Higher Order Universal One-Way Hash Functions from the Subset Sum Assumption

Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang

Dept. of Computing, Macquarie University, North Ryde, Australia
{rons,josef,hwang}@comp.mq.edu.au
http://www.ics.mq.edu.au/acac/

Abstract. Universal One-Way Hash Functions (UOWHFs) may be used in place of collision-resistant functions in many public-key cryptographic applications. At Asiacrypt 2004, Hong, Preneel and Lee introduced the stronger security notion of higher order UOWHFs to allow construction of long-input UOWHFs using the Merkle-Damgård domain extender. However, they did not provide any provably secure constructions for higher order UOWHFs.
We show that the subset sum hash function is a kth order Universal One-Way Hash Function (hashing n bits to $m < n$ bits) under the Subset Sum assumption for $k = O(\log m)$. Therefore we strengthen a previous result of Impagliazzo and Naor, who showed that the subset sum hash function is a UOWHF under the Subset Sum assumption. We believe our result is of theoretical interest; as far as we are aware, it is the first example of a natural and computationally efficient UOWHF which is also a provably secure higher order UOWHF under the same well-known cryptographic assumption, whereas this assumption does not seem sufficient to prove its collision-resistance. A consequence of our result is that one can apply the Merkle-Damgård extender to the subset sum compression function with 'extension factor' $k+1$, while losing (at most) about k bits of UOWHF security relative to the UOWHF security of the compression function. The method also leads to a saving of up to $m \log(k+1)$ bits in key length relative to the Shoup XOR-Mask domain extender applied to the subset sum compression function.
Keywords: hash function, provable security, subset sum

1 Introduction

Motivation. Universal One-Way Hash Functions (UOWHFs), introduced by Naor and Yung [14] (also known as 'Target Collision Resistant' functions), achieve weaker security than collision-resistant hash functions, but still suffice for important cryptographic applications – in particular they suffice for hashing long messages prior to signing with a digital signature scheme [14, 3, 16] (and even can be used to construct digital signature schemes).

A common methodology for designing hash functions consists of two stages. In the first stage, one designs an (efficient) *compression function* f which hashes a (relatively short) n-bit string to a shorter m-bit string (e.g. a compression function may hash a $n = 600$ bit input to a $m = 400$ bit output, compressing by $n - m = 200$ bits). The compression function f is designed to achieve some well defined security property (such as UOWHF security). Then in the second stage, one specifies a *domain extender* algorithm, which uses the compression function f to build a hash function f' hashing ℓ-bit inputs (for $\ell > n$) to an m-bit output. The domain extender is designed to ensure that if f satisfies its security property, then the extended function f' will satisfy the desired security property (e.g. UOWHF security).

The simplest and most natural domain extender is the well-known Merkle-Damgård (MD) extender [11, 5]. It was shown in [11, 5] that the MD extender preserves the collision-resistance security of the compression function, i.e. the MD extended function f' is collision-resistant if the compression function f is collision-resistant. However, as pointed out in [3], efficient collision-resistant compression functions seem difficult to design, and weakening the security requirement on the compression function is desirable.

A typical example that we focus on in this paper is the *subset sum* compression function, a computationally efficient function which was shown in [9] to achieve UOWHF security under the well known subset sum assumption (while the collision-resistance of this function depends on a less known and potentially much easier 'weighted knapsack' problem). It is natural to attempt to apply the MD extender to the subset sum compression function, and hope that the resulting function also achieves UOWHF security. Unfortunately, it was shown in [3] that the MD extender is not guaranteed to preserve UOWHF security of a compression function. Thus the result of [9] does not guarantee the security of the MD extended subset sum hash function, even assuming the subset sum assumption. Although other domain extenders exist [14, 3, 16] which do preserve the UOWHF property of the compression function, they are less simple than the MD extender and also (at least slightly) increase the length of the hash function key depending on the extension input length ℓ.

A possible way to use the MD extender for building UOWHF functions was proposed at Asiacrypt 2004 by Hong, Preneel and Lee [7]. They defined a stronger security property for compression functions called *higher order* UOWHF security. The 0th order UOWHF property is just the normal UOWHF property, but for $k > 0$, a kth order UOWHF is a stronger requirement than UOWHF. They showed that if a compression function f has the stronger kth order UOWHF property, then the MD extended function f' is guaranteed to have the UOWHF property, as long as the MD 'extension factor' is at most $k + 1$. However, it is known that there exist UOWHFs which are not kth order UOWHFs for any $k > 0$, so it is dangerous in general to simply take an UOWHF and assume that it is also a higher order UOWHF - in particular, the security loss as a function of k is unknown. Motivated by this concern in applying this result to the MD extended subset sum function, we were led to the following natural questions: Does the subset sum compression function satisfy the kth order UOWHF property for some $k > 0$, assuming only the subset sum assumption? If so, can we give an upper bound on the security lost as a function of k?

Our Results. We show that the subset sum hash function is a kth order UOWHF family (hashing n bits to $m < n$ bits) under the Subset Sum assumption for $k = O(\log m)$. Thus our result strengthens the one of Impagliazzo and Naor [9], who showed that the subset sum hash function is a UOWHF (i.e. UOWHF of order $k = 0$) under the Subset Sum assumption. Concretely, we show that the function's security as a kth order UOWHF deteriorates by (at most) about k bits (relative to the UOWHF case $k = 0$). Combined with the result of [7], we conclude that one can apply the MD extension to the subset sum compression function with 'extension factor' $k+1$, while losing (at most) about k bits of UOWHF security relative to the UOWHF security of the compression function (which is almost equivalent to the subset sum problem). We believe our result is of theoretical interest; in particular, as far as we are aware, our result is the first example of a natural UOWHF which is also a provably secure higher order UOWHF under the same well-known cryptographic assumption (while this assumption does not seem sufficient to prove its collision-resistance). In addition to showing that the natural MD extender can be applied to the subset sum compression function for small extension factors, our result also allows to shorten the key length of the extended hash function (compared with the total key length of the most efficient known UOWHF domain extender due to Shoup [16]).
Organization. The paper is organized as follows. In Section 2, we recall the definition of hash function security properties (in particular UOWHFs and higher order UOWHFs), and the construction of the subset sum compression function. Section 3 contains our main result on the kth order UOWHF security of the subset sum function. In Section 4, we discuss the application of our result to the extended subset sum function. Section 5 concludes the paper. Due to page limits, proofs of some claims in the paper were omitted – they can be found in the full version of the paper, available on the authors' web page.

2 Preliminaries

Collision-Resistant Hash Functions (CRHFs). Ideally, we would like a hash function to satisfy the strong security notion of collision-resistance, which is defined as follows.

Definition 1 (CRHFs). *A (t, ϵ) Collision-Resistant Hash Function (CRHF) family is a collection $\mathcal{F}$ of functions $f_K : \{0,1\}^n \to \{0,1\}^m$ indexed by a key $K \in \mathcal{K}$ (where $\mathcal{K}$ denotes the key space), and such that any attack algorithm* A *running in time t has success probability at most ϵ in the following game:*

- ***Key Sampling.*** *A uniformly random key $K \in \mathcal{K}$ is chosen and revealed to* A.
- A ***Collides.*** A *runs (on input K) and outputs a pair of hash function inputs $s_1, s_2 \in \{0,1\}^n$.*

We say that A succeeds *in the above game if it finds a valid collision for f_K, i.e. if $s_1 \neq s_2$ but $f_K(s_1) = f_K(s_2)$.*

Universal One-Way Hash Functions (UOWHFs). Naor and Yung [14] (see also [3]) showed that for several important cryptographic applications (such as hashing prior to signing a message with a digital signature scheme) one can weaken the collision-resistance requirement on a hash function, to a notion called Universal One-Way Hash Function (UOWHF), which is defined as follows.

