A Sufficient Condition and an Optimal Domain Extension of UOWHF

Mridul Nandi
Applied Statistics Unit
Indian Statistical Institute

February 2, 2004

Abstract

In this paper we will provide a non-trivial sufficient condition for UOWHF-preserving domain extension which will be very easy to verify. Using this result we can prove very easily that all known domain extension algorithms are valid. This will be a nice technique to prove a domain extension is valid. We also propose an optimal (w.r.t. both time complexity and key size) domain extension algorithm based on an incomplete binary tree. In Asiacrypt'03 [6] (also in [5]) author proposed a binary tree based domain extension of UOWHF. We will show that the binary tree based construction [5] is optimal in a subclass of full binary tree based domain extension. A full binary tree based algorithm is maximum possible parallel algorithm and very easy to implement.

Keywords : Hash function, UOWHF, Domain Extension Algorithm, strongly even-free masking assignment.

1 Introduction

Intuitively, a UOWHF (Universal One-Way Hash Function) is a family of hash functions $\{h_k\}_{k\in\mathcal{K}}$ with $h_k : \{0,1\}^n \to \{0,1\}^m$ where the following task is hard: adversary has to commit a n-bit string x and then given a random k he has to find another n-bit string x' such that $h_k(x) = h_k(x')$ and $x \neq x'$. Trivially, UOWHF is a weaker notion than CRHF or Collision Resistant Hash Function as a hash family is a UOWHF whenever it is a CRHF but the converse need not be true. A hash family is CRHF if given a random key k it is hard to find two distinct messages $x \neq x'$ such that $h_k(x) = h_k(x')$. In fact, Simon [12] proved that there exists an oracle relative to which UOWHF exists but CRHF does not exist. Bellare and Rogaway [1] constructed a secure signing algorithm using a hash family which is UOWHF. We say the above hash family $\{h_k\}_{k\in\mathcal{K}}$ is (n, m, K) hash family if $\mathcal{K} = \{0,1\}^K$. Here, we are mainly interested in valid (or UOWHF-preserving) domain extension which means that given a (n, m, K) hash family which is UOWHF we want to construct another (N, m, P) hash family which is also UOWHF where, $N > n$. Bellare and Rogaway [1] proved that a standard domain extension for CRHF, MD construction (Merkeley and Damgard [2]) is not valid domain extension for UOWHF.

In this paper we will study a general class of domain extension algorithm and provide a non trivial sufficient condition for valid domain extension. We also show that all known valid domain extensions satisfy the sufficient condition. In some case this condition is also equivalent to necessary condition. This sufficient condition will be very easy tool to prove a domain extension is valid. We also propose an optimal valid domain extension. So, this paper has both theoretical and practical interest.

1.1 Brief History

To sign a big message it is necessary to compress the message first and then run the signing algorithm on the compressed message. To have the security of signature scheme we need a collision resistant hash function. But, Bellare and Rogaway (BR) [1] constructed a generic signature scheme where a UOWHF is sufficient to prove the security of the signature scheme. UOWHF is first introduced by Naor and Young [7]. As the security notion is much weaker that that of CRHF it is believed that UOWHF can be constructed more easily than CRHF. Now, an immediate question comes that how do we extend the domain of UOWHF securely ? Bellare and Rogaway in the same paper showed that the standard domain extension algorithm i.e. MD construction for CRHF will not work. They gave binary tree based construction with the notion of XOR-ing masks. Then Shoup [11] constructed a sequential domain extension and Mironov [4] proved that it is optimum in key size over all sequential construction. Sarkar [9] gave another binary tree based construction where the number of masks or key size is less than that of BR but it is more than that of Shoup. But this algorithm (also the BR algorithm) can be implemented in parallel. In that case it will be much faster than Shoup's algorithm. Nandi [5] modified the Sarkar's algorithm with less number of keys but it still needs more masks. Lee et al [6] constructed an optimum algorithm (in key size) but parallelism is much smaller than binary tree based algorithm as they used incomplete l-ary tree. Finally in this paper we have an algorithm which has maximum possible parallelism and minimum key size.

2 Preliminaries

A (n, m) hash function (or compression function) h is an easily computable function from all n-bit strings to m-bit strings where, $n > m$ i.e. $h : \{0,1\}^n \to \{0,1\}^m$. A family of hash functions $\{h_k\}_{k\in\mathcal{K}}$ is called (n, m, K) hash family if $\mathcal{K} = \{0,1\}^K$ and for each $k \in \mathcal{K}$, h_k is a (n, m) hash function. $\mathcal{K}$ is called the *key-space* and K is the *key-size* of the hash family. The members of the *key-space* are known as *key*. To define UOWHF (Universal One-Way Hash Function) consider the following game for a (n, m, K) hash family $\{h_k\}_{k\in\mathcal{K}}$:

1. Adversary $\mathcal{A}$ chooses a n-bit string x called initial value. (In notation, $\mathcal{A}^{\text{guess}} \to (x, s)$ where s is state information which may be used in later stage).
2. $k \in_R \mathcal{K}$ (i.e. k is chosen randomly from $\mathcal{K}$) is given to the adversary $\mathcal{A}$.
3. Adversary $\mathcal{A}$ has to produce another n-bit string x'. (In notation, $\mathcal{A}^{\text{find}}\,(x, k, s) \to x'$).

An (ϵ, η, t)-*strategy* $\mathcal{A}$ for the above game is a probabilistic algorithm with runtime at most t which will find a *collision* (i.e. $h_k(x) = h_k(x')$ but $x \neq x'$) with probability at least ϵ and it invokes h_k at most η times. $\{h_k\}_{k\in\mathcal{K}}$ is an (ϵ, η, t)-UOWHF if there is no (ϵ, η, t)-strategy for $\{h_k\}_{k\in\mathcal{K}}$. We can also define UOWHF in asymptotic sense in which case we have to consider an infinite sequence of hash family (see [7] for more detail).

In this paper we are mainly interested in UOWHF preserving domain extension. Thus given a (n, m, K) hash family $\{h_k\}_{k \in \mathcal{K}}$ (*base hash family*) we would like to construct another (N, m, P) hash family $\{H_p\}_{p \in \mathcal{P}}$ (*extended hash family*) where $N > n$. The method of extending the domain of hash family is known as *domain extension algorithm.* We can define that $\mathcal{A}$ is an (ϵ, η, t)-*strategy* for $\{H_p\}_{p \in \mathcal{P}}$ if $\mathcal{A}$ find a collision with probability at least ϵ in time t and it invokes h_k at most η times. In studying strategies of $\{H_p\}_{p \in \mathcal{P}}$ we are interested in number of invocation of h_k as H_p is built using h_k. An extended hash family $\{H_p\}_{p \in \mathcal{P}}$ is (ϵ, η, t)-UOWHF if there is no (ϵ, η, t)-strategy for $\{H_p\}_{p \in \mathcal{P}}$. In this paper we always use the notations $\{h_k\}_{k \in \mathcal{K}}$ for (n, m, K) base hash family and $\{H_p\}_{p \in \mathcal{P}}$ for (N, m, P) extended hash family.

