Small Superset and Big Subset Obfuscation

Steven D. Galbraith and Trey Li

Department of Mathematics, University of Auckland, Auckland, New Zealand
{s.galbraith,trey.li}@auckland.ac.nz

Abstract. Let $S = \{1, \ldots, n\}$ be a set of integers and X be a subset of S. We study the boolean function $f_X(Y)$ which outputs 1 if and only if Y is a small enough superset (resp., big enough subset) of X. Our purpose is to protect X from being known when the function is evasive, yet allow evaluations of f_X on any input $Y \subseteq S$. The corresponding research area is called function obfuscation. The two kinds of functions are called small superset functions (SSF) and big subset functions (BSF), respectively. In this paper, we obfuscate SSF and BSF in a very simple and efficient way. We prove both input-hiding security and virtual black-box (VBB) security based on the subset product problem.
We also give a proof of input-hiding based on the discrete logarithm problem (DLP) for the conjunction obfuscation by Bartusek et al. [5] (see Appendix A) and propose a new conjunction obfuscation based on SSF and BSF obfuscation (see Appendix B). The security of our conjunction obfuscation is from our new computational problem called the *twin subset product problem*.

1 Introduction

Let n be a positive integer and $S = \{1, \ldots, n\}$ be the set of integers from 1 to n. Let X be a subset of S. A small superset function (SSF) (resp., big subset function (BSF)) is a function $f_X(Y)$ which takes as input a set $Y \subseteq S$ and accepts if Y is a small superset of X (resp., big subset of X), or rejects otherwise. For example, let $S = \{1, \ldots, 1000\}$, let $X \subseteq S$ be a randomly chosen subset of size 800, and let $t = 900$ (resp., $t = 700$) be a threshold value. Then $f_X(Y) = 1$ if and only if $Y \supseteq X$ and the size of Y is at most 900 (resp., if and only if $Y \subseteq X$ and the size of Y is at least 700). Our goal is to protect X from being known when the function is evasive, yet allow the users to be able to determine whether Y is a small superset (resp., big subset) of X, for any $Y \subseteq S$.

The research area is called function obfuscation. A simple example of function obfuscation is point function obfuscation (think about password checkers), of which the goal is to hide a point $x \in \{0,1\}^n$, yet allow determinations on whether $x = y$, for any given input $y \in \{0,1\}^n$. A simple obfuscation for point functions is to hash x and to evaluate by comparing the hash of x and the hash of y.

Generally speaking, the goal of function obfuscation is to prevent a function from being recovered while preserving its functionality and time complexity. Due to the impossibility of general purpose obfuscation [2], special purpose obfuscation aims at obfuscating restricted classes of functions. An interesting class of

functions is the class of evasive functions [17]. They are the kind of functions that are hard to find an accepting input by random sampling. Examples of evasive functions include point functions [9,19], conjunctions [7,5], fuzzy Hamming distance matching [13], hyperplane membership functions [10], compute-and-compare functions [20,14], etc. Section 5 of [1] gives an impossibility result for obfuscating all evasive functions.

Previous works for SSF or BSF obfuscation are [4] and [6]. SSF was first introduced in [4] to construct public-key function-private encryption, while BSF was first introduced in [6] to better analyze and obfuscate conjunctions. Bartusek et al. [4] obfuscate SSF using similar techniques to [5]. The obfuscator in [5] is a dual scheme of Bishop et al.'s conjunction obfuscator [8]. However, the security proof of [4] is somewhat complicated and it lacks a discussion on input-hiding. Beullens and Wee [6] obfuscate BSF from a new knowledge assumption called the KOALA assumption, which is very strong.

In this paper we use the subset product problem to obfuscate SSF and BSF. This is a much more simple and trustworthy assumption. This gives the main motivation of the paper. The other motivation is to provide a construction that is simple, efficient, and has simpler security proofs than [4] and [6].

1.1 Technical Overview

In the rest of the paper we focus on SSF since BSF can be converted into SSF. To see this, let f_X with threshold t be a BSF. Then $f_{S\backslash X}$ with threshold $n-t$ is an SSF, where $S\backslash X$ is the complement of X. Also, for simplicity, we consider function families where the sets have fixed size w.