Definition 2 (UOWHFs). *A (t, ϵ) Universal One-Way Hash Function (UOWHF) family [14] is a collection $\mathcal{F}$ of functions $f_K : \{0,1\}^n \rightarrow \{0,1\}^m$ indexed by a key $K \in \mathcal{K}$ (where $\mathcal{K}$ denotes the key space), and such that any attack algorithm A running in time t has success probability at most ϵ in the following game:*

– ***Key Sampling.*** *A uniformly random key $K \in \mathcal{K}$ is chosen (but not yet revealed to A).*
– A ***Commits.*** A *runs (with no input) and outputs a hash function input $s_1 \in \{0,1\}^n$.*
– ***Key Revealed:*** *The key K is given to A.*
– A ***Collides.*** A *continues running and outputs a second hash function input $s_2 \in \{0,1\}^n$.*

We say that A succeeds *in the above game if it finds a valid collision for f_K, i.e. if $s_1 \neq s_2$ but $f_K(s_1) = f_K(s_2)$.*

Higher Order UOWHFs. Hong, Preneel and Song [7] strengthened the definition of UOWHFs (while still being weaker than the CRHF requirement) by allowing the attacker to query an oracle for the hash function k times before commiting to the first input. A function that is secure even under this stronger attack is called a kth order UOWHF.

Definition 3 (kth Order UOWHFs). *A (t, ϵ) kth order Universal One-Way Hash Function family [7] is a collection $\mathcal{F}$ of functions $f_K : \{0,1\}^n \rightarrow \{0,1\}^m$ indexed by a key $K \in \mathcal{K}$ (where $\mathcal{K}$ denotes the key space), and such that any attack algorithm A running in time t has success probability at most ϵ in the following game:*

– ***Key Sampling.*** *A uniformly random key $K \in \mathcal{K}$ is chosen (but not yet revealed to A).*
– ***Oracle Queries.*** A *runs (with no input) and makes k adaptive queries $q_1, \ldots, q_k$ (with $q_i \in \{0,1\}^n$ for $i = 1, \ldots, k$) to an oracle for $f_K(\cdot)$, receiving answers $y_1, \ldots, y_k$ (where $y_i = f_K(q_i)$ for $i = 1, \ldots, k$).*
– A ***Commits.*** A *outputs a hash function input $s_1 \in \{0,1\}^n$.*
– ***Key Revealed:*** *The key K is given to A.*
– A ***Collides.*** A *continues running and outputs a second hash function input $s_2 \in \{0,1\}^n$.*

We say that A succeeds *in the above game if it finds a valid collision for f_K, i.e. if $s_1 \neq s_2$ but $f_K(s_1) = f_K(s_2)$.*

Note that a 0th order UOWHF is just a UOWHF, and a kth order UOWHF is also a rth order UOWHF for any $r \leq k$, but a UOWHF is not necessarily a higher order UOWHF; indeed, there exist UOWHFs which are not even first order UOWHFs [3].

The Subset-Sum Problem. This is defined as follows.

Definition 4 (Subset Sum Problem $\mathsf{SubSum}(n, m, p)$). *Let n and $m < n$ be positive integers, and let p denote a positive integer satisfying $2^{m-1} < p \leq 2^m$. The $\mathsf{SubSum}(n, m, p)$ problem is the following: Given p, a vector of n uniformly random integers $\mathbf{a} = (\mathbf{a}[1], \ldots, \mathbf{a}[n]) \in_R \mathbb{Z}_p^n$ and an independent uniform target integer $T \in_R \mathbb{Z}_p$, find a subset $\mathbf{s} = (\mathbf{s}[1], \ldots, \mathbf{s}[n])$ with $\mathbf{s}[i] \in \{0, 1\}$ for $i = 1, \ldots, n$ such that $\sum_{i=1}^{n} \mathbf{s}[i] \cdot \mathbf{a}[i] \equiv T \pmod{p}$.*

We say that problem $\mathsf{SubSum}(n, m, p)$ is (t, ϵ)-hard if, any algorithm A for $\mathsf{SubSum}(n, m, p)$ having run-time at most t has success probability at most ϵ, where the probability is over the uniformly random choice of $\mathbf{a} \in \mathbb{Z}_p^n$, $T \in \mathbb{Z}_p$ and the random coins of A.

A related, but possibly easier problem than Subset Sum is the *Weighted Knapsack* problem.

Definition 5 (Weighted Knapsack Problem $\mathsf{WKnap}(n, m, p)$). *Let n and $m < n$ be positive integers, and let p denote a positive integer satisfying $2^{m-1} < p \leq 2^m$. The $\mathsf{WKnap}(n, m, p)$ problem is the following: Given p, a vector of n uniformly random integers $\mathbf{a} = (\mathbf{a}[1], \ldots, \mathbf{a}[n]) \in_R \mathbb{Z}_p^n$ and an independent uniform target integer $T \in_R \mathbb{Z}_p$, find a weight vector $\mathbf{s} = (\mathbf{s}[1], \ldots, \mathbf{s}[n])$ with $\mathbf{s}[i] \in \{-1, 0, 1\}$ for $i = 1, \ldots, n$ such that $\sum_{i=1}^{n} \mathbf{s}[i] \cdot \mathbf{a}[i] \equiv T \pmod{p}$.*

We say that problem $\mathsf{WKnap}(n, m, p)$ is (t, ϵ)-hard if, any algorithm A for $\mathsf{WKnap}(n, m, p)$ having run-time at most t has success probability at most ϵ, where the probability is over the uniformly random choice of $\mathbf{a} \in \mathbb{Z}_p^n$, $T \in \mathbb{Z}_p$ and the random coins of A.

A decision variant of the subset sum problem was one of the first problems to be proven NP Complete [10]. The problem is well known in cryptography (also known as the knapsack problem) due to its role in the early history of public-key cryptosystems. The security of the Merkle-Hellman public key cryptosystem [12] was intended to based on the hardness of subset sum, but was later broken [15] due to the special non-random choice of the knapsack integers $\mathbf{a}[1], \ldots, \mathbf{a}[n]$. Later attacks based on lattice reduction work even for random knapsack integers, but only when m is sufficiently larger than n (i.e. when the function is used in *expansion* mode). According to [9], the best known provable lattice attack of this type [4] succeeds with high probability over a random choice of $\mathbf{a}[1], \ldots, \mathbf{a}[n]$, assuming a perfect lattice shortest vector oracle is available, whenever $m > 1.0629 \cdot n$.

Let us make a few other remarks:

- We use $m < n$ in our hash functions, which avoids the above-mentioned direct lattice attacks. However, one can still pick the (say) first $n' \leq m/1.0629$

integers $\mathbf{a}[1], \ldots, \mathbf{a}[n']$ and try to use the method of [4] to find a solution involving only those integers (i.e. set the $n - n'$ remaining weights to zero). A solution involving only the first n' integers is expected to exist with probability $1/2^{m-n'}$, so to make this probability at most $2^{-\delta}$ we need $m - n' \geq \delta$. It follows that we need $m \geq (1.0629/0.0629)\delta$, e.g. for $\delta = 80$, we need $m \geq 1352$ bit.
– A series of papers, starting from [1, 6] and up to the recent [13] have given reductions showing that the average-case weighted knapsack problem is as hard as various worst-case lattice problems, such as SVP approximation problems with a small polynomial approximation factor. However, although the average-case to worst-case connections exhibited in these papers are theoretically impressive, the concrete complexity of these 'polynomial approximation factor' lattice problems (even in the worst case) is currently unknown, and they may turn out to be substantially easier than subset sum due to the good performance of lattice reduction algorithms in practice.
– The Weighted knapsack problem may also be easier than the subset sum problem (see [2] for more discussion). Hence the subset sum hash function may not be as secure a collision-resistant function as it is as a UOWHF (or as we show, as a higher order UOWHF).

The Subset Sum Hash Function.