Definition 1 (UOWHF-preserving domain extension *or* **valid)**

A domain extension algorithm is called UOWHF-preserving (or valid) if for any (ϵ, η, t)-strategy for $\{H_p\}$ there is an (ϵ', η', t')-strategy for $\{h_k\}$ where η' and t' are not much larger than η and t respectively and ϵ' is not significantly lesser than ϵ. In other words $\{H_p\}_{p \in \mathcal{P}}$ is (ϵ, η, t)-UOWHF (or UOWHF in asymptotic case) whenever $\{h_k\}_{k \in \mathcal{K}}$ is (ϵ', η', t')-UOWHF (or UOWHF in asymptotic case).

Key expansion of a domain extension algorithm is $(P - K)$. We will be interested in valid domain extensions where key expansion is as small as possible. Also we will try to reduce the time complexity by considering parallel domain extension algorithms.

3 Class of Tree based Domain Extension Algorithm

It is very natural that to extend the domain of a function we have to apply the function repeatedly. The question is how we will combine this iteration i.e. how will output of one invocation of a function be fed into input of another invocation of the function. Here we will briefly describe a general class of domain extension algorithm based on a rooted directed tree introduced by Sarkar [8]. We use the notation $\mathcal{C}$ for the general class. Let us first study a rooted directed tree.

1. Let $T = (V, E, q)$ be a rooted directed tree where $V = [1, r] := \{1, \ldots, r\}$ is the set of vertices, E is the set of arcs and $q \in V$ is a special vertex called the *root* of the tree with the property that, $outdeg(q) = 0$ whereas $outdeg(i) = 1$ for other vertices i. $outdeg(i)$ is defined by $|\{j : (i, j) \in E\}|$. We also define $son(i) = \{j : (j, i) \in E\}$, $indeg(i) = |son(i)|$ and $\delta(T) = max_{i \in V}\ indeg(i)$.

2. In any rooted directed tree $T = (V, E, q)$, from any vertex i there is one and only one path from that vertex i to the root q. So, we can define $l(i)$ (called *level* of the vertex i) by the number of vertices in the unique path from i to q. Note, $l(q) = 1$. Write, $V[k] = \{i \in V : l(i) = k\}$ for each $k \geq 1$. Define $h(T)$ (called *height* of the tree T) by $max_{i \in V}\ l(i)$.

3. For each $i \neq q$, we use the notation $e_i = (i, j) \in E$. It is well-defined as $outdeg(i) = 1$ for every vertex i except the root q. Note that, $E = \{e_i : i \in V - \{q\}\}$.

4. A sub-tree T_1 of T is the tree induced by a subset of the vertex set V. Root of a sub-tree $T_1 = (V_1, E_1)$ is the vertex with minimum level. More precisely, i is called a root of the sub-tree T_1 if $i \in V_1$ and $l(i) = min_{j \in V_1} l(j)$. If $i \in V$, define $V(i)$ by the set of all vertices from which there is a path to the vertex i i.e. $V(i) = \{j \in V :$ there is a path from j to $i\}$. We will say the induced full sub-tree rooted at i by the sub-tree induced by $V(i)$ (in notation $T(i)$). Note that i becomes the root of the sub-tree $T(i)$.

Definition 2 (Masking assignment)

ψ is called a masking assignment *on a rooted tree $T = (V, E, q)$ if $\psi : V - \{q\} \to [1, l]$. Here, $[1, l]$ is thought as a set of integers. We also say that ψ is a l-masking assignment on T. We can also think ψ as a function on E where $\psi(e_i) = \psi(i)$.*

Let ψ be a l-masking assignment on $T = (V, E, q)$ where $V = [1, r]$ for some positive integers l and r. We want to define (N, m, P) hash family $\{H_p\}_{p \in \mathcal{P}}$ given a (n, m, K) hash family $\{h_k\}_{k \in \mathcal{K}}$ where, $N = (n - m)r + m$ and $P = K + m.l$. Write, $p = k||\mu_1|| \ldots ||\mu_l$ and $X = x_1|| \ldots ||x_r$ where, $|p| = P$, $|k| = K$, $|\mu_i| = m$, $|X| = N$ and $|x_i| = n - indeg(i) \times m$. We need to assume that, $n \geq \delta(T) \times m$. Note, $|X| = \sum_{i=1}^{r} |x_i| = (n - m)r + m$. Here, p is a key of extended hash family and X is any input of that hash family. We will treat k as a key of base hash family. We use the term *mask* for $\mu_i^{\prime}s$. Now we are ready to define $H_p(X)$ using the hash function h_k.

Algorithm $H_p(X)$

1. For $i \in V[t]$ $(t = h(T))$
   Compute $z_i = h_k(x_i)$.
2. For $j = t - 1$ down to 1
   Do in parallel For $i \in V[j]$
   $z_i = h_k((z_{i_1} \oplus \mu_{\psi(i_1)})|| \ldots ||(z_{i_d} \oplus \mu_{\psi(i_d)})||x_i)$ where $son(i) = \{i_1, \ldots, i_d\}$ and $i_1 < \ldots < i_d$.
3. z_q is the output of $H_p(X)$.

To understand the above algorithm better we can think the vertices of the tree as a function h_k. The masking assignment $\psi(i) = k$ means that the output of h_k at vertex i is XOR-ed by the mask μ_k and then it is fed into the input of h_k at vertex j where $e_i = (i, j) \in E$. To make input size of h_k at vertex i as n, we concatenate a part of the message X (we called it by x_i) with outputs of previous invocation of h_k.

It is clear that the above algorithm is completely determined by the rooted directed tree and the masking assignment on it. We will say the above algorithm is based on (T, ψ). The pair (T, ψ) is known as ***structure*** of the domain extension algorithm. The class $\mathcal{C}$ consists of all such above algorithm based on any pair (T, ψ) where ψ is a masking assignment on a rooted tree T.