We represent a set $X \subseteq \{1,\ldots,n\}$ by its characteristic vector $x \in \{0,1\}^n$. Hence an SSF is a function $f_x : \{0,1\}^n \to \{0,1\}$ such that $f_x(y) = 1$ if and only if $y - x \in \{0,1\}^n$ and $|x| = w \leq |y| \leq t$ (where $|y|$ denotes the Hamming weight of y).

We explain the obfuscation as follows. The high level idea is to encode $x \in \{0,1\}^n$ as a subset product $X = \prod_{i=1}^n p_i^{x_i} \pmod q$ with respect to some small primes $p_1,\ldots,p_n$ and a larger prime modulus q so that if and only if an input $y \in \{0,1\}^n$ is a small superset of x (which implies that x and y have many bits in common) the product $\prod_{i=1}^n p_i^{y_i - x_i}$ is smaller than q and thus

$$YX^{-1} \pmod q = \prod_{i=1}^n p_i^{y_i - x_i} \pmod q = \prod_{i=1}^n p_i^{y_i - x_i}$$

factors over $\{p_1,\ldots,p_n\}$, where $Y = \prod_{i=1}^n p_i^{y_i} \pmod q$. We explain the idea explicitly as follows.

Let $n, t \in \mathbb{N}$ with $t < n$. Let $x = (x_1,\ldots,x_n) \in \{0,1\}^n$ with Hamming weight w and $r = t - w$. We require $r \leq n/2$. To obfuscate, the obfuscator samples n different small primes $p_1,\ldots,p_n$ from $\{2,\ldots,B\}$ for some sufficiently large $B \in \mathbb{N}$, and a safe prime q such that $B^r < q < (1+o(1))B^r$. It then computes the product $X = \prod_{i=1}^n p_i^{x_i} \pmod q$ and publishes $(p_1,\ldots,p_n,q,X)$ as the obfuscated function.

To evaluate with input $y = (y_1, \ldots, y_n) \in \{0,1\}^n$, the obfuscated function firstly checks if $w \leq |y| \leq t$. If not then it terminates and outputs 0. If $w \leq |y| \leq t$, then it further computes $Y = \prod_{i=1}^{n} p_i^{y_i} \pmod q$ and $E = YX^{-1} \pmod q = \prod_{i=1}^{n} p_i^{y_i - x_i} \pmod q$, and tries to factor E by dividing by the primes $p_1, \ldots, p_n$ one by one. If $y - x \in \{0,1\}^n$ then E factors over $\{p_1, \ldots, p_n\}$ and the function outputs 1. Otherwise, if $y - x \notin \{0,1\}^n$, which means y is not a superset of x, then with high probability E will not factor over $\{p_1, \ldots, p_n\}$ and the function outputs 0.

1.2 Organization

In Section 2 we introduce basic notions that are used in this paper and define evasive functions and function obfuscation. In Section 3 we define SSF and BSF and discuss their evasiveness. In Section 4 we define the subset product problem and reduce the discrete logarithm problem to both high and low density subset product problems. In Section 5 we present our obfuscation for SSF and BSF. In Section 6 we prove distributional virtual black-box (VBB) security and input-hiding security of our scheme based on the subset product problems. We discuss techniques for potential attacks to our scheme in Section 7. Section 8 is a brief conclusion.

2 Preliminaries

Let $S = \{1, \ldots, n\}$ be a set of positive integers from 1 to n, where $n \in \mathbb{N}$. Let X be a subset of S. The binary string $x \in \{0,1\}^n$ whose 1's indicate the elements of X is called the characteristic vector of X. In this paper we call a characteristic vector x a set, by which we mean the set X that it represents.

Let $x \in \{0,1\}^n$, by $|x|$ we mean the Hamming weight of x, which represents the size of the set X that x represents. Let C be a circuit, by $|C|$ we mean the size of C. Let $a \in \mathbb{R}$, by $|a|$ we mean the absolute value of a. We denote continuous intervals in the usual way as (a,b), $[a,b)$, $(a,b]$, or $[a,b]$, for $a, b \in \mathbb{R}$. We denote discrete intervals in the usual way as $\{a, \ldots, b\}$, for $a, b \in \mathbb{N}$. We denote the natural logarithm as $\ln a$, for $a \in \mathbb{R}$. We call a rational number a proper rational if it is not an integer. Let $f, g : \mathbb{N} \to \mathbb{R}$ be two functions. By $f \sim g$ we mean $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$ and by $f \prec g$ we mean $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

We say two distributions D_λ and E_λ are computationally indistinguishable for every probabilistic polynomial time (PPT) algorithm A, for every $\lambda \in \mathbb{N}$, if there exists a negligible function $\mu(\lambda)$ such that

$$\left| \Pr_{x \leftarrow D_\lambda} [A(x) = 1] - \Pr_{x \leftarrow E_\lambda} [A(x) = 1] \right| \leq \mu(\lambda),$$

denoted $D_\lambda \overset{c}{\approx} E_\lambda$. To be concrete, in the rest of the paper we take $\mu = 1/2^\lambda$.

