Self Masking for Hardening Inversions

(Preliminary Version)

Paweł Cyprys[1], Shlomi Dolev[1], and Shlomo Moran[2]

[1] Ben-Gurion University of the Negev
[2] Technion Israel Institute of Technology
October 2, 2022

Abstract. The question whether one way functions (i.e., functions that are easy to compute but hard to invert) exist is arguably one of the central problems in complexity theory, both from theoretical and practical aspects. While proving that such functions exist could be hard, there were quite a few attempts to provide functions which are one way "in practice", namely, they are easy to compute, but there are no known polynomial time algorithms that compute their (generalized) inverse (or that computing their inverse is as hard as notoriously difficult tasks, like factoring very large integers).
In this paper we study a different approach. We provide a simple heuristic, called self masking, which converts a given polynomial time computable function f into a self masked version $[f]$, which satisfies the following: for a random input x, $[f]^{-1}([f](x)) = f^{-1}(f(x))$ w.h.p., but a part of $f(x)$, which is essential for computing $f^{-1}(f(x))$ is *masked* in $[f](x)$. Intuitively, this masking makes it hard to convert an efficient algorithm which computes f^{-1} to an efficient algorithm which computes $[f]^{-1}$, since the masked parts are available in $f(x)$ but not in $[f](x)$.
We apply this technique on variants of the subset sum problem which were studied in the context of one way functions, and obtain functions which, to the best of our knowledge, cannot be inverted in polynomial time by published techniques.

1 Introduction

The question whether one way functions (i.e., functions that are easy to compute but hard to invert) exist is arguably one of the central problems in complexity theory, both from theoretical and practical aspects. e.g., it is known that the existence of one way functions implies, and is implied by, the existence of pseudo random number generators (see e.g. [6] for a constructive proof of this equivalence).

While proving that one way functions exist could be hard (since it would settle affirmatively the conjecture that $P \neq NP$), there were quite a few attempts to provide functions which are one way "in practice" – namely, they are easy to compute, but there are no known polynomial time algorithms which compute their (generalized) inverses.

In this paper we suggest a heuristic, called *self masking*, to cope with published attacks on previous attempts to construct one way functions. Specifically, the self masking versions of polynomial time computable functions "hide" in the outputs of these functions parts which are essential for computing their inverse.

1.1 Preliminaries

To make the presentation self contained and as short as possible, we present only definitions which are explicitly used in our analysis. For a more comprehensive background on one way functions and related applications see, e.g., [7; 6].

The notation $x \in_{\mathcal{U}} D$ indicates that x is a member of the (finite) set D, and that for probabilistic analysis we assume a uniform distribution on D.

Following [6], we define one way functions using the notion of *polynomial time function ensembles.*

Definition 1 *A polynomial time function ensemble $f = (f_k)_{k=1}^{\infty}$ is a polynomial time computable function that, for a strictly increasing sequence $(n_k)_{k=1}^{\infty}$ and a sequence $(m_k)_{k=1}^{\infty}$, f_k maps $\{0,1\}^{n_k}$ to $\{0,1\}^{m_k}$. Both n_k and m_k are bounded by a polynomial in k and are computable in time polynomial in k. The domain of f_k is denoted by $D_k = \{0,1\}^{n_k}$.*[3]

Definition 2 *Let $f = (f_k)_{k=1}^{\infty}$ be a polynomial time function ensemble. Then f is* one way function *if for any polynomial time algorithm AL, and for all but finitely many k's, the probability that $AL(f_k(x)) \in f_k^{-1}(f_k(x))$ for $x \in_{\mathcal{U}} D_k$ is negligible (i.e., asymptotically smaller than $|x|^{-c}$ for any $c > 0$) .*

1.2 Previous work

Quite a few attempts to construct one way functions - typically in the context of public key cryptosystems - are based on the hardness of variants of the subset sum problem. However, algorithmic attacks which compute the inverses of the suggested functions in expected polynomial time were later found for all these attempts.

