = H(K, x)$, then he would obtain the key K and can make a collision easily.

4 Merkle-Damgård Construction Based on Higher Order UOWHF

Suppose we have a hash function family $H : \Sigma^k \times \Sigma^{c+m} \rightarrow \Sigma^c$, where m is a positive integer. The Merkle-Damgård construction of H with variable initial value gives a hash function family MD$[H] : \Sigma^k \times (\Sigma^c \times \Sigma_m^+) \rightarrow \Sigma^c$. For each key $K \in \Sigma^k$ and any message $x = x_0x_1...x_n$, where $x_0 \in \Sigma^c$ and $x_i \in \Sigma^m$ for $i = 1, ..., n$, MD$[H]$ is defined as follows.

Algorithm MD$[H](K, x)$
$n \leftarrow (|x| - c)/m$
$y_0 \leftarrow x_0$
for $i = 1, ..., n$ **do**:
  $y_i \leftarrow H_K(y_{i-1}||x_i)$

return y_n

If MD[H] only takes $(c + nm)$-bit messages for a fixed n, it would always have n rounds. In that case we use the notation $\mathrm{MD}_n[H]$ instead of MD[H]. In the following theorem, we say that if H is a UOWHF(r), the $(r + 1)$-round $\mathrm{MD}_{r+1}[H]$ is a UOWHF.

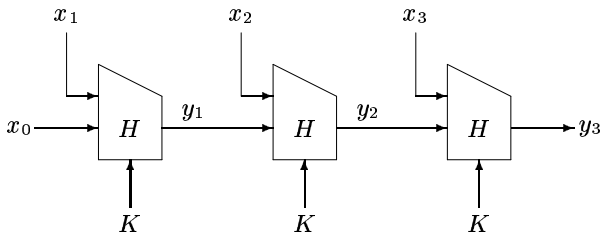

Fig. 1. 3-round Merkle-Damgård construction $\mathrm{MD}_3[H]$

Theorem 1. *Let* $H : \Sigma^k \times \Sigma^{c+m} \to \Sigma^c$ *be a* (t', ε')*-UOWHF*(r)*. Then,* $\mathrm{MD}_{r+1}[H] : \Sigma^k \times \Sigma^{c+rm} \to \Sigma^c$ *is a* (t, ε)*-UOWHF, where* $\varepsilon = (r+1)\varepsilon'$ *and* $t = t' - \Theta(r)(T_H + m + c)$.

Proof. Let $x, x' \in \Sigma^{c+(r+1)m}$ be a collision for $\mathrm{MD}_{r+1}[H](K, \cdot)$. We observe that there exists an index $j \in \{1, ..., r+1\}$ such that

$$\begin{aligned} \mathrm{MD}[H](K, x_0 x_1 \cdots x_j) &= \mathrm{MD}[H](K, x'_0 x'_1 \cdots x'_j) \\ \mathrm{MD}[H](K, x_0 x_1 \cdots x_{j-1}) || x_j &\neq \mathrm{MD}[H](K, x'_0 x'_j \cdots x'_{j-1}) || x'_j. \end{aligned} \tag{1}$$

We will exploit this below.

Assume that $A = (A_1, A_2)$ is an adversary who breaks $\mathrm{MD}_{r+1}[H]$ with inputs of equal-length in the UOWHF sense. We use it to make the adversary $B = (B_1, B_2)$ who works in the **Game**(UOWHF(r),B, H) as follows.

Game(UOWHF(r),B, H)
$K \xleftarrow{R} \Sigma^k$; $Q \leftarrow \emptyset$
if $r > 0$ **do**:
  for $j = 1, ..., r$ **do**:
    $B_1(Q)$ **do**:
      if $j = 1$ **do**:
        $A_1(null) \to (x, State_A)$
        $y_0 \leftarrow x_0$
        query $y_0 || x_1$ **to** $\mathcal{O}^{H(K,\cdot)}$
      if $j > 1$ **do**:
        query $y_{j-1} || x_j$ **to** $\mathcal{O}^{H(K,\cdot)}$