Main parameters of the above domain extensions :

1. Key expansion is most important parameter in practical point of view. For above type of domain extension algorithm (key expansion) = (number of masks) $\times$ (size of range). So we need to have valid domain extension with smallest possible number of masks. In [8] author showed that at least $[log_2(r+1)]$ many masks are necessary to have a valid domain extension where r is the total number of invocation i.e. number of vertices in the tree.
2. The algorithm from above class can be implemented in parallel (unless the tree is a sequential tree i.e. a path). If we run the algorithm in parallel, number of rounds is same as the height of the tree. If we have $n \geq 2m$ then we can consider binary trees. So number of rounds should be at least $[log_2(r+1)]$ where r is the number of vertices of the the binary tree. A complete binary tree of height t has $2^t - 1$ many vertices and hence a domain extension algorithm based on a complete binary tree needs t rounds only. Note that, this will have maximum possible parallelism if we only assume that $n \geq 2.m$.

4 Sufficient Condition of UOWHF-preserving Domain Extension

In [8] it was proved that every valid domain extension (See Definition 1) should be based on even-free masking assignment (See Definition 3). This condition (i.e. even-free) provides a lower bound for number of masks which turns out to be $[log_2(r+1)]$ (see [8] for more detail). In this section we will give a sufficient condition for valid domain extension. We also show that all known valid domain extensions satisfy this sufficient condition.

Definition 3 ((Strongly) even-free masking assignment)
A l-masking assignment ψ on $T=(V,E)$ is called (strongly) even-free masking assignment if for any non-trivial sub-tree $T_1=(V_1,E_1)$ of T there exists $i \in [1,l]$ such that i appears (only once) odd many times in the multi-set $\psi(E_1)=\{\psi(e) : e \in E_1\}$. This i is called a single man for that sub-tree T_1.

Now, we will prove that a domain extension algorithm based on any strongly even-free masking assignment is always valid. Strongly even-free is little more stronger than the necessary condition i.e even-free. But in some case they become equivalent with respect to number of masks. More precisely, we will show in the section 6 that in some cases existence of even free l-masking assignment implies existence of strongly even-free l-masking assignment.

First we state and prove our main theorem. Idea of the proof to construct a (ϵ', η', t') strategy $\mathcal{B}$ for base hash family $\{h_k\}$ given a (ϵ, η, t) strategy $\mathcal{A}$ for extended hash family $\{H_p\}$.

Theorem 4 *If a domain extension algorithm is based on a strongly even-free masking assignment then it is a valid domain extension. More precisely, if for any (ϵ, η, t)-strategy for $\{H_p\}$ there is an (ϵ', η', t')-strategy for $\{h_k\}$ where $\eta' = \eta + 2(r-1)$, $\epsilon' = \epsilon/r$ and $t' = t + O(r)$.*

Proof. Let the structure (See Section 3) of the domain extension algorithm be (T, ψ) where ψ is strongly even-free masking assignment and height of tree is h. Let $\mathcal{A}$ be a strategy for $\{H_p\}$. Now we will define a strategy $\mathcal{B}$ for $\{h_k\}$.

$\mathcal{B}^{\mathbf{guess}}$ **:**

1. $(X, s) \leftarrow \mathcal{A}^{\text{guess}}$. $(|X| = N)$
2. Choose $i \in_R V = [1, r]$ (i is chosen randomly from $[1, r]$).
   (a) If $i \in V[h]$, set $y = x_i$, $s = (s', i, y)$. Output (y, s) and stop.
   (b) else $r_{i_1}, \ldots r_{i_d} \in_R \{0,1\}^m$ (randomly) where $son(i) = \{i_1, \ldots, i_d\}$, $y = r_{i_1}||\ldots||r_{i_d}||x_i$ and $s = (s', i, y)$ where, $i_1 < \ldots < i_d$. 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 $s = (s', i, y)$.)

1. If $i \in V[h]$ then define the masks $\mu_1, \ldots, \mu_l$ randomly.
2. Else $\mu_1||\ldots||\mu_l \leftarrow MDef(X, k, i, r_{i_1}||\ldots||r_{i_d})$ (see the algorithm $MDef$).

3. $X' \leftarrow \mathcal{A}^{\text{find}}(X, p, s')$ where $p = k||\mu_1||\ldots||\mu_l$. Let y' be the input to processor at node i while computing $H_p(X')$. Output y'.

Now we will define the mask-defining algorithm $MDef$. Here, we add two more parameters for defining the algorithm recursively.

Algorithm $Mdef(X, k, i, r_{i_1}||\ldots||r_{i_d}, T, \psi)$ (Note $|r_i| = m$ and $d = indeg(i)$)

1. We can assume that $i = q$ the root of the tree (otherwise we can do the same thing for the induced full sub-tree rooted at i, $T(i)$). Suppose, $\psi(e_u) = l$ (say) is a single man for T. Let $T' = T - (T(u) \cup \{e_u\}) = (V', E')$. If $r' = |E'|$ then, $r' < r$. Assume, $u \in son(j)$ and $son(j) = \{j_1, \ldots, j_c = u\}$ where $j_1 < \ldots < j_c$. If $j = q$ then $R = r_u = r_{i_d}$ otherwise it is a random string. Define, $x'_j = R||x_j, x'_k = \varepsilon$ (empty string) if $k \in V' - \{j\}$, otherwise $x'_k = x_k$. Now, we define $X' = x'_1||\ldots x'_r$. Run recursively $MDef(X', k, q, r_{i_1}||\ldots||r_{i_{d-1}}, T', \psi')$ if $j = q$ or $MDef(X', k, q, r_{i_1}||\ldots||r_{i_d}, T', \psi')$ if $j \neq q$ to define the masks where ψ' is ψ restricted on T'. Note $MDef$ will always define those masks which are in the range of masking assignment. When we call $MDef(., ., \psi')$ it will not define μ_l as l is not in the range of ψ'.
2. If $|E| = 1$ then $\mu = \mu_1$ where $\mu_1 = h_k(x) \oplus r_u$.
3. Define all other yet undefined masks except l by random strings. Compute the output at vertex u, call it by z. Define $\mu_l = R \oplus z$. This will completely define all masks.

Note that we assume that $j_c = u$ which need not be true. To avoid this problem we can redefine the general tree based domain extension by considering an order of the input. Previously the input at node i is $s_{i_1}||\ldots||s_{i_c}||x_i$ but we can modify it to $\sigma_i(s_{i_1}||\ldots||s_{i_c}||x_i)$ where σ_i is some permutation of n-bit strings. In that case we have to redefine the permutation accordingly when we recursively call $MDef$. For simplicity of the proof we can ignore this.

