情報保護學會論文誌
第14券 第2號, 2004. 4

UOWHF 구성방법: 최적의 키길이를 가지는 새로운 병렬 도메인 확장기

이 원 일[†‡], 장 동 훈
고려대학교 정보보호기술연구센터

Construction of UOWHF: New Parallel Domain Extender with Optimal Key Size

Wonil Lee[†‡], Donghoon Chang
CIST, Korea University

요 약

본 논문에서는 UOWHF의 도메인을 확장하기 위한 새로운 병렬 처리 알고리즘을 제안한다. 제시되는 알고리즘은 non-complete l-ary tree에 기반을 두고 있으며 현재까지 최적의 키 길이를 가진 유일한 알고리즘인 Shoup의 알고리즘과 동일한 최적의 키 길이를 가진다. 또한 Sarkar의 결과[7]를 이용하여 본 논문에서 제시되는 알고리즘이 Shoup의 알고리즘과 함께 Sarkar가 제시한 도메인 확장 알고리즘들의 커다란 집합 중에서 가장 최적화된 키 길이를 가짐을 증명한다. 그러나 제안 알고리즘의 병렬처리능력은 complete tree에 기반한 구성 방법들 보다 약간 비효율적이다. 그러나 만약 l이 점점 커진다면 알고리즘의 병렬처리능력도 complete tree에 기반한 방법들에 가까워진다.

ABSTRACT

We present a new parallel algorithm for extending the domain of a UOWHF. Our algorithm is based on non-complete l-ary tree and has the same optimal key length expansion as Shoup's which has the most efficient key length expansion known so far. Using the recent result [8], we can also prove that the key length expansion of this algorithm and Shoup's sequential algorithm are the minimum possible for any algorithms in a large class of "natural" domain extending algorithms. But its parallelizability performance is less efficient than complete tree based constructions. However if l is getting larger then the parallelizability of the construction is also getting near to that of complete tree based constructions.

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

Ⅰ. Introduction

Naor and Yung [6] introduced the notion of universal one-way hash function (UOWHF) to prove that secure digital signatures can be based on any 1-1 one-way function. A UOWHF is a family of functions $\{h_k\}_{k\in K}$ for which the following task of the adversary is computationally infeasible. The adversary has to choose a

접수일: 2003년 12월 26일; 채택일: 2004년 4월 14일
* 본 연구는 정보통신부 대학 IT연구센터 육성·지원사업의 연구결과로 수행 되었습니다.
† 주저자, ‡ 교신저자 : wonil@cist.korea.ac.kr

x from the domain, and then given a random $k \in K$, he has to find a y such that $x \neq y$ but $h_k(x) = h_k(y)$. Intuitively, a UOWHF is a weaker primitive than a collision resistant hash function (CRHF), since the task of the adversary is more difficult, i.e., the adversary has to commit to the string x before knowing the actual hash function h_k for which the collision has to be found. Furthermore, Simon[10] had shown that there is an oracle relative to which UOWHFs exist but not CRHFs.

A UOWHF is an attractive alternative to a CRHF because it seems that building an efficient and secure UOWHF is easier than building an efficient and secure CRHF, and in many applications, most importantly for building digital signature schemes, a UOWHF is sufficient. In addition, as mentioned in [1], the birthday attack does not apply to UOWHFs. Hence the size of the message digest can be significantly shorter.

A reasonable approach to designing a UOWHF that hashes messages of arbitrary and variable length is to first design a compression function, that is, a UOWHF that hashes fixed-length messages, and then design a method for composing these compression functions so as to hash arbitrary and variable messages. The present paper deals with the second problem, that of composing compression functions. We will call the composite method *construction* or *domain extender* for the most part in this paper. The main technical problem in designing such domain extender is to keep the key length of the domain extender from getting too large.

Most practical signature schemes follow "hash-and-sign" paradigm. They take a message M of an arbitrary length and hash it to obtain a constant length string, which is then fed into a signing algorithm. Many schemes use CRHFs to hash a message x, but as it was first pointed out in [1] a UOWHF suffices for that purpose. Indeed, if $\{h_k\}_{k \in K}$ is a UOWHF, then to sign a message x, the signer chooses a random key k, and produces the signature $(k, \sigma(k, h_k(x)))$, where σ is the underlying signing function for short messages. As key k is a part of input of signing algorithm σ we should keep key size as small as possible. So, whenever a new construction will be proposed we have to keep in the mind the following :

1. Minimizing the key length expansion: This is certainly a very important aspect of any domain extending algorithm.
2. Parallel implementation: From an implementation point of view parallelizability is also an important aspect of any domain extending algorithm.