Definition 6 (Subset Sum Hash Function Family $\mathcal{F}_{SS}(n, m, p)$). *Let n and $m < n$ be positive integers, and let p denote a positive integer satisfying $2^{m-1} < p \leq 2^m$. The subset sum hash function family $\mathcal{F}_{SS}(n, m, p)$ is defined as follows. The key space is $\mathcal{K} = \mathbb{Z}_p^n$. Given a key $\mathbf{a} = (\mathbf{a}[1], \ldots, \mathbf{a}[n]) \in \mathbb{Z}_p^n$, the associated hash function $f_{\mathbf{a}} : \{0,1\}^n \to \{0,1\}^m$ is defined by $f_{\mathbf{a}}(\mathbf{s}) = \sum_{i=1}^{n} \mathbf{s}[i] \cdot \mathbf{a}[i] \bmod p \in \{0,1\}^m$ for $\mathbf{s} = (\mathbf{s}[1], \ldots, \mathbf{s}[n]) \in \{0,1\}^n$.*

We observe that that the subset sum hash function is a *public coin* function (see [8]), since the key consists of uniformly random integers in $\mathbb{Z}_p$.

3 The Security of the Subset Sum Hash Function

It is easy to see that the subset sum hash function family $\mathcal{F}_{SS}(n, m, p)$ is a CRHF family assuming the hardness of the weighted knapsack problem $\mathsf{WKnap}(n, m, p)$. However, as discussed above, the problem $\mathsf{WKnap}(n, m, p)$ may be easier than the subset sum problem $\mathsf{SubSum}(n, m, p)$. It is therefore desirable to have a hash family whose security relies only on the hardness of $\mathsf{SubSum}(n, m, p)$. With this motivation, Impagliazzo and Naor [9] relaxed their requirement from CRHF to a UOWHF, and showed that the subset sum hash function family $\mathcal{F}_{SS}(n, m, p)$ is a UOWHF assuming only the hardness of the subset sum problem $\mathsf{SubSum}(n, m, p)$. When translated to our concrete notation, the result of [9] can be stated as follows.

Theorem 1 (Impagliazzo-Naor). *If the Subset Sum problem $\mathsf{SubSum}(n, m, p)$ is (t, ϵ)-hard, then the the Subset Sum hash function family $\mathcal{F}_{SS}(n, m, p)$ is a*

(t', ϵ') *Universal One-Way Hash Function (UOWHF) family, where:*

$$t' = t - O(m \cdot n) \qquad \textit{and} \qquad \epsilon' = 2n \cdot \epsilon.$$

In this section we strengthen Theorem 1 by showing that the subset sum hash function family $\mathcal{F}_{SS}(n, m, p)$ is actually a kth order UOWHF for small $k = O(\log m)$, still assuming only the hardness of the subset sum problem $\mathsf{SubSum}(n, m, p)$. More concretely, we bound the way the security of $\mathcal{F}_{SS}(n, m, p)$ as a kth order UOWHF deteriorates with increasing k.

To begin with, we observe that for $k \geq m+2$, the security of $\mathcal{F}_{SS}(n, m, p)$ as a kth order UOWHF already deteriorates to the hardness of a weighted knapsack problem, i.e. the collision resistance of a related subset sum function.

Proposition 1. *For $k \geq m+2$, if the subset sum hash family $\mathcal{F}_{SS}(n, m, p)$ is a (t, ϵ) kth order UOWHF then the weighted knapsack problem* $\mathsf{WKnap}(\min(k, n) - 1, m, p)$ *is (t', ϵ') hard, where:*

$$t' = t - O(n) \qquad \textit{and} \qquad \epsilon' = \epsilon.$$

Proof. Let A' be an attacker for weighted knapsack problem $\mathsf{WKnap}(n', m, p)$ for $n' = \min(k, n) - 1$ with run-time/succ. prob. (t', ϵ'). Consider attacker A against the kth order UOWHF notion of the subset sum hash family $\mathcal{F}_{SS}(n, m, p)$ which runs as follows.

After a random key $\mathbf{a} = (\mathbf{a}[1], \ldots, \mathbf{a}[n]) \in \mathbb{Z}_p^n$ is chosen, A queries to $f_{\mathbf{a}}(.)$ the $\min(k, n)$ singleton subsets $\mathbf{q}_i$ for $i = 1, \ldots, n'+1$, where $\mathbf{q}_i[j] = 1$ if $j = i$ and $\mathbf{q}_i[j] = 0$ for $j \neq i$. Thus A obtains answers $y_i = \mathbf{a}[i]$ for $i = 1, \ldots, n'+1$. Now A runs A' on input modulus p, knapsack vector $\mathbf{a}' = (\mathbf{a}[1], \ldots, \mathbf{a}[n'])$ and target $T = \mathbf{a}[n'{+}1]$. After time t and with probability ϵ, A' returns $\mathbf{s} = (\mathbf{s}[1], \ldots, \mathbf{s}[n']) \in \{-1, 0, 1\}^{n'}$ satisfying $\sum_{i=1}^{n'} \mathbf{s}[i] \cdot \mathbf{a}[i] \equiv \mathbf{a}[n'+1] \pmod{p}$. So A has a collision $f_{\mathbf{a}}(\mathbf{s}_1) = f_{\mathbf{a}}(\mathbf{s}_2)$, where for $i = 1, \ldots, n'-1$, $\mathbf{s}_1[i] = 1$ if and only if $\mathbf{s}[i] = 0$, $\mathbf{s}_2[i] = 1$ if and only if $\mathbf{s}[i] = -1$, $(\mathbf{s}_1[n'], \mathbf{s}_2[n']) = (0, 1)$ (so $\mathbf{s}_1 \neq \mathbf{s}_2$) and for $i \geq n'+1$ we set $\mathbf{s}_1[i] = \mathbf{s}_2[i] = 0$. A outputs $\mathbf{s}_1$ and then $\mathbf{s}_2$ as his collision pair and breaks kth order UOWHF notion of $\mathcal{F}_{SS}(n, m, p)$. The attacker A has run-time $t = t' + O(n)$ and success probability $\epsilon = \epsilon'$. The proposition follows. □

For $k \leq m+1$, the reduction of Proposition 1 continues to hold, but in this case the associated weighted knapsack instance $\mathsf{WKnap}(k-1, m, p)$ has a solution with probability at most $3^{k-1}/p \leq 3^{k-1}/2^{m-1}$, which for fixed m decreases exponentially as k decreases towards 0. Thus for k sufficiently smaller than m we may hope that the subset sum hash family $\mathcal{F}_{SS}(n, m, p)$ is secure as a kth order UOWHF even if the weighted knapsack problem is easy to solve when a solution exists. Indeed, we next show that for $k = O(\log m)$ the subset sum hash function is a kth order UOWHF assuming only the hardness of subset sum. For technical reasons we also restrict in this result the modulus p to be *prime*.

Theorem 2. *Let n and $m < n$ be positive integers, let p denote a prime satisfying $2^{m-1} < p \leq 2^m$, and $k < \log_3(p) - 1$. If the Subset Sum problem $\mathsf{SubSum}(n, m, p)$ is (t, ϵ)-hard, then the Subset Sum hash function family $\mathcal{F}_{SS}(n, m, p)$ is a (t', ϵ') kth order Universal One-Way Hash Function (UOWHF) family, where:*

$$t' = t - O(k^2 n T_M(p)) \qquad \text{and} \qquad \epsilon' = 2^{k+1} \cdot (n-k) \cdot \epsilon + \frac{3^{k+1}}{2^m},$$

and $T_M(p)$ denotes the time to perform a multiplication modulo p.

Proof. Let A' be a kth order UOWHF attacker against the subset sum hash function family $\mathcal{F}_{SS}(n, m, p)$ with run-time/succ. prob. (t', ϵ'). We show how to use A' to construct an attacker A against subset sum problem $\mathsf{SubSum}(n, m, p)$ with run time $t = t' + O(k^2 n T_M(p))$ and succ. prob. $\epsilon \geq \frac{1}{2^{k+1} \cdot (n-k)} \cdot (\epsilon' - \frac{3^{k+1}}{2^m})$, which establishes the claimed result.

