'UOWHF에 대한 다차원 구성 방법'에 대한 고찰: 유한개의 프로세서를 사용한 경우

장동훈* 이원일* 이상진* 성수학**

*고려대학교 정보보호대학원 **배재대학교 응용수학과

An Inquiry into 'Multi-Dimensional Construction for UOWHF': In case of using Finite Processors

Donghoon Chang* Wonil Lee* Sangjin Lee* Soohak Sung**

*Center for Information Security Technologies(CIST), Korea University

**Applied Mathematics, Paichai University

요 약

지금까지 여러 암호학자들에 의해 UOWHF에 대한 구성 방법들, 즉 BLH[1], XLH[1], BTH[1], XTH[1], Shoup 구성 방법[8], Sarkar 구성 방법[5], 다차원 구성 방법[2] 등이 제안되었다. 이중에 BLH, XLH, Shoup 구성 방법은 오직 하나의 프로세서를 이용한다. 반면 BTH, XTH, Sarkar의 구성 방법, 다차원 구성 방법은 병렬 처리 구성 방법으로 처리 속도 측면에서 효율적인 구성 방법들이다. 하지만 BTH, XTH, Sarkar의 구성 방법, 다차원 구성 방법은 입력 메시지의 길이에 따라 필요한 프로세서의 수와 메모리 크기의 증가를 필요로 한다. Sarkar는 [6]에서 유한개의 프로세서와 한정된 메모리만 갖고서도 병렬처리할 수 있는 구성 방법(PUA)을 처음으로 제안하였다. 하지만 [6]에서 제안된 구성 방법 PUA는 키 확장 길이 측면에서 Shoup의 구성 방법에 비해 비효율적이다. 본 논문에서는 유한개의 프로세서와 한정된 메모리를 갖고서도 병렬처리(parallel processing)할 수 있으며, 동시에 키 확장 길이 측면에서 Shoup 구성 방법과 동일한 '유한개의 프로세서를 사용한 다차원 구성 방법'을 처음으로 제안한다.

I. 서론

1. UOWHF의 소개

Naor 와 Yung에 의해 소개된 UOWHF[4]는 일반적으로 널리 알려진 충돌 저항 해쉬 함수(Collision resistant hash function, 이하 CRHF라 표기함) 보다 더 약한 암호학적 기본 요소이다.

CRHF의 경우: 어떤 고정된 해쉬 함수에 대한 충돌쌍을 찾아야 하는 것이 공격자의 목표이다.

UOWHF의 경우: 공격자는 먼저 어떤 메시지(입력값)를 자신이 선택하여 내놓아야만 한다. 이 과정이 끝난 후에 특정한 해쉬 함수가 공격자에게 주어지게 된다. 그러면 이 해쉬 함수에 대하여, 먼저 내놓은 메시지와 충돌을 발생시키는 다른 메시지를 찾아야 하는 것이 공격자의 목표이다.

이와 같이, UOWHF에 대한 공격자는 자신이 공격할 구체적인 대상이 되는 해쉬 함수가 정해지기 전에 메시지를 선택하여 내놓아야만 하므로 공격자의 공격 과정이 CRHF에 대한 공격자보다 더욱 어렵게 된다. 따라서 UOWHF가 CRHF보다 더욱 약한 암호학적 기본요소가 되는 것이다. UOWHF가 CRHF의 대안으로써 크게 주목받고 있는 이유는 다음과 같다.

- UOWHF를 구성하는 것이 CRHF를 구성하는 것보다 훨씬 쉬워 보인다는 것이다. 이것은 MD4와

SHA-0, PKC'98에서 제안된 해쉬 함수 등의 (CRHF 관점에서)공격당한 예를 통하여 알 수 있다.

