# Masking Based Domain Extenders for UOWHFs: Bounds and Constructions

Palash Sarkar

Cryptology Research Group
Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road
Kolkata 700108, India
palash@isical.ac.in

**Abstract.** We study the class of masking based domain extenders for UOWHFs. Our first contribution is to show that any correct masking based domain extender for UOWHF which invokes the compression UOWHF $s$ times must use at least $\lceil \log_2 s \rceil$ masks. As a consequence, we obtain the key expansion optimality of several known algorithms among the class of *all* masking based domain extending algorithms. Our second contribution is to present a new parallel domain extender for UOWHF. The new algorithm achieves asymptotically optimal speed-up over the sequential algorithm and the key expansion is almost everywhere optimal, i.e., it is optimal for almost all possible number of invocations of the compression UOWHF. Our algorithm compares favourably with all previously known masking based domain extending algorithms.
**Keywords :** UOWHF, domain extender, parallel algorithm.

## 1 Introduction

A universal one-way hash function (UOWHF) is a function family $\{h_k\}_{k \in \mathcal{K}}$ with $h_k : \{0,1\}^n \rightarrow \{0,1\}^m$, for which the following task of the adversary is computationally infeasible: the adversary chooses an $n$-bit string $x$, is then given a $k$ chosen uniformly at random from $\mathcal{K}$ and has to find a $x'$ such that $x \neq x'$ and $h_k(x) = h_k(x')$. The notion of UOWHF was introduced in [9].

Intuitively, a UOWHF is a weaker primitive than a collision resistant hash function (CRHF), since the adversary has to commit to the string $x$ before knowing the actual hash function $h_k$ for which a collision has to be found. In fact, Simon [15] has shown that there is an oracle relative to which UOWHFs exist but CRHFs do not exist. Further, as pointed out in [1], the birthday paradox does not apply to the UOWHF and hence the message digest can be smaller. Thus a construction for UOWHF may be faster than a construction for CRHF.

There is a second and perhaps a more important reason to prefer UOWHF over CRHF. A protocol built using a UOWHF maybe "more" secure than a protocol built using CRHF. The intuitive reason being that even if it is possible to find a collision for a hash function, it might still be difficult to find a collision for it when considered as a UOWHF. This situation is nicely summed up in [1]: "Ask less of a hash function and it is less likely to disappoint!"

The important paper by Bellare and Rogaway [1] provides the foundation for the recent studies on UOWHFs. They introduce the notion of domain extender for UOWHF; show that the classical Merkle-Damgård algorithm does not work for UOWHFs; provide several new constructions for UOWHF domain extenders and finally provide a secure digital signature scheme based on UOWHF in the hash-then-sign paradigm.

The study in [1] shows that extending the domain usually requires an associated increase in key length. One of the major new ideas behind their domain extending algorithm is "masking" the outputs of intermediate invocations by random strings. This idea of masking based algorithms have been later pursued by several authors [14, 3, 13, 12, 8, 7]. We would like to point out that [1] also presents other (i.e., non-masking type) techniques for domain extension. However, the key expansion for these techniques is more than the masking type techniques. Consequently, subsequent work, including the current one, have concentrated only on masking type domain extenders.

Our Contributions: The contribution of this paper is twofold.

We start by formalizing the class $\boldsymbol{\mathcal{A}}$ of *all* masking based domain extending algorithms. This class includes all known efficient domain extending algorithms [14, 3, 13, 12, 8, 7]. Any masking based algorithm in $\boldsymbol{\mathcal{A}}$ proceeds by XORing the output of any intermediate invocation of the compression UOWHF by a fixed length bit string called a mask.

Suppose $\{h_k\}_{k\in\mathcal{K}}$, $h_k : \{0,1\}^n \rightarrow \{0,1\}^m$ is a compression UOWHF whose domain is to be extended. Further, suppose that an algorithm in $\boldsymbol{\mathcal{A}}$ makes $s$ invocations of $h_k$ (for some $k \in \mathcal{K}$) and uses a total of $\rho$ masks. In Proposition 1, we show that the length of any string in the extended domain is $n+(s-1)(n-m)$. The resulting amount of key expansion is $\rho m$ and hence the key expansion is totally determined by the number of masks.

Our main result on class $\boldsymbol{\mathcal{A}}$ is to obtain a necessary condition for any algorithm in $\boldsymbol{\mathcal{A}}$ to be a correct domain extending algorithm. Using this necessary condition, we obtain a non-trivial lower bound on the number of masks used by any correct algorithm in $\boldsymbol{\mathcal{A}}$. More precisely, in Theorem 1, we show that $\rho \geq \lceil \log_2 s \rceil$. Based on this lower bound, we define the masking efficiency, $\mathsf{ME}$ of an algorithm which uses $\rho$ masks and makes $s$ invocations of the compression UOWHF to be $\mathsf{ME} = \rho - \lceil \log_2 s \rceil$. In the case $\mathsf{ME} = 0$, we say that the algorithm achieves optimal masking. Our lower bound immediately shows the masking optimality of the sequential algorithm of Shoup [14] and the parallel algorithms of [3, 7].

The basic unit of operation of a domain extending algorithm is one invocation of the compression UOWHF. The number of operations made by any sequential algorithm is equal to the number of invocations of the compression UOWHF. On the other hand, in a parallel algorithm, several invocations of the compression UOWHF is done in parallel and thus the number of parallel rounds will be lower. Suppose an algorithm makes $s$ invocations of the compression UOWHF and uses $N_p$ processors to complete the computation in $N_r$ rounds. Since there are $s$ invocations and $N_p$ processors, at least $\lceil s/N_p \rceil$ parallel rounds will be required and hence $N_r \geq \lceil s/N_p \rceil$. We define the parellelism efficiency, $\mathsf{PE}$ to be equal to $s/N_r$. In general, $\mathsf{PE} \leq N_p$ and in the case $\mathsf{PE} = N_p$ we say that the algorithm achieves optimal parallelism.

Our second contribution is to obtain a parallel domain extending algorithm. The basic idea of the algorithm is to divide the input message into several parts, hash each part separately and then combine the different parts using a binary tree. This idea has already been suggested for collision resistant hash functions by Damgård in [2]. Our contribution is to add a suitable masking strategy. The result is a simple and efficient parallel domain extending algorithm for UOWHF. The masking efficiency $\mathsf{ME}$ is almost always zero and in very few cases it is one. Hence we say that the masking efficiency of our algorithm is almost always optimal. Further, the parallelism efficiency $\mathsf{PE}$ is asymptotically optimal. Thus our algorithm provides a satisfactory parallel domain extender for UOWHF and to a certain extent completes the line of research on obtaining efficient domain extenders for UOWHFs which was started in [1].