The basic idea of the reduction at a high level and its relation to the one in [9] is as follows. Given its subset sum instance $(\mathbf{a}, T)$, A runs A', answering its oracle queries using key $\mathbf{a}$ to obtain the first colliding input $\mathbf{s}_1$, but then reveals a different key $\mathbf{a}' \equiv_p \mathbf{a} + \mathbf{d}$ to A'. The new key $\mathbf{a}'$ is chosen by A based on $\mathbf{s}_1$ and the target sum T. In the reduction of [9], $\mathbf{d}$ is chosen to have Hamming weight 1 (in a random bit position) and such that $\sum_i \mathbf{s}_1[i] \cdot \mathbf{a}'[i] \equiv_p T$. This implies that a successful colliding $\mathbf{s}_2$ will be a solution to subset sum instance $(\mathbf{a}, T)$ if $\mathbf{s}_2$ has a zero in the position where $\mathbf{d}$ is non-zero. The authors in [9] are able to argue that such a zero position in $\mathbf{s}_2$ will exist (and equal the randomly chosen non-zero position in $\mathbf{d}$ with probability $1/n$). In our case, however, $\mathbf{a}'$ must also be consistent with the k earlier oracle query answers. This implies that $\mathbf{d}$ is restricted to be a solution of a linear system of rank $k+1$, so the minimum allowable Hamming weight of $\mathbf{d}$ increases to $k+1$, and the proof of [9] seems difficult to extend – we need that certain $k+1$ bits of $\mathbf{s}_2$ are *zero* (e.g. such bits may not exist). Instead, we use an alternative approach which only requires *guessing* the values (whatever they are) of the $k+1$ bits of $\mathbf{s}_2$ in positions where $\mathbf{d}$ is non-zero (hence we succeed with probability $1/2^{k+1}$). To do this, we choose $\mathbf{d}$ of weight $k+1$ such that $\sum_i \mathbf{s}_1[i] \cdot (\mathbf{a}[i] + \mathbf{d}[i]) \equiv_p T + \sum_i \widehat{\mathbf{s}}_2[i] \cdot \mathbf{d}[i]$ (where we use our guesses $\widehat{\mathbf{s}}_2$ for the $k+1$ bits of $\mathbf{s}_2$ on the right hand side) – note that this requirement is equivalent to equation (4) in the proof below. Then a colliding $\mathbf{s}_2$ gives $\sum_i \mathbf{s}_2[i] \cdot (\mathbf{a}[i] + \mathbf{d}[i]) \equiv_p T + \sum_i \widehat{\mathbf{s}}_2[i] \cdot \mathbf{d}[i]$ which implies that $\mathbf{s}_2$ is a solution to instance $(\mathbf{a}, T)$ if our guesses of $k+1$ bits of $\mathbf{s}_2$ were right (note the simplified discussion above ignores some other issues handled by the proof).

We now present the detailed reduction game.

1. **Subset Sum Instance Generation.** A random subset sum instance $(\mathbf{a}, T)$ (where $\mathbf{a} \in_R \mathbb{Z}_p^n$ and $T \in_R \mathbb{Z}_p$) is generated and given to A.
2. **Oracle Queries.** A runs A' with no input. When A' makes its ith oracle query $\mathbf{q}_i \in \{0,1\}^n$, A responds with answer $y_i = f_{\mathbf{a}}(\mathbf{q}_i) = \sum_{j=1}^n \mathbf{q}_i[j] \cdot \mathbf{a}[j] \bmod p$ (for $i = 1, \ldots, k$). A also stores the queries $\mathbf{q}_1, \ldots, \mathbf{q}_k$ for later use.

3. A' **Commits.** A' outputs hash function input $\mathbf{s}_1 \in \{0,1\}^n$.
4. **Key Revealed.** A samples a difference vector $\mathbf{d} \in \mathbb{Z}_p^n$ (using the algorithm detailed below) and gives A' the key $\mathbf{a}' = \mathbf{a} + \mathbf{d} \bmod p$. The difference vector $\mathbf{d}$ is sampled by A as follows:
   (a) A uses the stored queries of A' to build a $k \times n$ matrix Q having $\mathbf{q}_i$ as its ith row for $i = 1, \ldots, k$. *Remark*: The difference vector $\mathbf{d}$ will satisfy the matrix equation $Q \cdot \mathbf{d} \equiv \mathbf{0} \pmod p$, which implies that $Q \cdot \mathbf{a}' \equiv Q \cdot \mathbf{a} \pmod p$, i.e. $\mathbf{a}'$ is consistent with the answers to queries $\mathbf{q}_1, \ldots, \mathbf{q}_k$.
   (b) A performs Gaussian elimination on the matrix Q (by performing $O(k^2)$ elementary row operations over the field $\mathbb{Z}_p$ and $O(k)$ column swapping operations). Let $\widehat{Q}$ be the resulting $k \times n$ matrix (with entries in $\mathbb{Z}_p$) which is in reduced row echelon form:

$$\widehat{Q} = \begin{pmatrix} 1\ 0 \cdots 0 & \mathbf{q}'_1[k+1] & \cdots & \mathbf{q}'_1[n] \\ 0\ 1 \cdots 0 & \mathbf{q}'_2[k+1] & \cdots & \mathbf{q}'_2[n] \\ \vdots\ \vdots\ \ddots\ \vdots & \vdots & \cdots & \vdots \\ 0\ 0 \ldots 1 & \mathbf{q}'_k[k+1] & \cdots & \mathbf{q}'_k[n] \end{pmatrix}. \tag{1}$$

   A also keeps track of the column swapping operations to compute the corresponding column permutation $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ such that $\mathbf{d} \in \mathbb{Z}_p^n$ satisfies $Q\mathbf{d} \equiv 0 \pmod p$ if and only if $\widehat{\mathbf{d}} = (\mathbf{d}[\pi(1)], \ldots, \mathbf{d}[\pi(n)])^T$ satisfies $\widehat{Q}\widehat{\mathbf{d}} \equiv 0 \pmod p$. *Remark*: We assume, without loss of generality, that the k query vectors $\mathbf{q}_1, \ldots, \mathbf{q}_k$ are linearly independent over $\mathbb{Z}_p$ – If some query vector $\mathbf{q}_i$ of A' is a linear combination of the $i-1$ previous query vectors (the linear combination coefficients can be efficiently computed by Gaussian elimination over $\mathbb{Z}_p$), A' can itself answer the query by the same linear combination of the $i-1$ previous query answers. Hence we can always modify A' so that it always makes k linearly independent queries, without affecting the success probability of A'.
   (c) A picks a uniformly random integer $\ell \in_R \{k+1, \ldots, n\}$, and $k+1$ independent uniformly random bits $(\widehat{\mathbf{s}}[1], \ldots, \widehat{\mathbf{s}}[k]) \in \{0,1\}^k$ and $\widehat{\mathbf{s}}[\ell] \in \{0,1\}$. A defines $\widehat{\mathbf{s}}[j] = 0$ for $j \notin \{1, \ldots, k\} \cup \{\ell\}$ and computes (as detailed later) the unique vector $\mathbf{d} \in \mathbb{Z}_p^n$ (if it exists) satisfying

$$\mathbf{d}[\pi(j)] = 0 \text{ for } j \in \{k+1, \ldots, n\} \setminus \{\ell\}. \tag{2}$$

   and

$$Q \cdot \mathbf{d} \equiv \mathbf{0} \pmod p \tag{3}$$

   and

$$\sum_{j=1}^{n} (\widehat{\mathbf{s}}[j] - \mathbf{s}_1[\pi(j)]) \cdot \mathbf{d}[\pi(j)] \equiv T' - T \pmod p, \tag{4}$$

   where $T' = \sum_{j=1}^{n} \mathbf{s}_1[j] \cdot \mathbf{a}[j] \bmod p$. If no solution $\mathbf{d} \in \mathbb{Z}_p^n$ satisfying (2), (3) and (4) exists or if the solution exists but is not unique (because $T' - T \equiv 0 \pmod p$), then A sets $\mathbf{d} = \mathbf{0}$.