2.1 Obfuscation

We use circuits to represent functions. By a circuit we always mean a circuit of minimal size that computes a specified function. The size complexity of a circuit of minimal size is polynomial in the time complexity of the function it computes.

Evasive Functions Evasive functions are the kind of Boolean functions that have small fiber of 1 compared to the domain of the function.

Definition 1 (Evasive Circuit Collection [1]). *A collection of circuits $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$, where C_λ takes $n(\lambda)$-bit input, is evasive if there exists a negligible function $\mu(\lambda)$ such that for all $\lambda \in \mathbb{N}$ and all $x \in \{0,1\}^{n(\lambda)}$,*

$$\Pr_{C\leftarrow C_\lambda}[C(x) = 1] \leq \mu(\lambda),$$

where the probability is taken over the random sampling of C_λ.

Input-Hiding Obfuscation The intuition of input-hiding is that given the obfuscated Boolean function, it should be inefficient for any PPT algorithm to find an element in the fiber of 1. We call elements of the fiber of 1 of a Boolean function *accepting inputs*.

Definition 2 (Input-Hiding [1]). *Let $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ be a circuit collection and $\mathcal{D}$ be a class of distribution ensembles $\mathcal{D} = \{D_\lambda\}_{\lambda\in\mathbb{N}}$, where D_λ is a distribution on C_λ. A probabilistic polynomial time (PPT) algorithm O is an input-hiding obfuscator for the family $\mathcal{C}$ and the distribution $\mathcal{D}$ if the following three conditions are met.*

1. Functionality Preserving: There is some negligible function $\mu(\lambda)$ such that for all $n \in \mathbb{N}$ and for all circuits $C \in \mathcal{C}$ with input size n we have

$$\Pr[\forall x \in \{0,1\}^n : C(x) = C'(x) \mid C' \leftarrow O(1^\lambda, C)] \geq 1 - \mu(\lambda),$$

where the probability is over the coin tosses of O.

2. Polynomial Slowdown: For every n, every circuit $C \in \mathcal{C}$, and every possible sequence of coin tosses for O, there exists a polynomial p such that the circuit $O(C)$ runs in time at most $p(|C|)$, i.e., $|O(C)| \leq p(|C|)$, where $|C|$ denotes the size of the circuit C.

3. Input-hiding Property: For every PPT adversary $\mathcal{A}$, for every $\lambda \in \mathbb{N}$ and for every auxiliary input $\alpha \in \{0,1\}^{poly(\lambda)}$ to $\mathcal{A}$, there exists a negligible function $\mu(\lambda)$ such that

$$\Pr_{C\leftarrow D_\lambda}[C(\mathcal{A}(O(C), \alpha)) = 1] \leq \mu(\lambda),$$

where the probability is taken over the random sampling of D_λ and the coin tosses of $\mathcal{A}$ and O.

Note that input-hiding is particularly defined for evasive functions. This is because non-evasive functions always leak accepting inputs. Also note that input-hiding is incomparable with VBB [1].

Virtual Black-Box Obfuscation The intuition of VBB obfuscation is that anything one can efficiently compute from the obfuscated function, one should be able to efficiently compute given just oracle access to the function [2]. It attempts to hide everything about a circuit without affecting the usage of the function it computes. We use the following variant of VBB.