Related Work: We have already mentioned that UOWHF was introduced by Naor and Yung [9] and the important work done by Bellare and Rogaway [1]. There are several direct constructions of UOWHFs based on general assumptions [10, 4]. However, as noted in [1] these are not very efficient. Subsequent to the work in [1], Shoup [14] described a nice domain extending algorithm which is a modification of the Merkle-Damgård construction. Shoup's algorithm is a sequential algorithm and Mironov [6] proved that the algorithm achieves minimal key length expansion among all *sequential* masking based domain extending algorithms. (As opposed to this, our lower bound shows that Shoup's algorithm is optimal among *all* masking based domain extending algorithms.) Later work [13, 8, 3, 12, 7] have provided different parallel constructions of domain extending algorithms with varying trade-off between degree of parallelism and key length expansion. These are summarized in Tables 1 and 2.

We note that none of the previous constructions simultaneously achieve optimal parallelism and optimal key expansion. (In [7], it is claimed that their algorithm achieves optimal parallelism. This claim is not correct: In [7], $s = 2^T$ and the number of parallel rounds is $N_r = T + 1$. This requires $N_p = 2^{T-1}$ and hence $\mathsf{PE} \approx N_p/\log_2 N_p$; as explained above, for optimal parallelism we should have $\mathsf{PE} = N_p$.)

Note that the algorithms in [1, 13, 8, 7, 3] can also be executed with a fixed number of processors by a level-by-level simulation of the large binary tree. However, this simulation will require storing the results of all the invocations at one level and hence will push up the memory requirement. In contrast, for our algorithm, the required number of buffers is exactly equal to the number of processors.

In [7], a sufficient condition for the correctness of any algorithm in $\boldsymbol{\mathcal{A}}$ is presented. Essentially, this condition states that, if, for any subtree, there is at least one mask which occurs *exactly once* in that subtree, then the construction

**Table 1.** Comparison of masking efficiency. Here $s$ is the number of invocations of the compression UOWHF.

| construction | [1] | [13] | [8] | [12] | [14, 3, 7] | ours |
|---|---|---|---|---|---|---|
| ME | $\approx \log_2 s$ | $\approx \log_2 \log_2 s$ | $O(\log^* s)$ | const | 0 | 0 or $1^\dagger$ |

†: the value is almost always 0.

**Table 2.** Comparison of parallelism efficiency. Here $N_p$ is the number of processors.

| construction | [1, 13, 8, 7] | [3] | [12], ours |
|---|---|---|---|
| PE | $\approx \frac{N_p}{\log N_p}$ | $\approx N_p^{1-1/l}$, $l$ const. | $\approx N_p$ |

is correct. In contrast, our necessary condition states that for any correct construction, for any subtree, there must be at least one mask which occurs an *odd number of times.* Though these two combinatorial conditions are close, they are not the same and they have not yet been proved to be equivalent. In fact, it is also possible that they *cannot* be proved to be equivalent.

Our necessary condition yields a tight lower bound on the number of masks, whereas the sufficient condition in [7] is used to verify the correctness of some previous constructions. However, it is not easy to apply the sufficient condition of [7] to prove the correctness of the construction in [12] and the construction presented here. On the other hand, for small examples, it is possible to verify that both the construction in [12] and the one presented here satisfy the sufficient condition of [7]. Thus our necessary condition and the sufficient condition of [7] are actually different and are of separate interest. It could be an interesting research problem to obtain a single necessary and sufficient condition for correct domain extension for any algorithm in $\boldsymbol{\mathcal{A}}$.

The rest of the paper is organized as follows. In Section 2, we describe the necessary preliminaries. In Section 3, we describe the formal model for masking based domain extenders and study its properties. In this section, we also obtain the necessary condition and the lower bound on the number of masks. The new construction of a parallel domain extending algorithm is described in Section 4. Finally, Section 5 concludes the paper. Due to lack of space, most of the proofs are omitted. These can be found in the full version of the paper and in the technical report [11].

## 2 Preliminaries

All logarithms in this paper are to the base 2. The notation $x \in_r A$ denotes the (uniformly at) random choice of the element $x$ from the set $A$. Also $\boldsymbol{\lambda}$ denotes the empty string. By an $(n, m)$ function $f$ we will mean a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$. A formal definition for UOWHF is given in [9]. In this paper we will be interested in "securely" extending the domain of a given UOWHF.

Our proof technique will essentially be a reduction. We formalize this as a reduction between two suitably defined problems.

Let $\mathbf{F} = \{h_k\}_{k \in \mathcal{K}}$ be a keyed family of hash functions, where each $h_k$ is an $(n, m)$ function, with $n > m$. Consider the following adversarial game $\mathcal{G}(\mathbf{F})$ for the family $\mathbf{F}$.

1. Adversary chooses an $x \in \{0,1\}^n$.
2. Adversary is given a $k$ which is chosen uniformly at random from $\mathcal{K}$.
3. Adversary has to find $x'$ such that $x \neq x'$ and $h_k(x) = h_k(x')$.

The problem $\mathcal{G}(\mathbf{F})$-$\mathsf{UOWHF}$ for the family $\mathbf{F}$ is to win the game $\mathcal{G}(\mathbf{F})$.

A strategy $\mathcal{A}$ for the 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 $\mathsf{state}$. In the second stage $\mathcal{A}^{\text{find}}(x, k, \mathsf{state})$, the adversary either finds a $x'$ which provides a collision for $h_k$ or it reports failure. Both $\mathcal{A}^{\text{guess}}$ and $\mathcal{A}^{\text{find}}(x, k, \mathsf{state})$ are probabilistic algorithms. The success probability of the strategy is measured over the random choices made by $\mathcal{A}^{\text{guess}}$ and $\mathcal{A}^{\text{find}}(x, k, \mathsf{state})$ and the random choice of $k$ in Step 2 of the game. We say that $\mathcal{A}$ is an $(\epsilon, q)$-strategy for $\mathcal{G}(\mathbf{F})$-$\mathsf{UOWHF}$ if the success probability of $\mathcal{A}$ is at least $\epsilon$ and it invokes some hash function from the family $\mathbf{F}$ at most $q$ times. Informally, we say that $\mathbf{F}$ is a UOWHF if there is no "good" winning strategy for the game $\mathcal{G}(\mathbf{F})$.