$y_j \leftarrow \mathcal{O}^{H(K,y_{j-1}||x_j)}$
$Q \leftarrow \{(y_{j-1}||x_j, y_j)\} \cup Q$
$B_1(Q)$ **do**:
$i \xleftarrow{R} \{1, ..., r+1\}$
output $(y_{i-1}||x_i, State_B)$
$B_2(K, y_{i-1}||x_i, State_B)$ **do**:
$A_2(K, x, State_A) \to x'$
$y'_{i-1} \leftarrow \mathrm{MD}[H](K, x'_0 x'_1 \cdots x'_{i-1})$
output $y'_{i-1}||x'_i$

The adversary B works in the game as follows. When $j = 1$, Q is empty and B_1 runs A_1 to obtain $(x, State_A)$ where $x = x_0 x_1 \cdots x_{r+1}$. Then B_1 sets y_0 to x_0 and sends a query $y_0||x_1$ to the oracle $\mathcal{O}^{H(K,\cdot)}$. When $j \neq 1$, Q is nonempty and B_1 sends a query $y_{j-1}||x_j$ to the oracle $\mathcal{O}^{H(K,\cdot)}$. After collecting r adaptive query-answer pairs, B_1 selects $i \in \{1, ..., r+1\}$ at random, and then outputs $y_{i-1}||x_i$ and $State_B = (i, x, State_A)$ as the target message and an additional state information for B_2, respectively. On input $(K, y_{i-1}||x_i, State_B)$, B_2 runs A_2 by giving $(K, x, State_A)$. Once A_2 outputs its sibling message x', B_2 computes y'_{i-1} and outputs $y'_{i-1}||x'_i$ as its sibling message.

Now we must bound the probability that $(y_{i-1}||x_i, y'_{i-1}||x'_i)$ is a collision for $H(K, \cdot)$ in the game. Note that i was chosen at random, so if (x, x') is a collision for $\mathrm{MD}_{r+1}[H](K, \cdot)$ then we have $i = j$ with probability $1/(r+1)$, where j is the value of Equation (1). So, $\varepsilon' > \varepsilon/(r+1)$.

The running time of B is that of A plus the overhead. This overhead is $\Theta(r)(T_H + m + c)$. The choice of t in the theorem statement makes all this at most t', from which we conclude the result. □

Assume that a domain $\mathcal{D}$ and a range $\mathcal{R}$ are fixed. We regard the notion of *UOWHF(r)* as the class of all UOWHF(r). We also consider the class of all the hash functions which do not lose universal one-wayness (upto equal-length collisions) until they are extended to (r+1)-round Merkle-Damgård construction, *UOW-MD(r)*. From Proposition 1 and Theorem 1, it is easy to see that these classes forms two different chains between the classes *CRHF* and *UOWHF* with the same domain $\mathcal{D}$ and the same range $\mathcal{R}$, and that for each integer $r > 0$, *UOWHF(r)* implies *UOW-MD(r+1)* (see Fig. 2).

We can generalize Theorem 1 to MD[H] taking inputs of variable length. We assume that the message is always padded such that its length is a multiple of m. There are many padding methods but we don't mention any specific one. We use the notation $(t, \mu_1, \mu_2, \varepsilon)$-UOWHF instead of (t, ε)-UOWHF. μ_1 is the bound on the length of the target message and μ_2 is the bound on the length of the sibling message. Note that the only restriction on μ_2 is that the algorithms on the sibling message should be computable in polynomial time.

Theorem 2. *Suppose* $H : \Sigma^k \times \Sigma^{c+m} \to \Sigma^c$ *be a* (t', ε')*-UOWHF*(r). *Suppose* $\mu_1 - c$ *and* $\mu_2 - c$ *are multiples of* m. *Then, for* $\mu_1 \leq c + (r+1)m$ *and a proper* μ_2, $\mathrm{MD}[H] : \Sigma^k \times (\Sigma^c \times \Sigma_m^+) \to \Sigma^c$ *is a* $(t, \mu_1, \mu_2, \varepsilon)$*-UOWHF, where* $\varepsilon = \sigma_{\min}\varepsilon'$