The public key cryptosystem of Merkle and Hellman [10] uses an easy to solve variant of the subset sum problem, in which the input sequence is super increasing, which is transformed to a sequence in which the super increasing structure is concealed. This cryptosystem was first broken by Shamir in [12], and subsequently more sophisticated variants of it were broken too [2].

Super increasing sequences are a special case of *low density* instances of the subset sum problem. These low density instances were also solved efficiently [8; 1; 3]. A comprehensive survey of these methods and of the corresponding attacks can be found in [11].

---

[3] For definiteness, inputs whose length ℓ is different from m_k for all k are mapped to 1^{ℓ}.

1.3 Contribution

The basic variant of the self masking technique replaces a (polynomial time computable) function f by a self masking version, denoted $[f]$, as follows: Let $y = f(x)$ for arbitrary x in the domain of f, and let $|x|$ denote the length of x. Then a self masked version $[y] = [f](x)$ is obtained by replacing two "critical" substrings, z_1 and z_2, of y, of length $|x|^{\Omega(1)}$, by $z_1 \oplus z_2$ [4]. Intuitively, z_1 and z_2 are critical in the sense that they are essential for computing $f^{-1}(y)$.

An immediate concern raised by this method is that it might significantly increase the number of preimages associated with the masked output value $[f](x)$, e.g. that $[f]^{-1}([f](x))$ may contain exponentially many preimages of $[f](x)$ even if $f^{-1}(f(x))$ contains only few elements. We cope with this difficulty by showing that, by carefully selecting the parameters of the transformation, this is not the case, and in fact that we can guarantee that, w.h.p., $[f]^{-1}([f](x)) = \{x\}$, i.e. $[f]$ is univalent.

We demonstrate this technique on functions associated with variants of the subset sum problem, which were widely used in the context of one way functions (see eg [10; 8; 7; 9]).

Organization of the paper. Section 2 introduces the self masked subset sum problem, and proves that this problem is NP hard.
Section 3 presents function ensembles associated with the self masked subset sum problem, and present conditions under which the resulted functions are univalent w.h.p.
Section 4 demonstrates that applying the self masking technique on super increasing instances of the subset sum problem produces function which cannot be inverted by the known attacks on cryptosystems based on super increasing sequences.
Section 5 extends this result by showing that applying the self masking technique on low density instances of the subset sum problem provides functions which cannot be inverted by the known algorithms for solving low density instances of the (unmasked) subset sum problem.
Section 6, which is only sketched in this version, discusses extension of the self masking technique to high density instances of the subset sum problem.
Finally, Section 7 summarizes the results of this paper and discusses possible extensions.

2 Subset sum with self masking

The subset sum problem of dimensions k and ℓ, to be denoted $SS(k, \ell)$, is defined as follows: Let $\mathcal{A}_{k,\ell} = \{(a_1, \ldots, a_k) : a_i \in [0, 2^\ell - 1]\}$. An input to $SS(k, \ell)$ is a pair (A, b), where $A \in \mathcal{A}_{k,\ell}$ and b is an additional integer. It is required to

[4] $z_1 \oplus z_2$ denotes bitwise XOR of the binary representations of z_1 and z_2; leading zeros are assumed when these representations are of different lengths.

decide if there is a binary vector $\alpha = (\alpha_1, \ldots, \alpha_k)$ s.t. $A\alpha^T = \sum \alpha_i a_i = b$.

We use the subset sum problem to define the following function ensemble $(f_{k,\ell})_{k,\ell \in \mathbb{N}}$: For each k and ℓ, $n_{k,\ell} = k(\ell+1)$, $m_{k,\ell} = \ell(k + \lceil log(k) \rceil)$. An input $x \in \{0,1\}^{n_{k,\ell}}$ represents a sequence $x = (A, \alpha) = ((a_1, \ldots, a_k), \alpha)$, where each a_i is encoded by ℓ bits, and α is a binary vector with k bits. $f_{k,\ell}(x) = y$ is given by

$$f_{k,\ell}(x) = f_{k,\ell}(A, \alpha) = (A, A\alpha^T) = y \ , \tag{1}$$

where $A\alpha^T < k2^\ell$ is encoded by $\ell \lceil log(k) \rceil$ bits.