Definition 3 (Distributional Virtual Black-Box Obfuscator (DVBB) [20]). *Let $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of polynomial size circuits. Let $\mathcal{D}$ be a class of distribution ensembles $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$, where D_λ is a distribution on C_λ and some polynomial size auxiliary information α. Let $\lambda \in \mathbb{N}$ be the security parameter. A PPT algorithm O is a VBB obfuscator for the distribution class $\mathcal{D}$ over the circuit family $\mathcal{C}$ if it satisfies the functionality preserving and polynomial slowdown properties in Definition 2 and the following property: For every (non-uniform) polynomial size adversary $\mathcal{A}$, there exists a (non-uniform) PPT simulator $\mathcal{S}$, such that for every distribution ensemble $D = \{D_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{D}$, and every (non-uniform) polynomial size predicate $\varphi : \mathcal{C} \to \{0,1\}$, there exists a negligible function $\mu(\lambda)$ such that:*

$$\left| \Pr_{(C,\alpha)\leftarrow D_\lambda}[\mathcal{A}(O(C),\alpha) = \varphi(C)] - \Pr_{(C,\alpha)\leftarrow D_\lambda}[\mathcal{S}^C(1^\lambda,\pi,\alpha) = \varphi(C)] \right| \leq \mu(\lambda), \quad (1)$$

where the first probability is taken over the coin tosses of $\mathcal{A}$ and O, the second probability is taken over the coin tosses of $\mathcal{S}$, π is a set of parameters associated to C (e.g., input size, output size, circuit size, etc.) which we are not required to hide, and $\mathcal{S}^C$ has black-box access to the circuit C.

Note that black-box access to evasive functions is useless. Hence it makes sense to consider a definition that does not give the simulator black-box access to the circuit C.

Definition 4 (Distributional-Indistinguishability [20]). *An PPT algorithm O for the distribution class $\mathcal{D}$ over a family of circuits $\mathcal{C}$, satisfies distributional-indistinguishability, if there exists a (non-uniform) PPT simulator $\mathcal{S}$, such that for every distribution ensemble $D = \{D_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{D}$, where D_λ is a distribution on $C_\lambda \times \{0,1\}^{poly(\lambda)}$, we have that*

$$(O(1^\lambda, C), \alpha) \overset{c}{\approx} (\mathcal{S}(1^\lambda,\pi),\alpha),$$

where $(C,\alpha) \leftarrow D_\lambda$, and α is some auxiliary information. I.e., there exists a negligible function $\mu(\lambda)$ such that:

$$\left| \Pr_{(C,\alpha)\leftarrow D_\lambda}[\mathcal{B}(O(1^\lambda,C),\alpha) = 1] - \Pr_{(C,\alpha)\leftarrow D_\lambda}[\mathcal{B}(\mathcal{S}(1^\lambda,\pi),\alpha) = 1] \right| \leq \mu(\lambda), \quad (2)$$

where the first probability is taken over the coin tosses of $\mathcal{B}$ and O, the second probability is taken over the coin tosses of $\mathcal{B}$ and $\mathcal{S}$.

Distributional-indistinguishability with auxiliary information $\alpha' = (\alpha, \varphi(C))$ implies DVBB with auxiliary information α [20], where $\varphi(C)$ is an arbitrary 1-bit predicate of the circuit. To state the theorem, we need the following definition of *predicate augmentation*, which allows to add an arbitrary 1-bit predicate of the circuit to the auxiliary information.

Definition 5 (Predicate Augmentation [3,20]). *For a distribution class $\mathcal{D}$, we define its augmentation under predicates, denoted $aug(\mathcal{D})$, as follows. For any (non-uniform) polynomial-time predicate $\varphi : \{0,1\}^* \to \{0,1\}$ and any $D = \{D_\lambda\} \in \mathcal{D}$ the class $aug(\mathcal{D})$ indicates the distribution ensemble $D' = \{D'_\lambda\}$ where D'_λ samples $(C, \alpha) \leftarrow D_\lambda$, computes $\alpha' = (\alpha, \varphi(C))$ and outputs (C, α').*

Theorem 1 (Distributional-Indistinguishability implies DVBB [20]). *For any family of circuits $\mathcal{C}$ and a distribution class $\mathcal{D}$ over $\mathcal{C}$, if an obfuscator O satisfies distributional-indistinguishability for the class of distributions $\mathcal{D}' = aug(\mathcal{D})$, i.e., if there exists a (non-uniform) PPT simulator $\mathcal{S}$, such that for every PPT distinguisher $\mathcal{B}$, for every distribution ensemble $D' = \{D'_\lambda\}$ where D'_λ samples $(C, \alpha) \leftarrow D_\lambda$ with $C \in \mathcal{C}$, computes $\alpha' = (\alpha, \varphi(C))$ and outputs (C, α'),*

$$\left| \Pr_{(C,\alpha') \leftarrow D'_\lambda} [\mathcal{B}(O(1^\lambda, C), \alpha') = 1] - \Pr_{(C,\alpha') \leftarrow D'_\lambda} [\mathcal{B}(\mathcal{S}(1^\lambda, \pi), \alpha') = 1] \right| \leq \mu(\lambda), \quad (3)$$

then it also satisfies DVBB security for the distribution class $\mathcal{D}$ (Definition 3).

