.

New Parallel Domain Extenders of UOWHF

W. Lee, D. Chang, S. Lee, S. Sung
Center for Information Security Technologies
and
M. Nandi
Indian Statistical Institute

Hash Function :

$\{h_k\}_{k\in\mathcal{K}}$: (n, m, K) keyed family of hash functions. For each $|k| = K$
$h_k : \{0,1\}^n \to \{0,1\}^m, n > m.$

• Applications : Digital Signature Schemes, Public Encryption Schemes etc.

• Requirement : Collision Resistant (i.e. hard to find $x \neq x'$ but $h_k(x) = h_k(x')$) or Target Collision Resistant .

Collision Resistant Hash Function (CRHF) :

$\{h_k\}_{k\in\mathcal{K}}$: (n, m, K) keyed family of hash functions. Consider the following adversarial game.

1. Adversary is given a k which is chosen uniformly at random from $\mathcal{K}$.
2. Adversary has to find x and x' such that $x \neq x'$ but $h_k(x) = h_k(x')$.

$\{h_k\}_{k\in\mathcal{K}}$ is a **CRHF** if the adversary has a negligible probability of success with respect to any probabilistic polynomial time strategy.

Universal One-Way Hash Function (UOWHF):

- Introduced by Naor and Yung in 1989.

$\{h_k\}_{k \in \mathcal{K}}$: (n, m, K) keyed family of hash functions. Consider the following adversarial game.

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 $x \neq x'$ but $h_k(x) = h_k(x')$.

$\{h_k\}_{k \in \mathcal{K}}$ is a **UOWHF** if the adversary has a negligible probability of success with respect to any probabilistic polynomial time strategy.

Why UOWHF ?

1. It is weaker primitive than CRHF.
2. In building digital signature scheme, a UOWHF is sufficient.
3. Birthday attack will not work for UOWHF.
4. Simon had shown that there is an oracle relative to which UOWHFs exist but CRHF does not exist.

Domain Extension

Given (n, m, K) *base* hash family $\{h_k\}_{k \in \mathcal{K}}$, construct (N, m, P) *extended* hash family $\{H_p\}_{p \in \mathcal{P}}$ where, $N = (n - m)r + m$, $P = K + m.l$.

• Key Expansion : $(P - K) = m$, l = Number of masks

• **Valid domain extension** : $\{H_p\}_{p \in \mathcal{P}}$ is UOWHF whenever $\{h_k\}_{k \in \mathcal{K}}$ is UOWHF.

• **Aim** : to construct a valid domain extension such that :

1. Minimizing the key length expansion (reducing the space complexity).
2. Parallel implementation (reducing the time complexity)

Sequential Domain Extension Algorithm

- $[a,b] = \{a, a+1, \ldots, b\}$.
- $\psi : [1, r-1] \to [1, l]$ - **masking assignment**.

$y = y_1||\ldots||y_r$ and $p = k||\mu_1||\mu_2||\ldots||\mu_l$ where, $|y_1| = n, |y_2| = \ldots = |y_r| = (n-m), |k| = K$ and $|\mu_i| = m$.

Computation of $H_p(y)$:

1. **Input:** y and p
2. $z_1 = h_k(y_1)$.
3. For $2 \leq i \leq r$, $z_i = h_k((z_{i-1} \oplus \mu_{\psi(i)})||y_i)$.
4. **Output:** $z_r = H_p(y)$.

Sequential Domain Extension

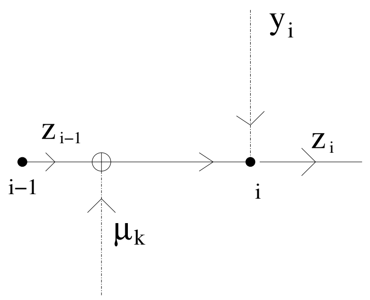

Figure 1:

The output of $(i-1)$-node XOR-ed by a mask is fed into the input of i^{th} node.

$\psi(i-1) = k$ means that the mask μ_k is XOR-ed with output of $(i-1)$-node.

- Valid masking assignment.

