A New Tree based Domain Extension of UOWHF

Mridul Nandi
Applied Statistical Unit
Indian Statistical Institute
203, B.T.Road , Kolkata

21st June , 2003

Abstract

We present a new binary tree based parallel algorithm for extending the domain of a UOWHF. The key length expansion is $m(t + O(log^*(t)))$ bits. In particular, the key length expansion is $2m$ bits for $t = 2$; $m(t + 1)$ bits for $3 \leq t \leq 6$ and $m(t + 2)$ bits for $7 \leq t \leq 134$, where m is the length of the message digest and $t \geq 2$ is the height of the binary tree. The previously best known binary tree algorithm required a key length expansion of $m \times (t + \lfloor \log_2(t-1) \rfloor)$ bits. We also give a sufficient condition for valid domain extension in sequental domain extension.

Keywords : UOWHF, hash function, masking assignment, sequential construction, tree based construction.

1 Introduction

Nowadays, digital signature scheme is very important cryptographic primitive. To speed up the signature generation and verification algorithm it is always better to compress or hash the message. However, The compression function should satisfy some properties for signature scheme to be secure. First of all it should compress the message of arbitrary length. Moreover, it should be collision resistant, i.e. it is hard to find two different messages whose compressed images are same. The standard hash functions are SHA-1, MD4, MD5, RIPEMD etc. They are all treated as CRHF. The above hash functions have bounded domain. So, we need to extend the domain. One such method is MD construction. But, Bellare and Rogaway [1] showed that MD construction is not valid in case of UOWHF(Universal One Way Hash Function) which is introduced by Naor and Yung [5]. Then, why should we work with UOWHF? We put several reasons of considering UOWHF.

1. It is weaker primitive than CRHF so, it seems easier to build an efficient and secure UOWHF than to build an efficient and secure CRHF. We have already found an attack for MD4 when it is treated as CRHF but there is no known attack when it is treated as UOWHF. Furthermore Simon [10] had shown that there is an oracle relative to which UOWHF exists but CRHF do not exist.

2. In many applications, like building digital signature scheme, a UOWHF is sufficient.

In the light of the above discussion one should try to extend the domain of UOWHF. There are several known constructions but, in most of the cases sizes of keys grow sufficiently with message length. So, it is important to look for construction that extends the domain of a UOWHF for which the resulting increase in key length is minimum possible.

There are two approaches for this. One is sequential and another is tree based algorithm. Shoup [9] constructed an optimal sequential algorithm [9] and optimality is proved by Mironov [4]. But there is no known optimal construction for tree-based algorithm. We will present a tree based domain extension whose key length expansion is significantly less. In fact, we have got a lot of evidences[?] that the our algorithm will be optimal in the class of binary tree based algorithm. Tree based construction is more preferable than sequential algorithm as it has a capability of parallel processing. So, whenever parallel processing is practically feasible and not costly the tree-based algorithm can be used as it is much faster than sequential algorithm. In our algorithm total time is order of logarithm of total time of shoup's algorithm if parallelism is done. But only disadvantage is that here we need more masks (part of the key) than sequential algorithm. But it is not too large as in all practical situations we need two more masks than Shoup's algorithm.

We also give a sufficient condition for valid domain extension for sequential construction and it is likely that the condition is necessary. So, that will characterize the valid domain extension for sequential construction. In[?] we have also shown that the same condition becomes sufficient for general tree based domain extension.

Route of the paper : After giving an introduction of UOWHF and its domain extension we will generalize Shoup's sequential domain extension. We also provide a sufficient condition for valid domain extension. Then we will present our tree based domain extension and will give a proof of validness of the extension. Finally we will study about time (number of parallel steps) and space complexity (only key size) of our algorithm

2 Preliminaries

The following notations are used in this paper.