Note that $f_{k,\ell}(x)$ is necessarily a solvable instance of the subset sum problem, and $f_{k,\ell}^{-1}(f_{k,\ell}(x))$ is the nonempty set of the solutions to this instance.

The self masking version of subset sum, denoted *self masked subset sum*, consists of two independent instances of the problem, which mask each other (so it actually uses the 2-dimensional subset sum problem [5]).

Specifically, the input is a triplet (A_1, A_2, b), where A_1 and A_2 are k dimensional vectors of positive integers, and b is s positive integer. It is needed to decide if there is a binary k-vector α and integers b_1, b_2, s.t.

$$A_1\alpha^T = b_1, \ A_2\alpha^T = b_2, \quad \text{and} \quad b = b_1 \oplus b_2.$$

Lemma 1. *The self masked subset sum problem is NP-Hard.*

Proof. We prove the lemma by presenting a polynomial time reduction from the subset sum problem to the self-masked subset sum problem. Let (A, b) be an input to the subset sum of dimensions k and ℓ (for arbitrary k and ℓ). We reduce (A, b) to an input (C, D, e) to the self masked subset sum, where $C = A$ and D and e are defined as follows: Let $n = \lceil \log(\sum_{i=1}^{k} a_i) \rceil$. Then $D = 2^n C = (2^n a_1, \ldots, 2^n a_k)$, and $e = (2^n + 1)b$.

Since n is linear in the input length, the reduction can be performed in polynomial time. To prove its correctness, we need to show that there is a binary vector α satisfying $A\alpha^T = b$ if and only if there is a binary vector β satisfying $C\beta^T \oplus D\beta^T = e$.

Let $\alpha = (\alpha_1, \ldots, \alpha_k)$ be an arbitrary non-zero binary vector. Observe that in the binary representation of the integer $D\alpha^T$, the n least significant bits are all zeros, while in the binary representation of $C\alpha^T$ (with possible leading zeros), the only non-zero bits are among the n least significant bits. This implies that, in this case, the XOR operation coincides with integer addition, i.e.

$$C\alpha^T \oplus D\alpha^T = C\alpha^T + D\alpha^T = (2^n + 1)C\alpha^T = (2^n + 1)A\alpha^T.$$

We conclude that if $A\alpha^T = b$ for some α, then $C\alpha^T \oplus D\alpha^T = (2^n + 1)b = e$, and vice versa - if, for some β, $C\beta^T \oplus D\beta^T = e$, then $A\beta^T = e/(2^n + 1) = b$. This completes the correctness proof.

3 Function ensembles associated with self masked subset sum

Given k and ℓ, the function ensemble of dimensions k and ℓ associated with the self masked subset sum is denoted by $[f]_{k,\ell}$. An input x to $[f]_{k,\ell}$ is a triple (A_1, A_2, α), and

$$[f]_{k,\ell}(x) = [f]_{k,\ell}(A_1, A_2, \alpha) = \left(A_1, A_2, A_1\alpha^T \oplus A_2\alpha^T\right) \tag{2}$$

That is, in $[f]_{k,\ell}(x)$ the values of b_1 and b_2 mask each other by $b_1 \oplus b_2$.

Lemma 2. *Assume a uniform distribution on $\mathcal{A}_{k,\ell}$, and let $\alpha \in \{0,1\}^k \setminus \{0^k\}$. Then the random variable r_α on $\mathcal{A}_{k,\ell}$ defined by*

$$\forall A \in \mathcal{A}_{k,\ell}, \quad r_\alpha(A) = A\alpha^T (\text{mod } 2^\ell).$$

defines the uniform distribution on $[0, 2^\ell - 1]$.