5. A' **Collides.** A' continues running and outputs a second hash function input $\mathbf{s}_2 \in \{0,1\}^n$.
6. A **Output.** A outputs $\mathbf{s}_2$ as its solution to the subset sum instance $(\mathbf{a}, T)$.

This completes the description of A. For clarity, and also for reference in later analysis, we now give more details on how A efficiently computes a unique $\mathbf{d} \in \mathbb{Z}_p^n$ satisfying (2), (3) and (4) (or determines that such $\mathbf{d}$ does not exist or is not unique). Using (1), the conditions (3) and (4) are equivalent to requiring that $\widehat{\mathbf{d}} = (\mathbf{d}[\pi(1)], \ldots, \mathbf{d}[\pi(n)])^T$ satisfies the $(k+1) \times n$ linear system

$$\widehat{Q}' \cdot \widehat{\mathbf{d}} \equiv \mathbf{t} \pmod{p}, \tag{5}$$

where $\widehat{Q}'$ is the $(k+1) \times n$ matrix having $\widehat{Q}$ as its first k rows and the row vector $\mathbf{s}' = (\widehat{\mathbf{s}}[1] - \mathbf{s}_1[\pi(1)], \ldots, \widehat{\mathbf{s}}[n] - \mathbf{s}_1[\pi(n)])$ as the $(k+1)$th row, and $\mathbf{t} = (0, 0, \ldots, 0, T' - T)^T$. By adding the multiple $-(\widehat{\mathbf{s}}[j] - \mathbf{s}_1[\pi(j)])$ of row j to row $k+1$ for $j = 1, \ldots, k$, A transforms the linear system (5) to the equivalent system

$$\widehat{Q}'' \cdot \widehat{\mathbf{d}} \equiv \mathbf{t} \pmod{p}, \tag{6}$$

where $\widehat{Q}''$ is a $(k+1) \times n$ matrix having $\widehat{Q}$ as its first k rows and its last row $\mathbf{s}''$ has its first k entries equal to 0 (i.e. $\mathbf{s}''[j] = 0$ for $j = 1, \ldots, k$). Now there are two cases. In the case $\mathbf{s}''[\ell] \equiv 0 \pmod{p}$, clearly either there are no solutions to (6) satisfying (2) (if $T' - T \not\equiv 0 \pmod{p}$), or the solution is not unique (if $T' - T \equiv 0 \pmod{p}$), so A sets $\mathbf{d} = \mathbf{0}$. In the second case $\mathbf{s}''[\ell] \not\equiv 0 \pmod{p}$, A uses back substitution to compute the unique solution $\mathbf{d}$ to (6) satisfying (2), i.e from the $(k+1)$th row of (6):

$$\mathbf{d}[\pi(\ell)] = \mathbf{s}''[\ell]^{-1} \cdot (T' - T) \bmod p \tag{7}$$

and from the first k rows:

$$\mathbf{d}[\pi(j)] = -\mathbf{q}'_j[\ell] \cdot \mathbf{d}[\pi(\ell)] \bmod p \text{ for } j = 1, \ldots, k. \tag{8}$$

The running-time of A is $t = t' + O(k^2 n T_M(p))$ as claimed. Now we analyse the success probability ϵ of A. Let us define several events in the above game:

1. SucA': A' succeeds, i.e. $\mathbf{s}_2 - \mathbf{s}_1 \neq 0$ and
$$\sum_{i=1}^{n} (\mathbf{s}_2[i] - \mathbf{s}_1[i]) \cdot \mathbf{a}'[i] \equiv 0 \pmod{p}. \tag{9}$$
2. SucA'_1: SucA' occurs and $\mathbf{s}_2 - \mathbf{s}_1$ is linearly independent of $\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$ over $\mathbb{Z}_p$.
3. SucA'_2: SucA' occurs and $\mathbf{s}_2 - \mathbf{s}_1$ is a linear combination of $\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$ over $\mathbb{Z}_p$.

Notice that events SucA'_1 and SucA'_2 partition the event SucA' so

$$\Pr[\mathsf{SucA}'] = \Pr[\mathsf{SucA}'_1] + \Pr[\mathsf{SucA}'_2]. \tag{10}$$

Claim 1. *If event* SucA'_1 *occurs then there exist 'good' values* $(\ell^*, \widehat{\mathbf{s}}^*, \widehat{\mathbf{s}}^*[\ell^*]) \in \{k+1, \ldots, n\} \times \{0,1\}^k \times \{0,1\}$ *such that if* A *correctly guessed those values when choosing its random variables* $(\ell, \widehat{\mathbf{s}}, \widehat{\mathbf{s}}[\ell])$ *(i.e. if* $(\ell, \widehat{\mathbf{s}}, \widehat{\mathbf{s}}[\ell]) = (\ell^*, \widehat{\mathbf{s}}^*, \widehat{\mathbf{s}}^*[\ell^*])$*) then* A *succeeds in solving its subset sum instance (i.e.* $\sum_{i=1}^{n} \mathbf{s}_2[i] \cdot \mathbf{a}[i] \equiv T \pmod p$*).*

Proof. If SucA'_1 occurs, then substituting $\mathbf{a}' \equiv \mathbf{a} + \mathbf{d} \pmod p$ and the definition of T' in (9) we obtain

$$\sum_{i=1}^{n} \mathbf{s}_2[i] \cdot \mathbf{a}[i] - T' \equiv \sum_{i=1}^{n} -(\mathbf{s}_2[i] - \mathbf{s}_1[i]) \cdot \mathbf{d}[i].$$

Hence if $\mathbf{d}$ satisfies

$$\sum_{i=1}^{n} (\mathbf{s}_2[i] - \mathbf{s}_1[i]) \cdot \mathbf{d}[i] \equiv T' - T \pmod p \tag{11}$$

then $\sum_{i=1}^{n} \mathbf{s}_2[i] \cdot \mathbf{a}[i] \equiv T \pmod p$ and A succeeds as claimed.

Now consider the equations (2),(3) and (4) and suppose for a moment that we had $\widehat{\mathbf{s}}[i] = \mathbf{s}_2[\pi(i)]$ for *all* $i = 1, \ldots, n$ (i.e. A correctly guessed *all* the n bits of $\mathbf{s}_2$). Because $\mathbf{s}_2 - \mathbf{s}_1$ is linearly independent of $\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$ over $\mathbb{Z}_p$, we know that the last row $\mathbf{s}''$ of the reduced matrix $\widehat{Q}''$ in (6) has a non-zero entry $\mathbf{s}''[\ell^*] \not\equiv 0 \pmod p$ where $\ell^* \in \{k+1, \ldots, n\}$, so if $\ell = \ell^*$ then a unique solution $\mathbf{d} = \mathbf{d}^*$ satisfying (2),(3) and (4) exists. Now observe that because of (2), the solution $\mathbf{d}^*$ depends only on ℓ^* and a subset of $k+1$ bits of $\mathbf{s}_2$, namely the bits $\mathbf{s}_2[\pi(1)], \ldots, \mathbf{s}_2[\pi(k)]$ and $\mathbf{s}_2[\pi(\ell^*)]$. So if A correctly guesses just those values (i.e. $\ell = \ell^*$ and $\widehat{\mathbf{s}}[i] = \mathbf{s}_2[\pi(i)]$ for $i \in \{1, \ldots, k\} \cup \{\ell^*\}$ with $\widehat{\mathbf{s}}[i] = 0$ for all other values of i) then $\mathbf{d} = \mathbf{d}^*$ is still a unique solution satisfying (2),(3) and (4) which is computed by A' (using (7) and (8)), so from (2) and (4) we conclude that (11) is satisfied and A succeeds as claimed. This completes the proof of the claim. □

Claim 2. *In the above game,* A *perfectly simulates the distribution of the view of* A' *as in the real kth order UOWHF attack game. Furthermore, the simulated view of* A' *is statistically independent of the random choices* $(\ell, \widehat{\mathbf{s}}, \widehat{\mathbf{s}}[\ell])$ *made by* A*.*

Proof. See full version of the paper.