In this paper, we are interested in extending the domain of a UOWHF. Let $\mathbf{F} = \{h_k\}_{k \in \mathcal{K}}$, where each $h_k$ is an $(n, m)$ function. For $i \geq 1$, let $n_i = n + (i-1)(n-m)$. Define $\mathbf{F}_0 = \mathbf{F}$ and for $i > 0$, define $\mathbf{F}_i = \{H_{p_i}\}_{p_i \in \mathcal{P}_i}$, where each $H_{p_i}$ is an $(n_i, m)$ function. The family $\mathbf{F}_i$ is built from the family $\mathbf{F}$. In fact, as shown in Proposition 1, a function in $\mathbf{F}_i$ is built using exactly $i$ invocations of some function in $\mathbf{F}$.

We consider the problem $\mathcal{G}(\mathbf{F}_i)$-UOWHF. We say that the adversary has an $(\epsilon, q)$-strategy for $\mathcal{G}(\mathbf{F}_i)$-$\mathsf{UOWHF}$ if there is a strategy $\mathcal{B}$ for the adversary with probability of success at least $\epsilon$ and which invokes some hash function from the family $\mathbf{F}$ at most $q$ times. Note that $\mathbf{F}_i$ is built using $\mathbf{F}$ and hence while studying strategies for $\mathcal{G}(\mathbf{F}_i)$ we are interested in the number of invocations of hash functions from the family $\mathbf{F}$.

The correctness of our construction will essentially be a Turing reduction. We will show that if there is an $(\epsilon, q)$-strategy for $\mathcal{G}(\mathbf{F}_i)$, then there is an $(\epsilon_1, q_1)$-strategy for $\mathbf{F}_i$, where $\epsilon_1$ is not "significantly" less than $\epsilon$ and $q_1$ is not "significantly" more than $q$. In fact, we will have $\epsilon_1 = \epsilon/i$ and $q_1 = q + 2i$. Since $\mathbf{F}_i$ invokes a hash function from $\mathbf{F}$ a total of $i$ times, we "tolerate" a reduction in success probability by a factor of $1/i$. (This is also true for other constructions such as in [1].) The intuitive interpretation of the reduction is that if $\mathbf{F}$ is a UOWHF then so is $\mathbf{F}_i$ for each $i \geq 1$.

The key length for the base hash family $\mathbf{F}$ is $\lceil \log |\mathcal{K}| \rceil$. On the other hand, the key length for the family $\mathbf{F}_i$ is $\lceil \log |\mathcal{P}_i| \rceil$. Thus increasing the size of the input from $n$ bits to $n_i$ bits results in an increase of the key size by an amount $\lceil \log |\mathcal{P}_i| \rceil - \lceil \log |\mathcal{K}| \rceil$.

## 3 Lower Bound on Key Expansion

In this section, we consider the problem of minimising the key expansion while securely extending the domain of a UOWHF. More precisely, we are interested in obtaining a lower bound on key length expansion. Obtaining a complete answer to this problem is in general difficult. Thus we adopt a simpler approach to the problem. We fix a class of possible domain extending algorithms and obtain a lower bound on key expansion for any algorithm in this class. (Note that in computer science, this is the usual approach for proving lower bounds on algorithmic problems. For example, the lower bound of $O(n \log n)$ for sorting $n$ elements is obtained for the class of all *comparison based algorithms.*)

The usefulness of our lower bound depends on the class of algorithms that we consider. The class that we consider consists of all masking based domain extending algorithms. (We make this more precise later.) All previously known masking based algorithms [1, 14, 13, 12, 8, 3, 7] belong to this class. We consider this to be ample evidence for the usefulness of our lower bound. However, we would like to point out that our lower bound does not hold for *any* domain extending algorithm. Thus it might be possible to achieve lower key expansions. However, any such algorithm must adopt an approach which is different from masking based algorithms. One possible approach could be to develop the technique of using separate keys for the compression functions (see [1]).

Let $\mathbf{F} = \{h_k\}_{k \in \mathcal{K}}$, where each $h_k$ is an $(n, m)$ function. We are interested in the class $\boldsymbol{\mathcal{A}}$ of masking based domain extension algorithms. We do not want the algorithm to be dependent on the structure of the UOWHF; in fact it should work for all UOWHF's which can "fit" into the structure of the algorithm. Any algorithm $A \in \boldsymbol{\mathcal{A}}$ behaves in the following manner.

1. It invokes some function $h_k \in \mathbf{F}$ a finite number of times.
2. The outputs of all but one invocation of $h_k()$ is masked by XORing with an $m$-bit string selected from the set $\{\mu_0, \ldots, \mu_{\rho-1}\}$.
3. The invocations of $h_k()$ whose outputs are masked are called *intermediate* invocations and the invocation whose output is not masked is called the *final* invocation.
4. The entire output of any intermediate invocation is fed into the input of exactly one separate invocation of $h_k()$.
5. Each bit of the message $x$ is fed into exactly one invocation of $h_k()$.
6. The output of the final invocation is the output of $A$.

We emphasize that all previously known masking based algorithms [1, 14, 13, 12, 8, 3, 7] belong to $\boldsymbol{\mathcal{A}}$. In the following we make a general study of any algorithm in $\boldsymbol{\mathcal{A}}$, with particular emphasis on obtaining a lower bound on key expansion made by any algorithm in $\boldsymbol{\mathcal{A}}$.

**Proposition 1.** *Let $A \in \boldsymbol{\mathcal{A}}$ be such that $A$ invokes $h_k()$ a total of $s$ times. Then the length of the message which is hashed is equal to $n + (s-1)(n-m)$.*

Thus the number of invocations of $h_k()$ and the parameters $n$ and $m$ determine the length of the message to be hashed irrespective of the actual structure of the algorithm. Hence any algorithm $A \in \boldsymbol{\mathcal{A}}$ which invokes $h_k()$ a total of $s$ times defines a family $\mathbf{F}^{(A,s)} = \{H_p^{(A,s)}\}_{p \in \mathcal{P}}$, where $\mathcal{P} = \{0,1\}^{|k|+m\rho}$ and each $H_p^{(A,s)}$ is an $(n + (n-m)(s-1), m)$ function. The structure of any algorithm $A \in \boldsymbol{\mathcal{A}}$ which makes $s$ invocations of $h_k()$ is described by a labelled directed graph $D_s^A = (V_s, E_s, \psi_s)$, where

1. $V_s = \{v_1, \ldots, v_s\}$, i.e., there is a node for each invocation of $h_k()$.
2. $(v_i, v_j) \in E_s$ if and only if the output of the $i$th invocation is fed into the input of the $j$th invocation.
3. $\psi_s$ is a map $\psi : E_s \to \{\mu_0, \ldots, \mu_{\rho-1}\}$, where $\psi(v_{i_1}, v_{i_2}) = \mu_j$ if the output of the $i_1$-th invocation of $h_k()$ is masked using $\mu_j$.