Note that the auxiliary informations α in input-hiding is some global information for the whole function family, while the α in DVBB and distributional-indistinguishability are some local information about the specific function being sampled. The other commonly used names for global and local auxiliary information are independent and dependent auxiliary information, respectively.

3 Small Superset and Big Subset Functions

3.1 Function Definition

We define small superset and big subset functions in the following.

Definition 6 (Small Superset Function, SSF). *Let $x \in \{0,1\}^n$ be a characteristic vector of a subset of $\{1, \ldots, n\}$. A small superset function with respect to x is a function $f_x : \{0,1\}^n \to \{0,1\}, y \mapsto f_x(y)$ such that $f_x(y) = 1$ if and only if $y - x \in \{0,1\}^n$ and $|y| \leq t$, where $t \in \mathbb{N}$ with $0 \leq t \leq n$ is a threshold indicating "small".*

Definition 7 (Big Subset Function, BSF). *Let $x \in \{0,1\}^n$ be a characteristic vector of a subset of $\{1, \ldots, n\}$. A big subset function with respect to x is a function $f_x : \{0,1\}^n \to \{0,1\}, y \mapsto f_x(y)$ such that $f_x(y) = 1$ if and only if $x - y \in \{0,1\}^n$ and $|y| \geq t$, where $t \in \mathbb{N}$ with $0 \leq t \leq n$ is a threshold indicating "big".*

Note that we only need to study SSF obfuscation since BSF obfuscation can be reduced to SSF obfuscation. To see this, let f_x with threshold t be a BSF. Then $f_{\bar{x}}$ with threshold $n-t$ is an SSF, where $\bar{x}$ is the complement of x. So if we can obfuscate SSF we can also obfuscate BSF by firstly converting BSF to SSF.

Also, for simplicity and to simplify the security analysis, we consider function families where the sets have the same size. I.e., all x in the function family have the same Hamming weight.

3.2 Evasive Function Family

Denote by $B_{n,w}$ the set of binary strings of length n and Hamming weight w. Denote by $U_{n,w}$ the uniform distribution on $B_{n,w}$.

Only "evasive" SSF are interesting to obfuscate since x is immediately leaked once a small superset y of x is leaked. The attack is as follows. Let y be a small superset of x. The attacker flips the 1's of y one by one and queries the obfuscated function of f_x. If the y with a 1-position flipped is still a small superset of x, then the corresponding position of x is a 0; otherwise it is 1. Running through all 1's in y the attacker learns x. In particular, if all the flipped y's are rejected, then the attacker learns that $x = y$.

We define evasive SSF families in the following.

Definition 8 (Evasive SSF Family). *Let λ be the security parameter and n, t, w with $t \leq n$, $w \leq n$ be polynomial in λ. Let $\{X_n\}_{n\in\mathbb{N}}$ be an ensemble of distributions over $B_{n,w}$. The corresponding SSF family is said to be evasive if there exists a negligible function $\mu(\lambda)$ such that for every $\lambda \in \mathbb{N}$, and for every $y \in \{0,1\}^n$:*

$$\Pr_{x\leftarrow X_n}[f_x(y) = 1] \leq \mu(\lambda). \tag{4}$$

Now we consider parameters for evasive function family. Let us start with uniform distributions $\{X_n\}_{n\in\mathbb{N}} = \{U_{n,w}\}$, remembering that n and w are polynomials in λ.

If $|y| < w$ or $|y| > t$, then y will never be a small superset of any x with Hamming weight w hence Inequality (4) always holds. If $w \leq |y| \leq t$, then there are at most $\binom{t}{w}$ many x with Hamming weight $|x| = w$ such that y is a superset of x, Inequality (4) holds if and only if

$$\binom{t}{w}/\#B_{n,w} = \binom{t}{w}/\binom{n}{w} \leq 1/2^{\lambda}. \tag{5}$$

An asymptotic way to see this inequality is $t^w/n^w \leq 1/2^{\lambda}$. Also note that this is the most basic requirement for t in the sense that it is obtained under the best possible (i.e., highest entropy) distributions.