- 대부분의 증명 가능한 전자 서명을 구성하는 데 있어서 가장 중요한 역할을 하는 해쉬 함수가 UOWHF이기만 해도 전자 서명의 안전성을 보장할 수 있다는 사실이다.[1]

그렇다면 임의의 길이의 메시지들을 해쉬하기 위한 UOWHF를 어떻게 하면 효율적으로 구성 할 수 있을 것인가? 이에 대한 가장 합리적인 접근 방법은 고정된 길이를 해쉬하는 압축 함수 (Compression function)를 먼저 구성하고 그 후 이 압축 함수를 이용하여 임의의 길이의 메시지를 해쉬 할 수 있는 확장된 해쉬 함수(Extended hash function)를 구성하는 것이다. 본 논문에서는 UOWHF인 압축 함수가 존재한다는 가정 하에 이 압축 함수를 이용하여 UOWHF인 확장된 해쉬 함수를 구성하는 데에 관심이 있다. 이렇게 확장시키는 방법을 표현하는 용어로써 우리는 앞으로 '구성 방법(construction method)'이라는 용어를 사용하도록 한다.(참고로 어떤 논문은 합성법(composite scheme)이라는 용어를 사용한다.)

확장된 해쉬 함수의 구성 방법에 관한 연구는 대표적으로 다음 두 가지에 의하여 진행되어 왔다.

- 이제까지 제안된 확장된 해쉬 함수를 구성하는 방법들은 키 확장 길이의 증가를 필요로 한다는 것이다. 따라서 이러한 키 확장 길이를 낮출 수 있는 구성 방법을 찾기 위한 연구에 초점이 맞추어져 왔다.
- 둘째는 해쉬 값을 얻기 위하여 필요한 처리 속도를 병렬 처리(parallel process)를 통하여 빠르게 하는 방법에 대한 연구이다.

다음절에서는 지금까지 제안된 구성 방법들을 소개한다.

2. UOWHF 구성 방법 소개

· Bellare-Rogaway 구성 방법[1]: CRHF인 확장된 해쉬 함수를 구성하기 위한 방법으로써 Merkle-Damgard 구성 방법은 매우 널리 잘 알려진 방법이다. 그러나 Bellare와 Rogaway는 이러한 Merkle-Damgard 구성 방법이 UOWHF의 경우에는 적용되지 않음을 반례를 제시하여 보였다. 또한 그들은 BLH(Basic linear hash scheme), XLH(XOR linear hash scheme), BTH(Basic Tree hash scheme), XTH(XOR Tree hash scheme)와 같은 새로운 구성 방법들을 제시하고 이들에 관한 안전성과 효율성을 분석하였다.(BLH와 XLH는 메시지 길이와 상관없이 오직 하나의 프로세서를 이용한다. 그런데 Bellare와 Rogaway는 이들의 논문에서 병렬처리(parallel processing)이라는 용어를 쓰지 않는다. 따라서 본 논문에서는 구성 방법 간의 처리 속도를 비교하기 위해 BTH와 XTH는 메시지 길이가 커짐에 따라 필요한 프로세서의 수도 증가하는 것으로 가정하자. 여기서 '프로세서의 수'란 최대로 동시에 처리할 수 있는 '압축 함수의 수'를 가리킨다.)