1. $[a, b] = \{a, a+1, \dots, b\}$ where a and b are integers.
2. Suppose A is a finite set. By $a \in_R A$ we mean that a is a uniform discrete random variable taking values from A.
3. $\nu_2(i) = j$ if $2^j | i$ and $2^{j+1} \not| \, i$.
4. For $t \geq 1$, $log^m(t)$ means that the function log applies m many times on t. $log^*(t) = m$ if $log^m(t) \leq 1$ but $log^{m-1}(t) > 1$.
5. Let $T_t = (V_t, A_t)$ be the full binary tree where $V_t = \{1, 2, \dots, 2^t - 1\}$ and $A_t = \{a_i; 2 \leq i \leq 2^t - 1\}$ where $a_i = (i, \lfloor i/2 \rfloor)$. $ht_t(i) = j$ means that $2^{t-j} \leq i \leq 2^{t+1-j} - 1$. So, root has ht t and all leave have ht 1. We can define level of a vertex by $t - ht(v)$. For any vertex i define, $V[i]$ by the complete binary sub-tree rooted at i.

A keyed family of hash function $\{h_k\}_{k \in \mathcal{K}}$ is called CRHF (Collision Resitant Hash Function) if given $k \in_R \mathcal{K}$ it is hard to find two different messages x and x' such that $h_k(x) = h_k(x')$. In case of UOWHF (Universal One-Way Hash Function) a strategy $\mathcal{A}$ of adversary runs in two stages. In the first stage $\mathcal{A}^{\text{guess}}$, the adversary finds the x to which he has to commit in Step 1. It also produces some auxiliary state information s. In the second stage $\mathcal{A}^{\text{find}}(x, k, s)$, the adversary either finds a x'

which provides a collision for h_k or it reports failure. Both $\mathcal{A}^{\text{guess}}$ and $\mathcal{A}^{\text{find}}(x,k,s)$ are probabilistic algorithms. The success probability of the strategy is measured over the random choices made by $\mathcal{A}^{\text{guess}}$ and $\mathcal{A}^{\text{find}}(x,k,s)$ and the random choice of k in step 2 of the game. We say that $\mathcal{A}$ is an (ϵ,a)-strategy if the success probability of $\mathcal{A}$ is at least ϵ and it invokes the hash function h_k at most a times. In this case we say that the adversary has an (ϵ,a)-strategy for $\{h_k\}_{k\in\mathcal{K}}$. Note that we do not include time as an explicit parameter though it would be easy to do so.

In this paper we will use the following conventions.

1. $\{h_k\}_{k\in\mathcal{K}}$ is always the base hash family where where $\mathcal{K}=\{0,1\}^K$ and $h_k:\{0,1\}^n\to\{0,1\}^m$. In case of sequential construction $n>m$ and in case of tree based construction $n>2m$.

2. We will construct extended $\{H_p\}_{p\in\mathcal{P}}$ where, $p=k||\mu_1||\mu_2||\ldots||\mu_l$ and μ_i are m-bit binary strings called masks and $|k|=K$. Each $H_p:\{0,1\}^N\to\{0,1\}^m$. Here, $N=n+r(n-m)$ in case of sequential algorithm and $N=n2^{t-1}+(n-2m)(2^{t-1}-1)$ in case of binary tree based algorithm.

3. In sequential construction input of H_p is written as $y=y_0||y_1||\ldots||y_r$ and in case of Tree based construction input of H_p is written as $x=x_1||\ldots||x_{2^t}-1$ where $|y_0|=n$, $|y_i|=n-m$ for $1\le i\le r$ and $|x_i|=n-2m$ for $1\le i\le 2^{t-1}-1$, $|y_i|=n$ for $2^{t-1}\le i\le 2^t-1$.

4. Let $i\in V_t$ and x be a message of length N. We define, $x(i)=x_i||x_{2i}||x_{2i+1}||x_{4i+1}||\ldots$ i.e. concatenating all x_j in ascending order of j where j runs in $V[i]$. In other words the part of the message used in the complete binary sub-tree rooted at i.

5. Let $\mu:[1,l]\to\{0,1\}^m$ be any function called mask funcion. Define, $\mu[j]=\mu(1)||\ldots\mu(j)$ where $1\le j\le l$. $\mu[j]$ can be thought of μ restricted on $[1,l]$. Given the key $p=k||\mu_0||\mu_1||\ldots||\mu_{l-1}$ we define masks function $\mu:[1,l]\to\{0,1\}^m$ such that $\mu(i)=\mu_i$. So, μ can also be written as $\mu[l]$.