Now we consider general distributions $\{X_n\}_{n\in\mathbb{N}}$. We first define the following entropy.

Definition 9 (Conditional Small Superset Min-Entropy). *Let $0 \leq t \leq n \in \mathbb{N}$. The small superset min-entropy of a random variable X conditioned on a correlated variable Y is defined as*

$$H_{Sup,\infty}(X \mid Y) = -\ln\left(\mathbb{E}_{y \leftarrow Y}[\max_{y \in \{0,1\}^n} \Pr[y - X \in \{0,1\}^n, |y| \leq t \mid Y = y]]\right).$$

Now let us see what exactly Inequality (4) means. In words, it means that for every $y \in \{0,1\}^n$, an x sampled from the distribution X_n has negligible probability to have y as a small superset. Intuitively, this requires that in the space $\{0,1\}^n$, the number of points x representing SSF is large enough and at the same time they are "well spread out" in the sense that small superset relations between points occur sparsely and evenly in the space. Rigorously, the following requirement implies Inequality (4). where we now include auxiliary information.

Definition 10 (Small Superset Evasive Distribution). *Let $X = \{(X_n, \alpha_n)\}_{n \in \mathbb{N}}$ be an ensemble of distributions on $B_{n,w} \times \{0,1\}^{poly(\lambda)}$. We denote a sample from X_n as (x, α) where $\alpha \in \{0,1\}^{poly(\lambda)}$ is considered to be auxiliary information about x. We say that X is small superset evasive if the conditional small superset min-entropy of x conditioned on α (as in Definition 9) is at least λ.*

Note that asking for a small superset is a stronger question than asking for a "close" set. Hence the above requirement is somehow looser than the evasiveness requirement for fuzzy Hamming distance matching. Intuitively, in the case of fuzzy Hamming distance matching, we require that the points in the Hamming space are spread out such that their Hamming balls do not overlap too seriously; while in the case of SSF, the Hamming balls can overlap more seriously. For example, let $x = (01||c)$ and $y = (10||c)$ be two strings with only the first two bits different, where $c \in \{0,1\}^{n-2}$. We can see that x and y have very small Hamming distance $|x \oplus y| = 2$, but neither of them is a superset of the other.

This means that in the same space $\{0,1\}^n$, there are more evasive SSF distributions than evasive fuzzy Hamming distance matching distributions.

Nonetheless, our obfuscation for SSF has to work under the stronger requirement of evasive Hamming distance matching. This is because an attacker can always recover the secret x in our scheme by merely finding a "close" set and not necessarily a small superset. We therefore use the following definition 12 for evasiveness of SSF.

Definition 11 (Conditional Hamming Ball Min-Entropy [13,12]). *The Hamming ball min-entropy of random variables X conditioned on a correlated variable Y is*

$$H_{Ham,\infty}(X \mid Y) = -\ln\left(\mathbb{E}_{y \leftarrow Y}[\max_{y \in \{0,1\}^n} \Pr[|X \oplus y| \leq r \mid Y = y]]\right),$$

where $r < n \in \mathbb{N}$.

Definition 12 (Hamming Distance Evasive Distribution [13]). *Let λ be the security parameter and n, t, w with $t \leq n$, $w \leq n$ be polynomial in λ. Let $X =$*

$\{X_n\}_{n\in\mathbb{N}}$ be an ensemble of distributions on $B_{n,w} \times \{0,1\}^{poly(\lambda)}$. We say that the distribution X_n is Hamming distance evasive if for all $\lambda \in \mathbb{N}$, the conditional Hamming ball min-entropy of x conditioned on α (as in Definition 11 with $r := t - w$) is at least λ.

4 Subset Product Problem

This section is about the computational problem that our obfuscation is based on.

Let us keep in mind that all parameters are functions in λ with $\lambda \leq n$. The subset product problem is the following.

Definition 13 (Subset Product Problem, SP [13]). *Given $n+1$ distinct primes $p_1, \ldots, p_n, q$ and an integer $X \in \mathbb{Z}_q^*$, find a vector $(x_1, \ldots, x_n) \in \{0,1\}^n$ (if it exists) such that $X = \prod_{i=1}^n p_i^{x_i} \pmod q$.*

The decisional version is the following.