Proof. We need to show that for each integer $c \in [0, 2^l - 1]$, it holds that $Prob[r(A) = c] = 2^{-\ell}$. For this we assume WLOG that $\alpha_1 = 1$, and we let β be the vector obtained from α by setting α_1 to 0. Then, for each $A = (a_1, \ldots, a_k)$, $r_\alpha(A) = A\alpha^T = A\beta^T + a_1$. Hence we get, using arithmetic modulus 2^ℓ:

$$\begin{aligned} Prob[r_\alpha(A) = c] &= \sum_{j \in [0, 2^\ell - 1]} \left(Prob[A\beta^T = j] \cdot Prob[a_1 = (c - j)]\right) \\ &= \left(\sum_{j \in [0, 2^\ell - 1]} (Prob[A\beta^T = j]\right) \cdot 2^{-\ell} \;=\; 2^{-\ell}, \end{aligned}$$

where the second equality holds since for all j, $Prob[a_1 = c - j \pmod{2^\ell}] = 2^{-\ell}$. □

Our proof uses the following variant of lemma 2

Lemma 3. *Assume a uniform distribution on $\mathcal{A}_{k,\ell}$, and let $\alpha, \beta \in \{0,1\}^k$ s.t. $\alpha \neq \beta$. Then the random variable $r_{\alpha,\beta}$ on $\mathcal{A}_{k,\ell}$ defined by*

$$\forall A \in \mathcal{A}_{k,\ell}, \quad r_{\alpha,\beta}(A) = [A\alpha^T (\text{mod } 2^\ell)] \;\oplus\; [A\beta^T (\text{mod } 2^\ell)]$$

defines the uniform distribution on $[0, 2^\ell - 1]$.

Proof. Assume WLOG that $\alpha_1 = 0$ and $\beta_1 = 1$. Let $c = (c_2, \ldots, c_k)$, where the c_i's are arbitrary elements in $[0, \ldots, 2^\ell - 1]$. Let $\mathcal{A}_c$ be the subset of all vectors $(a_1, \ldots, a_k) \in \mathcal{A}_{k,\ell}$ in which $a_2 = c_2, \ldots, a_k = c_k$. Then, on $\mathcal{A}_c$, $A\alpha^T$ is fixed (since $\alpha_1 = 0$ and hence it is independent of the value of a_1), and $A\beta^T$ mod (2^ℓ) distributes uniformly on $[0, \ldots, 2^\ell - 1]$ (since a_1 distributes uniformly in $[0, 2^\ell - 1]$). So the lemma holds for $\mathcal{A}_c$. The lemma follows by observing that $\mathcal{A}_{k,\ell}$ is a disjoint union of $\mathcal{A}_c$, where c varies over all $2^{\ell(k-1)}$ possible combinations of $(k-1)$ tuples.

Lemma 4. *Let* $(A_1, A_2) \in_{\mathcal{U}} [\mathcal{A}_{k,\ell}]^2$*, and let* $\alpha \in \{0,1\}^k$*. Then the probability that there exists* $\beta \in \{0,1\}^k, \beta \neq \alpha$*, s.t.*

$$A_1\alpha^T (\text{mod } 2^\ell) \oplus A_2\alpha^T (\text{mod } 2^\ell) = A_1\beta^T (\text{mod } 2^\ell) \oplus A_2\beta^T (\text{mod } 2^\ell) \qquad (3)$$

is at most $2^{k-\ell}$.

Proof. Fix A_1 for now. Let $\beta \in \{0,1\}^k, \beta \neq \alpha$ be given. Denote for brevity $A_1\alpha^T = b_1$ and $A_1\beta^T = b_2$. Then (3) can be written as: $b_1 \oplus A_2\alpha^T = b_2 \oplus A_2\beta^T$, which is equivalent to $b_1 \oplus b_2 = A_2\alpha^T \oplus A_2\beta^T$. By Lemma 3, the probabilty of this last equality to hold for a random A_2 is $2^{-\ell}$. The lemma for fixed A_1 follows by applying the union bound on all $\beta \in \{0,1\}^k \setminus \{\alpha\}$. Since A_1 was arbitrary, the lemma is proven.