Lemma 5 *$MDef(x, k, i, r_{i_1}||\ldots||r_{i_d})$ returns a random string $\mu_1||\ldots||\mu_l$ whenever $r_{i_1}||\ldots||r_{i_d}$ is a random string. Also input of node v while computing $H_p(x)$ is $r_{i_1}||\ldots||r_{i_d}||x_v$ if $v \notin V[t]$.*

Proof. We can check easily that all the masks are random strings as either they are random strings or they are XOR of two strings one of which is a random string. So, we can prove the first part of the lemma by using induction on size of the tree. We also prove the second part of the lemma by induction. So, if input at node j contributed by node u is R then input of node i will be $r_{i_1}||\ldots||r_{i_d}||x_i$ by the induction hypothesis. The mask μ_l is defined by $\mu_l = z \oplus R$. As output of node u be z (see step-3 in the algorithm $MDef$) the input at node j contributed by node u is R . Note that μ_l is a single-man so it is not used any where else. This proves the lemma. ■

So, by above lemma input of node i while computing $H_p(X)$ will be y which is already committed in $\mathcal{B}^{\text{guess}}$. We now lower bound the winning probability. Now note that p is a randomly chosen key from the set $\mathcal{P}$ as both k and μ_i's are random strings. Suppose X and X' collides for the function H_p. Then there must be a $v \in V$ such that at vertex v 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 $i = v$ is $\frac{1}{|V|}$. Hence, if the winning probability of $\mathcal{A}$ is at least ϵ, then the winning probability of $\mathcal{B}$ is at least $\frac{\epsilon}{|V|}$. Also the number of invocation of h_k by $\mathcal{B}$ is equal to the number of invocation of h_k by $\mathcal{A}$ plus at most $2|V|$. (the number $2|V|$ is coming from the fact that in $MDef$ algorithm we need at most $|V|$ many invocation and we may need at most $|V|$ many invocation of h_k again to compute y'). This completes the proof of the theorem. ■

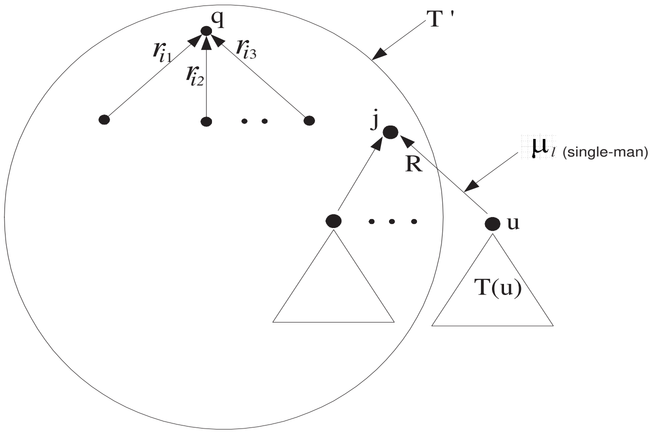

Figure 1: Recursive call of $MDef(.,T',.)$ in the algorithm $MDef(.,T,.)$

4.1 Sufficient Condition and some of known previous constructions

One can check easily that all previously known domain extension algorithm belong to the class $\mathcal{C}$ based on a rooted tree. We will prove some of previous constructions are valid using the above sufficient condition. The same technique will also work for other known secure domain extensions. So, we reduce a problem of computational reduction to verifying strongly even-free property of a function (i.e. whether a masking assignment is strongly even-free or not) which will be more easy task.

We list some known algorithms in terms of their structures.

1. **Shoup** [11] : $V = [1, r]$, $E = \{(i, i+1) : 1 \le i \le r-1\}$, $q = r$ is the root and $\psi(i) = \nu_2(i) + 1$. $\nu_2(i) = j$ means that $2^j | i$ but $2^{j+1} \not| i$.

2. **Bellare-Rogaway** [1] , **Sarkar** [9], **Nandi** [5]: The tree is full binary tree of height t and the masking assignments are given below. The complete binary tree of height t has a set of vertices $[1, 2^t - 1]$ and a set of edges $E = \{e_i = (i, [i/2]) : 2 \le i < 2^t\}$.

3. **Lee et al** [6] : In their paper a 4-dimensional construction is given which can be generalized to l-dim construction. Here we will describe 2-dimensional construction for simplicity. For integer t, $g(t) = (a, b)$, where $a = \lfloor t/2 \rfloor$, $b = \lfloor (t+1)/2 \rfloor$. $T_t = (V_t, E_t, 1)$ be a rooted binary tree, where $V_t = [1, 2^t]$ and $E_t = \{e_i : 2 \le i \le 2^t\}$ where $e_i = (i, i-1)$ for $2 \le i \le 2^a$, $e_i = (i, i - 2^a)$ for $2^a < i \le 2^{a+b}$.

   The authors defined two functions α_t, β_t as follows.

   (a) $\alpha_t : [1, 2^a - 1] \to [1, a]$ is defined by $\alpha_t(i) = 1 + \nu_2(2^a - i)$.

   (b) $\beta_t : [1, 2^b - 1] \to [a+1, a+b]$ is defined by $\beta_t(i) = a + 1 + \nu_2(2^b - i)$.

Their masking assignment $\psi_t(e_i)$ is defined as follow:

(a) $\psi_t(e_i) = \alpha_t(j)$ if $2 \le i \le 2^a$ and $j = i - 1$.

(b) $\psi_t(e_i) = \beta_t(j)$ if $2^a < i \le 2^{a+b}$ and $j2^a < i \le (j+1)2^a$.

To define the masking assignment used in [1, 9, 5] we have to define level-uniform masking assignments.

Definition 6 (Level-uniform masking assignment)
A masking assignment ψ is said to be a **level-uniform** *masking assignment on a complete binary tree $T_t = (V_t, E_t)$ of height t if there are two functions α_t and β_t : $[2,t] \to [1,l]$ such that $\psi(2i+1) = \alpha(j)$ and $\psi(2i) = \beta(j)$ where $2^{t-j} \le i < 2^{t-j+1}$ i.e. $l(i) = t - j + 1$.*

In every complete binary tree all nodes excepts leave (i.e the vertices i where $2^{t-1} \le i < 2^t$) have two sons called left or right son. So, a level-uniform masking assignments depends on the level of the vertex and type of son i.e. whether it is a left or right son of its father. The masking assignments for Sarkar [9], BR [1] and Nandi [5] are level-uniform. The following are masking assignments used by Sarkar, Bellare-Rogaway and Nandi .