· Shoup 구성 방법[8]: Shoup은 Eurocrypt 2000에서 Bellare와 Rogaway가 제시한 XLH와 거의 유사한 방법을 제시하였다.(Shoup 구성 방법은 메시지 길이와 상관없이 오직 하나의 프로세서만 사용한다) XLH와 다른 점은 키 확장 길이를 가능한 최대로 줄이기 위하여 마스크 키들을 중복하여 사용했다는 점이다. 이제부터 그의 구성 방법을 Shoup 구성 방법이라 부르기로 하자. Eurocrypt 2001에서 Milonov는, Shoup 보다 더욱 효율적인 증명 방법을 통하여 Shoup 구성 방법에 대한 안전성을 증명하였다.[3] 게다가 Milonov는 Shoup 구성 방법이 키 확장 길이의 관점에서 (그가 제시한 구성 방법이 취한 구조(uniary tree) 안에서는) 가장 최적화된 방법임을 증명하였다. 현재까지도 Shoup 구성 방법은 키 길이 확장의 측면에서 가장 효율적인 방법이 되고 있다. 그러나 Milonov도 언급했다시피 Shoup 구성 방법의 최적합성(optimality)이 더 적은 키 확장 길이를 요구하는 (uniary tree가 아닌) 구성 방법의 존재 가능성을 소멸시키는 것은 아니다. 또한 Sarkar가 [5]에서 언급했듯이 Shoup 구성 방법 안에 쓰이는 초기값 IV을 메시지 블록으로 대치시키고 IV와 XOR되는 첫 번째 마스크 키도 생략할 수 있음을 알 수 있다. 본 논문에서는 앞으로, 초기값 IV와 첫 번째 압축 함수의 입력 값에 적용된 마스크 키를 생략한 경우의 Shoup 구성 방법만을 고려할 것이다.

· Sarkar 구성 방법[5]: Sarkar는 자신이 제안한 구성 방법을 완전 이진 트리(Full binary tree)를 이용하여 표현하였다. 그는 Shoup이 사용한 마스크 키 대응 방법과 XTH에서 Bellare와 Rogaway가 사용한 마스크 키 대응 방법을 동시에 자신의 구성 방법에 적용하였다. 이러한 그의 구성 방법을 Sarkar 구성 방법이라 부르기로 하자. (Sarkar의 구성 방법은 메시지 길이가 커짐에 따라 필요한 프로세서의 수와 메모리 크기가 증가한다) Sarkar 구성 방법은 XTH 중에서 완전 이진 트리로 표현되는 구성 방법보다 더욱 적은 키 확장 길이를 필요로 한다. 그러나 Shoup 구성 방법보다는 약간 더 많은 키 확장 길이가 필요하다. 처리 속도 관점에서 볼 때는, Sarkar 구성 방법은 XTH 중에서 완전 이진 트리로 표현되는 구성 방법과 같고 Shoup 구성 방법보다는 매우 효율적이다. Sarkar는 그의 구성 방법이 안전하기 위하여 필요한 키 확장 길이의 하한 값(lower bound)은 제시하였지만 그가 사용한 마스크 키 대응 함수에 의한 키 확장 길이가 이 하한 값과 일치한다는 것을 보이는 데에는 실패하였다(단, 작은 길이의 메시지에 대해서는 하한 값과 일치한다). 따라서 일반적으로 말하면 Sarkar 구성 방법이 요구하는 키 확장 길이는 최적이라고 말할 수 없다. 게다가 Sarkar는 XTH와는 다르게 자신의 구성 방법을 완전 이진 트리인 경우에만 국한시켜 제시하였으며 그의 구성 아이디어가, 임의의 $l>2$ 에 대한 완전 l-ary 트리(Full l-ary tree)인 경우에 적용되기는 어렵게 보인다.

· 다차원 구성 방법[2]: 이원일, 장동훈은 'Shoup 구성 방법'과 동일한 키 확장 길이를 필요로 하면서, 동시에 처리 속도 관점에서 'Shoup 구성 방법'보다 효율적인 구성 방법을 처음으로 제안하였고 이를 '다차원 구성 방법'이라 명칭하였다. (다차원 구성 방법은 메시지 길이가 커짐에 따라 필요한 프로세서의 수와 메모리 크기가 증가한다. 여기서 차원을 나타내는 수인 l 은 압축 함수 $H: \Sigma^k \times \Sigma^m \rightarrow \Sigma^m$의 입력 메시지의 길이 n과 출력 길이 m에 다음과 같이 의존한다. 만일 $n \geq lm$ 이 성립하면 l 차원 구성 방법이 정의될 수 있다.) 동시에 이에 관한 안전성과 키 확장 길이의 최적합성을 증명하였다. 하지만 처리 속도 측면에서 보았을 때 XTH와 Sarkar의 구성 방법보다는 비효율적이다.