From the above Claims we obtain the following lower bound on the success probability ϵ of A:

$$\begin{aligned} \epsilon &\geq \Pr[\mathsf{SucA}'_1 \wedge (\ell, \widehat{\mathbf{s}}, \widehat{\mathbf{s}}[\ell]) = (\ell^*, \widehat{\mathbf{s}}^*, \widehat{\mathbf{s}}^*[\ell^*])] \text{ using Claim 1} \\ &\geq \frac{1}{(n-k)2^{k+1}} \cdot \Pr[\mathsf{SucA}'_1] \text{ using independence Claim 2} \\ &\geq \frac{1}{(n-k)2^{k+1}} \cdot (\epsilon' - \Pr[\mathsf{SucA}'_2]) \text{ using (10) and Claim 2.} \end{aligned} \tag{12}$$

The following claim therefore completes the proof of the theorem's lower bound on the success probability of A. It is obtained by an information theoretic argument based on the fact that the answers y_i to the oracle queries of A' are independent and uniformly random in $\mathbb{Z}_p$ (over the random choice of $\mathbf{a}$).

Claim 3. $\Pr[\mathsf{SucA}'_2] \leq \frac{3^{k+1}}{2^m}$.

Proof. See full version of the paper.

Plugging the bound of Claim 3 in (12) establishes the claimed lower bound $\epsilon \geq \frac{1}{(n-k)2^{k+1}} \cdot \left(\epsilon' - \frac{3^{k+1}}{2^m}\right)$ on A's success probability, completing the proof of the theorem.

□

4 Application to Construction of Long-Input UOWHFs

In this section we discuss the application of our result to constructing UOWHFs used to hash long messages using a subset-sum compression function, in conjunction with the results of [7].

Let us suppose we wish to use the compression function family $\mathcal{F}_{SS}(n, m, p)$ (hashing n bits to $m < n$ bits) to construct a hash function family $\mathcal{F}'_{SS}(\ell, m)$ hashing a long l-bit message to m bits, where ℓ could be much larger than n. We want to ensure that $\mathcal{F}'_{SS}(\ell, m)$ is a UOWHF family, assuming that the underlying family $\mathcal{F}_{SS}(n, m, p)$ is a UOWHF family (or a higher order UOWHF family). A well-known and natural 'domain-extension' method is the Merkle-Damgård (MD) transform [11, 5], which works as follows. We assume for simplicity that $\ell = m + \mathcal{L} \cdot (n - m)$ for a positive integer $\mathcal{L}$. Then the MD family $\mathcal{F}'_{SS}(\ell, m)$ is defined as follows. A key $\mathbf{a} \in \mathbb{Z}_p^n$ of $\mathcal{F}'_{SS}(\ell, m)$ is just a uniformly random key of $\mathcal{F}_{SS}(n, m, p)$. An input message $M \in \{0,1\}^\ell$ is hashed using $f'_{\mathbf{a}}$ as follows:

1. Split $M \in \{0,1\}^\ell$ into one m-bit block $x_0 \in \{0,1\}^m$ and $\mathcal{L} = (\ell - m)/(n - m)$ $(n - m)$-bit blocks $(M[0], \ldots, M[\mathcal{L} - 1])$.
2. For $i = 0, \ldots, \mathcal{L} - 1$, compute $x_{i+1} = f_{\mathbf{a}}(x_i, M[i])$. Return $x_{\mathcal{L}} \in \{0,1\}^m$.

It has been proved in [11, 5] that if the compression family $\mathcal{F}_{SS}(n, m, p)$ is collision-resistant, then so is the MD family $\mathcal{F}'_{SS}(\ell, m)$. However, as discussed above, the collision-resistance of $\mathcal{F}_{SS}(n, m, p)$ relies on the hardness of the weighted knapsack problem $\mathsf{WKnap}(n, m, p)$, which may be substantially easier than the subset sum problem $\mathsf{SubSum}(n, m, p)$. So, using the fact that UOWHF security is enough for many hashing applications, and in order to rely only on the hardness of $\mathsf{SubSum}(n, m, p)$, one could hope to use Theorem 1, which shows that $\mathcal{F}_{SS}(n, m, p)$ is a (0th order) UOWHF family assuming only the hardness of $\mathsf{SubSum}(n, m, p)$. Unfortunately, as shown in [3], the MD construction does not preserve the UOWHF property in general, i.e. the fact that $\mathcal{F}_{SS}(n, m, p)$ is a UOWHF family does not imply that $\mathcal{F}'_{SS}(\ell, m)$ is a UOWHF family.
However, Hong, Preneel and Lee [7] have shown that if $\mathcal{F}_{SS}(n, m, p)$ is a (t, ϵ) kth order UOWHF for some $k > 0$ and $\mathcal{L} \leq k+1$, then the MD family $\mathcal{F}'_{SS}(\ell, m)$ is approximately a $(t, \mathcal{L} \cdot \epsilon)$ UOWHF. Combined with our result (Theorem 2), we conclude that for $k = O(\log m)$, the MD family $\mathcal{F}'_{SS}(\ell, m)$ is a UOWHF for $\mathcal{L} \leq k + 1$, assuming only the hardness of $\mathsf{SubSum}(n, m, p)$. More precisely, if subset sum problem $\mathsf{SubSum}(n, m, p)$ is (t, ϵ)-hard for some large time bound

t, then $\mathcal{F}'_{SS}(\ell,m)$ is approximately a $(t, 2^{k+1}(n-k)\mathcal{L}\cdot\epsilon)$-UOWHF. Comparing with Theorem 1, we see that the proven kth order UOWHF security of $\mathcal{F}_{SS}(n,m,p)$ (defined as the log of attacker's run-time/success probability ratio) is at most about $k+\log(\mathcal{L})$ bits lower than the proven UOWHF security of $\mathcal{F}_{SS}(n,m,p)$ (which in turn, by Theorem 1, is essentially equivalent to the hardness of $\mathsf{SubSum}(n,m,p)$).

4.1 Comparison with Shoup XOR-Mask UOWHF Domain Extender

Besides the basic MD construction, several other domain extenders for UOWHF hash families are known [14, 3, 16] which do preserve the UOWHF security of the underlying compression family; however, unlike the MD extension above, they all have the property that the length of key increases with the length of the message. The most efficient (in terms of key length) known extender of this type is the Shoup XOR-Mask variant of MD [16]. Let us denote this construction (hashing $\ell = m + \mathcal{L}\cdot(n-m)$ bits to m bits for a positive integer $\mathcal{L}$) by $= \mathcal{F}''_{SS}(\ell,m)$. It is built from the compression family $\mathcal{F}_{SS}(n,m,p)$ as follows. A key for family $\mathcal{F}''_{SS}(\ell,m)$ consists of a key $\mathbf{a}\in\mathbb{Z}_p^n$ for $\mathcal{F}_{SS}(n,m,p)$ and $\lfloor\log(\mathcal{L})\rfloor+1$ random 'masks' $\mathbf{K}^* = (K^*[0],\ldots,K^*[\lfloor\log(\mathcal{L})\rfloor])$, where $K^*[i]\in\{0,1\}^m$ for all i and $\mathcal{L} = (\ell-m)/(n-m)$. To hash an input message $M\in\{0,1\}^\ell$ using $f''_{\mathbf{a},\mathbf{K}^*}$,

1. Split $M\in\{0,1\}^\ell$ into one m-bit block $x_0\in\{0,1\}^m$ and $\mathcal{L} = (\ell-m)/(n-m)$ blocks of $(n-m)$-bit each, $(M[0],\ldots,M[\mathcal{L}-1])$.
2. For $i = 0,\ldots,\mathcal{L}-1$, compute $x_{i+1} = f_{\mathbf{a}}(x_i\oplus K^*[\nu_2(i+1)], M[i])$, where $\nu_2(i)$ denotes the largest integer ν such that 2^ν divides i. Return $x_{\mathcal{L}}\in\{0,1\}^m$.