Bellare and Rogaway[1] suggested the XOR tree hash (XTH) construction in order to reduce the key length expansion. Since XTH is based on the complete (or full) l-ary tree($l \geq 2$), it has also an efficiency regarding the parallelizability (the processing speed). Shoup [9] proposed a sequential construction (can not be implemented by parallel algorithm) which is more efficient than XTH with regard to the key length expansion. Furthermore, Mironov [4] had shown that the key length expansion needed in Shoup's construction is the minimum possible for any sequential algorithm. In other words, there is no sequential algorithm which has more efficient key length expansion than Shoup's. Sarkar modified the XTH

construction which reduces the key size together with same parallel performance as XTH. However, his construction does not have the same key length expansion as Shoup's one.

Note that all the previously proposed parallel algorithms took more key length expansion than that of Shoup's sequential algorithm. So an important question is whether this is true in general of any parallel algorithm. Our new construction shows that this is not the case. The new construction has the same key length expansion as Shoup's one. But the construction does not have the same parallelizable performance with Sarkar's construction (See Table 1). The construction will be called l-DIMensional construction (l-DIM, $l \geq 2$).

We think it is difficult to say that which one is more important than the other between the key length expansion and the parallel implementation. Of course, it would be very nice to have a regular parallel structure something like the complete tree which also minimizes the key length expansion. But at this point, we do not have any such algorithm. Hence, in our opinion, we should separately consider both the above- mentioned two points of view with the same importance. And the present work is important in regarding the former point of view. Particularly, the l-DIM and Shoup's one are the only two known algorithms which minimize the key length expansion. In addition that, the reason why the l-DIM has more meaning is that it is a parallel algorithm which has the same key length expansion as Shoup's sequential algorithm and this is the very first trial in designing the parallel algorithms. Using the recent result[8], we can also prove that the key length expansion of our new parallel construction and Shoup's sequential construction are the minimum possible for any constructions in a large class of "natural" domain extenders includ- ing all the previously proposed methods.

Table 1. Specific comparison of domain extenders for UOWHF.

	Shoup[9]	l-DIM	Sarkar[7]
seq/par	sequential	parallel	parallel
message length	$2^t n - (2^t-1)m$	$2^t n - (2^t-1)m$	$(2^t-1)n - (2^t-2)m$
# invocations of h_k	2^t	2^t	2^t-1
# masks	t	t	$t + \lceil \log_2 t \rceil - 1$
# rounds	2^t	$l2^{t/l} - l + 1$ ($t \equiv 0 \bmod l$)	t
speed-up	1	$\frac{2^t}{l2^{t/l} - l + 1}$ ($t \equiv 0 \bmod l$)	$\frac{2^t - 1}{2^t}$
ranking of key size	1	1	3
ranking of parallelizability	3	2	1

II. Preliminaries

The following notations are used in this paper.

1. $[a, b] = \{a, a+1, \ldots, b\}$ 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 $2^{j+1} \nmid i$.
4. In the 4-dimensional construction, for integer t, $g(t) = (a, b, c, d)$, where
   $a = \lfloor t/4 \rfloor + \lfloor ((t \bmod 4) + 3)/4 \rfloor$,
   $b = \lfloor t/4 \rfloor + \lfloor ((t \bmod 4) + 2)/4 \rfloor$,
   $c = \lfloor t/4 \rfloor + \lfloor ((t \bmod 4) + 1)/4 \rfloor$, and
   $d = \lfloor t/4 \rfloor$.
   Here $t \bmod 4 = t - \lfloor t/4 \rfloor n$.
5. In the 4-dimensional construction let

$T_t=(V_t, E_t)$ be a non-complete 4-ary tree, where $V_t=\{1,2,\ldots,2^t\}$ 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 $2^a< i\le 2^{a+b}$, $e_i=(i, i-2^{a+b})$ for $2^{a+b}< i\le 2^{a+b+c}$, and $e_i=(i, i-2^{a+b+c})$ for $2^{a+b+c}\le i\le 2^t$. Here a, b, c, and d are such that $g(t)=(a, b, c, d)$.

Let $\{h_k\}_{k\in K}$ be a keyed family of hash functions, where each $h_k: \{0,1\}^n\rightarrow\{0,1\}^m$, $n>m$. Consider the following adversarial game.

1. $A^{guess}()$ outputs an $x\in\{0,1\}^n$ and some state information σ.
2. $A^{find}(k, x, \sigma)$ outputs x' such that $x\neq x'$ and $h_k(x)=h_k(x')$ or reports failure, where $k\in_R K$.

We say that A is an (ε, η)-strategy for $\{h_k\}_{k\in K}$ if the success probability of A is at least ε and it invokes the hash function h_k at most η times. In this case we say that the adversary has an (ε, η)-strategy for $\{h_k\}_{k\in K}$. Note that we do not include time as an explicit parameter though it would be easy to do so. Informally, we say that $\{h_k\}_{k\in K}$ is a UOWHF if the adversary has a negligible probability of success with respect to any probabilistic polynomial time strategy. Here, the security parameter is length of the message i.e., the length of the input.