For given k and ℓ, let $[f]_{k,\ell}$ be the self masking function associated with the subset sum problem as defined in the beginning of Section 3, and let $n_{k,\ell}$ be the length of the corresponding inputs. As an immediate application of lemma 4 we get:

Corollary 1 *Let c be a positive constant. If* $\ell > k + c\log n_{k,\ell} = k + c\log(k(\ell+1))$*, then the probability that for a random input* x_1 *to* $[f]_{k,\ell}$*, there exists* $x_2 \neq x_1$ *satisfying* $[f]_{k,\ell}(x_1) = [f]_{k,\ell}(x_2)$ *is smaller than* $(n_{k,\ell})^{-c}$.

Proof. Let $x_1 = (A_1, A_2, \alpha)$ be a random input to $[f]_{k,\ell}$. Then the probability that there exists $x_2 \neq x_1$ s.t. $[f]_{k,\ell}(x_1) = [f]_{k,\ell}(x_2)$ is equal to the probability that there exists $\beta \neq \alpha$ satisfying Equation (3). By Lemma 4 this probability is smaller than $2^{k-\ell}$. The result follows since by our assumption $k - \ell < -c\log n_{k,\ell}$.

Note that the premises of Corollary 1 hold for almost all ℓ provided that $\ell \geq (1+\varepsilon)k$ for some fixed $\varepsilon > 0$ - i.e. for low density instances of the subset sum problem.

4 Subset sum with super increasing sequences

A sequence $A = (a_1, \dots, a_k)$ is *super increasing* if:

$$\text{for } i = 2, \dots k, \quad \sum_{j=1}^{i-1} a_j \; < \; a_i \; .$$

A subset sum instance (A, b) is easily solved in polynomial time when A is super increasing: start with an empty subset S, and at each stage add to S the largest element in A which is not yet in S, provided that the sum of the elements in S does not exceed b. Nevertheless, few cryptographic schemes are based on solving instances with super increasing sequences, by concealing their super increasing nature. [11] provides a detailed survey of these methods, and then describes the efficient attacks that eventually broke them. In this section we observe that these attacks must use a value which is hidden by the self masking technique, implying that the self masked version of subset sum with super increasing sequences are likely to be immune to these attacks.

The super increasing variant of subset sum of dimensions k and ℓ, denoted $SS^{si}(k,\ell)$, is defined by associating with each input sequence $A = (a_1, \dots, a_k) \in \mathcal{A}_{k,\ell}$ a super increasing sequence $A^{si} = (a_1^{si} \dots, a_k^{si})$, where $a_1^{si} = a_1, a_2^{si} = 2^\ell + a_2, a_3^{si} = 3 \cdot 2^\ell + a_3$, and in general $a_i^{si} = (2^{i-1} - 1)2^\ell + a_i$. It is easy to check that A^{si} is super increasing. The functions $f_{k,\ell}^{si}$ are obtained from $f_{k,\ell}$ in Equation 1, by replacing $A\alpha^T$ by $A^{si}\alpha^T$:

$$f_{k,\ell}^{si}(x) = f_{k,\ell}^{si}(A, \alpha) = (A, A^{si}\alpha^T)$$

Since A^{si} is super increasing, inverting the function $f_{k,\ell}^{si}$ by finding the unique vector α satisfying $A^{si}\alpha^T = b$, as outlined above, is easy. We now argue that, for k and ℓ satisfying the premises of Corollary 1, the self masking version of $f_{k,\ell}^{si}$ is likely to be immune to this inversion method. For this, we observe that $f_{k,\ell}^{si}$ is univalent w.h.p.:

For all A and α it holds that $A\alpha^T (\bmod\ 2^\ell) = A^{si}\alpha^T (\bmod\ 2^\ell)$. Hence Lemma 4 remains valid if, in eq. (3), we replace $A_1 (A_2)$ by $A_1^{si} (A_2^{si}$ resp.). and hence the following analogue of Corollary 1 for super increasing sequences holds.