[표 1]은 구성 방법들 간에 필요한 마스크 키의 개수와 처리 속도 등을 비교해 놓았다. 여기서 처리 속도는 매 단계당 사용되는 압축 함수의 평균 횟수를 의미한다.

[표 1] 구성 방법들 비교

	개선된 Shoup 구성방법	l-차원구성 방법($l>1$)	Sarkar 구성 방법	XTH
순차/병렬 처리	순차	병렬	병렬	병렬
메시지길이	$2^t n-(2^t-1)m$	$2^t n-(2^t-1)m$	$(2^t-1)n-(2^t-2)m$	$(2^t-1)n-(2^t-2)m$
압축함수 사용 횟수	2^t	2^t	2^t-1	2^t-1
마스크 수	t	t	$t+\lceil \log_2 t \rceil -1$	$2(t-1)$
처리단계 수	2^t	$\frac{t2^{t/l}-l+1}{(t \equiv 0 \bmod l)}$	t	t
처리속도	1	$\frac{2^t}{t2^{t/l}-l+1}$ $(t \equiv 0 \bmod l)$	$\frac{2^t-1}{t}$	$\frac{2^t-1}{t}$

· PUA(Parallel UOWHF Algorithm)[6]: 지금까지 제안된 병렬 처리 가능한 UOWHF 구성 방법들인 XTH, Sarkar 구성 방법, 다차원 구성 방법은 메시지 길이가 커짐에 따라 필요한 프로세서의 수와 메모리 크기가 함께 증가한다. 이와 같이 프로세서의 수와 메모리 크기에 제한을 두지 않는 가정은 비현실적이다. Sarkar는 이를 인식하고 유한개의 프로세서와 한정된 메모리를 이용하여 병렬 처리가 가능한 'PUA'라는 UOWHF 구성 방법을 처음으로 제안하였다. 이는 앞에 소개된 Sarkar의 구성 방법[5]과 [7]에 소개된 아이디어를 접목시킨 새로운 구성 방법이다. 하지만 키 확장 길이 측면에서 앞서 제안한 Sarkar의 구성 방법에 비해 더 많은 키 확장 길이를 필요로 한다는 점에서 비효율적이다.

본 논문에서는 유한개의 프로세서와 한정된 메모리만 갖고서도 병렬처리(parallel processing)할 수 있으며, 동시에 키 확장 길이 측면에서 Shoup의 구성 방법과 동일한 '유한개의 프로세서를 사용한 다차원 구성 방법'을 처음으로 제안한다.

II. 유한개의 프로세서를 사용한 다차원 구성 방법

본 논문에서 제안하고자 하는 '유한개의 프로세서를 사용한 다차원 구성 방법'은 [2]에서 제안된 '다차원 구성 방법'을 유한개의 프로세서를 사용한 경우로 수정한 구성 방법이다. 본 논문에는 '유한개의 프로세서를 사용한 4차원 구성 방법'을 먼저 제시하고 이를 '유한개의 프로세서를 사용한 다차원 구성 방법'으로 일반화하고자 한다.

1. '유한개의 프로세서'를 사용한 4차원 구성 방법

2^t개의 프로세서와 $2^t n-(2^t-1)m$ 길이의 메시지가 주어졌다고 하자.($t, t \in Z^+$) 이때 '2^t개의 프로세서를 사용한 4차원 구성 방법'의 아이디어는 [2]에서 제시된 '4차원 구성 방법'에서 사용된 g 함수를 g_t로 수정하는 것이다. 즉, 정의 2와 같이 g_t 함수를 정의하고 $g(t)=g_t(t)$를 4차원 구성 방법에 적용하는 것이 '2^t개의 프로세서를 사용한 4차원 구성 방법'인 것이다.