In this paper we are interested in extending the domain of a UOWHF. More specifically, given a UOWHF $\{h_k\}_{k\in K}$, $h_k: \{0,1\}^n\rightarrow\{0,1\}^m$, $n>m$, we would like to construct another extended UOWHF $\{H_p\}_{p\in P}$ with $H_p: \{0,1\}^N\rightarrow\{0,1\}^m$, where $n<N$.

We say that B is an (ε, η)-extended strategy for $\{H_p\}_{p\in P}$ if the success probability of B is at least ε and it invokes the hash function h_k at most η times. In this case we say that the adversary has an (ε, η)-extended strategy for $\{H_p\}_{p\in P}$. Note that H_p is built using h_k and hence while studying strategies for H_p we are interested in the number of invocations of the hash function h_k.

The correctness of our construction will essentially be a Turing reduction. We will show that if there is an (ε, η)-extended strategy B for $\{H_p\}_{p\in P}$, then there is an (ε', η')-strategy A for $\{h_k\}_{k\in K}$, where ε' is not significantly lesser than ε and η' is not much larger than η. This shows that if $\{h_k\}_{k\in K}$ is a UOWHF, then so is $\{H_p\}_{p\in P}$. In this case, we say that the domain extension is valid.

The key length for the base hash family $\{h_k\}_{k\in K}$ is $\lceil \log_2|K| \rceil$. On the other hand, the key length for the extended hash family $\{H_p\}_{p\in P}$ is $\lceil \log_2|P| \rceil$. Thus increasing the size of the input from n bits to N bits results in an increase of the key size by an amount $\lceil \log_2|P| \rceil - \lceil \log_2|K| \rceil$. From a practical point of view it is very important to minimize this increase in the key length.

For the remainder of this paper we assume the following conventions.

1. $\{h_k\}_{k\in K}$ is always the base hash family, where $K=\{0,1\}^K$ and $h_k: \{0,1\}^n\rightarrow\{0,1\}^m$. In case of sequential construction $n>m$, and in case of 4-dimensional construction $n>4m$.
2. We will construct $\{H_p\}_{p\in P}$, $H_p: \{0,1\}^N\rightarrow\{0,1\}^m$ using the base hash family

$\{h_k\}_{k\in K}$, where $p=k\|\mu_1\|\mu_2\|\ldots\|\mu_l$ for some l and each μ_i is m-bit binary string called *mask* and $|k|=K$. Here, in case of sequential algorithm $N=n(r+1)-mr$ and in case of 4-dimensional construction $N=n2^t-m(2^t-1)$. Let us define $\mu[i,j]=\mu_i\|\ldots\|\mu_j$, where $1\le i\le j\le l$. We will use μ_j instead of $\mu[1,j]$ for $j\le l$ and define $\mu[0]$ to be empty string.

3. In sequential construction input of H_p is written as $y=y_0\|y_1\|\ldots\|y_r$ where $|y_0|=n$ and $|y_i|=n-m$ for $1\le i\le r$. In 4-dimensional construction input of H_p is written as $x=x_1\|\ldots\ x_{2^t}$ where $|x_i|=n-4m$ for $1\le i<2^a$, $|x_i|=n-3m$ for $2^a\le i\le 2^a(2^b-1)$, $|x_i|=n-2m$ for $2^a(2^b-1)<i\le 2^{a+b}(2^c-1)$, $|x_i|=n-m$ for $2^{a+b}(2^c-1)<i\le 2^{a+b+c}(2^d-1)$, and $|x_i|=n$ for $2^{a+b+c}(2^d-1)<i\le 2^t$. Here a,b,c, and d are such that $g(t)=(a,b,c,d)$.

Ⅲ. Sequential Construction

The best known sequential algorithm is given by Shoup[9]. Let $\psi[1,r]\to[1,l]$ be any function called a **masking assignment**. Fix a masking assignment ψ, $H_p(y)$, the extended hash function, is computed by the following algorithm.

1. Input: $y=y_0\|y_1\|\ldots\|y_r$ and $p=k\|\mu_1\|\mu_2\|\|\mu_l$.
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 $\psi=\nu_2+1$ and $l=1+\lfloor\log_2 r\rfloor$ (in his paper ν_2 is masking assignment but that makes no difference). We will write $s(i,y,k,\mu)$, $z(i,y,k,\mu)$ for s_i and z_i respectively (in the algorithm with input (y,p), where $p=k\|\mu$). Now we will define some terms related with masking assignment and domain extension.