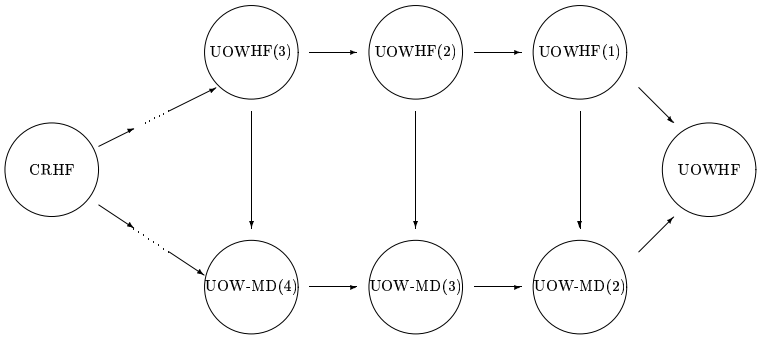

Fig. 2. Two chains between CRHF and UOWHF. Each arrow means the implication.

and $t = t' - \Theta(\sigma_{\max})(T_H + m + c)$ *for* $\sigma_{\min} = \min\{(\mu_1 - c)/m, (\mu_2 - c)/m\}$ *and* $\sigma_{\max} = \max\{(\mu_1 - c)/m, (\mu_2 - c)/m\}$.

Proof. The proof is similar to that of Theorem 1.

5 Tree Construction Based on Higher Order UOWHF

If we are given a UOWHF family $H : \Sigma^k \times \Sigma^m \to \Sigma^c$ and m is a multiple of c, we can extend it more efficiently by using a tree structure. If $m = dc$ for a positive integer d, we can use a d-ary tree structure. Since a tree construction consists of parallel procedures, it can be more efficient than the Merkle-Damgård construction if multiple processors are available.

Firstly, we define the parallel construction $\mathrm{PA}[H]$ on H. $\mathrm{PA}_n[H]$ consists of n components $\mathrm{PA}_n[H]_1, ..., \mathrm{PA}_n[H]_n$. For $K \in \Sigma^k$, each component function $\mathrm{PA}_n[H]_i$ is

$$\mathrm{PA}_n[H]_i(K, x) = \begin{cases} H(K, x_i) & \text{if } |x| = dc \\ x_i & \text{if } |x| = c \end{cases}$$

That is, the domain of $\mathrm{PA}_n[H]_i(K, \cdot)$ is $\Sigma^c \cup \Sigma^{dc}$, while the domain of $H(K, \cdot)$ is Σ^c.

We assume that we are given $x = x_1 \cdots x_n$ for each $x_i \in \Sigma^c \cup \Sigma^{dc}$. For a key $k \in \Sigma^k$, $\mathrm{PA}_n[H](K, x)$ is defined as follows.

Algorithm $\mathrm{PA}_n[H](K, x)$
$n \leftarrow \log_{dc} |x|$
for $i = 1, ..., n$ **do**:
  $y_i \leftarrow \mathrm{PA}_n[H]_i(K, x_i)$
return $y_1 || \cdots || y_n$

Now we define a tree construction $\mathrm{TR}[H]$ based on H. We begin with the message space $\Sigma_c^{d^l}$. We denote the tree construction on H to hash only messages

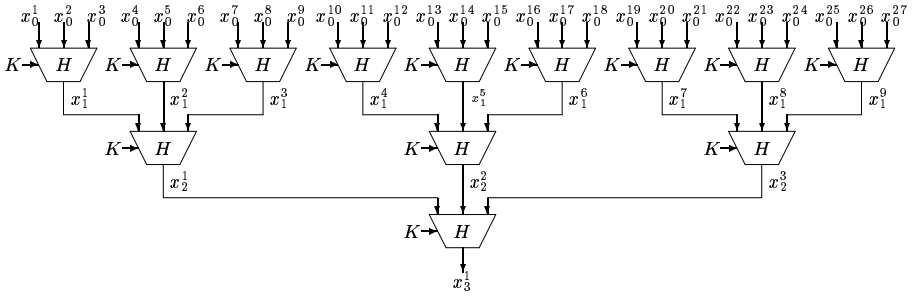

Fig. 3. 3-level tree construction $\mathrm{TR}_3[H]$ for the case of $l = 3$

in $\Sigma_c^{d^l}$ as $\mathrm{TR}_l[H]$. For each key $K \in \Sigma^k$ and any message $x \in \Sigma_c^{d^l}$, $\mathrm{TR}_l[H]$ is defined according to:

Algorithm $\mathrm{TR}_l[H](K, x)$
$Level[0] \leftarrow x$
for $i = 1, ..., l$ **do**
  $Level[i] \leftarrow \mathrm{PA}_{d^{l-i}}[H](K, Level[i-1])$
return $Level[l]$

Write $Level[0] = x$ as $x_0 = x_{0,1}x_{0,2}\cdots x_{0,d^l}$ where $x_{0,i} \in \Sigma^c$ for $i = 1, ..., d^l$. Then, we can see $\mathrm{TR}_l[H](K, x)$ is computed as follows.

$$\begin{aligned} Level[0] &= x_0^1 x_0^2 \ \cdots \ \cdots \ \cdots \ x_0^{d^l} \\ Level[1] &= x_1^1 x_1^2 \ \cdots \ \cdots \ x_1^{d^{l-1}} \\ &\vdots \\ Level[l-1] &= x_{l-1}^1 \ \cdots \ x_{l-1}^d \\ Level[l] &= x_l^1 \end{aligned}$$

where $x_i^j = H(K, x_{i-1}^{(j-1)d+1} x_{i-1}^{(j-1)d+2} \cdots x_{i-1}^{jd})$.

The following theorem states that if H is a UOWHF(r) and $r = (d^l - 1)/(d - 1)$, then $\mathrm{TR}_l[H]$ is a UOWHF.

Theorem 3. *Let $H : \Sigma^k \times \Sigma_c^d \to \Sigma^c$ be a (t', ε')-UOWHF(r) and $r = (d^l - d)/(d-1)$. Then $\mathrm{TR}_l[H] : \Sigma^k \times \Sigma_c^{d^l} \to \Sigma^c$ is a (t, ε)-UOWHF, where $\varepsilon = (r+1)\varepsilon'$, and $t' = t + \Theta(d^l)(T_H + dc)$.*

Proof. Assume that $x, y \in \Sigma_c^{d^l}$ is a collision for $\mathrm{TR}_l[H](K, \cdot)$. We observe that there exist $\alpha \in \{1, ..., l\}$ and $\beta \in \{1, ..., d^{l-\alpha}\}$ such that

$$\begin{aligned} x_\alpha^\beta &= y_\alpha^\beta \\ x_{\alpha-1}^{(\beta-1)d+1} || \cdots || x_{\alpha-1}^{\beta d} &\neq y_{\alpha-1}^{(\beta-1)d+1} || \cdots || y_{\alpha-1}^{\beta d}. \end{aligned} \tag{2}$$

We will exploit this below.

Assume that $A = (A_1, A_2)$ is an adversary who breaks $\mathrm{TR}_l[H]$ with inputs of equal-length in the UOWHF sense. We use it to make the adversary $B = (B_1, B_2)$ who works in **Game**(UOWHF(r),B, H) as follows.

Game(UOWHF(r),B, H)
$K \xleftarrow{R} \Sigma^k$; $Q \leftarrow \emptyset$
if $r > 0$ **do**:
  for $u = 1, ..., l-1$ **do**:
    for $v = 1, ..., d^{l-u}$ **do**:
      $B_1(Q)$ **do**:
        if $(u, v) = (1, 1)$ **do**:
          $A_1(null) \to (x, State_A)$
          query $x_0^1 || \cdots || x_0^d$ **to** $\mathcal{O}^{H(K,\cdot)}$
        if $(u, v) \neq (1, 1)$ **do**:
          query $x_{u-1}^{(v-1)d+1} || \cdots || x_{u-1}^{vd}$ **to** $\mathcal{O}^{H(K,\cdot)}$
        $x_u^v \leftarrow \mathcal{O}^{H(K, x_{u-1}^{(v-1)d+1} || \cdots || x_{u-1}^{vd})}$
        $Q \leftarrow \{(x_{u-1}^{(v-1)d+1} || \cdots || x_{u-1}^{vd}, x_u^v)\} \cup Q$
$B_1(Q)$ **do**:
  $i \xleftarrow{R} \{1, ..., l\}$; $j \xleftarrow{R} \{1, ..., d^{l-i}\}$
  output $(x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}, State_B)$
$B_2(K, x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}, State_B)$ **do**:
  $A_2(K, x, State_A) \to y$
  output $y_{i-1}^{(j-1)d+1} || \cdots || y_{i-1}^{jd}$