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

2. **Sarkar** [9] : $\alpha_t(i) = i - 1$ and $\beta_t(i) = t + \nu(i-1)$. In [9] it was shown that ψ_t is valid. Here, we need $t + \lfloor log_2(t-1) \rfloor$ masks.

3. **Nandi** [5, 6] : Define two sequences $\{l_k\}_{k\ge0}$ and $\{m_t\}_{t\ge2}$ as follow : $l_{k+1} = 2^{l_k+k} + l_k$ where, $l_0 = 2$ and $m_2 = 2$ and if $k \ge 1$, $m_t = t + k$ for all $t \in [l_{k-1}+1, l_k]$. Note that, both l_k and m_t are strictly increasing sequences and if $t = l_k$ for some k then $m_{t+1} = m_t + 2$ and if for some k, $l_k < t < l_{k+1}$ then $m_{t+1} = m_t + 1$. It is proved in [5] that $m_t = t + O(\log_2^* t)$. Here, $\log_2^* t = j$ means that after applying log, j many times for t it becomes less than 1 for first time. The level uniform m_t-masking assignment ψ_t is based on the functions α_t, β_t : $[2,t] \to [1, m_t]$, $t \ge 2$ where they are defined as follow

   (a) $\alpha_2(2) = 2$ and $\beta_2(2) = 1$.

   (b) For $t \ge 3$, $\alpha_t(i) = \alpha_{t-1}(i)$ and $\beta_t(i) = \beta_{t-1}(i)$ whenever $2 \le i \le t-1$.

   (c) If $t \ge 3$ and $t - 1 = l_k$ for some k then $\alpha_t(t) = \alpha_{t-1}(t-1) + 2$ and $\beta_t(t) = \alpha_{t-1}(t-1) + 1$ and if $l_k < t - 1 < l_{k+1}$ then $\alpha_t(t) = \alpha_{t-1}(t-1) + 1$ and $\beta_t(t) = \nu_2(t - 1 - l_k) + 1$.

Now we can use the above theorem to prove the UOWHF-preserving property for the domain extension algorithms presented in this Section. In fact, one can check that all other valid domain extensions are based on strongly even-free masking assignments.

Theorem 7 *The domain extension algorithms [1, 11, 9, 5, 6] presented above are valid domain extensions. In fact, all these domain extension algorithms are based on strongly even-free masking assignments.*

Proof. We only prove that all these domain extensions are based on strongly even-free masking assignments. So the theorem follows from Theorem 4. For a complete binary tree $(2i+1, i)$ (or $(2i, i)$) will be known as α-edge (or β-edge).

(1) **(Shoup [11]):** Here any nontrivial sub-tree is an interval $[a, b]$ where $1 \leq a < b \leq r$. Now there exists unique $c \in [a, b-1]$ such that $\nu_2(c)$ is maximum in that interval as there can not be more than one c's with maximum $\nu_2(c)$. So $j = \nu_2(c) + 1 = \psi(c)$ is a single-man for the sub-tree $[a, b]$. So, it is a strongly even-free masking assignment. ∎

(2) **(Bellare-Rogaway [1]) :** Let $T' = (V', E')$ be any sub-tree rooted at i. Now either $\psi(2i)$ or $\psi(2i+1)$ is a single-man for T'. ∎

(3) **(Sarkar [1]) :** Let $T' = (V', E')$ be any sub-tree. Consider the α edge $(i, 2i+1)$ in E' so that i is minimum. If there is one such i then $\psi(2i+1) = j - 1 = t - l(i)$ is a single-man where $j = h(i)$. So, assume there is no α edge in E'. Then T' is a sequential tree consists of β edges. But along β edges the masking assignment is same as that of Shoup. So, in that case also we have a single man. ∎

(4) **(Nandi [5]) :** Take a sub-tree say S rooted at height t' and $l_{k+1} \geq t' \geq l_k + 1$ then from height l_{k+1} to $l_k + 1$ no α-edge can be in S (otherwise first such one i.e. the α edge having maximum height will be a single-man). Hence, T' can contain at most one β edge at height $l_k + 1$. If it contain that then $\beta_t(l_k + 1)$ is a single-man for that sub-tree. So, if S does not contain that β edge then again it is a sequential sub-tree consists of only β-edges from height t' to at least height $l_k + 1$. But on that tree masking assignment is define by ν_2 function which is itself strongly even-free masking assignment. So, the above masking assignment is strongly even-free. ∎

(5)**(Lee et al $(l-dim)$) [6] :** Let $T' = (V', E')$ be any sub-tree. Note that, $[1, 2^a] \cap V'$ is an interval say, $[c, d]$. If $d > c$ then from the definition of the masking assignment it is clear that on $[1, 2^a]$ is same as Shoup's assignment which is strongly even-free. Also note that the masks used on $[1, 2^a]$ are totally different with the masks used in other parts. So the single man on $[c, d]$ is also single man of T'. Now if $c = d$ or $[1, 2^a] \cap V' = \phi$ then T' is a sequential sub-tree of the tree induced by the vertices $\{i, i + 2^a, \ldots, 2^a(2^b - 1) + i\}$ for some $2 \leq i \leq 2^a$. Again along this tree the masking assignment is determined by β_t which is strongly even-free (which is same as Shoup's assignment). So the masking assignment is strongly even-free. ∎

5 Optimality of Binary Tree based domain Extension

In [5] author has shown that in case of sequential tree minimum number of masks for strongly even-free masking assignment is same as minimum number of masks for even-free masking assignment. We will prove this partially for complete binary tree. In fact, under some assumption we can prove it completely. Let $T_t = (V_t, E_t)$ be a full binary tree of height t and ψ be a masking assignment on it. If ψ is level-uniform (see Definition 6) then ψ is determined by two functions α and β. We will write α_i or β_i instead of $\alpha(i)$ or $\beta(i)$ respectively. Define, $M[i] = \{\alpha_i, \beta_i\}$ and $M[i, j] = \bigcup_{k=i}^{j} M[k]$. We will say a mask k is new at i if $k \in M[i]$ but k is not in $M[2, i-1]$. Define, $New(i)$ by the set of masks which are new at i. It can be empty set. For any non-trivial sub-tree S, $supp(S) = \{i : vec_\psi(S)_i$ is odd $\} \subset [1, l]$ where $vec_\psi(S)_i =$ the number of times the mask i appears in the multi-set $\psi(S)$. For each v define, $S(v) = \{A : A = supp(S), \text{root of } S \text{ is } v\}$. Define, $N(v) = |S(v)|$. If ψ is level-uniform then $S(v_1) = S(v_2)$ whenever $h(v_1) = h(v_2)$ and hence we can write $N(v)$ by N_i where $i = h(v)$. If ψ is even-free (See Definition **??**) then $supp(S)$ is non-empty for any non-trivial sub-tree. We will use the notation m_t for each t which is defined in Nandi's domain extension (Section 4.1).