Definition 14 (Decisional Subset Product Problem, d-SP [13]). *Given $n+1$ distinct primes $p_1, \ldots, p_n, q$ and an integer $X \in \mathbb{Z}_q^*$, decide if there exists a vector $(x_1, \ldots, x_n) \in \{0,1\}^n$ such that $X = \prod_{i=1}^n p_i^{x_i} \pmod q$.*

In order to define hard SP, we avoid parameters that will make the problem trivial.

If $q \geq \prod_{i=1}^n p_i^{x_i}$, then x_i is immediately leaked by checking whether $p_i \mid X$. Hence we require $q < \prod_{i=1}^n p_i^{x_i}$. In particular, we can set q to lie between a length r prime product and a length $r+1$ prime product, for some suitably chosen $r < n$.

Now if $r \geq n/2$, the problem is still trivial. One can just sample a uniform $y \in \{0,1\}^n$ and decode x from $XY^{-1} \pmod q$ using the naive and improved attacks discussed in Section 7, where $Y = \prod_{i=1}^n p_i^{y_i} \pmod q$. The naive attack works when the Hamming distance between x and y is $\leq r$. Note that a uniform y is expected to be $n/2$ away from x. Hence if $r \geq n/2$, the naive attack is expected to work. To avoid this, we require negligible probability of y being r-close to x. I.e., $\Pr_{y \leftarrow \{0,1\}^n}[|x \oplus y| \leq r] \leq 1/2^\lambda$, where $\oplus$ denotes the XOR operation. For uniformly sampled x and y, this gives

$$r \leq \frac{n}{2} - \sqrt{\lambda n \ln 2}. \tag{6}$$

For a proof of this, see Lemma 2 in [13].

Again, if x is from a low entropy distribution, finding a point y close to x is easy. For example, suppose all points cluster together. Then one can just find y by searching in the cluster. To avoid this, we require the distribution of x to have conditional Hamming ball min-entropy (as defined by Definition 11) λ.

Now we are ready to define the hard SP distribution.

Definition 15 ((n, r, B, X_n)-SP Distribution). *Let λ, n, r, B be positive integers with $n \geq \lambda$ polynomial in λ, r satisfies Inequality (6), and B larger than the n-th prime. Let X_n be a distribution over $\{0,1\}^n$ with Hamming ball min-entropy λ. Let $(x_1 \ldots, x_n) \leftarrow X_n$. Let $p_1, \ldots, p_n$ be distinct primes uniformly sampled from the primes in $\{2, \ldots, B\}$. Let q be a uniformly sampled safe prime in $\{B^r, \ldots, (1 + o(1))B^r\}$. Then we call the distribution $(p_1, \ldots, p_n, q, X)$ with $X = \prod_{i=1}^{n} p_i^{x_i} \pmod q$ the (n, r, B, X_n)-SP distribution.*

The hard SP and hard d-SP are the following.

Assumption 2 (Hard SP) *Let λ, n, r, B, X_n satisfy the conditions in Definition 15. Then for every PPT algorithm $\mathcal{A}$ and every $\lambda \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that the probability that $\mathcal{A}$ solves SP of instances sampled from the (n, r, B, X_n)-SP distribution is not greater than $\mu(\lambda)$.*

Assumption 3 (Hard d-SP) *Let λ, n, r, B, X_n satisfy the conditions in Definition 15. Let $D_0 = (p_1, \ldots, p_n, q, X)$ be the (n, r, B, X_n)-SP distribution and let D_1 be D_0 with $X = \prod_{i=1}^{n} p_i^{x_i} \pmod q$ replaced by a random element in $\mathbb{Z}_q^*$. Then for every PPT algorithm $\mathcal{A}$ and every $\lambda \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that*

$$\left| \Pr_{d_0 \leftarrow D_0} [\mathcal{A}(d_0) = 1] - \Pr_{d_1 \leftarrow D_1} [\mathcal{A}(d_1) = 1] \right| \leq \mu(\lambda). \tag{7}$$

By the search-to-decision reductions in [16] and [18], one can show that d-SP is at least as hard as SP. In the following we show that SP is at least as hard as DLP for certain parameter ranges. This gives evidence that SP is hard. An informal statement of this result for high density SP was given in [13], where the density of an SP instance is defined as $d := n/\log_2 q$. In the following we give a rigorous proof for both high and low density SP.