(strongly) even-free masking assignment

For each $[a, b] \subseteq [1, r-1]$ there exists $c \in [a, b]$ such that $\psi(c)$ occurs exactly odd many times (or once) in the sequence $\psi(a), \psi(a+1), \ldots, \psi(b)$.

• Mironov proved that valid masking assignment $\Rightarrow$ even-free masking assignment.

Theorem 1 ***(Sufficient Condition for Valid Sequential Extension)***
strongly even-free masking assignment $\Rightarrow$ valid masking assignment.

• Shoup's masking assignment : $\psi(i) = \nu_2(i)$ $(= j$ means that $2^j | i$ but $2^{j+1} \not| i)$.

Binary Tree Based Construction

• $T_t = (V_t, E_t)$: full binary tree where $V_t = [1, 2^t - 1]$, $E_t = \{(i, \lfloor i/2 \rfloor)\}$. At each node i there is a processor P_i which is capable of computing base hash function h_k.

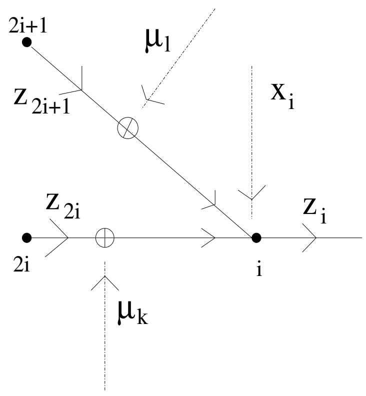

Figure 2: The output of $2i^{th}$ and $(2i+1)$ node XOR-ed by masks are fed into the input of i^{th} node.

• *masking assignment :* $\psi : [2, 2^t - 1] \to [1, l]$. $\psi(i) = j$ means that output of the node i is XOR-ed by the mask μ_j. The output of node 1 is the final output.

• Note, number of rounds $= t$

Previous Constructions :(Binary Tree based)

1. **Bellare-Rogaway**: $l = 2(t-1)$ masks.
2. **Sarkar**: $l = t + \lfloor log_2(t-1) \rfloor$ masks.

New Constructions :

1. **IBTC** (Improved Binary Tree Based Construction) : $l = t + O(log^* t)$ many masks.
2. l**-dim** (Incomplete l-ary tree based) : $l = t$ many masks.

Theorem 2 *The New Domain extension algorithms are valid.*

Theorem 3 *The speed-up of IBTC over the sequential algorithm (Shoup's) is by a factor of* $\frac{2^t-1}{t}$.

Theorem 4 *The speed-up of* l*-dim over the sequential algorithm (Shoup's) is by a factor of* $\frac{2^t}{2^{2+t/4}-3}$ *if* $t \equiv 0$ *mod* 4.

Comparison to Known Constructions

Method	Used Tree	Ranking of Key expansion	Ranking of Parallelizability	Seq /Par
BLH [1]	Unary	7	3	seq
XLH [1]	Unary	6	3	seq
Shoup [10]	Unary	1	3	seq
BTH [1]	Full l-ary ($l \geq 2$)	5	1	par
XTH [1]	Full l-ary ($l \geq 2$)	4	1	par
Sarkar [8]	Full Binary	3	1	par
IBTC(this paper)	Full Binary	2	1	par
l-**DIM**(this paper)	Non-Full l-ary ($l \geq 2$)	1	2	par

Table 1: Comparison of domain extenders for UOWHF.

Comparison to Known Constructions

Parameter	Shoup [10]	l-**DIM**$(l \geq 2)$	**IBTC**	Sarkar [8]
seq/par	sequential	parallel	parallel	parallel
message length	$2^t n$ $-(2^t-1)m$	$2^t n$ $-(2^t-1)m$	$(2^t-1)n$ $-(2^t-2)m$	$(2^t-1)n$ $-(2^t-2)m$
# invo. of h_k	2^t	2^t	2^t-1	2^t-1
# masks	t	t	t+ $O(\log_2^* t)$	$t+\lceil \log_2 t \rceil - 1$
# rounds	2^t	$l2^{t/l}-l+1(t \equiv 0 \bmod l)$	t	t
speed-up	1	$\frac{2^t}{l2^{t/l}-l+1}(t \equiv 0 \bmod l)$	$\frac{2^t-1}{t}$	$\frac{2^t-1}{t}$