We say that the adversary has an (ϵ,a) winning strategy $\mathcal{B}$ for $\{H_p\}_{p\in\mathcal{P}}$ if it has succes probability at least ϵ and it invokes h_k at most a times.

We say that the base hash family $\{h_k\}_{k\in\mathcal{K}}$ is UOWHF if for all polynomial time adversary success probility is negligible. Here, the security parameter is length of the message i.e. the length of the input.

We say that the extension or the rule of masking assignment which extends the hash family is valid if $\{h_p\}_{p\in\mathcal{P}}$ is UOWHF whenever $\{h_k\}_{k\in\mathcal{K}}$ is UOWHF.

3 Sequential Construction

The best known sequential algorithm is given by Shoup [9]. We will generalize the idea of the construction. We also give the sufficient condition for valid sequential construction. Let $\psi:[1,r]\to[1,l]$ be any function called a **masking assignment**. Fix a masking assignment ψ, $H_p(y)$ is computed by the following algorithm.

1. **Input:** $y=y_0||y_1||\ldots||y_r$ and $p=k||\mu_1||\mu_2||\ldots||\mu_l$.

2. $z_0=h_k(y_0)$

3. For $1 \le i \le r$, define $s_i = z_{i-1} \oplus \mu_{\psi(i)}$ and $z_i = h_k(s_i||y_i)$.

4. **Output:** z_r.

We say that the sequential construction is based on the masking assignment ψ. In Shoup's algorithm $\psi = \nu_2 + 1$ (in his paper ν_2 is masking assignment but that makes no difference). So, $l = 1 + \lfloor \log r \rfloor$. We will write $s(i, y, k, \mu) = s_i$ (in the algorithm with input (y, p) where, $p = k||\mu$).

Definition 1 *We say that ψ is* ***correct*** *at i where $1 \le i \le r$ if for each $C \in \{0,1\}^m$, $y \in \{0,1\}^N$ and for any hash function h_k mapping n-bits to m-bits, there is an algorithm called $KR_{seq}(i, y, k, C)$ which outputs $\mu = \mu_1||\mu_2||\ldots||\mu_l$ such that $s(i, y, k, \mu) = C$. We say ψ is* ***correct*** *if for each i, $1 \le i \le r$, ψ is correct at i. $KR_{seq}()$ is called a Key-Reconstruction algorithm. Any algorithm based on a corect masking assignment is called correct domain extension.*

A masking assignment is **totally correct** *if it is correct and there is a Key-Reconstruction algorithm which will generate random μ whenever C is a random string and other inputs are fixed.*

Definition 2 *We say that a domain extension algorithm is* **valid** *if $\{H_p\}$ is UOWHF whenever $\{h_k\}$ is UOWHF. In case of sequential construction if valid domain extension algorithm is based on a masking assignment ψ then we say that the masking assignment is* **valid**.

Definition 3 *A masking assignment $\psi : [1, r] \to [1, l]$ is* ***strongly even-free*** *(***even-free***) if for each $[a, b] \subset [1, r]$ there exists $c \in [a, b]$ such that $\psi(c)$ occurs exactly once (resply, odd times) in the sequence $\psi(a), \psi(a+1), \ldots, \psi(b)$. Call this c (also $\psi(c)$) a* **single-man** *for the interval $[a, b]$.*

Theorem 4 ***(Sufficient Condition for Valid Sequential Extension)***
If ψ is strongly even-free then ψ is totally correct.

Proof. We will define the key reconstructing algorithm $KR_{seq}(i, y, k, C, \psi)$. We add the masking assignment ψ as a parameter of KR.

1. If $i = 1$ then define $\mu_{\psi(1)} = C \oplus h_k(y_0)$.

2. If $i > 1$ then choose any c which is a single-man for the interval $[1, i]$. Compute $j \leftarrow i - c$, If $j = 0$ then goto step 4.

3. Let $\psi' : [1, j] \to [1, l]$ be a masking assignment such that $\psi'(n) = \psi(n + c)$ where $n \in [1, j]$. Take a random string D and then define, $y' = y'_0||\ldots||y'_j$ where, $y'_n = y_{n+c}$ when $n > 1$ and $y'_0 = D||y_c$. Run $KR_{seq}(j, y', k, C, \psi')$.