The adversary B works in the game as follows. When $(u, v) = (1, 1)$, Q is empty and B_1 runs A_1 to obtain $(x, State_A)$ where $x = x_0^1 || \cdots || x_0^{d^l} \in \Sigma_c^{d^l}$. Then, B_1 sends a query $x_0^1 || \cdots || x_0^d$ to the oracle $\mathcal{O}^{H(K,\cdot)}$. When $(u, v) \neq (1, 1)$, Q is nonempty. B_1 sends a query $x_{u-1}^{(v-1)d+1} || \cdots || x_{u-1}^{vd}$ to the oracle $\mathcal{O}^{H(K,\cdot)}$. After collecting r adaptive query-answer pairs, B_1 randomly selects i and j from $\{1, ..., l\}$ and $\{1, ..., d^{l-i}\}$, respectively. The B_1 outputs $x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}$ as the target message and $State_B = (i, j, x, State_A)$ as an additional state information for B_2. On the input $(x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}, State_B)$, B_2 runs A_2 by giving $(K, x, State_A)$. Once A_2 outputs its sibling message x', B_2 computes and outputs $y_{i-1}^{(j-1)d+1} || \cdots ||\ y_{i-1}^{jd}$ as its sibling message.

Now we must bound the probability that $x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}, y_{i-1}^{(j-1)d+1} || \cdots || y_{i-1}^{jd}$ is a collision for $H(K, \cdot)$. The number of possibilities for (i, j) is at most $d^0 + \cdots + d^{l-1} = (d^l - 1)/(d - 1)$. Note that i and j were chosen randomly and independently, so if x, y is a collision for $\mathrm{TR}_l[H](K, \cdot)$ then we have $(i, j) = (\alpha, \beta)$ with probability $(d - 1)/(d^l - 1)$, where (α, β) is the pair in Equation (2). So, $\varepsilon' > \varepsilon(d - 1)/(d^l - 1)$.

The running time of B is that of A plus the overhead, which is equal to $\Theta(d^l)(T_H + dc)$. □

The tree construction can hash the messages of variable lengths like the Merkle-Damgård construction. We assume that we are given a message x. When $\lceil \log_d \frac{|x|}{c} \rceil = l$, we pad x such that the length of the padded message x^* is the smallest value larger than $|x|$ of the form $|x^*| = (d^l - qd + q)c$ for some integer $0 < q < d^{l-1}$. Then, the number of applications of the underlying hash function is $1 + d + d^2 + \cdots + d^{l-1} - q = \frac{d^l - 1}{d-1} - q$. See Fig. 4, 5, 6 and 7 for the case of $l = 2, d = 4$. We denote the set of the padded messages in such way by

$$\mathcal{S}(c,d) = \{x \in \Sigma^* || x| = (d^l - qd + q)c \text{ for some integers } l > 0 \text{ and } 0 \le q < d^{l-1}\}$$

Now we generalize Theorem 3.