Corollary 2 *If $\ell > k + c \log n_{k,\ell} = k + c \log(k(\ell + 1))$, the probability that there are x_1, x_2 with $f_{k,\ell}^{si}(x_1) = f_{k,\ell}^{si}(x_2)$ is smaller than $(n_{k,\ell})^{-c}$.*

Proof. Let $\left[f^{si}\right]$ be the self masking version of f^{si}. Corollary 2 implies that if $\ell > k + c\log(n_{k,\ell})$, then w.h.p., $\left[f^{si}\right](A_1, A_2, \alpha) = \{(A_1, A_2, \alpha)\}$. In this scenario, inverting $\left[f^{si}\right](A_1, A_2, \alpha)$ is at least as hard as reconstructing the integers $b_1 = A_1^{si}\alpha$ and $b_2 = A_2^{si}\alpha$ from their xor $b = b_1 \oplus b_2$ and the sequences A_1, A_2. This last task appears to be non trivial.

Other variants based on super increasing sequences. As noted above, few cryptosystems are based on subset sum with super increasing sequences. We briefly survey them below (for a more comprehensive exposition we refer again to [11]).

The most known variant is due to Merkle and Hellman [10]: Given a super increasing sequence $A = (a_1, \dots, a_n)$ and b, select relatively prime integers W, M, where $W < M$ and $M > b$, and then define $a_i' = Wa_i \pmod M, b' = Wb \pmod M$. The original super increasing sequence A is then replaced by a random permutation of $A' = (a_1', \dots, a_n')$, and b is replaced by b'. The resulting sequence is not super increasing, and reconstructing the original super increasing sequence (when W and M are not given) is not straightforwards. This process can be iterated few times, yielding the multiply iterated Merkle Hellman system.
The first polynomial time algorithm which solves the original (singly iterated) Merkel Hellman system was given by Shamir in [12] (this attack assumes certain restrictions on the ratio between M and k, which are implied by properties of the associated cryptosystem). Shamir's attack was later followed by algorithms

solving more sophisticated variants of such systems (eg [2]). These algorithms essentially reconstruct the original sequence A and b from the hidden versions A' and b', and in particular the value of b' must be given for applying these attacks. Since this value is hidden by our self masking technique, it appears that these attacks cannot be directly applied to the self masked variants of Merkle and Hellman systems, as well to their extensions.

5 The low density variant

The subset sum problem of dimensions k and ℓ, $SS(k,\ell)$, is said to be of low density if ℓ is larger than k. Polynomial time algorithms for inverting low density $f_{k,\ell}$ were first obtained in [1; 8]. These algorithms reduce the inversion of $f_{k,\ell}$ to finding a shortest vector in an integer lattice. A detailed survey of these algorithms and later improvements can be found in [11]. We briefly describe below the algorithm of [8], as described in [3].

Let $x = (A,\alpha)$ be an input to $f_{k,\ell}$, where $A = (a_1,\dots,a_k) \in_{\mathcal{U}} \mathcal{A}_{k,\ell}$ and $\alpha = (\alpha_1,\dots,\alpha_k) \in_{\mathcal{U}} \{0,1\}^k$. Let further $y = f_{k,\ell}(x) = (A,b)$, where $b = A\alpha^T$. The algorithm of [8] reduces the computation of $f^{-1}_{k,\ell}(y)$ to the problem of finding a shortest vector in the $k+1$ dimensional integer lattice $L(y) = L(A,b)$ defined by the basis

$$
\begin{aligned}
v_1 &= (1,0,\dots,0,-Ka_1),\\
v_2 &= (0,1,0,\dots,0,-Ka_2)\\
&\cdots\\
v_k &= (0,\dots,0,1,-Ka_k),\\
v_{k+1} &= (0,\dots,0,Kb).
\end{aligned}
$$

where K is any integer larger than $\sqrt{k}$. Given that basis, each binary vector $\beta = (\beta_1,\dots,\beta_k)$ is mapped to a lattice-vector $w(\beta)$ given by

$$w(\beta) = \sum \beta_i v_i + v_{k+1} = \left(\beta_1,\beta_2,\dots,\beta_k, b - A\beta^T\right);$$