4. Define all yet undefined masks (i.e. after running KR some masks may not be defined as ψ' may not be onto or j can be 0) randomly. Compute $\mu_{\psi(c)} = z(c-1, y, k, \mu) \oplus D$. Note that to compute $z(c-1, y, k, \mu)$ we do not need the mask $\mu_{\psi(c)}$ as c is a single-man.

Note that, the above recurssive algorithm will always stop as $j < i$. The masking assignment ψ is nothing but ψ restricted at $[c, i]$. So, if $s(c, y, k, \mu) = D||y_c$ then by induction $s(i, y, k, \mu) = C$. But, $s(c, y, k, \mu) = D||y_c$ is true by definition of μ_c. It proves the correctness of ψ. If C is a random string then all masks μ is a random string as they are randomly defined (in step-4) or they are obtained by XOR-ing with a random string (in step-1). So, it is totally correct. ∎

Now we will try to characterize all valid masking assignments. From Mironov's paper we have seen that every valid masking assignment is even-free. Now we will prove that every **strongly**

even-free masking assignment is totally correct and hence valid. Totally correct implies valid is a basic idea of proving an extension is valid. In all known papers the same idea is used for proving validness of extension. We will give a proof in case of binary tree based domain extension in sec[?]. The same idea will carry through in case of sequential construction.

Theorem 5 ***(Sufficient Condition for Valid Sequential Extension)*** *If a sequential domain extension is based on a strongly even-free masking assignment ψ then the domain extension is valid.*

Proof. By the above lemma ψ is totally correct. The proof of totally correct implies valid is given in case of complete binary tree domain extension in section [?]. The same idea will carry through in case of sequential construction. So, we omit this proof. ■

Remark : Strongly even-free is sufficient condition for correct masking assignment. For example $\nu_2 + 1$. One can feel that the condition may be necessary. So, we may conjecture that, under the assumpition that the output of the base hash function is random, correctness of masking assignment is equivalent to strongly even-free property.

4 Binary Tree based Construction

In the previous section we study about sequential construction. Now, we will first define the generic algorithm based on full binary tree of height t. Let $T_t = (V_t, E_t)$ be the full binary tree where $V_t = \{1, 2, \ldots, 2^t - 1\}$ and $E_t = \{e_i; 2 \leq i \leq 2^t - 1\}$, $e_i = (i, \lfloor i/2 \rfloor)$. Like sequential construction any function $\psi : E_t \to [1, l]$ is a masking assignment. Let $x = x_1 || \ldots || x_{2^t - 1}$ be the input message of length N. Given $\psi, x, p = k||\mu$, $H_p(x)$ is computed by the following algorithm.

1. **Input:** $y = x_1||x_2|| \ldots ||x_{2^t-1}$ and $p = k||\mu_1||\mu_2|| \ldots ||\mu_l$.
2. If $2^{t-1} \leq i \leq 2^t - 1$ then, $z(i, x, k, \mu, t) = h_k(x_i)$ else if $1 \leq i < 2^{t-1}$ then $z(i, x, k, \mu, t) = h_k(s(2i, x, k, \mu, t)||s(2i + 1, x, k, \mu, t)||x_i)$ where, $s(i, x, k, \mu, t) = z(i, x, k, \mu, t) \oplus \mu_{\psi(e_i)}$.
3. **Output:** $z(1, x, k, \mu, t)$.

Define, the input of i^{th} node $input(i, x, k, \mu, t) = s(2i, x, k, \mu, t)||s(2i + 1, x, k, \mu, t)||x_i$. Output of node i is $z(i, x, k, \mu, t)$. We say that, the above complete binary tree based domain extension is based on the masking assignment ψ. Like Sequential algorithm we say that ψ is correct if there is a key reconstructing algorithm $KR_{tree}(i, x, k, t, r_0, r_1)$ where $|r_0| = |r_1| = m$ which outputs $\mu = \mu_1|| \ldots ||\mu_l$ such that $s(2i, x, k, \mu, t) = r_0$ and $s(2i + 1, x, k, \mu, t) = r_1$. ψ is totally correct if the output μ of teh key reconstructing algorithm is random string provided r_0, r_1 are random strings and other inputs are fixed.