Definition 1. We say that ψ is **correct** if for all $1\le i\le r$, $C\in\{0,1\}^m$, $y\in\{0,1\}^N$ and for any hash function h_k there is an algorithm called $Mdef_{seq}(i,y,k,C,\psi)$ which outputs $\mu=\mu_1\|\mu_2\|\ \ldots\|\mu_l$ such that $s(i,y,k,\mu)=C$. $Mdef_{seq}(i,y,k,C,\psi)$ is called a mask defining algorithm. A sequential construction based on a correct masking assignment is called a correct domain extension. A masking assignment is **totally correct** if there is a mask defining algorithm $Mdef_{seq}(i,y,k,C,\psi)=\mu=\mu_1\|\mu_2\|\ldots\|\mu_l$ for any i,y,k,C as above such that $s(i,y,k,\mu)=C$ holds and μ is a random string whenever C is a random string and other inputs are fixed.

Definition 2. We say that a domain extension algorithm is **valid** if $\{H_p\}_{p\in P}$ is a UOWHF whenever $\{h_k\}_{k\in K}$ is a 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** (or **even-free**) if for each $[a,b]\subseteq[1,r]$ there exists $c\in[a,b]$ such that $\psi(c)$ occurs exactly once (respectively, odd times) in the sequence $\psi(a),\psi(a+1),\ldots,\psi(b)$. Call this c (also the mask $\psi(c)$) a **single-man** for the interval $[a,b]$.

Note that Shoup's construction has the strongly even-free property. And Shoup's construction can be proved by the following mask defining algorithm which is totally correct.

We can define the mask defining algorithm $Mdef_{seq}(i, y, k, C, \psi)$ in order to prove Shoup's construction. (See [5] for details. In fact, [5] also include sufficient condition for valid sequential extension.)

$Mdef_{seq}(i, y, k, C, \psi)$

1. If $i=1$ then define $\mu_{\psi(1)}=C\oplus h_k(y_0)$ and define all yet undefined masks randomly and quit.
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 $\psi':[1, j]\rightarrow[1, l]$ 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 $n\geq 1$ and $y_0'=D||y_c$. Run $Mdef_{seq}(j, y', k, C, \psi')$.
4. Define all yet undefined masks except $\mu_{\psi(c)}$ (i.e. after running $Mdef_{seq}$ some masks may not be defined as ψ' may not be onto or j can be 0) randomly. Compute $\mu_{\psi(c)}=z(c-1, y, k, \mu)\oplus D$ and quit.

Ⅳ. Non-Complete t-ary Tree based Construction

In this section we present a new parallel construction for a UOWHF based on a 4-ary directed tree which is not complete. We will first define the generic algorithm based on the 4-ary directed tree $T_t=(V_t, E_t)$ for $t\geq 4$. For $t=2$ and $t=3$, we can define the algorithm based on the binary and 3-ary tree based construction (See Section 4.3), respectively.

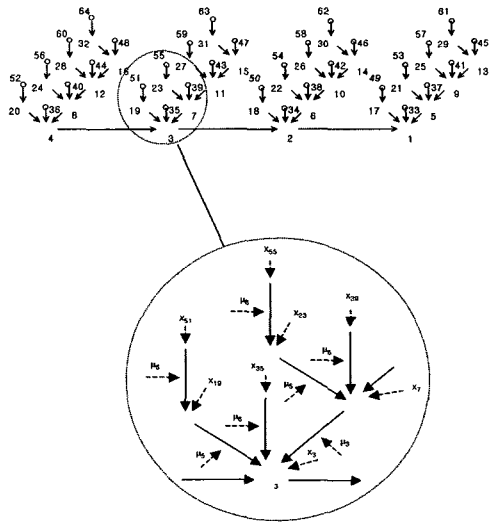

Fig. 1 4-dimensional parallel algorithm ($t=6$ and $x=x_1||\ldots||x_{26}$. Note that $g(6)=(2,2,1,1)$ and $|x_1|=\cdots=|x_3|=n-4m$, $|x_4|=\cdots=|x_{12}|=n-n-3m$, $|x_{13}|=\cdots=|x_{16}|=n-2m$, $|x_{17}|=\cdots=|x_{32}|=n-m$, and $|x_{33}|=\cdots=|x_{26}|=n$. · means $h_k(\cdot)$.)

Like sequential construction, any function $\psi_t: E_t\rightarrow[1, l]$ is a masking assignment. Let $x=x_1||x_2||\ldots||x_{2^t}$ be the input message of length N. Given ψ_t, x, and $p=k||\mu$, $H_p(x)$ is computed by the following algorithm. This is depicted in Figure 1. In this section a, b, c, and d denote the output of $g(t)$.

1. **Input**: $x=x_1||x_2||\ldots||x_{2^t}$ and $p=k||\mu_1||\mu_2||\ldots||\mu_l$.
2. If $2^{a+b+c}(2^d-1)<i\leq 2^t$ then $z_i=h_k(x_i)$.
3. If $d=1$ then goto step 4.
   a. For $j=2^d-2$ down to 1 do
      For $j2^{a+b+c}<i\leq(j+1)2^{a+b+c}$,
      $z_i=h_k(s_{i+2^{a+b+c}}||x_i)$ where $s_i=z_i\oplus\mu_{\psi_t(e_i)}$
      (This notation is also same in the following procedure).
4. For $2^{a+b}(2^c-1)<i\leq 2^{a+b+c}$, $z_i=h_k(s_{i+2^{a+b+c}}||x_i)$.
5. If $c=1$ go to step 6.

a. For $j=2^c-2$ down to 1 do
For $j2^{a+b}<i\le(j+1)2^{a+b}$,
$z_i=h_k(s_{i+2^{a+b}}\|s_{i+2^{a+b}+1}\|x_i)$.