Observe that $A\beta^T = b$ iff $w(\beta) = (\beta_1,\dots,\beta_k,0)$. The main ingredient in the correctness proof of the algorithm of [8] is showing that if $k < 1.54725\ell$, then w.h.p. α is the only vector satisfying $A\alpha^T = b$, and $w(\alpha)$ is the unique shortest vector in $L(y)$. [3] uses a similar proof technique, but reduces the vector $y = (A,b)$ to a different lattice $L'(y)$, which enables to improve the required density to $k < 1.0639\ell$. For our sake it is sufficient to note that the use of the sum $b = A\alpha^T$ in the definition of the basis vector v_{k+1} is crucial in the above reductions.

Consider now the self masking function $[f]_{k,\ell}$

$$[f]_{k,\ell}(A_1,A_2,\alpha) = (A_1,A_2,b_1 \oplus b_2), \quad \text{where} \quad b_1 = A_1\alpha^T, b_2 = A_2\alpha^T.$$

In order to compute the inverse of $[f]_{k,\ell}(A_1,A_2,\alpha) = (A_1,A_2,b_1 \oplus b_2)$ it is necessary to compute from $(A_1,A_2,b_1 \oplus b_2)$ two integers b'_1 and b'_2 s.t.: (i) $b'_1 \oplus b'_2 = b_1 \oplus b_2$, and (ii) for some binary vector α' it holds that $b'_1 = A_1\alpha'^T, b'_2 = A_2\alpha'^T$.

6 The high density variant

The subset sum problem of dimensions k and ℓ, is said to be of high density if $k > \ell$. In this case the corresponding self masking function $[f]_{k,\ell}$ is w.h.p. not univalent. Nevertheless, self masking functions which are univalent w.h.p. can be obtained also for each high density variant of the subset sum problem, $SS(k,\ell)$, by using a d dimensional self masking, $\left[f^{(d)}\right]$, where $d-1 > \frac{k+c\log(n_k)}{\ell}$, as sketched below.

An input x to $f^{(d)}_{k,\ell}$ contains d independent instances of the problem, i.e. $x = (A_1, A_2, \ldots, A_d, \alpha)$, and the self masked version of $f^{(d)}$ is

$$\left[f^{(d)}\right]_{k,\ell}(x) = \left[f^{(d)}\right]_{k,\ell}(A_1, A_2, \ldots, A_d, \alpha) =$$
$$\left(A_1, A_2, \ldots, A_d, A_1\alpha^T \oplus A_2\alpha^T, A_1\alpha^T \oplus A_3\alpha^T, \ldots, A_1\alpha^T \oplus A_d\alpha^T\right),$$

Lemma 5. *Let $A_1, \ldots, A_d$ be mutually independent random vectors from $\mathcal{A}_{k,\ell}$, and let $c_2, \ldots, c_d$ be arbitrary integers. Then the probability that there exists a vector $\alpha \in \{0,1\}^k$ and integers $b_1, b_2, \ldots, b_d$ s.t. for each $i \in \{2, \ldots, d\}$ it holds that $c_i = b_1 \oplus b_i$ and $A_i \cdot \alpha^T = b_i$, is at most $2^{k-(d-1)\ell}$.*

Proof. First we note that $A_i \cdot \alpha^T = b_i$ iff $A_1\alpha^T \oplus A_i\alpha^T = c_i,\ i = 2, \ldots, d$. As in the proof of Lemma 4, we first fix A_1. Then we get that for a given α, and for each $i \in \{2, \ldots, d\}$:

$$Prob[A_1\alpha^T \oplus A_i\alpha^T = c_i \mid A_1, \alpha] \le 2^{-\ell}.$$

Since the A_i are mutually independent, we get that for a fixed α the probability that this equality holds for all $i \in \{2, \ldots, d\}$ is at most $2^{-\ell(d-1)}$. The lemma for a fixed A_1 follows by the union bound. Since A_1 is arbitrary, the lemma holds. □