정의 1. 함수 $g: Z^+ \rightarrow Z^+ \times Z^+ \times Z^+ \times Z^+$ 를 다음과 같이 정의하자. 즉, 임의의 $t \in Z^+$에 대하여

$$g(t)=\begin{cases}(i,i,i,i) & \text{if } t=4i\\ (i+1,i,i,i) & \text{if } t=4i+1\\ (i+1,i+1,i,i) & \text{if } t=4i+2\\ (i+1,i+1,i+1,i) & \text{if } t=4i+3\end{cases}$$

이다. g를 초기화 함수(initialize function)라고 하며, $g(t)=(a,b,c,d)$ 이면 항상 $t=a+b+c+d$ 가 성립한다.

정리 1. 메시지 x의 길이가 $|x|=2^t n-(2^t-1)m$ 이고 $g(t)=(a,b,c,d)$이면 메시지 x를 4차원 구성 방법으로 해쉬하는데 필요한 프로세서의 수는 2^{a+b+c}이다.

정의 1과 정리 1를 이용하여 다음과 같이 함수 $g_t(t)$를 정의한다.

정의 2. 함수 $g_t: Z^+ \rightarrow Z^+ \times Z^+ \times Z^+ \times Z^+$ 를 다음과 같이 정의하자. 즉, 임의의 $t', t \in Z^+$에 대하여

$$\text{if } a+b+c \le t' \text{ s.t. } g(t)=(a,b,c,d)$$

$$\text{then } g_{t'}(t)=g(t)$$

$$\text{if } a+b+c > t' \text{ s.t. } g(t)=(a,b,c,d)$$

$$\text{then } g_{t'}(t)=\begin{cases}(1,t-1,0,0) & \text{if } t'=1\\ (1,1,t-2,0) & \text{if } t'=2\\ (i,i,i,t-3i) & \text{if } t'=3i,\ i>0\\ (i+1,i,i,t-3i-1) & \text{if } t'=3i+1,\ i>0\\ (i+1,i+1,i,t-3i-2) & \text{if } t'=3i+2,\ i>0\end{cases}$$

이다. 여기서 $g_t(t)=(a,b,c,d)$ 이면 항상 $t=a+b+c+d$ 가 성립한다.

본 논문에서 제안하는 '2^t개의 프로세서를 사용한 4차원 구성 방법'의 안전성과 키 확장 길이의 최적합성에 대한 증명은 [2]에서 제안된 '4차원 구성 방법'의 안전성과 키 확장 길이의 최적합성에 대한 증명과 동일하기 때문에 본 논문에서는 생략한다.

2. '유한개의 프로세서'를 사용한 다차원 구성 방법

실제로 '2^t개의 프로세서를 사용한 4차원 구성 방법'에 대한 아이디어는 임의의 l 차원에도 적용될 수 있다. 이 때 정의 2와 비슷하게 초기화 함수 $g_t(t)=(a_1,\ldots,a_l)$를 정의하고, $g(t)=g_t(t)$을 [2]에서 제시된 '다차원 구성 방법'에 적용하면 된다. 또한 '2^t개의 프로세서를 사용한 l 차원 구성 방법'의 안전성 및 키 확장 길이의 최적합성도 증명할 수 있다.

3. '유한개의 프로세서를 사용한 다차원 구성 방법'과 'PUA'의 비교 분석

[표 2]는 최대 2^t개의 프로세서의 사용이 가능할 때 '2^t개의 프로세서를 사용한 다차원 구성 방법'과 PUA[6] 사이의 필요한 마스크 키의 개수와 처리 속도 등을 비교해 놓았다. 여기서 처리 속도는 매 단계당 사용되는 압축 함수의 평균 횟수를 의미한다. [표 2]에서 t'는 $t'=(l-1)i,\ i>0$ 이고 $a+b+c>t'$ s.t. $g(t)=(a,b,c,d)$ 라고 가정한다. '2^t개의 프로세서를 사용한 다차원 구성 방법'에 적용된 메시지의 길이는 $2^t n-(2^t-1)m$이고 'PUA'에 적용된 메시지의 길이는 $(2^t-1)n-(2^t-2)m$ 이라고 가정한다.