6. For $2^a(2^b-1)<i\le 2^{a+b}$, $z_i=h_k(s_{i+2^{a+b}}\|s_{i+2^{a+b}+1}\|x_i)$.
7. If $b=1$ go to step 8.
a. For $j=2^b-2$ down to 1 do
For $j2^a<i\le(j+1)2^a$,
$z_i=h_k(s_{i+2^a}\|s_{i+2^{a+1}}\|s_{i+2^{a+b}+1}\|x_i)$.
8. For $i=2^a$, $z_i=h_k(s_{i+2^a}\|s_{i+2^{a+1}}\|s_{i+2^{a+b}+1}\|x_i)$.
9. For $i=2^a-1$ down to 1, $z_i=h_k(s_{i+1}\|s_{i+2^a}\|s_{i+2^{a+1}}\|s_{i+2^{a+b}+1}\|x_i)$.
10. **Output**: z_1.

We say that, the above non-complete 4-ary tree based construction is based on the masking assignment ϕ_t. Here, we need some definitions in order to consider the correctness of ϕ_t.

1. We will write $s(i,x,k,\mu,t)$, $z(i,x,k,\mu,t)$ for s_i and z_i, respectively.
2. ϵ means the empty string.
3. For each node $1\le i\le 2^{a+b+c}(2^d-1)$,
a. Define $s^0(i,x,k,\mu,t)$ as $s(i+1,x,k,\mu,t)$ for $1\le i<2^a$ and as ϵ for $2^a\le i\le 2^{a+b+c}(2^d-1)$.
b. Define $s^1(i,x,k,\mu,t)$ as $s(i+2^a,x,k,\mu,t)$ for $1\le i\le 2^a(2^b-1)$ and as ϵ for $2^a(2^b-1)<i\le 2^{a+b+c}(2^d-1)$.
c. Define $s^2(i,x,k,\mu,t)$ as $s(i+2^{a+b},x,k,\mu,t)$ for $1\le i\le 2^{a+b}(2^c-1)$ and as ϵ for $2^{a+b}(2^c-1)<i\le 2^{a+b+c}(2^d-1)$.
d. Define $s^3(i,x,k,\mu,t)$ as $s(i+2^{a+b+c},x,k,\mu,t)$ for $1\le i\le 2^{a+b+c}(2^d-1)$.

Therefore the input of i^{th} node can be represented by $s^0(i,x,k,\mu,t)\|s^1(i,x,k,\mu,t)\|s^2(i,x,k,\mu,t)\|s^3(i,x,k,\mu,t)\|x_i$ for $1\le i\le 2^{a+b+c}(2^d-1)$.

We will say that ϕ_t is **correct** if, for each $1\le i\le 2^{a+b+c}(2^d-1)$, there is an algorithm

$Mdef_{4\dim}(i,x,k,t,r_0,r_1,r_2,r_3,\phi_t)$, where r_0 is a m-bit string if $1\le i<2^a$ and ϵ if $2^a\le i\le 2^a(2^b-1)$, r_1 is a m-bit string if $1\le i\le 2^a(2^b-1)$ and ϵ if $2^a(2^b-1)<i\le 2^{a+b}(2^c-1)$, r_2 is a m-bit string if $1\le i\le 2^{a+b}(2^c-1)$ and ϵ if $2^{a+b}(2^c-1)<i\le 2^{a+b+c}(2^d-1)$, and r_3 is a m-bit string for $1\le i\le 2^{a+b+c}(2^d-1)$ which outputs $\mu=\mu_1\|\mu_2\|\ldots\|\mu_l$ such that $s^j(i,x,k,\mu,t)=r_j$ for $0\le j\le 3$. ϕ_t is **totally correct** if the output μ of the mask defining algorithm is random string provided r_0, r_1, r_2 and r_3 are random strings and other inputs are fixed.

4.1 4-Dimensional Domain Extender

Our parallel construction uses the following masking assignment $\phi_t: E_t\to[1,t]$. The map represents the assignment of masks to the directed edges. Here we present our definition of ϕ_t which needs t masks for 4-dimensional construction. Intuitively, the map ϕ_t is made from expanding the mask assigning method of Shoup's sequential construction into four directions. At first, we define four functions α_t, β_t, γ_t, and δ_t as follows.