Table 2: Specific comparison of domain extenders for UOWHF

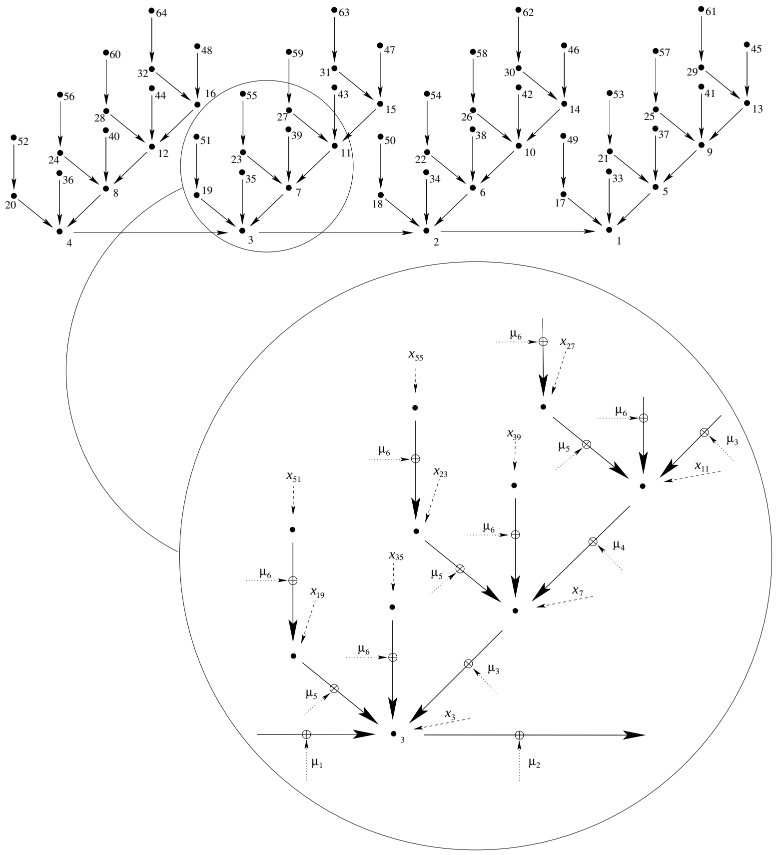

Figure 3: 4-dimensional parallel algorithm. $t = 6$ and $x = x_1||\ldots||x_{2^6}$. Note that $g(6) = (2,2,1,1)$ and $|x_1| = \cdots = |x_3| = n-4m$, $|x_4| = \cdots = |x_{12}| = n-3m$, $|x_{13}| = \cdots = |x_{16}| = n-2m$, $|x_{17}| = \cdots = |x_{32}| = n-m$, and $|x_{33}| = \cdots = |x_{2^6}| = n$.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

l-ary Tree based Domain Extension

Key Size Optimality of 4-Dimensional Domain Extender

- Mironov[4]: Among all the sequential algorithms, Shoup's algorithm reuses the masks as much as possible.
- We can also ask whether the masks can be re-used even more in the 4-dimensional domain extender.

Key Size Optimality of 4-Dimensional Domain Extender

• Sarkar[8]: provided a generic lower bound on the key length expansion required for securely extending the domain of a UOWHF.

- Defined the large class $\mathcal{A}$ of "natural" domain extending algorithms.
- Proved that for any $A \in \mathcal{A}$ such that A is correct for s invocations of h_k the number of masks required by A is at least $\lceil \log_2 s \rceil$.

• Note that Shoup's algorithm is an element of the class $\mathcal{A}$. Therefore, it follows that Shoup's algorithm is optimal for the class $\mathcal{A}$.

Key Size Optimality of 4-Dimensional Domain Extender

- Our 4-dimensional domain extender is also an element of the class $\mathcal{A}$.
  - 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 $\mathcal{A}$.

<u>l-Dimensional Domain Extender</u>

• The above 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 also consider the security and optimality of the l-dimensional domain extender as in the case of 4-dimensional domain extender.

Thank You