[표 2] 본 논문의 구성 방법과 PUA의 비교

	2^t 개의 프로세서를 사용한 l-차원 구성 방법	PUA[6]
프로세서의 수	2^t	2^t
메시지 길이	$2^t n-(2^t-1)m$	$(2^t-1)n-(2^t-2)m$
압축함수 사용 횟수	2^t	2^t-1
마스크 수	t	$\simeq t+2+\lfloor \log_2(t-1) \rfloor$
처리속도	$\frac{2^t}{(l-1)2^{t/(l-1)}+2^{t-t}-l+1}$	$\simeq \frac{2^t-1}{1+2^{t-(t+2)}}$

다음의 [표 3]은 t=1,2,3,4,5 일 때, '2^t개의 프로세서를 사용한 4차원 구성 방법'과 PUA를 비교해 놓았다. $a+b+c>t$ s.t. $g(t)=(a,b,c,d)$ 라고 가정한다. 또한 '2^t개의 프로세서를 사용한 다차원 구성 방법'에 적용된 메시지의 길이는 $2^t n-(2^t-1)m$ 이고 'PUA'에 적용된 메시지의 길이는 $(2^t-1)n-(2^t-2)m$ 라고 가정한다.

[표 3] 프로세서의 수에 따른 비교

t	2^t 개의 프로세서를 사용한 4차원 구성 방법	PUA[6]
1	$\frac{2^t}{\frac{1}{2}2^t+1}$	$\simeq \frac{2^t}{2^t+6}$
2	$\frac{2^t}{\frac{1}{4}2^t+2}$	$\simeq \frac{2^t}{\frac{1}{3}2^t+\frac{16}{3}}$
3	$\frac{2^t}{\frac{1}{8}2^t+3}$	$\simeq \frac{2^t}{\frac{1}{7}2^t+\frac{40}{7}}$
4	$\frac{2^t}{\frac{1}{16}2^t+5}$	$\simeq \frac{2^t}{\frac{1}{15}2^t+\frac{32}{5}}$
5	$\frac{2^t}{\frac{1}{32}2^t+7}$	$\simeq \frac{2^t}{\frac{1}{31}2^t+\frac{224}{31}}$
⋮	⋮	⋮

[표 3]은 '2^t개의 프로세서를 사용한 4차원 구성 방법'과 PUA의 각각에 $2^t n-(2^t-1)m$과 $(2^t-1)n-(2^t-2)m$ 길이의 메시지가 입력될 때, 처리 속도를 비교해 놓은 것이다.

참고문헌

[1] M. Bellare, and P. Rogaway, "Collision -resistent hashing: towards making UOWHFs practical." , *Proceedings of CRYPTO'97*, pp 470-484, 1997.

[2] 이원일, 장동훈, "UOWHF에 대한 다차원 구성 방법", 제 5회 정보보호 우수논문집, 한국 정보보호진흥원, 2002, pp. 5-38.

[3] I. Mironov. "Hash functions: from Merkle-Damgard to Shoup." , *Proceedings of EU -ROCRYPT 2001*, pp 166-181, 2001.

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

[5] P. Sarkar, "Construction of UOWHF: Tree Hashing Revisited" ,*Cryptology ePrint*, 2002.

[6] P. Sarkar, "Domain Extenders for UOWHF: A Generic Lower Bound on Key Expansion and a Finite Binary Tree Algorithm", *Crypt -ology ePrint*, 2003.

[7] P. Sarkar, Paul J. Schellenberg "A Paralleliz -able Design Principle for Cryptographic Hash Functions", *Cryptology ePrint*, 2002.

[8] V. Shoup. "A composition theorem for univ -ersal one-way hash functions." , *Proceedin -gs of EUROCRYPT 2000*, pp 445-452,2000.