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

4. $\delta_t:[1,2^d-1]\rightarrow[a+b+c+1,t]$ is defined by $\delta_t(i)=a+b+c+1+\nu_2(2^d-i)$.

Our masking assignment $\phi_t(e_i)$ is defined as follow:

1. $\phi_t(e_i)=\alpha_t(j)$ if $2\le i\le 2^a$ and $j=i-1$.
2. $\phi_t(e_i)=\beta_t(j)$ if $2^a<i\le 2^{a+b}$ and $j2^a<i\le(j+1)2^a$.
3. $\phi_t(e_i)=\gamma_t(j)$ if $2^{a+b}<i\le 2^{a+b+c}$ and $j2^{a+b}<i\le(j+1)2^{a+b}$.
4. $\phi_t(e_i)=\delta_t(j)$ if $2^{a+b+c}<i\le 2^t$ and $j2^{a+b+c}<i\le(j+1)2^{a+b+c}$.

Now we will prove that the above ϕ_t is totally correct.

Theorem 1. The masking assignment ϕ_t based on four functions α_t, β_t, γ_t, and δ_t as above is totally correct.

Proof. We will define the mask defining algorithm $Mdef_{4\text{dim}}$.

Input: $k,x,i,r_0,r_1,r_2,r_3,\phi_t$

output: $\mu=\mu_1\|\mu_2\|\ldots\|\mu_t$ such that $s^j(i,x,k,\mu,t)=r_j$ for $0\le j\le 3$.

We can define $Mdef_{4\text{dim}}$ for each case $j\in\{1,2,3,4\}$ where

1. $1\le i<2^a$.
2. $2^a\le i\le 2^a(2^b-1)$.
3. $2^a(2^b-1)<i\le 2^{a+b}(2^c-1)$.
4. $2^{a+b}(2^c-1)<i\le 2^{a+b+c}(2^d-1)$.

But we will present the specific procedure of $Mdef_{4\text{dim}}$ for only case 1 since the other cases are very similar and much simpler than case 1. Let $1\le i<2^a$.

1. a. Let $D=2^d-1$. Let $\psi':[1,D]\rightarrow[1,t]$ be a masking assignment such that $\psi'(j)=\phi_t(e_{i+(D+1-j)2^{a+b+c}})$ where $j\in[1,D]$.
   b. Let $y^3=y^3_0\|y^3_1\|\ldots\|y^3_D$ where, $y^3_v=x_{i+(D-v)2^{a+b+c}}$ for $0\le v\le D-1$ and $y^3_D=r_0\|r_1\|r_2\|x_i$. Note that $|y^3_0|=n$ and $|y^3_j|=n-m$ for $1\le j\le D$.
   c. Run $Mdef_{seq}(D,y^3,k,r_3,\psi')$ to get an output $\mu=[a+b+c+1,t]$.
   d. Set $\mu=\mu[t]=\mu'[a+b+c]\|\mu[a+b+c+1,t]$, where $\mu'[a+b+c]$ is the $m(a+b+c)$-bit zero string.
2. a. Let $C=2^c-1$. Let $\psi'':[1,C]\rightarrow[1,t]$ be a masking assignment such that $\psi''(j)=\phi_t(e_{i+(C+1-j)2^{a+b}})$ where $j\in[1,C]$.
   b. Let $y^2=y^2_0\|y^2_1\|\ldots\|y^2_C$ where, $y^2_v=s^3(i+(C-v)2^{a+b},x,k,\mu,t)\|x_{i+(C-v)2^{a+b}}$ for $0\le v\le C-1$ and $y^2_C=r_0\|r_1\|r_3\|x_i$.
   c. Run $Mdef_{seq}(C,y^2,k,r_2,\psi'')$ to get an output $\mu=[a+b+1,a+b+c]$.
   d. Set $\mu=\mu[t]=\mu'[a+b]\|\mu[a+b+1,a+b+c]\|\mu[a+b+c+1,t]$, where $\mu'[a+b]$ is the $m(a+b)$-bit zero string.
3. a. Let $B=2^b-1$. Let $\psi''':[1,B]\rightarrow[1,t]$ be a masking assignment such that $\psi'''(j)=\phi_t(e_{i+(B+1-j)2^a})$ where $j\in[1,B]$.
   b. Let $y^1=y^1_0\|y^1_1\|\ldots\|y^1_B$ where, $y^1_v=s^2(i+(B-v)2^a,x,k,\mu,t)\|s^3(i+(B-v)2^a,x,k,\mu,t)\|x_{i+(B-v)2^a}$ for $0\le v\le B-1$ and $y^1_B=r_0\|r_2\|r_3\|x_i$.
   c. Run $Mdef_{seq}(B,y^1,k,r_1,\psi''')$ to get an output $\mu=[a+1,a+b]$.
   d. Set $\mu=\mu[t]=\mu'[a]\|\mu[a+1,a+b]\|\mu[a+b+1,a+b+c]\|\mu[a+b+c+1,t]$, where $\mu'[a]$ is the ma-bit zero string.
4. a. Let $u=2^a-i$ and $A=2^a-1$. Let $\psi'''':[1,A]\rightarrow[1,t]$ be a masking

assignment such that $\psi'''(j) = \phi_t(e_{A+2\ j})$ where $j \in [1, A]$.