The nodes corresponding to the intermediate invocations are called intermediate nodes and the node corresponding to the final node is called the final node. Without loss of generality we assume the final node to be $v_s$. Nodes with indegree zero are called leaf nodes and the others are called internal nodes. Define $\delta(D_s^A) = \max\{\mathsf{indeg}(v) : v \in V_s\}$. We call $\delta(D_s^A)$ to be the fan-in of algorithm $A$ for $s$ invocations.

**Proposition 2.** *The outdegree of any intermediate node in $D_s^A$ is 1 and the outdegree of the final node is 0. Hence there are exactly $(s-1)$ arcs in $D_s^A$. Consequently, $D_s^A$ is a rooted directed tree where the final node is the root of $D_s^A$.*

**Proposition 3.** *If $\delta = \delta(D_s^A)$, then $n \geq \delta m$.*

Thus an algorithm $A$ with fan-in $\delta$ cannot be used with all UOWHFs. The value of fan-in places a restriction on the values of $n$ and $m$. However, given this restriction the actual structure of $D_s^A$ does not depend on the particular family $\mathbf{F}$.

Let $T$ be a non-trivial subtree of $D_s^A$. Denote by $\mathsf{vec}_\psi(T)$ the $\rho$-tuple

$$(\mathsf{num}_{\mu_0}(T) \bmod 2, \ldots, \mathsf{num}_{\mu_{\rho-1}}(T) \bmod 2),$$

where $\mathsf{num}_{\mu_i}(T)$ is the number of times the mask $\mu_i$ occurs in the tree $T$. We say that $D_s^A$ is *null-free* if $\mathsf{vec}\psi(T) \neq (0, \ldots, 0)$ for each non-trivial subtree of $D_s^A$.

We now turn to the task of obtaining a lower bound on key expansion made by any algorithm $A$ in $\boldsymbol{\mathcal{A}}$. This consists of two tasks. Firstly, we show that for any "correct UOWHF preserving domain extender" $A$ which invokes some function from the compression UOWHF exactly $s$ times, the DAG $D_s^A$ must be null-free. This translates the problem into a combinatorial one. Our second task is to use use this combinatorial property to obtain the required lower bound.

The intuitive idea behind the first part is as follows. Given $D_s^A$ and a family $\mathbf{F}'$ with suitable parameters, we construct a family $\mathbf{F}$ such that if $\mathbf{F}'$ is a UOWHF, then so is $\mathbf{F}$. Then we extend the domain of $\mathbf{F}$ using $D_s^A$ to obtain the family $\mathbf{F}^{(A,s)}$ and show that if $D_s^A$ is not null-free then it is possible to exhibit a collision for every function in $\mathbf{F}^{(A,s)}$. Now we argue as follows. If $\mathbf{F}'$ is a UOWHF and

$A$ is correct for $s$ invocations, then $\mathbf{F}^{(A,s)}$ must also be a UOWHF and hence $D_s^A$ must be null-free. This intuitive argument is now formalized in terms of reductions.

Let $A \in \boldsymbol{\mathcal{A}}$ and $D_s^A$ be the DAG corresponding to $s$ invocations of the compression family by $A$. We set $\delta = \delta(D_s^A)$. Let $\mathbf{F}' = \{h'_k\}_{k\in\mathcal{K}}$ where each $h'$ is an $(n, m')$ function with $\mathcal{K} = \{0,1\}^K$, $m = m' + K$ and $n = \delta m + \delta + 1$. For $z \in \{0,1\}^n$ write

$$z = z_{1,1}||z_{1,2}||z_{2,1}||z_{2,2}||\ldots||z_{i,1}||z_{i,2}||\ldots||z_{\delta,1}||z_{\delta,2}||y||b$$

where $|z_{i,1}| = m$, $|z_{i,2}| = K$ for $1 \le i \le \delta$, $|y| = \delta$ and $b \in \{0,1\}$. We write $y = y(z)$ and $b = b(z)$ to show the dependence of $y$ and $b$ on $z$. Given $z \in \{0,1\}^n$, define $\mathsf{KLst} = \{z_{1,2}, z_{2,2}, z_{3,2}, \ldots, z_{\delta,2}\}$. Given $z \in \{0,1\}^n$ and $k \in \mathcal{K}$, define a Boolean function $\phi(z,k)$ to be true ($\mathsf{T}$) if and only if $k = \bigoplus_{w\in S} w$ for some $\emptyset \ne S \subseteq \mathsf{KLst}(z)$. We define the family of functions $\mathbf{F} = \{h_k\}_{k\in\mathcal{K}}$, where each $h_k$ is an $(n,m)$ function in the following manner.

$$\left.\begin{aligned} h_k(z) &= h'_k(z)||k \quad \text{if } b = 1 \text{ and } \phi(z,k) = \mathsf{F};\\ &= h'_k(z)||0^K \ \text{if } b = 0, y = 0^\delta \text{ and } \phi(z,k) = \mathsf{F};\\ &= h'_k(z)||S_y \ \ \text{if } b = 0, y \ne 0^\delta \text{ and } \phi(z,k) = \mathsf{F};\\ &= 1^m \qquad\quad\ \ \phi(z,k) = \mathsf{T}. \end{aligned}\right\} \tag{1}$$

Here $y = y(z)$ and $S_y = \oplus_{y_i=1} z_{i,2}$, i.e., the XOR's of the $z_{i,2}$'s for which the $i$th bit of $y$ is 1.

**Proposition 4.** *Suppose there is an $(\epsilon, q)$-strategy for $\mathcal{G}(\mathbf{F})$. Then there is an $(\epsilon - \frac{1}{2^K}, q)$-strategy for $\mathcal{G}(\mathbf{F}')$.*

Intuitively, this means that if $\mathbf{F}'$ is a UOWHF, then so is $\mathbf{F}$. In the next result we show that if $D_s^A$ is not null-free, then it is possible to exhibit a collision for each function in $\mathbf{F}^{(A,s)}$.