Hence, for $\mathcal{L}\leq k+1$, the key length for the Shoup XOR-Mask extension $\mathcal{F}''_{SS}(\ell,m)$ is $len_{\mathcal{F}''} = n\cdot m + (\lfloor\log(\mathcal{L})\rfloor+1)\cdot m$ compared to $len_{\mathcal{F}'} = n\cdot m$ for the MD extension discussed above, so the MD extension achieves a saving of up to $(\lfloor\log(k+1)\rfloor+1)\cdot m$ bits by taking advantage of our result (Theorem 2). The MD extension method is also simpler. On the other hand, because the key length $n\cdot m$ for the compression family $\mathcal{F}_{SS}(n,m,p)$ dominates, the relative saving in *total* key length is small, and is only about $\frac{(\lfloor\log(k+1)\rfloor+1)}{n}$. However, as we explain in the next section, the total key length is not so important in applications and more significant relative savings in *per use* key length can be achieved in certain cases by combining our result with the 'XOR Mask Transform'.

Hashing Longer Messages. One can also take advantage of our result for hashing longer messages of arbitrary length $\ell > (k+1)\cdot(n-m)$. To do so (still assuming only the kth order UOWHF security of the compression family $\mathcal{F}_{SS}(n,m,p)$), it is possible to combine the MD extension with the Shoup extension. Namely, first apply the MD extension to $\mathcal{F}_{SS}(n,m,p)$ to construct the UOWHF family $\mathcal{F}'_{SS}((k+1)\cdot(n-m)+m, m)$ (hashing $(k+1)\cdot(n-m)+m$ bits to m bits), then apply the Shoup XOR-Mask extension to the compression family $\mathcal{F}'_{SS}(\ell,m)$ to hash ℓ bits to m bits. Compared to applying the Shoup extension directly to $\mathcal{F}_{SS}(n,m,p)$, this 'combined' method reduces the number of blocks in the Shoup extension by a factor of $k+1$, leading to a saving in key length by an additive amount of $\log(k+1)\cdot m$ bits.

4.2 Using the 'Semi Public-Key' XOR Mask Transform

In this section we show that more significant relative savings in UOWHF key length can be achieved in certain cases by combining our result with the 'Semi Public-Key XOR Mask Transform'.

The Semi-Public Key XOR Mask Transform. As remarked in [9], UOWHF hash families have the following useful property, namely that the UOWHF property is preserved by what we call the 'Semi-Public Key XOR-Mask Transform'. First, let us define the 'XOR-Mask Transform'.

Definition 7 ('XOR-Mask Transform'). *Let $\mathcal{F}(n, m)$ be a hash family (hashing n bits to m bits). Define the XOR-Mask Ttransform hash family $\mathcal{F}'(n, m)$ (hashing n bits to m bits) as follows. A key of $\mathcal{F}'(n, m)$ consists of a key $\mathbf{a}$ of $\mathcal{F}(n, m)$ and a random 'mask' $K \in \{0, 1\}^n$. An input $M \in \{0, 1\}^n$ is hashed using key $(\mathbf{a}, K)$ as follows $f'_{\mathbf{a},K}(M) = f_{\mathbf{a}}(M \oplus K)$.*

We call the XOR-Mask Transform a 'Semi-Public Key' transform, if the portion $\mathbf{a}$ of the key $(\mathbf{a}, K)$ of $f'_{\mathbf{a},K}$ is published before the attacker commits to its first collision input. Then we have the following simple but useful result.

Lemma 1. *['Semi-Public Key XOR-Mask Transform' Preserves UOWHF Security] Let $\mathcal{F}(n, m)$ be a hash family (hashing n bits to m bits), and let $\mathcal{F}'(n, m)$ denote the corresponding XOR-Mask transform of $\mathcal{F}(n, m)$. If $\mathcal{F}(n, m)$ is a (0th order) UOWHF family, then $\mathcal{F}'(n, m)$ is a (0th order) UOWHF family, even against 'Semi-Public Key' UOWHF attacks on $\mathcal{F}'(n, m)$, in which the random key $\mathbf{a}$ of $\mathcal{F}(n, m)$ is given to the attacker before committing to the first colliding input $\mathbf{s}_1 \in \{0, 1\}^n$ (i.e. only the 'XOR-Mask' $K \in \{0, 1\}^n$ is kept hidden from the attacker until he commits to $\mathbf{s}_1$).*

As remarked in [9], the practical implication of Lemma 1 for hash function applications (e.g. hashing a message prior to signing with a digital signature scheme) is that one can publish the long key $\mathbf{a}$ of $\mathcal{F}(n, m)$ once and for all (e.g. in the public key of a signature scheme, or in a hashing standard document), and then each use of the hash function (e.g. hashing and signing a message) only requires appending (to the signature) a relatively short fresh 'mask key' $K \in \{0, 1\}^n$.

Key Savings with the XOR Mask Transform. To construct a long ℓ-bit input UOWHF function (with $\ell = m + \mathcal{L} \cdot (n - m)$ for integer $\mathcal{L}$) from the subset sum compression family $\mathcal{F}(n, m, p)$ using the XOR Mask Transform, the standard method is to apply the Semi-Public Key XOR-Mask transform to $\mathcal{F}(n, m, p)$ (with mask key length n bit) and then the Shoup XOR-Mask domain extender from the previous section. Note that an m-bit part of the XOR transform mask key K can be 'absorbed' into the Shoup mask keys. Hence the result is a UOWHF family $\mathcal{F}'(\ell, m)$ mapping $\{0, 1\}^l$ to $\{0, 1\}^m$ with 'per-use' key length $l' = (\lfloor \log(\mathcal{L}) \rfloor + 1) \cdot m + (n - m) = n + \lfloor \log(\mathcal{L}) \rfloor \cdot m$. In terms of provable security, combining the reduction in [16] with Theorem 1, we obtain that if subset sum problem $\mathsf{SubSum}(n, m, p)$ is (t, ϵ)-hard, then $\mathcal{F}'(\ell, m)$ is approximately a $(t, 2n\mathcal{L} \cdot \epsilon)$ UOWHF.

We now show that one can shorten the 'per use' key length of the standard method using our result, if the compression ratio $\tau = n/m$ of the building block subset sum compression function family $\mathcal{F}(n,m,p)$ is close to 1 (the relative saving increases as τ gets close to 1 and decreases with increasing message length). We remark that the hardness of subset sum can only improve as τ gets close to 1, and indeed some efficient attacks are known which exploit a large value of $\tau > 1$ (see [9]); therefore the use of τ close to 1 may be necessary to achieve sufficient security.

Assume that $k+1$ is a divisor of $\mathcal{L}$ so $\mathcal{L} = \mathcal{L}' \cdot (k+1)$ for positive integer $\mathcal{L}'$. We first apply the MD extender with extension factor $k+1$ to $\mathcal{F}(n,m,p)$ to obtain a UOWHF family $\mathcal{F}^2$ mapping $\{0,1\}^{m+(k+1)\cdot(n-m)}$ to $\{0,1\}^m$. Next we apply the Semi-Public XOR Mask Transform to $\mathcal{F}^2$ to obtain UOWHF $\mathcal{F}^3$ with same domain and range and XOR mask key length $m + (k+1)\cdot(n-m)$ bit. Finally we apply the Shoup XOR-Mask extender with $\mathcal{L}'$ blocks to $\mathcal{F}^3$ obtain UOWHF $\mathcal{F}''(\ell, m)$ mapping $\{0,1\}^{\ell = m+\mathcal{L}\cdot(n-m)}$ to $\{0,1\}^m$, with 'per-use' key length $l'' = (\lfloor \log(\mathcal{L}')\rfloor + 1)\cdot m + (k+1)\cdot(n-m) = n + \lfloor\log(\mathcal{L}')\rfloor \cdot m + k\cdot(n-m)$. In terms of provable security, we combine the reductions in [16] and [7] with our Theorem 2 to obtain that if subset sum problem $\mathsf{SubSum}(n,m,p)$ is (t,ϵ)-hard then $\mathcal{F}''(\ell,m)$ is approximately a $(t, 2^{k+1}(n-k)\mathcal{L}\cdot\epsilon)$ UOWHF, so our method's provable security is about 2^k times lower than the standard method.