Definition 6 *A masking assignment $\psi : E_t \to [1, l]$ is a* **level uniform** *masking assignment if there are two functions $\alpha, \beta : [2, t] \to [1, l]$ such that, $\psi(i) = \alpha(j)$ if i is odd and $\psi(i) = \beta(j)$ if i is even where, $j = ht_t(i) + 1$.*

We will first briefly state some standard Tree based constructions all of which are based on level-uniform masking assignment.

1. **Bellare-Rogaway** [1]

   $\alpha_t(i) = ht_t(i) - 1$ and $\beta_t(i) = t + ht_t(i) - 1$. In [1] it was shown that ψ is valid. Here, we need $2(t-1)$ masks.

2. **Sarkar** [8]

   $\alpha_t(i) = ht_t(i) - 1$ and $\beta_t(i) = t + \nu_2(ht_t(i) - 1)$. In [8] it was shown that ψ is valid. Here, we need $t + \lfloor log(t-1\rfloor)$ masks.

Now, we will propose our binary tree based construction which needs lesser number of masks than Sarkar's.

4.1 Improved Tree Based Construction

Define the sequence $\{l_k\}_{k\geq 0}$ as follow : $l_0 = 2$ and $l_{k+1} = 2^{l_k+k} + l_k$. So, $l_1 = 6$, $l_2 = 134$, $l_3 = 2^{136} + 134$ and so on. Note that, if $l_k < t < l_{k+1}$ then, $\lfloor log_2(t - l_k)\rfloor < l_k + k$.

Define another sequence $\{m_t\}_{t\geq 2}$ based on $\{l_k\}_{k\geq 0}$ as follow : since l_k is increasing and $l_0 = 2$ define for all $t \in [l_k, l_{k+1} - 1]$, $m_t = t + k$. So, $m_2 = 2$, $m_3 = 4,\ldots,m_6 = 7$, $m_7 = 9$, and so on. Now check the following.

1. $m_{t+1} = m_t + 2$ if $t = l_k$ for some k.
2. $m_{t+1} = m_t + 1$ if for some k, $l_k < t < l_{k+1}$.
3. Hence, m_t is increasing.

Now, we will define a **masking assignment** $\psi_{\mathbf{t}}$ which needs m_t many maskes for tree based construction of height t. At first we define two functions α_t and β_t recursively. Define, α_t, β_t : $[2,t] \to [1, m_t]$, $t \geq 2$ as follow.

1. $\alpha_2(2) = m_2 = 2$ and $\beta_2(2) = 1$.
2. After defining α_t and β_t
   (a) $\alpha_{t+1}(i) = \alpha_t(i)$ and $\beta_{t+1}(i) = \beta_t(i)$ whenever $2 \leq i \leq t$.
   (b) If $t = l_k$ for some k then $\alpha_{t+1}(t+1) = \alpha_t(t)+2 = m_t+2 = m_{t+1}$ and $\beta_{t+1}(t+1) = m_t+1$
   (c) If $l_k < t < l_{k+1}$ then $\alpha_{t+1}(t+1) = \alpha_t(t)+1 = m_t+1 = m_{t+1}$ and $\beta_{t+1}(t+1) = \nu_2(t-l_k)+1$

From the definition it is clear that

1. $\alpha_{t+1}(t+1) = m_{t+1}$ and $\beta_{t+1}(t+1) = \nu_2(t - l_k) + 1 \leq l_k + k = m_{l_k}$.
2. So, α_t and β_t are well-defined functions which map into $[1, m_t]$. In fact, $\alpha_t([2,t]) \cup \beta_t([2,t]) = [1, m_t]$. So, we need m_t many masks.

Finally, the masking assignment $\psi_t(i)$ is defined as follow :
Let $2 \leq i \leq 2^t - 1$ and $j = ht_t(i) + 1$

1. $\psi_t(i) = \alpha_t(j)$ if i is odd and

2. $\psi_t(i) = \beta_t(j)$ if i is even.