b. Let $y^0 = y_0^0 || y_1^0 || \ldots || y_A^0$ where, $y_v^0 = s^1(A+1-v, x, k, \mu, t) || s^2(A+1-v, x, k, \mu, t) || s^3(A+1-v, x, k, \mu, t) || x_{A+1-v}$ for $0 \le v \le A-1$ and $y_A^0 = 1^{3m} || x_1$.

c. Run $Mdef_{seq}(u, y^0, k, r_0, \psi'''')$ to get an output $\mu = [a]$.

5. Output $\mu[t] = \mu[a] \,||\, \mu[a+1, a+b] ||\, \mu[a+b+1, a+b+c] ||\, \mu[a+b+c+1, t]$

It is easy to check that $s^j(i, x, k, \mu, t) = r_j$ for $0 \le j \le 3$. Therefore $Mdef_{4\dim}$ is correct for $1 \le i \le 2^a - 1$. If r_0, r_1, r_2 and r_3 are random strings then so is the output μ and hence ϕ_t is totally correct for $1 \le i \le 2^a - 1$.

The other cases are very similar. So we omit the proof for these cases.

The following theorem shows that if $\{h_k\}_{k \in K}$ is a UOWHF, then so is $\{H_p\}_{p \in P}$. Using the fact that ϕ_t is totally correct, we can prove this theorem in a much similar way given in [7]. So we omit this proof.

Theorem 2. (Validness of domain extension)

In case of 4-dimensional domain extension a totally correct masking assignment is always valid. More precisely, if there is an (ε, η)-extended strategy for $\{H_p\}_{p \in P}$ then there is an $(\frac{\varepsilon}{2^t}, \eta + 2^{t+1})$-strategy for $\{h_k\}_{k \in K}$ whenever $\{H_p\}_{p \in P}$ is based on a totally correct masking assignment.

We now show the speed-up of 4-dimensional construction over the sequential construction. For the sake of simplicity we do not describe the case of $t \not\equiv 0 \bmod 4$.

Theorem 3. The speed-up of 4-dimensional construction over the sequential construction in Section~\ref{sec-seq} is by a factor of $\frac{2^t}{2^{2+t/4}-3}$ if $t \equiv 0 \bmod 4$.

Proof. 4-dimensional construction hashes a message of length $n2^t - m(2^t - 1)$ into a digest of length m using $2^a + 2^b + 2^c + 2^d - 3$ parallel rounds. Therefore, if $t \equiv 0 \bmod 4$ then $4 \times 2^{t/4} - 3$ parallel rounds are need to hash a message of length $n2^t - m(2^t - 1)$. 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 requre 2^t invocations of the hash function h_k on a message of length $n2^t - m(2^t - 1)$. Hence, the speed-up of the 4-dimensional construction over the sequential construction is by a factor of $\frac{2^t}{2^{2+t/4}-3}$ if $t \equiv 0 \bmod 4$.

By the definition of the masking assignment of 4-dimensional construction, the following theorem is clear.

Theorem 4. The number of masks for 4-dimensional construction is t.

4.2 Optimality of the 4-Dimensional Domain Extender

In [4] Mironov proved that among all the sequential algorithms Shoup's algorithm reuses the masks as much as possible. This means that among all the sequential algorithms there is no algorithm which has a more smaller key expansion than Shoup's algorithm.

In [7] Sarkar provided a generic lower

bound on the key length expansion required for securely extending the domain of a UOWHF. He first defined the large class Λ of "natural" domain extending algorithms. Then he proved that for any $A \in \Lambda$ such that A is correct for s invocations of h_k the number of masks required by A is at least $\lceil \log_2 s \rceil$. (Details can be found in section 4 of [7].) Note that Shoup's algorithm is an element of the class Λ. Therefore, it follows that Shoup's algorithm is optimal for the class Λ.

On the other hand the 4-dimensional domain extender is also an element of the class Λ. And note that for 2^t invocations of h_k the 4-dimensional domain extender uses $t(=\lceil \log_2 2^t \rceil)$ masks to securely extend the domain of a UOWHF. Hence this shows that the 4-dimensional domain extender is also optimal for the class Λ.

4.3 l-Dimensional Domain Extender

In the above we provided the 4-dimensional domain extender and considered the security and optimality of key length expansion. In fact the construction idea can be generalized to any l-dimensional domain extender ($l \geq 2$). If $n \geq lm$, we can define the l-dimensional domain extender. We can start to define the l-dimensional domain extender with setting the function $g(t) = (a_1, \ldots, a_l)$ exactly in the similar way as we did for 4-dimensional. And the whole specification of l-dimensional domain extender can be similarly defined by using the description method of the 4-dimensional domain extender. We can also consider the security and optimality of the l-dimensional domain extender as in the case of 4-dimensional domain extender.