We first state a property of level-uniform masking assignment. An immediate corollary of the property says that for every level-uniform even-free masking assignment there is a new mask at each i i.e. for all i, $New(i) \notin \phi$.

Lemma 8 *Suppose ψ is any level-uniform masking assignment on a complete binary tree T_t. Then, for any subset $A \subset M[2,i]$ containing $New(i)$ and for any $v \in V$ with height i, $A \in S(v)$. In particular, if ψ is even-free then at each height there is a new mask, otherwise we have a nontrivial tree whose support is empty which contradicts the assumption that ψ is even-free.*

Proof. By induction on i. Suppose the result is true for $i-1$. We will prove the case when only α_i is new at i. Let $A' = ((A - \{\alpha_i\}) \cup New(i-1))\Delta\beta_i$. By induction hypothesis (note that A' contains $New(i-1)$ as $\beta_i \notin New(i-1)$) there is a sub-tree S_1 rooted at v_1 where v_1 and v_2 are two sons of v such that $supp(S_1) = A'$. Now add the edges $(v_1, v), (v_2, v)$ with S_1 and call it by S_2. Now it is easy to check that $supp(S)\Delta A \subset New(i-1)$. Let v_3 and v_4 be two sons of v_2. So, depending on $supp(S)\Delta A$ add (v_3, v_2) and (v_4, v_2) edges with S_2 and call it by S (If $supp(S)\Delta A = \alpha_{i-1}$ then $S = S_1 \cup (v_3, v_2)$ where (v_3, v_2) is a α-edge). Note v is the root of S. The other cases are similar to this case so we leave those cases. ■

Corollary 9 *If ψ is level-uniform even-free masking assignment then for each i, $|New(i)| \geq 1$. If $m'_i = |M[2,i]|$ i.e. the number of masks used in the full binary sub-tree of height i then, $N_{i+1} = (N_i + 1) + 2^{m'_i}$ when $|New(i)| = 1$ and $N_{i+1} = 3 \times 2^{m'_i}$ when $|New(i)| = 2$.*

Proof. In the theorem choose $A = \phi$ if $New(i) = \phi$. So there will be a non-trivial tree with empty support which contradicts the even-free assumption. When $New(i) = \{k\}$ for any subset A of $[1, m'_i]$, we have a non trivial sub-tree rooted at v of height $i+1$ such that $supp(S) = A \cup \{k\}$. Also there $N_i + 1$ sets in $S(v)$ which does not contain k (consider all sub-tree rooted at v not containing the new-edge, the edge containing the new mask) ■

If ψ is even-free then, $|New(i)|$ is either 1 or 2 and more importantly, $2^{m'_i - 1} \leq N_i < 2^{m'_i}$. The first inequality can be proved by induction using the above corollary. Now we have a lemma :

Theorem 10 *If ψ is level-uniform even-free masking assignment then for any $l_k + 1 \leq i \leq l_{k+1}$, $N_i \geq 2^{m_i} - j \geq 2^{m_i - 1}$ where, $j = l_{k+1} - i + 1$. In particular, $m'_i \geq m_i$. So, Nandi's domain extension is optimal in all valid level-uniform domain extensions.*

Proof. By induction on i. Suppose the result is true for i. First assume $i = l_k$ then, $N_i \geq 2^{m_i} - 1$. Now if $m'_i \leq m_i$ then $N_i \geq 2^{m'_i} - 1$ (hence $N_i = 2^{m'_i} - 1$ and $m_i = m'_i$).Now, $|New(i+1)| = 2$ as all non-empty subsets of $[1, m_i]$ occur. So we can apply the corollary [?] for $|New(i+1)| = 2$. If $m'_i > m_i$ then also the result is true for $i+1$ (apply the corollary for both possible values of $|New(i+1)|$).

If $j > 1$ then $N_{i+1} \geq (N_i + 1) + 2^{m'_i} \geq 2^{m_i + 1} - j + 1 = 2^{m_{i+1}} + j - 1$ (since $i \neq l_k$). For smaller values of i say $i = 2$ the result is trivial. ■

Theorem 11 *The minimum number of masks for a strongly even-free masking assignment on a complete binary tree of height t is m_t. So if a valid full binary tree based domain extension is based on a strongly even-free masking assignment then number of masks is at least m_t which is the number of masks for Nandi's domain extension algorithm.*

Proof. Let $L_{i,j}(v)$ be a union of two tree S_i (sequential tree of length i) and T_j (complete binary tree of height j) joined at v where v is the root of T_j as well as one end of S_i. We will prove that if $\psi : L_{i,j}(v) \to [1,l]$ is strongly even-free masking assignment then there exists $\psi' : L_{i,j} \to [1,l]$ which is level uniform even-free (i.e. it is level uniform on T_j). In particular when $i = 1$, $L_{1,j} = T_j$. So, we can change a strongly even-free masking assignment to a level uniform even-free masking assignment on full binary tree using same number of masks and hence by previous theorem minimum number of masks for strongly even-free masking assignment is m_t.

We will prove the result by induction on $|L_{i,j}(v)|$. Suppose $e = (u_1, v_1)$ is an edge so that $\psi(e) = l$ (say) is a single-man for the subtree $L_{i,j}(v)$. If $e \in S_i$ then removing e we have $L_{i',j}(u_1)$ as one component and the other component is sequential tree. As $i' < i$ by induction hypothesis we can change ψ restricted at $L_{i',j}(u_1)$ to level uniform even-free ψ' using same set of masks. On the rest of the tree ψ' is same as ψ. If $e \in T_j$ then $L_{i+1,j-1}(u)$ is a subtree of bigger component of $L_{i,j}(v) - \{e\}$ where u is a son of v. Again, change ψ on that part to level uniform even-free ψ' (by induction hypothesis). Then define $\psi'(u', v)$ by $\psi(e)$ and ψ' on the full binary tree rooted at u' is same as ψ on rooted subtree at u. It is easy to check that, ψ' is level uniform masking assignment. ∎

Remark : So, to have a valid construction based on binary tree using lesser number of masks than that of Nandi's one has to construct a non level-uniform, non strongly even-free masking assignment which should be at least even-free and valid. It is not likely that there exists one such. The above results and empirical verifications are enough evidences to make a conjecture like below.