**Lemma 1.** *Let $A \in \boldsymbol{\mathcal{A}}$ and $\mathbf{F}$ be defined as in (1). For $s > 0$, let $\mathbf{F}^{(A,s)}$ be the family obtained by extending the domain of $\mathbf{F}$ using $D_s^A$. If $D_s^A$ is not null-free, then it is possible to define two strings $x, x'$ such that $x \ne x'$ and $H_p^{(A,s)}(x) = H_p^{(A,s)}(x')$ for any $H_p^{(A,s)} \in \mathbf{F}^{(A,s)}$,*

We now translate Lemma 1 into a lower bound on the number of masks.

**Definition 1.** *Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be two subtrees of $D_s^A$. We denote by $T_1 \Delta T_2$ the subtree of $D_s^A$ induced by the set of arcs $E_1 \Delta E_2$, where $E_1 \Delta E_2$ is the symmetric difference between $E_1$ and $E_2$.*

**Definition 2.** *Let $\mathcal{F}$ be a family of non-trivial subsets of $D_s^A$ such that for any $T_1, T_2 \in \mathcal{F}$, the tree $T_1 \Delta T_2$ is also a non-trivial subtree of $D_s^A$. We call $\mathcal{F}$ a connected family of $D_s^A$.*

**Lemma 2.** *Let $D_s^A$ be null-free and let $\mathcal{F}$ be a connected family of $D_s^A$. Then*

1. *For any $T \in \mathcal{F}$, $\mathsf{vec}_\psi(T) \ne (0, \ldots, 0)$.*
2. *For any $T_1, T_2 \in \mathcal{F}$, $\mathsf{vec}_\psi(T_1) \ne \mathsf{vec}_\psi(T_2)$.*

*Consequently, $2^\rho - 1 \geq |\mathcal{F}|$ or equivalently $\rho \geq \lceil \log_2(|\mathcal{F}|+1) \rceil$, where $\rho$ is the number of masks used by $A$ for $s$ invocations.*

Lemma 2 provides a lower bound on the number of masks in terms of sizes of connected families. Thus the task is to find a connected family of maximum size in $D_s^A$. We show the existence of a connected family of size $(s-1)$ in $D_s^A$. For each intermediate node $v \in D_s^A$, let $P_v$ be the path from $v$ to the final node of $D_s^A$. Define $\mathcal{F} = \{P_v : v \text{ is an intermediate node in } D_s^A\}$. It is easy to check that $\mathcal{F}$ is a connected family of size $(s-1)$. Hence we have the following result.

**Theorem 1.** *Let $s > 0$ and $A \in \boldsymbol{\mathcal{A}}$ be correct for $s$ invocations. Then the number of masks required by $A$ is at least $\lceil \log_2 s \rceil$.*

The bound in Theorem 1 is tight since Shoup's algorithm [14] meets this bound with equality. This also shows that Shoup's algorithm is optimal for the class $\boldsymbol{\mathcal{A}}$ . Also we would like to point out that the lower bound of Theorem 1 can be improved for particular algorithms.

**Lemma 3.** *Suppose $D_s^A$ is the full binary tree on $s = 2^t - 1$ nodes. If $t = 2$, there is a connected family of size 3 in $D_s^A$ and for $t \geq 3$, there is a connected family of size $5 \times 2^{t-2} - 2$ in $D_s^A$. Consequently, $\rho \geq 2$ for $t = 2$ and $\rho \geq t+1$ for $t \geq 3$.*

## 4 New Construction

For $t > 0$, let $\mathcal{T}_t$ be the binary tree defined as $\mathcal{T}_t = (V_t = \{P_0, \ldots, P_{2^t-2}\}, A_t)$, where $A_t = \{(P_{2j+1}, P_j), (P_{2j+2}, P_j) : 0 \leq j \leq 2^{t-1}-2\}$. The underlying digraph for our algorithm is a binary tree with sequential paths terminating at the leaf nodes of the tree. We define a digraph $\mathcal{G}_{t,i}$ which consists of the full binary tree $\mathcal{T}_t$ alongwith a total of $i$ nodes on the sequential paths. The precise definition of $\mathcal{G}_{t,i} = (V_{t,i}, A_{t,i})$ is

$$\left.\begin{array}{rl} V_{t,i} &= V_t \cup \{Q_0, \ldots, Q_{i-1}\} \\ A_{t,i} &= A_t \cup \{(Q_j, P_{2^{t-1}+j-1}) : 0 \leq j \leq 2^{t-1}-1\} \\ &\quad \cup \{(Q_j, Q_{j-2^{t-1}}) : 2^{t-1} \leq j \leq i-1\}. \end{array}\right\} \tag{2}$$

The total number of nodes in $\mathcal{G}_{t,i}$ is equal to $2^t - 1 + i$, where $2^t - 1$ nodes are in the binary tree part and $i$ nodes are in the sequential part. We define parameters $r_{t,i}$ and $s_{t,i}$ (or simply $r$ and $s$) in the following manner: If $i = 0$, then $r = s = 0$; if $i > 0$, then $r$ and $s$ are defined by the equation:

$$i = r2^{t-1} + s \tag{3}$$

where $s$ is a unique integer from the set $\{1, \ldots, 2^{t-1}\}$. For $i > 0$, we can write $i = (r+1) \times s + (2^{t-1} - s)r$. Thus in $\mathcal{G}_{t,i}$ there are $s$ sequential paths of length $(r+1)$ each and these terminate on the left most $s$ leaf nodes of $\mathcal{T}_t$. There are also $(2^{t-1} - s)$ sequential paths of length $r$ each and these terminate on the other $(2^{t-1} - s)$ leaf nodes of $\mathcal{T}_t$. Figure 1 shows $\mathcal{G}_{4,19}$.

We define $\rho_{t,i}$ or (simply $\rho$) to be the maximum length (counting only $Q$ nodes) of a path from a $Q$-node to a $P$-node. Hence $\rho = 0$ if $i = 0$ and $\rho = r + 1$ if $i > 0$.