Now, we will prove that the above ψ_t is correct for all $t \geq 2$. The proof is done in 3 steps. At first step, we will show that the function ψ_t is correct for all $t \geq 2$ iff ψ_t is correct at 1 for all $t \geq 2$. Then,we will divide the proof of ψ_t is correct at 1 in two cases. $t = l_k + 1$ and $t \neq l_k + 1$. In the proof one can check the randomness of output of KR and hence it is totally correct.

Theorem 7 *The function ψ_t is correct for all $t \geq 2$ iff ψ_t is correct at 1 for all $t \geq 2$.*

Proof: Let $i \in V_t$ with $ht_t(i) = t'$. Now, if we relabel $E_t(t')$ to $E_{t'}$ such that $i \to 1$, and if $j \to k$ then $2j \to 2k$, $(2j+1) \to 2k+1$ then, ψ_t on $E_t(t')$ is same as ψ_t on $E_{t'}$. The relabelling procedure does not change the height of vertices.

So, $s(i, x, k, \mu[m_t], t) = s(1, x(i), k, \mu[m_{t'}], t')$.
Now, given x, i, t, k, u consider $T_{t'}$ and get $\mu(m_{t'})$ such that $s(1, x(i), k, \mu[m_{t'}], t') = u$. Extend $\mu[m_{t'}]$ to $\mu[m_t]$ by concatenating remaining masks randomly.

So, $s(i, x, k, \mu[m_t], t) = s(1, x(i), k, \mu[m_{t'}], t') = u$. ■

Theorem 8 *The function ψ_t is correct at 1 when $t = l_k + 1$ for some k.*

Proof: Let $t' = t - 1$. Note that, $\psi_t(2) = m_{t'} + 1$ and $\psi_t(3) = m_{t'} + 2$. So, define $\mu[m_{t'}]$ by some random string. Given, x, k, r_0, r_1 compute the following :

1. $\mu_{m_{t'}} + 1 = z(1, x(2), k, \mu[m_{t'}], t-1) \oplus r_0 = z(2, x, k, \mu[m_t], t) \oplus r_0$.
2. $\mu_{m_{t'}} + 2 = z(1, x(3), k, \mu[m_{t'}], t-1) \oplus r_1 = z(3, x, k, \mu[m_t], t) \oplus r_0$.

From these facts correctness of ψ_t at 1 follows. ■
So, $KR_{tree}(i, x, k, t, r_0, r_1)$ is μ. And note that masks are either a random string or obtained from XOR with random string.

Theorem 9 *The function ψ_t is correct at 1 when $l_k + 1 < t \leq l_{k+1}$ for some k.*

Proof. We will first define the Key-Reconstruction algorithm KR().
Input: k, x, i, r_0, r_1.
Output: $\mu = \mu_1 || \dots || \mu_l$.
Here $l = m_t$. Let $t' = t - (l_k + 1)$ where $l_k + 1 < t \leq l_{k+1}$.
Define, $y_0 = b_0 || x_{2^{t'}}, y_1 = b_1 || x_{2^{t'}-1} \dots y_{t'} = b_{t'} || x_1$ and $y = y_0 || y_1 || \dots || y_{t'}$.
b_i's are random string with suitable length such that $|y_0| = n, |y_i| = n - m,\ 1 \leq i \leq t'$. Run $KR_{seq}(t', y, k, r_0)$ to get an output $\mu_1 || \dots || \mu_{l'}$ such that $s(t', x, k, \mu[l']) = r_0$ where $l' = \lfloor log_2(t') \rfloor + 1 \leq l_k + k$.
Define $\mu_{l'+1}, \dots, \mu_{l_k+k}$ randomly. Let $b_0 = b_{01} || b_{02}$ such that $|b_{01}| = |b_{02}| = m$. Compute,

1. $\mu_{l_k+k+1} = z(2^{t'}, x, k, \mu[m_{t'}]) \oplus b_{01}$,
2. $\mu_{l_k+k+2} = z((2^{t'} + 1), x, k, \mu[m_{t'}]) \oplus b_{02}$.
3. For $l_k + k + 3 \leq j \leq t - 1$ $\mu_j = z(2^{t-j}, x, k, \mu[j-1]) \oplus b_{j-(l_k+k+2)}$.