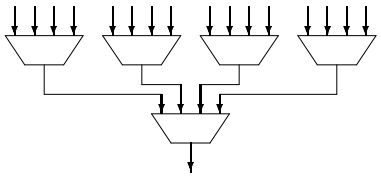

Fig. 4. $|x|/c = 16 = 4^2$

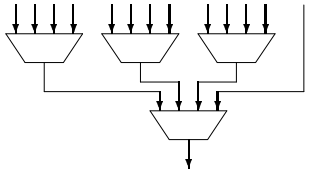

Fig. 5. $|x|/c = 13 = 4^2 - 1 \cdot 4 + 1$

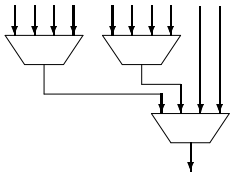

Fig. 6. $|x|/c = 10 = 4^2 - 2 \cdot 4 + 2$

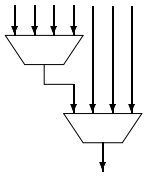

Fig. 7. $|x|/c = 7 = 4^2 - 3 \cdot 4 + 3$

Theorem 4. *Suppose* $H : \Sigma^k \times \Sigma_c^d \to \Sigma^c$ *be a* (t', ε')*-UOWHF*(r)*. Suppose* $\mu_i = d^{l_i} - q_i d + q_i$ *for* $i = 1, 2$*. Then for* $\mu_1 \le c(r(d-1) + d)$ *and a proper* μ_2*,* $\mathrm{TR}[H] : \Sigma^k \times \mathcal{S}(c,d) \to \Sigma^c$ *is a* (t, ε)*-UOWHF, where* $\varepsilon = \sigma_{\min}\varepsilon'$ *and* $t' = t + \Theta(\sigma_{\max})(T_H + dc)$ *for* $\sigma_{\min} = \min\{\frac{d^{l_1}-1}{d-1} - q_1, \frac{d^{l_2}-1}{d-1} - q_2\}$ *and* $\sigma_{\max} = \max\{\frac{d^{l_1}-1}{d-1} - q_1, \frac{d^{l_2}-1}{d-1} - q_2\}$.

Proof. The proof is similar to that of Theorem 3. □

6 Conclusion

We defined the order of a UOWHF family and showed how much the efficiency of known constructions for UOWHFs is improved by the notion of the order. Our main results are as follows.

- If the order of the underlying UOWHF H is r, then the $(r+1)$-round Merkle-Damgård construction $MD_{r+1}[H]$ is also a UOWHF. If the resulting function $MD_{r+1}[H]$ is used as a building block in existing constructions with linear structure, the key size can be reduced with at most a factor of $(r+1)$.
- If the order of the underlying UOWHF $H : \Sigma^k \times \Sigma_c^d \rightarrow \Sigma^c$ is $r = \frac{d^l - d}{d-1}$, then the l-level tree construction $TR_l[H]$ is also a UOWHF. If the resulting function $TR_l[H]$ is used as a building block in existing constructions with tree structure, the key size can be reduced with at most a factor of l.

References

1. M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical," In B.S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, LNCS 1294, Springer-Verlag, pages 470–484, 1997.
2. I. Damgård, "A Design Priciple for Hash Functions," In G. Brassard, editor, *Advances in Cryptology – Crypto'89*, LNCS 435, Springer-Verlag, pages 416–427, 1989.
3. R. Merkle, "One way hash functions and DES," In G. Brassard, editors, *Advances in Cryptology – Crypto'89*, LNCS 435, Springer-Verlag, pages 428–446, 1989.
4. I. Mironov, "Hash Functions: From Merkle-Damgård to Shoup," In B. Pfitzmann, editor, *Advances in Cryptology – Eurocrypt 2001*, LNCS 2045, Springer-Verlag, pages 166-181, 2001.
5. M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," *Proceedings of the Twenty-first ACM Symposium on Theory of Computing*, pages 33–43, 1989.
6. P. Rogaway and T. Shrimpton, "Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance," In B. Roy and W. Meier, editors, *Fast Software Encryption 2004*, LNCS 3017, Springer-Verlag, pages 371–388, 2004.
7. P. Sarkar, "Constuction of UOWHF: Tree Hashing Revisited," Cryptology ePrint Achive, http://eprint.iacr.org/2002/058.
8. P. Sarkar, "Domain Extenders for UOWHF: A Generic Lower Bound om Key Expansion and a Finite Binary Tree Algorithm," Cryptology ePrint Archive, http://eprint.iacr.org/2003/009.
9. V. Shoup, "A composite theorem for universal one-way hash functions," In B. Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, LNCS 1807, Springer-Verlag, pages 445–452, 2000.
10. D. Simon, "Finding collisions on a one-way street: can secure hash functions be based on general assumptions?," In K. Nyberg, editor, *Advances in Cryptology – Eurocrypt'98*, LNCS 1403, Springer-Verlag, pages 334–345, 1998.
11. W. Lee, D. Chang, S. Lee, S. Sung, and M. Nandi, "New Parallel Domain Extenders for UOWHF," *Advances in Cryptology – ASIACRYPT 2003*, LNCS 2894, Springer-Verlag, pages 208–227, 2003.
12. Y. Zheng, T. Matsumoto, H. Imai, "Connections between several versions of one-way hash functions," *Trans. IEICE E*, Vol. E73, No. 7, pp. 1092–1099, July 1990.