When $i = 0$, $\mathcal{G}_{t,i}$ is simply the full binary tree $\mathcal{T}_t$ and when $t = 1$, $\mathcal{G}_{t,i}$ is a dipath of length $r + 1$. These are the two extreme cases – one leading to a full binary tree and the other leading to a single dipath. In practical applications, $t$ will be fixed and there will be "long" dipaths terminating on the leaf nodes of $\mathcal{T}_t$. For implementation purpose, the number of processors required is $2^{t-1}$. Hence for practical applications, the value of $t \le 5$.
**Remark:** The idea of breaking a message into parts, hashing them independently and finally combining the outputs is present in Damgård [2] in the context of collision resistant hash functions. The current construction can be seen as a development of the "UOWHF version" of this idea.

## 4.1 Notation

We define a few notation for future reference.

1. $t$ is the number of levels in the binary tree $\mathcal{T}_t$.
2. $i$ is the total number of nodes in the sequential part of the algorithm.
3. $r$ and $s$ are as defined in (3).
4. $\rho = 0$ if $i = 0$ and $\rho = r + 1$ if $i > 0$.
5. $N = 2^t - 1 + i$ is the total number of nodes in $\mathcal{G}_{t,i}$.
6. For $U \in V_{t,i}$, define $\mathsf{nodenum}(U) = j$ if $U = P_j$ and $\mathsf{nodenum}(U) = j + 2^t - 1$ if $U = Q_j$.
7. For $U \in V_{t,i}$, we say that $U$ is a $P$-node (resp. $Q$-node) if $U = P_j$ (resp. $U = Q_j$) for some $j$.

For $U \in V_{t,i}$, we define $\mathsf{indeg}(U)$ (resp. $\mathsf{outdeg}(U)$) to be the indegree (resp. outdegree) of $U$. Note that other than $P_0$ each node $U$ has $\mathsf{outdeg}(U) = 1$. Thus for each node $U \neq P_0$ there is a unique out neighbour.

The concept of level is defined in the following manner. There are $L = \rho + t$ levels in $\mathcal{G}_{t,i}$ and the level number of each node is defined as follows.

$$\left.\begin{array}{ll}\mathsf{level}(P_j) = L - 1 - j_1 & \text{if } 2^{j_1} - 1 \le j \le 2^{j_1+1} - 2 \text{ and } 0 \le j_1 \le t - 1;\\ \mathsf{level}(Q_j) = \rho - j_1 - 1 & \text{if } j_1 2^{t-1} \le j \le (j_1 + 1)2^{t-1} - 1 \text{ and } 0 \le j_1 \le r - 1;\\ \mathsf{level}(Q_j) = 0 & \text{if } r2^{t-1} \le j \le r2^{t-1} + s.\end{array}\right\} \quad (4)$$

Note that if $\rho = 0$, there are no $Q$-nodes and hence the level numbers of $Q$-nodes are not defined. The root node of $\mathcal{T}_t$ has the highest level. Nodes with indegree zero can be at levels zero and one. Let $U \in V_{t,i}$ and $j = \mathsf{nodenum}(U)$: If $0 \le j \le 2^{t-1} - 2$ then we define $\mathsf{lchild}(U) = P_{2j+1}$ and $\mathsf{rchild}(U) = P_{2j+2}$; if $2^{t-1} - 1 \le j \le N - 1$, then we define predecessor of $U$ in the following manner:

$$\left.\begin{array}{ll}\mathsf{pred}(U) = Q_{j+2^{t-1}} & \text{if } 2^{t-1} - 1 \le j \le 2^{t-1} + i - 2;\\ \qquad\quad\;\; = \mathsf{NULL} & \text{if } 2^{t-1} + i - 1 \le j \le N - 1;\end{array}\right\} \quad (5)$$

For a node $U$, $\mathsf{pred}(U) = \mathsf{NULL}$ implies that the indegree of $U$ is zero.

### 4.2 Mask Assignment Algorithm

There are two disjoint sets of masks $\{\alpha_0, \ldots, \alpha_{l-1}\}$ and $\{\beta_0, \ldots, \beta_{t-2}\}$ where $l = \lceil \log(\rho + t) \rceil$. The mask assignment

$$\psi : A_{t,i} \to \{\alpha_0, \ldots, \alpha_{l-1}\} \cup \{\beta_0, \ldots, \beta_{t-2}\}.$$

is a function from the set of arcs of $\mathcal{G}_{t,i}$ to the set of masks. The definition of $\psi$ is as follows: Let $(U, V) \in A_{t,i}$ with $\mathsf{level}(U) = j - 1$ and $\mathsf{level}(V) = j$ for some $j \in \{1, \ldots, L-1\}$.

- If (($U$ is a $Q$-node) or ($U$ is a $P$-node and $U = \mathsf{lchild}(V)$)),
then $\psi(U, V) = \alpha_{\nu(j)}$.
- If ($U$ is a $P$-node and $U = \mathsf{rchild}(V)$) then $\psi(U, V) = \beta_{j-(\rho+1)}$.

Here $\nu(j)$ is defined to be the non negative integer $j_1$ such that $2^{j_1} | j$ and $2^{j_1+1} \not| j$. Also for the convenience of notation we write $\psi(U, V)$ instead of $\psi((U, V))$. The mask assignment for $\mathcal{G}_{4,19}$ is shown in Figure 1.

### 4.3 Optimality of Mask Assignment

The total number of masks used is equal to $t - 1 + \lceil \log(\rho + t) \rceil$. The total number of nodes in $\mathcal{G}_{t,i}$ is equal to $N = 2^t - 1 + i$. Using Theorem 1, at least $\mathcal{L}_{t,i} = \lceil \log(2^t - 1 + i) \rceil$ masks are required by any algorithm in class $\boldsymbol{\mathcal{A}}$. Our algorithm requires $\mathcal{R}_{t,i} = t - 1 + \lceil \log(\rho + t) \rceil$ masks. Define $\mathcal{D}_{t,i} = \mathcal{R}_{t,i} - \mathcal{L}_{t,i}$. We study $\mathcal{D}_{t,i}$.

**Proposition 5.**

$$\left.\begin{aligned} \mathcal{D}_{t,i} &= 0 && \textit{if } i = 0 \textit{ and } t = 1; \\ &= \lceil \log t \rceil - 1 && \textit{if } i = 0 \textit{ and } t > 1; \\ &= \lceil \log(r + 1 + t) \rceil - \left\lceil \log \left( r + 2 + \tfrac{s-1}{2^{t-1}} \right) \right\rceil && \textit{if } i > 0. \end{aligned}\right\} \quad (6)$$

*Furthermore, $\mathcal{D}_{t,i} = 0$ if and only if either $t = 1$; or ($t = 2$ and $i = 0$); or $2^j - 1 - \lceil (s-1)/2^{t-1} \rceil \le r \le 2^{j+1} - t - 1$ for some $j > 0$.*

For $t = 1$, the mask assignment algorithm reduces to the mask assignment algorithm of Shoup [14] and for $i = 0$, the mask assignment algorithm reduces to the mask assignment algorithm of Sarkar [13]. Hence we concentrate on the case $t > 1$ and $i > 0$. For practical parallel implementation, the value of $t$ will determine the number of processors and will be fixed whereas the value of $i$ can grow in an unbounded manner.