4. Randomly define all other undefined masks.

Note the folowing observations,

1. Along the path $1, 2, 2^2, \ldots, 2^{t'}$ this rule of masking assignment is same as Shoup's rule of masking assignment. In this path only $\mu[l_k + k]$ is involved.

2. $\mu_{l_k+k+1}, \mu_{l_k+k+2}, \ldots, \mu_{m_t}$ are the masks on the edges $a_3, a_{2^2+1} \ldots, a_{2^{t'}+1}, a_{2^{t'}}$.

It is again easy to check that $s(3, x, k, \mu, t) = r_1$ and $s(2, x, k, \mu, t) = r_0$ and hence it is corect. ∎

Again, $KR_{tree}(i, x, k, t)$ is μ. And note that masks are either a random string or obtained from XOR with random string. So, the masking assignment is totally correct. Now we will briefly sketch the proof of the statement *totally correct* $\Rightarrow$ *valid.* We will prove it for binary tree based masking assignment. The same idea will carry through for any other cases e.g. sequential tree based construction.

Theorem 10 ***(Validness of domain extension)*** *In case of binary tree based domain extension a totally correct masking assignment is always valid. More precisely, we have that, if there is an* (ϵ, a) *winning strategy* $\mathcal{A}$ *for* $\{H_p\}_{p\in\mathcal{P}}$ *then there is also an* $(\frac{\epsilon}{2^t-1}, a + 2(2^t - 1))$*-strategy* $\mathcal{B}$ *for* $\{H_k\}_{k\in\mathcal{K}}$ *whenever* ψ *is totally correct.*

Proof. We describe the two stages of the strategy $\mathcal{B}$ as follows.
$\mathcal{B}^{\mathbf{guess}}$ **:** (output (y, s), with $|y| = n$.)

1. Run $\mathcal{A}^{\text{guess}}$ to obtain $x \in \{0, 1\}^N$ and state information s'.

2. Choose an $i \in_R \{1, \ldots, 2^t - 1\}$.

3. Write $x = x_1||\ldots||x_{2^t-1}$, where $|x_1| = \ldots = |x_{2^{t-1}-1}| = n - 2m$ and $|x_{2^{t-1}}| = \ldots = |x_{2^t-1}| = n$.

4. If $2^{t-1} \leq i \leq 2^t - 1$, set $y = x_i$; u_1, u_2 to be the empty string and $s = (s', i, u_1, u_2)$. Output (y, s) and stop.

5. If $1 \leq i \leq 2^{t-1} - 1$, then choose two strings u_1 and u_2 uniformly at random from the set $\{0, 1\}^m$. Set $y = u_1||u_2||x_i$ and $s = (s', i, u_1, u_2, x)$. Output (y, s) and stop.

At this point the adversary is given a k which is chosen uniformly at random from the set $\mathcal{K} = \{0, 1\}^K$. The adversary then runs $\mathcal{B}^{\text{find}}$ which is described below.
$\mathcal{B}^{\mathbf{find}}(y, k, s)$ **:** (Note $s = (s', i, u_1, u_2)$.)

1. Define the masks $\mu_1, \ldots, \mu_{m_t}$ by executing algorithm $KR_{tree}(i, x, k, t, r_0, r_1)$ (Key-Reconstruction algorithm). This defines the key $p = k||\mu$ for the function H_p.

2. Run $\mathcal{A}^{\text{find}}(x, p, s')$ to obtain x'.

3. Let y' be the inputs to processor P_i corresponding to the string x'. Output y'.

We now lower bound the probability of success. By totally correctness p is a randomly chosen key from the set $\mathcal{P}$.

Suppose x and x' collides for the function H_p. Then there must be a j in the range $1 \le j \le 2^t - 1$ such that at vertex j there is a collision for the function h_k. (Otherwise it is possible to prove by a backward induction that $x = x'$.) The probability that $j = i$ is $\frac{1}{2^t-1}$ where i is a random number lying between 2 and $2^t - 1$. Hence if the success probability of $\mathcal{A}$ is at least ϵ, then the success probability of $\mathcal{B}$ is at least $\frac{\epsilon}{2^t-1}$. Also the number of invocations of h_k by $\mathcal{B}$ is equal to the number of invocations of h_k by $\mathcal{A}$ plus at most $2(2^t - 1)$. This completes the proof. ■

Theorem 11 *The speed-up of our algorithm over the sequential algorithm in Section* **??** *is by a factor of* $\frac{2^t}{t}$.