Conjecture 1 *Given any even free masking assignment $\psi : T \to [1,l]$ there exists another strongly even-free masking assignment using same number of masks where T is a complete binary tree.*

Remark : In the light of the above conjecture we make the comment as follow: as far as number of masks is concerned strongly even-free is same as even-free. Former one is sufficient condition for valid domain extension and later one is necessary condition for valid domain extension. In fact. one can easily construct optimal strongly even-free masking assignment recursively for any tree. That means we know an optimal tree based domain extension for any tree.

Assumption : Let $S, T \subset F_2^l$ (i.e. collections of l bit strings) such that $S \cap T = \{0\}$ (the zero vector). If $|S|, |T| \geq 2^k + r$ then $|S + T| \geq 2^{k+1} + 2^k + r$ where, $2^k > r \geq 1$ and $S + T = \{s \oplus t : s \in S, t \in T\}$. In particular if $|S| = |T| = 2^n - i$ then $|S + T| \geq 2^{n+1} - (i - 1)$.

We verify the assumptions for small values of l and k by computer and we did not find any counter-example. If the dimension of $S \cup T$ is $k + 2$ then the lower bound can achieve. Otherwise the size of $S + T$ is more then the proposed bound.

Now we will give a sketch of the proof of the above conjecture under this assumption. We have already defined $S(v), N(v)$ at vertex v. Define, $N(i) = min\{N(v) : h(v) = i\}$. When, ψ is level-uniform $N(i) = N(v)$ for any v with height i. We will prove by induction on k that for $i = l_k$, $N(i) \geq 2^{l_k + k} - 1$. If not, then say for $i = l_{k+1}$, $N(i) < 2^{l_{k+1}+k+1} - 1$. Then by above assumption $N(i - 1) < 2^{l_{k+1}+k} - 2$ and so on. So, we have $N(l_k) < 2^{l_k + k} - 1$ which is not true by induction hypothesis. One can write the prove in detail. Basic idea is that we are using the above assumption for $S(v) + \{u_i\}$ where $i = \psi(e_v)$.

Table 1: Specific comparison of domain extenders for UOWHF

Parameter	Shoup [11]	l-DIM($l \geq 2$)	Nandi	**Opt**
seq/par	sequential	parallel	parallel	parallel
message length	$2^t n$ $-(2^t-1)m$	$2^t n$ $-(2^t-1)m$	$(2^t-1)n$ $-(2^t-2)m$	$2^t n$ $-(2^t-1)m$
# invocation of h_k	2^t	2^t	2^t-1	2^t
# masks	t	t	t+ $O(\log_2^* t)$	t
# rounds	2^t	$l2^{t/l} - l + 1 (t \equiv 0 \bmod l)$	t	$t+1$
speed-up	1	$\frac{2^t}{l2^{t/l}-l+1}(t \equiv 0 \bmod l)$	$\frac{2^t}{t+1}$	$\frac{2^t}{t+1}$

6 Optimal Parallel Domain Extension

In this section we will construct valid and optimum with respect to both parallelism and key expansion (See Table 1) domain extension algorithm. Here, we will consider a rooted binary tree (not complete) instead of directed rooted binary tree. It is easy to correspond a directed binary tree to a binary tree and vice-versa. Let $T = (V, E, v_0)$ be a rooted binary tree i.e. $v_0 \in V$ and $deg(v) \leq 3$ for all $v \in V$ and $deg(v_0) \leq 2$. v_0 is called the root of the binary tree.

Definition 12 **(i-binary tree)**

$T = (V, E, v_1)$ is called i-binary tree if there exists a binary sub-tree $T_1 = (V_1, E_1, v_i)$ of T such that $E = E_1 \cup \{v_1v_2, \ldots v_{i-1}v_i\}$ and v_k are not in V_1 for $1 \leq k \leq i-1$. The path $v_1v_2 \ldots v_i$ is called the i-path of the i-binary tree T.

See examples of i-binary tree in fig[?]. A i-binary tree of size i is a sequential tree i.e. a path of length $i-1$. Given two disjoint binary tree (vertex sets are disjoint) $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ we can concatenate as follow: $T = T_1 +_{uv} T_2$ (notation) where, $T = (V, E)$, $V = V_1 \cup V_2$, $u \in V_1$, $v \in V_2$ and $E = E_1 \cup E_2 \cup \{uv\}$.

We know that if ψ is even-free m-masking assignment (so for strongly even-free) then, $2^m \geq |V|$. A m-masking assignment ψ on $T = (V, E)$ is called optimal masking assignment if it is strongly even-free and $2^m \geq |V| > 2^{m-1}$. So, an optimal masking assignment is a strongly even-free masking assignment whose size of image is minimum possible. One such example is given by Shoup[?] as follow : let T be a n-binary tree of size n i.e. it is a path of length $n-1$. More precisely, $V = \{v_1, v_2, \ldots v_n\}$ and $E = \{e_i = v_iv_{i+1} 1 \leq i \leq n-1\}$. Define ψ as follow: $\psi(e_i) = \nu_2(i) + 1$ where, $\nu_2(i) = j$ means that $2^j | i$ but $2^{j+1} \not| i$. It can be checked that the above masking assignment is optimal. It is called sequential n-optimal masking assignment.

Like concatenation of two trees we can define concatenation of two masking assignment as follow. Suppose, ψ_i is a k-masking assignment on T_i for $i = 1, 2$ then, we can define ψ a $(k+1)$-masking assignment on $T_1 +_{uv} T_2$ where, ψ on T_i is same as ψ_i on T_i and $\psi(uv) = k+1$. We will denote ψ as $\psi_1 +_{uv} \psi_2$. Note that, if both ψ_1 and ψ_2 are strongly even-free then so is ψ. In fact, if both ψ_1 and ψ_2 are optimal then so is ψ.

Definition 13 **((m, l, i)-optimal masking assignment)**

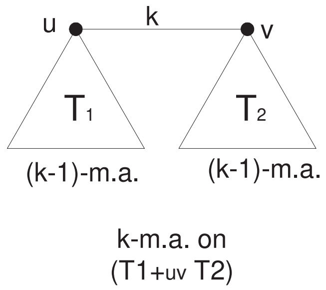

Figure 2: Concatenation of two masking assignments

A m-masking assignment is called (m, l, i)-optimal masking assignment if it is optimal masking assignment on a i-binary tree T such that $l(T) = l$ and $|V| = 2^m$.