4.4 Comparison to Known Algorithms

In Table 1 we compare the specific performance of the different known algorithms with l-dimensional domain extender. Note that the message length which can be handled varies with each of the known algorithms. For example, Shoup's and l-DIM can handle a $2^t n-(2^t-1)m$ bits message, however, Sakar's can not handle the same length message. Therefore, we can not fix a message length in order to compare the different known algorithms with l-DIM. Instead, we separately describe the message length for each of the algorithms as shown in Table 1.

The algorithms use one key for the base hash function and some number of m-bit mask keys. The number of masks described in Table 1 refers to the latter. The number of invocations of h_k is the total cost. The number of rounds reflects the parallelizability arising via tree-based constructions, and indicates the total time to completion. In Shoup's sequential construction it is equal to the number of invocations of h_k. Speed-up (over the sequential algorithm or Shoup) is the ratio of the number of invocations of h_k to that of rounds. For the sake of simplicity we do not describe the case of $t \not\equiv 0 \bmod l$ in the positions of the number of rounds and speed for our l-DIM.

Table 1 shows the key length expansion of l-DIM is the same as that of Shoup's and it doesn't have the same parallalizable performance with Sarkar's construction. But if l is getting larger, then the speed of the l-DIM is also getting near to the speed of Sarkar.

V. Conclusion

In this paper we have provided the new parallel domain extender l-DIM for UOWHF. It has an important theoretical meaning in the study of efficient domain extanding method for UOWHF. l-DIM has the same key length expansion as Shoup's. Furthermore, l-DIM and Shoup's construction are the minimum possible for any algorithms in a large class of "natural" domain extenders including all the previously proposed constructions. But l-DIM does not have the same parallelizability performance as complete l-ary ($l \geq 2$) tree based constructions. However, if l is getting larger, then the speed of the l-DIM is also getting near to the speed of Sarkar.

References

[1] M. Bellare and P. Rogaway, "Collision-resistant hashing: towards making UOWHFs practical," Advances in Cryptology - Crypto'97, Lecture Notes in Computer Science, Vol. 1294, Springer-Verlag, pp. 470-484, 1997.

[2] I. B. Damgard, "A design principle for hash functions," Advances in Cryptology - Crypto'89, Lecture Notes in Computer Sciences, Vol. 435, Springer-Verlag, pp. 416-427, 1989.

[3] R. Merkle, "One way hash functions and DES," Advances in Cryptology - Crypto'89, Lecture Notes in Computer Sciences, Vol. 435, Springer-Verlag, pp. 428-446, 1989.

[4] I. Mironov, "Hash functions: from Merkle-Damgard to Shoup," Advances in Cryptology - Eurocrypt'01, Lecture Notes in Computer Science, Vol. 2045, Springer-Verlag, pp 166-181, 2001

[5] M. Nandi, "A New Tree based Domain Extension of UOWHF," Cryptology ePrint Archive, http://eprint.iacr.org/2003/142.

[6] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, ACM Press, pp 33-43, 1989.

[7] P. Sarkar, "Construction of UOWHF: Tree Hashing Revisited," Cryptology ePrint Archive, http://eprint.iacr.org/2002 /058.

[8] P. Sarkar, "Domain Extenders for UOWHF: A Generic Lower Bound on Key Expansion and a Finite Binary Tree Algorithm," Cryptology ePrint Archive, http://eprint.iacr.org/2003/009.

[9] V. Shoup, "A composition theorem for universal one-way hash functions," Advances in Cryptology - Eurocrypt'00, Lecture Notes in Computer Science, Vol. 1807, Springer-Verlag, pp 445-452, 2000.

[10] D. Simon, "Finding collisions on a one-way street: can secure hash functions be based on general assumptions?," Advances in Cryptology - Eurocrypt'98, Lecture Notes in Computer Science, Vol. 1403, Springer-Verlag, pp 334-345, 1998.

〈著者紹介〉

이 원 일 (Wonil Lee) 정회원
1998년 2월: 고려대학교 수학과 학사
2000년 2월: 고려대학교 수학과 석사
2003년 8월: 고려대학교 수학과 박사
2000년 8월~현재: 고려대학교 정보보호기술연구센터 연구원
〈관심분야〉 해쉬 함수, 블록 암호, 스트림 암호, 암호 프로토콜

장 동 훈 (Donghoon Chang) 정회원
2001년 2월: 고려대학교 수학과 학사
2003년 2월: 고려대학교 정보보호대학원 석사
2003년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 암호이론, 해쉬 함수 및 블록 암호 설계 및 분석