Suppose $2^{\tau-1} < t - 1 \le 2^{\tau}$. For $j \ge 0$, define two intervals of integers in the following manner:

$$\left.\begin{aligned} I_j &= \{2^{\tau+j} - 1 - \left\lceil \tfrac{s-1}{2^{t-1}} \right\rceil, 2^{\tau+j} - \left\lceil \tfrac{s-1}{2^{t-1}} \right\rceil, \ldots, 2^{\tau+j+1} - t - 1\}; \\ J_j &= \{2^{\tau+j+1} - t, 2^{\tau+j+1} - t + 1, \ldots, 2^{\tau+j+1} - 2 - \left\lceil \tfrac{s-1}{2^{t-1}} \right\rceil\}. \end{aligned}\right\} \quad (7)$$

Clearly, $|I_j| = 2^{\tau+j} - t + 1 + \lceil (s-1)/2^{t-1} \rceil$, $|J_j| = t - 1 - \lceil (s-1)/2^{t-1} \rceil$ and $|I_j| + |J_j| = 2^{\tau+j}$.

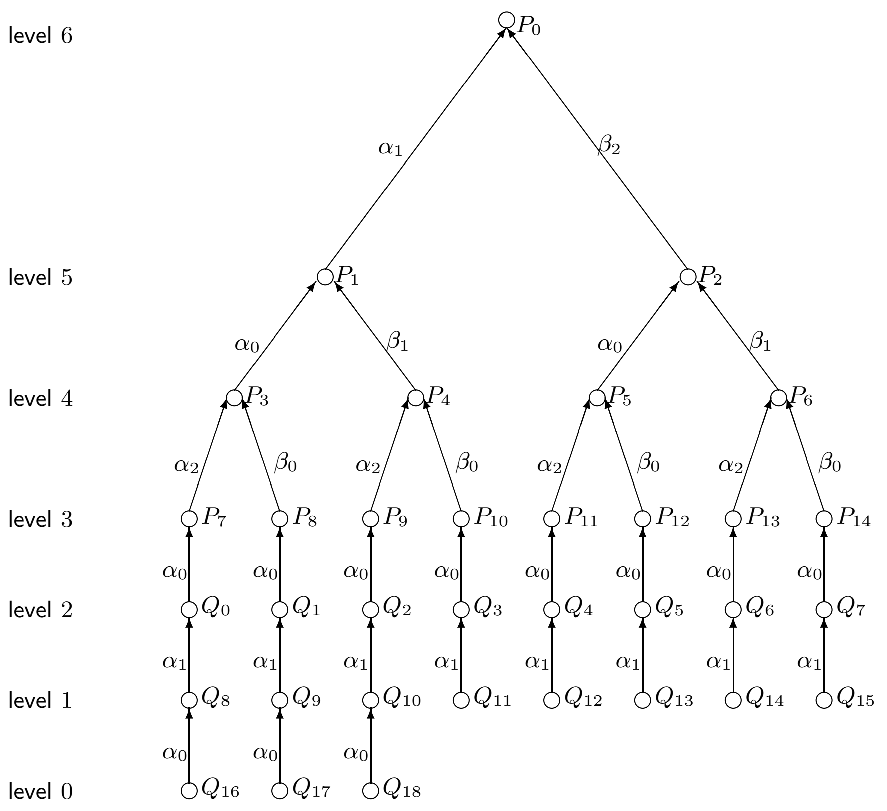
level 6
level 5
level 4
level 3
level 2
level 1
level 0
$P_0$
$\alpha_1$
$\beta_2$
$P_1$
$P_2$
$\alpha_0$
$\beta_1$
$P_3$ $P_4$ $P_5$ $P_6$
$\alpha_2$
$\beta_0$
$P_7$ $P_8$ $P_9$ $P_{10}$ $P_{11}$ $P_{12}$ $P_{13}$ $P_{14}$
$Q_0$ $Q_1$ $Q_2$ $Q_3$ $Q_4$ $Q_5$ $Q_6$ $Q_7$
$Q_8$ $Q_9$ $Q_{10}$ $Q_{11}$ $Q_{12}$ $Q_{13}$ $Q_{14}$ $Q_{15}$
$Q_{16}$ $Q_{17}$ $Q_{18}$

**Fig. 1.** Example of mask assignment for $t = 4$ and $i = 19$.

**Theorem 2.** *Suppose $i > 0$, $2^{\tau-1} < t - 1 \le 2^{\tau}$ and for $j \ge 0$, $I_j$ and $J_j$ are as defined in 7. Then $\mathcal{D}_{t,i} = 0$ if $r \in I_j$; and $\mathcal{D}_{t,i} = 1$ if $r \in J_j$.*

From Theorem 2, it follows that for $j \ge \tau$, in any interval $2^j + 1 \le r \le 2^{j+1}$, there are exactly $t-1-\lceil (s-1)/2^{t-1} \rceil$ points where the algorithm is suboptimal with respect to the lower bound. Moreover, at these points it requires exactly one extra mask over the lower bound. In any practical parallel implementation, the value of $t$ will be fixed, whereas the value of $r$ will grow. In such a situation, the ratio $\frac{1}{2^j}(t-1-\lceil (s-1)/2^{t-1} \rceil)$ approaches zero very fast and hence we can say that for $t \ge 2$, the algorithm achieves optimal key length expansion *almost everywhere*. (Note that for $t = 1$, the algorithm reduces to Shoup's algorithm and hence achieves optimal key length expansion.)

## 4.4 Computation of Message Digest

Let $\{h_k\}_{k \in \mathcal{K}}$, where each $h_k$ is an $(n, m)$ function, be the compression UOWHF whose domain is to be extended. *For $t > 1$, we require $n \ge 2m$.* The nodes of $\mathcal{G}_{t,i}$ represent the invocations of $h_k$. Thus $h_k$ is invoked a total of $N$ times. The output of $P_0$ is provided as output digest, whereas the outputs of all the other nodes are used as inputs to other invocations as defined by the arcs of $\mathcal{G}_{t,i}$. Using Proposition 1 of [12] we obtain the following: Suppose a message $x$ is hashed using $\mathcal{G}_{t,i}$ and the compression UOWHF $\{h_k\}_{k \in \mathcal{K}}$, where each $h_k$ is an $(n, m)$ function. Then $|x| = N(n-m)+m$.