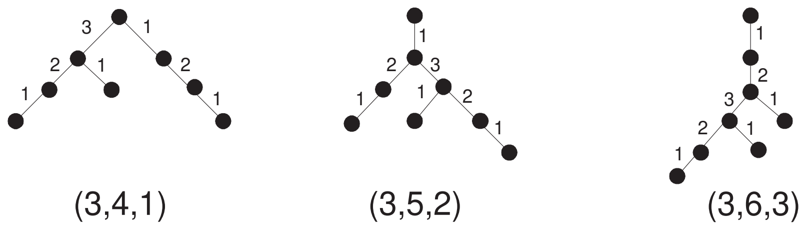

Figure 3: some optimal masking assignments

So, even-free masking assignment is necessary condition for valid domain extension and strongly even-free masking assignment is sufficient condition for valid domain extension.

Theorem 14 *There exists an $(n, n+i, i)$-optimal masking assignment if $i \leq 2^n$.*

Proof. Let $f(k) = 2^k + k + 1$ for $k \geq 0$. It is strictly increasing function, So, given positive integers n and i there exists a unique $k \geq 1$ such that $f(k) > (n+i) \geq f(k-1)$. We will prove the theorem by induction on $n+i$. For small values of $n+i$ we have shown some examples in fig[?]. Now given n and i we assume that the theorem is true for any i_1 and n_1 such that $i_1 \leq 2^{n_1}$ and $i_1 + n_1 < i + n$. Choose k as above for these n and i. First assume that, $f(k) > (n+i) > f(k-1)$. Let $j = (n+i) - f(k-1) \geq 1$. By induction hypothesis there is a $(k-1, k-1+j, j)$-optimal masking assignment ($2^{k-1} \geq j$ as $f(k) > n+i$). Call this by ψ_{k-1}. Now, ψ_{k-1} is a masking assignment on a j-binary tree $T_{k-1} = (V_{k-1}, E_{k-1}, v_1)$ where, $\{v_1, \ldots, v_j\}$ is the i-path. Now take the sequential $(k-1)$-optimal masking assignment ψ on $T = (V, E)$ and define $\psi_k = \psi_{k-1} +_{v_j v_{j+1}} \psi$ where, $V = \{v_{j+1}, \ldots v_{j+2^{k-1}}\}$. Now we can add optimal masking assignment one by one with ψ_k at v_i's. More precisely, let ψ'_l be a $(l-1, 1, l)$-optimal masking assignment on $T_l = (V_l, E_l, u_l)$ for $k+1 \leq l \leq n$. Define $\psi_l = \psi_{l-1} +_{v_{l-k+1} u_l} \psi'_l$ recursively for $k+1 \leq l \leq n$. Now it can be checked easily that ψ_n is $(n, n+i, i)$-optimal masking assignment.

We leave with other possible case where $n + i = f(k)$ for some k. In this case construct a k-sequential optimal masking assignment ψ on a sequential tree $T_k = (V_k, E_k)$ where, $V_k = \{v_1, \ldots, v_{2^k}\}$ For each l, $k \leq l \leq n-1$ we have $(l, l+1, 1)$-optimal masking assignment psi_l. We can concatenate ψ_l with ψ one by one. Finally we will have $(n, n+i, i)$-optimal masking assignment. ■

One immediate corollary is given below which tells that we have a domain extension algorithm which needs t many masks and $t+1$ many rounds for 2^t many invocation of base hash function. Note that both number of rounds and number of keys are minimum possible.

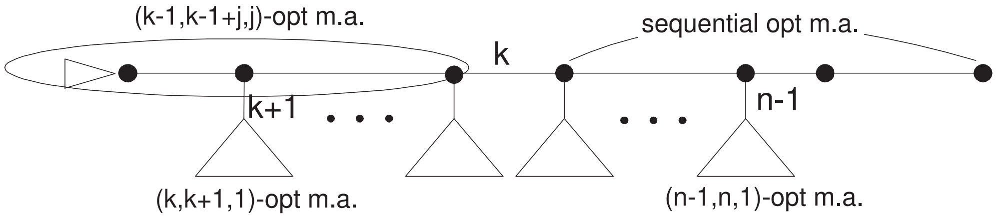

Figure 4: Construction of $(n, n+i, i)$-optimal masking assignment

Corollary 15 *There exists an $(m, m+1, 1)$-optimal masking assignment i.e. there exists a binary tree T of size 2^m with $l(T) = m+1$ which is minimum possible (a complete binary tree of level m has size $2^m - 1$) so that an optimal masking assignment ψ on T exists.*

Parallelizability:	**Opt**	=	Nandi	=	Sarkar	<	l-DIM	<	Shoup
Key length expansion:	**Opt**	=	***l*-DIM**	=	Shoup	<	Nandi	<	Sarkar

The results may be summarized as shown in Table 2 (Here, 'seq' means 'sequential' and 'par' means 'parallel').

Table 2: Comparison of domain extenders for UOWHF

Method	Used Tree	Ranking of Key expansion	Ranking of Parallelism	Seq /Par
BLH [1]	Unary	7	3	seq
XLH [1]	Unary	6	3	seq
Shoup [11]	Unary	1	3	seq
BTH [1]	Complete l-ary ($l \geq 2$)	5	1	par
XTH [1]	Complete l-ary ($l \geq 2$)	4	1	par
Sarkar [9]	Complete Binary	3	1	par
Nandi [5]	Complete Binary	2	1	par
l-DIM [6]	Incomplete l-ary ($l \geq 2$)	1	2	par
Opt(this paper)	Incomplete Binary	1	1	par

7 Conclusion

This paper has both theoretical and practical interest. Here, we will construct a UOWHF-preserving domain extension algorithm which is optimum in both key size and number of rounds. We also show that the construction given in [5] is optimum in a wide sub class of complete binary tree based algorithm. In this paper we also study how to check UOWHF-preserving property of a domain extension algorithm by just verifying a simple property called strongly even-free. It is very interesting to note that all known UOWHF-preserving domain extension algorithms satisfy the sufficient condition. So one can try to prove that the condition is a necessary condition. The sufficient condition make easy to construct a UOWHF-preserving domain extension algorithm.

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. Nandi. A New Tree based Domain Extension of UOWHF. *IACR e-print server http://eprint.iacr.org/2003/142.*

[6] M. Nandi, W. Lee, D. Chang, S. Lee and S. Sung. New Parallel Domain Extenders for UOWHF. *Lecture Notes in Computer Science* to appear in Advances in Cryptology - ASIACRYPT'03).

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

[8] 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.*

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

[10] P. Sarkar. A Practical Parallel Domain Extender for UOWHF . *IACR preprint server, http://eprint.iacr.org/2003/?.* .

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

[12] 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.