Similarly to the one dimensional case, Lemma 5 implies:

Corollary 3 *If $(d-1)\ell > k + c\log n_{k,\ell}$, the probability that two independent random inputs, x_1, x_2, to $f^{(d)}_{k,\ell}$, satisfy $f^{(d)}_{k,\ell}(x_1) = f^{(d)}_{k,\ell}(x_2)$, is smaller than $(n_{k,\ell})^{-c}$.*

7 Concluding Remarks

In this paper we introduced the self masking technique, which aims at making the inversion of various polynomial time computable functions harder (see the informal idea sketch suggested in [4]). In the basic version, $[f](x)$, the self masking version of $f(x)$, replaces two "critical" parts of $f(x)$ by their bitwise xor. A straight forwards approach for solving the resulted computational task of computing $[f]^{-1}(x)$ requires examining numerous possible pairs of candidates for the xored parts. Thus inversion is hard unless there is a way to bypass this straight forwards approach in an efficient way. Specifically, this task is likely to be difficult if, w.h.p., computing the inverse of $[f](x)$ requires to reconstruct the original critical parts from their bitwise xor, i.e. if $[f]^{-1}([f](x)) = \{x\}$. As will be discussed in the full version the invesion task remains hard when the univalence requirement is relaxed to the case when $[f]^{-1}([f](x))$ is of small cardinality. We note that, apriori, a self masking $[f]$ of f could be hard to invert even if f can be inverted in polynomial time.

We applied this technique on well studied functions based on variants of the subset sum problem, where the critical parts were the sums of two independent solvable inputs for this problem. As we discussed, these sums are indeed critical for the polynomial time inversion algorithms surveyed in [11]. Thus it appears that these inversion algorithms cannot be directly applied to the self masked versions of subset sum problems presented in this paper.

Possible extensions. It is interesting if the self masking technique can be shown to harden the inversion of other polynomial time computable functions.

The practicality of the self masking technique depends heavily on the hardness to reconstruct the self masked parts. Ideally we would like it to imitate xor with one time pad. A promising way to approach this goal is to use instances from different functions, e.g. to mask a critical part of a function defined by an instance to the subset sum problem by a critical part of an instance to a different problem.

References

1. Brickell, E.F.: Solving low density knapsacks. In: Advances in cryptology. pp. 25–37. Springer (1984)
2. Brickell, E.F.: Breaking iterated knapsacks. In: Proceedings of CRYPTO 84 on Advances in Cryptology. p. 342–358. Springer-Verlag, Berlin, Heidelberg (1985)
3. Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: Improved low-density subset sum algorithms. Computational complexity **2**(2), 111–128 (1992)
4. Dolev, H., Dolev, S.: Toward provable one way functions. IACR Cryptol. ePrint Arch. p. 1358 (2020), https://eprint.iacr.org/2020/1358
5. Emiris, I.Z., Karasoulou, A., Tzovas, C.: Approximating multidimensional subset sum and minkowski decomposition of polygons. Mathematics in Computer Science **11**(1), 35–48 (2017)
6. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal of Computing **28**, 12–24 (1999)
7. Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. Journal of cryptology **9**(4), 199–216 (1996)
8. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. Journal of the ACM (JACM) **32**(1), 229–246 (1985)
9. Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: Theory of Cryptography Conference. pp. 382–400. Springer (2010)

10. Merkle, R., Hellman, M.: Hiding information and signatures in trapdoor knapsacks. IEEE transactions on Information Theory **24**(5), 525–530 (1978)
11. Odlyzko, A.M.: The rise and fall of knapsack cryptosystems. In: In Cryptology and Computational Number Theory. pp. 75–88. A.M.S (1990)
12. Shamir, A.: A polynomial-time algorithm for breaking the basic merkle - hellman cryptosystem. IEEE Transactions on Information Theory **30**(5), 699–704 (1984). https://doi.org/10.1109/TIT.1984.1056964