The relative saving $S(k) \overset{\text{def}}{=} (l'-l'')/l'$ in 'per use' key length of our method over the standard method is

$$S(k) = \frac{(\lfloor \log(\mathcal{L}'\cdot(k+1))\rfloor - \lfloor\log(\mathcal{L}')\rfloor)\cdot m - k\cdot(n-m)}{n + \lfloor\log(\mathcal{L})\rfloor\cdot m}. \tag{13}$$

Dropping the floor functions and using $\tau = n/m$, we obtain the continuous approximation

$$S(k) \approx \frac{\log(k+1) - (\tau-1)\cdot k}{\log(\mathcal{L}) + \tau}.$$

It is clear that for fixed $\mathcal{L}$ and τ close to 1, there is an optimum choice k_o for k which maximises $S(k)$. Using the continuous approximation for $S(k)$ above it is easy to show that the optimum values are given by

$$k_o \approx \frac{1}{\ln(2)(\tau-1)} - 1, \qquad S(k_o) \approx \frac{\log(\frac{1}{\ln(2)(\tau-1)}) + \tau - 1 - 1/\ln(2)}{\log(\frac{\mathcal{L}'}{\ln(2)(\tau-1)}) + \tau}, \tag{14}$$

corresponding to an absolute additive saving in 'per use' key length of $l' - l'' \approx (\log(\frac{1}{\ln(2)(\tau-1)}) + \tau - 1 - 1/\ln(2))\cdot m$ bits. Because the total 'per use' key length l' of the Shoup method increases only logarithmically with the message length, this constant additive saving remains significant even for quite long message lengths. On the other hand, the above comparison does not take into account that the proven security of our method is lower than the standard method by a factor of about 2^k relative to the subset sum problem. Let $T(\tau, m)$ denote the security (run time to success probability ratio) of subset sum problem $\mathsf{SubSum}(\tau\cdot m, m, p)$. To compare the key length at equal proven security level, we may assume a larger modulus length $m' > m$ in our method (but same compression ratio $\tau = n'/m' = n/m$) chosen such that $T(\tau, m') = 2^k \cdot T(\tau, m)$. Assuming $T(\tau, m) = C(\tau) \cdot 2^{c \cdot m}$ for some function $C(\tau)$ and constant $c > 0$ (e.g. $c = 0.0629/(1.0629) \approx 0.059$ may be reasonable as discussed in Section 2), we obtain $m' = m + k/c$. This leads to a reduced relative key length saving (for equal length messages)

$$S'(k) \geq \left(1 + \frac{k/c}{m}\right) S(k) - \frac{k/c}{m}. \tag{15}$$

This relative saving is still significant for short messages when m is sufficiently large compared to k/c, although the saving decreases (and actually becomes negative) for very long messages. Table 1 shows an example of the achievable savings.

Msg Len (kbit)	**Key Len std (kbit)**	**Key Len our (kbit))**	**Savings (%)**
5.6	10.1	5.6	45.2
8.8	12.1	7.9	35.1
15.3	14.1	10.2	27.8
106	20.1	17.2	14.8
1661	28.1	26.5	6.0
6637	32.1	31.1	3.3

Table 1. Example of savings in 'per use' key length using our method combined with the Shoup method ('our' column), compared to the Shoup method alone ('std' column). The savings have been corrected for equal provable security as explained in the text, assuming parameter values $m = 2000$, $\tau = 1.07$, $k = 19$, $c = 0.059$, $m' = 2321$.

One could obtain slightly greater savings with our method if Lemma 1 could be generalized to higher order UOWHFs. However we point out that this is not true in general, and in particular, the kth order UOWHF property of the subset sum function is not preserved by the 'Semi-Public Key XOR-Mask Transform' – we refer the reader to the full version of the paper for more details.

5 Conclusion

We have shown that the subset sum hash function is a kth order UOWHF for $k = O(\log m)$. Concretely, we have shown that its security as a kth order UOWHF is at most about k bits lower than its security as a (0th order) UOWHF (which in turn is almost equivalent to the subset sum problem), and showed an application of this result to shortening the key length of long-input UOWHFs built from the subset sum compression function using the Shoup XOR-mask domain extender. An interesting research problem is to find other applications for higher order UOWHFs (for which UOWHFs are not sufficient).

Acknowledgements. This work was supported by Australian Research Council Discovery Grants DP0345366 and DP0451484.

References

1. M. Ajtai. Generating Hard Instances of Lattice Problems. In *Proc. 28th STOC*, pages 99–108, New York, 1996. ACM Press.
2. M. Bellare and D. Micciancio. A New Paradigm for Collision-free Hashing: Incrementality at Reduced Cost. In *EUROCRYPT '97*, volume 1233 of *LNCS*, pages 163–192, Berlin, 1997. Springer-Verlag.
3. M. Bellare and P. Rogaway. Collision-Resistant hashing: Towards making UOWHFs Practical. In *CRYPTO '97*, volume 1294 of *LNCS*, pages 470–484, Berlin, 1997. Springer-Verlag.
4. M.J. Coster, B.A. LaMacchia, A.M. Odlyzko, and C.P. Schnorr. An Improved Low-Density Subset Sum Algorithm. In *EUROCRYPT '91*, volume 547 of *LNCS*, pages 54–67, Berlin, 1991. Springer-Verlag.
5. I. Damgård. A Design Principle for Hash Functions. In *CRYPTO '89*, volume 435 of *LNCS*, pages 416–427, Berlin, 1989. Springer-Verlag.
6. O. Goldreich, S. Goldwasser, and S. Halevi. Collision-free hashing from lattice problems. Technical Report TR96-056, Electronic Colloquium on Computational Complexity (ECCC), 1996.
7. D. Hong, B. Preneel, and S. Lee. Higher Order Universal One-Way Hash Functions. In *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 201–213, Berlin, 2004. Springer-Verlag.
8. C. Hsiao and L. Reyzin. Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins? In *CRYPTO '04*, volume 3152 of *LNCS*, pages 92–105, Berlin, 2004. Springer-Verlag.
9. R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *Journal of Cryptology*, 9:199–216, 1996.
10. R. M. Karp. Reducibility among Combinatorial Problems. In R. E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computation.* Plenum, New York, 1972.
11. R. Merkle. One Way Hash Functions and DES. In *CRYPTO '89*, volume 435 of *LNCS*, pages 428–446, Berlin, 1989. Springer-Verlag.
12. R. Merkle and M. Hellman. Hiding Information and Signatures in Trapdoor Knapsacks. *IEEE Trans. on Information Theory*, 24:525–530, 1978.
13. D. Micciancio and O. Regev. Worst-Case to Average-Case Reductions based on Gaussian Measures. In *Proc. FOCS 2004*, pages 372–381. IEEE Computer Society Press, 2004.
14. M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Significance. In *Proc. 21st STOC*, pages 33–43, New York, 1989. ACM Press.
15. A. Shamir. A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem. *IEEE Trans. on Information Theory*, 30:699–704, 1984.
16. V. Shoup. A Composition Theorem for Universal One-Way Hash Functions. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 445–452, Berlin, 2000. Springer-Verlag.