Thus $\{h_k\}_{k\in\mathcal{K}}$ is extended to $\{H_p^{(t,i)}\}_{p\in\mathcal{P}}$ where each $H_p^{(t,i)}$ is an $(N(n-m)+m, m)$ function and

$$p = k||\alpha_0||\ldots||\alpha_{l-1}||\beta_0||\ldots||\beta_{t-2}.$$

The message $x$ of length $N(n-m)+m$ has to be formatted into small substrings and provided as input to the different invocations of $h_k$. Write $x = x_0||\ldots||x_{N-1}$, where the lengths of the $x_j$'s are as follows.

$$\left.\begin{array}{ll} |x_j| = n-2m & \text{if } 0 \le j \le 2^{t-1}-2; \\ \quad\;\; = n-m & \text{if } 2^{t-1}-1 \le j \le 2^{t-1}+i-2; \\ \quad\;\; = n & \text{if } 2^{t-1}+i-1 \le j \le 2^t+i-2. \end{array}\right\} \qquad (8)$$

The substring $x_j$ is provided as input to node $U$ with $\mathsf{nodenum}(U) = j$ and the $m$-bit output of $U$ is denoted by $z_j$. The outputs $z_1, \ldots, z_{N-1}$ are masked using the $\alpha$ and $\beta$ masks to obtain $m$-bit strings $y_1, \ldots, y_{N-1}$ in the following manner.

$$y_j = z_j \oplus \psi(U, V) \text{ if } \mathsf{nodenum}(U) = j. \qquad (9)$$

The inputs to the invocations of $h_k$ are formed from the $x$'s and the $y$'s in the following manner. There are $N$ invocations whose inputs are denoted by $w_0, \ldots, w_{N-1}$ and are defined as follows.

$$\left.\begin{array}{ll} w_j = x_j||y_{2j+1}||y_{2j+2} & \text{if } 1 \le j \le 2^{t-1}-2; \\ \quad\; = x_j||y_{j+2^{t-1}} & \text{if } r > 0 \text{ and } 2^{t-1}-1 \le j \le 2^{t-1}+i-2; \\ \quad\; = x_j & \text{if } 2^{t-1}+i-1 \le j \le 2^t+i-2. \end{array}\right\} \qquad (10)$$

Note that the length of each $w_j$ is $n$ and hence we can invoke $h_k$ on $w_j$ for all $j \in \{0, \ldots, N-1\}$. For any node $U \in V_{t,i}$ we define $x(U), y(U)$ and $w(U)$ to be the $x, y$ and $w$ strings associated to the node $U$ as defined respectively in (8), (9) and (10). Similarly the output of node $U$ will be denoted by $z(U)$.

Now we are ready to describe the digest computation algorithm. Most of the work has already been done, so that the description of the algorithm becomes simple. Suppose the compression UOWHF is $\{h_k\}_{k\in\mathcal{K}}$. We describe the digest computation of $H_p^{(t,i)}(x)$.

**Algorithm to compute $H^{(t,i)}(x)$**

```
1. for $j = 0$ to $L - 1$ do
2.     for all $U$ with $\mathsf{level}(U) = j$ do in parallel
3.         compute $z(U) = h_k(w(U))$;
4.     end do;
5. end do;
6. return $z_0$.
```

## 5 Conclusion

In this paper, we have formalized the model for masking based domain extending algorithms. Using this formal model, we obtained a lower bound on the minimum amount of key expansion required by any masking based algorithm. Our second contribution has been to develop a simple and efficient parallel domain extender. The key expansion of our algorithm is almost everywhere optimal whereas the efficiency of parallelism is asymptotically optimal.

## 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. *Proceedings of Crypto 1989*, Lecture Notes in Computer Science, volume 435 (1990), 416–427.
3. W. Lee, D. Chang, S. Lee, S. Sung and M. Nandi. New Parallel Domain Extenders for UOWHF. *Proceedings of Asiacrypt 2003*, Lecture Notes in Computer Science, pp 208–227.
4. R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes provably as secure as subset sum. *Journal of Cryptology*, volume 9, number 4, 1996.
5. R. C. Merkle. One way hash functions and DES. *Proceedings of Crypto 1989*, Lecture Notes in Computer Science, volume 435, 1990, pp 428–226.
6. I. Mironov. Hash functions: from Merkle-Damgård to Shoup. *Proceedings of Eurocrypt 2001*, Lecture Notes in Computer Science, volume 2045 (2001), Lecture Notes in Computer Science, pp 166–181.
7. M. Nandi. Optimal Domain Extension of UOWHF and a Sufficient Condition. *Proceedings of SAC 2004*, Lecture Notes in Computer Science, to appear.
8. M. Nandi. A New Tree based Domain Extension of UOWHF, Cryptology e-print archive, Report No. `http://eprint.iacr.org, 2003/142`.
9. 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.
10. J. Rompel. One-way functions are necessary and sufficient for digital signatures. *Proceedings of the 22nd Annual Symposium on Theory of Computing*, ACM, 1990.
11. P. Sarkar. Masking Based Domain Extenders for UOWHFs: Bounds and Constructions, Cryptology e-print archive, `http://eprint.iacr.org`, Report No. 2003/225.
12. P. Sarkar. Domain Extenders for UOWHFs: A Finite Binary Tree Algorithm, Cryptology e-print archive, `http://eprint.iacr.org`, Report No. 2003/009.
13. P. Sarkar. Construction of UOWHF: Tree Hashing Revisited, Cryptology e-print archive, `http://eprint.iacr.org`, Report No. 2002/058.
14. V. Shoup. A composition theorem for universal one-way hash functions. *Proceedings of Eurocrypt 2000*, pp 445–452, 2000.
15. D. Simon. Finding collisions on a one-way street: Can secure hash function be based on general assumptions?, *Proceedings of Eurocrypt 1998*, Lecture Notes in Computer Science, pp 334–345, 1998.
16. D. R. Stinson. Some observations on the theory of cryptographic hash functions. `http://www.cacr.math.uwaterloo.ca/~dstinson/papers/newhash.ps`.