Proof. This algorithm hashes a message of length $2^t(n-m)-(n-2m)$ into a digest of length m using t parallel rounds. The time taken by a single parallel round is proportional to the time required by a single invocation of the hash function h_k. The sequential construction require 2^t invocations of the hash function h_k on a message of length $2^t(n-m)-(n-2m)$. Hence, the speed-up of the binary tree algorithm over the sequential algorithm is by a factor of $\frac{2^t}{t}$. ■

Remark : The speed-up achieved by our algorithm is substantial even for moderate values of t. Such speed-up will prove to be advantageous for hashing long messages.

Theorem 12 *The number of masks for this algorithm is* $t + O(log^* t)$.

Proof: From the recurrence relation it is clear that $2^{2^{l_k}} > l_{k+1} > 2^{l_k}$. So, $log^*(l_k) \le log^*(l_{k+1}) \le log^*(l_k) + 2$ and hence $log^*(l_k) = \theta(k)$ i.e. $log^*(l_k)$ and k are of same order. So, for all $l_k \le t < l_{k+1}$, $m_t - t = k = O(log^* t)$.

5 Conclusion

In this paper we have considered the problem of extending the domain of a UOWHF using a binary tree algorithm. As shown in [1] this requires an expansion in the length of the key to the hash function. Our algorithm makes a key length expansion of $2m$ bits for $t = 2$; $m(t+1)$ bits for $3 \le t \le 6$ and $m(t+2)$ for $7 \le t \le 134$ using a binary tree of height t and a base hash function $h_k : \{0,1\}^n \to \{0,1\}^m$. The previous algorithm in [8] required a key length expansion of $m(t + \lfloor \log_2(t-1) \rfloor)$ with the same parameters. Hence the key length expansion in our algorithm is significantly lesser. In fact, it uses m-bits more than the lower bound proved in [8] and $2m$-bits more than the shoup's algorithm in all practical situations. But it is much faster than shoup's algorithm when parallelism is done.

6 Acknowledgement

I want to give a special thank to P. Sarkar who helped me to think in a right way.

References

[1] M. Bellare and P. Rogaway. Collision-resistant hashing: towards making UOWHFs practical. *Proceedings of CRYPTO 1997*, pp 470-484.

[2] I. B. Damgård. A design principle for hash functions. *Lecture Notes in Computer Science*, 435 (1990), 416-427 (Advances in Cryptology - CRYPTO'89).

[3] R. C. Merkle. One way hash functions and DES. *Lecture Notes in Computer Science*, 435 (1990), 428-226 (Advances in Cryptology - CRYPTO'89).

[4] I. Mironov. Hash functions: from Merkle-Damgård to Shoup. *Lecture Notes in Computer Science*, 2045 (2001), 166-181 (Advances in Cryptology - EUROCRYPT'01).

[5] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic aplications. *Proceedings of the 21st Annual Symposium on Theory of Computing*, ACM, 1989, pp. 33-43.

[6] B. Preneel. The state of cryptographic hash functions. *Lecture Notes in Computer Science*, 1561 (1999), 158-182 (Lectures on Data Security: Modern Cryptology in Theory and Practice).

[7] P. Sarkar. Domain Extender for UOWHF : A Generic Lower bound on key Expansion and a finite Binary Tree Algorithm . . *IACR preprint server, http://eprint.iacr.org.*

[8] P. Sarkar. Construction of UOWHF : Tree Hashing Revisited . *IACR preprint server, http://eprint.iacr.org/2002/058.* .

[9] V. Shoup. A composition theorem for universal one-way hash functions. *Proceedings of Eurocrypt 2000*, pp 445-452, 2000.

[10] D. Simon. Finding collisions on a one-way street: Can secure hash function be based on general assumptions?, *Lecture Notes in Computer Science - EUROCRYPT'98*, pp 334-345, 1998.

[11] D. R. Stinson. Some observations on the theory of cryptographic hash functions. IACR preprint server, http://eprint.iacr.org/2001/020/.