Definition 16 (Discrete Logarithm Problem, DLP). *Let G be a finite group of order N written in multiplicative notation. The discrete logarithm problem is given $g, h \in G$ to find a (if it exists) such that $h = g^a$.*

Assumption 4 (Hard DLP) *Let $\mathbb{Z}_q^*$ be the multiplicative group of integers modulo q, where $q = 2p + 1 \geq 2^\lambda$ is a safe prime for some prime p. If g is sampled uniformly from $\mathbb{Z}_q^*$ and a is sampled uniformly from $\{0, \ldots, q-2\}$, then for every PPT algorithm $\mathcal{A}$ and every $\lambda \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that the probability that $\mathcal{A}$ solves the DLP (g, g^a) is not greater than $\mu(\lambda)$.*

Two heuristics are needed for the reduction.

Heuristic 5 *The number of elements $X \in \mathbb{Z}_q^*$ being a subset product $\prod_{i=1}^{n} p_i^{x_i} \pmod q$ over the (n, r, B, U_n)-SP distribution $(p_1, \ldots, p_n, q, X)$ with $q \leq 2^n p(n)$ is $\geq q/p(n)$, for polynomial $p(n)$, where U_n is the uniform distribution.*

This means that if q is not larger than polynomial times 2^n, then a uniformly chosen X from $\mathbb{Z}_q^n$ is a subset product with noticeable probability. Also notice that the requirement $q \leq 2^n p(n)$ captures both high and low density SP.

Heuristic 6 *The number of random DLP group elements g^a needed for getting polynomially many SP solutions x that span $\mathbb{Z}_\ell^n$ for each prime factor ℓ of $q-1$ is polynomial, where $x \in \{0,1\}^n$ is such that $g^a = \prod_{i=1}^n p_i^{x_i} \pmod q$ and SP and DLP are as defined in Assumption 2 and 4.*

This means that when writing different DLP group elements g^a in terms of subset products $\prod_{i=1}^n p_i^{x_i} \pmod q$ with respect to some random primes $p_1, \ldots, p_n$, the exponent vectors $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ have high probability to give a full rank matrix over $\mathbb{Z}_\ell$, for each prime factor ℓ of $q-1$. This makes sense if we think about the randomness of the primes $p_1, \ldots, p_n$.

Also note that q is a safe prime, i.e., $q-1 = 2p$ has only two prime factors 2 and p. Hence it is not a serious requirement since there are only two spaces $\mathbb{Z}_2^n$ and $\mathbb{Z}_p^n$ needed to satisfy.

Theorem 7. *Assuming Heuristic 5 and 6, if there exists a PPT algorithm to solve SP (as defined in Assumption 2, with $q \leq 2^n p(n)$) with overwhelming probability in time T, then there exists an algorithm to solve DLP (as defined in Assumption 4) in expected time $O(t(\lambda)T)$, for some polynomial $t(\lambda)$.*

Proof. Let (g, h) be a DLP instance as defined in Assumption 4. Let $\mathcal{A}$ be a PPT algorithm that solves SP as defined in Assumption 2, with the same q as the DLP. We solve the DLP as follows. Sample a uniform a from $\{0, \ldots, q-2\}$, then call $\mathcal{A}$ to solve $(p_1, \ldots, p_n, q, g^a)$. If g^a is a subset product, then with overwhelming probability $\mathcal{A}$ can solve for an $x \in \{0,1\}^n$ such that $g^a = \prod_{i=1}^n p_i^{x_i} \pmod q$. Since $q \leq 2^n p(n)$, by Heuristic 5 we have that $n_{\mathrm{SP}} \geq q/p(n)$, where n_{SP} is the number of subset products in $\mathbb{Z}_q^*$. Hence the probability that g^a being a subset product is $\geq 1/p(n)$. We therefore expect that after $np(n)$ samples of a, we can solve for n vectors $x \in \{0,1\}^n$ such that $a \equiv \sum_{i=1}^n x_i \log_g(p_i) \pmod{q-1}$.

Also by Heuristic 6, with at most $np(n)p'(n)$ samples of a, we expect to be able to choose n vectors $x \in \{0,1\}^n$ to span $\mathbb{Z}_\ell^n$ for each prime factor ℓ of $q-1$, for some polynomial $p'(n)$. We therefore have n relations $a \equiv \sum_{i=1}^n x_i \log_g(p_i) \pmod \ell$ whose coefficient matrix is full rank, for each prime factor ℓ of $q-1$. Then we can solve the systems of equations for different ℓ respectively and use the Chinese remainder theorem to lift the solutions to $\mathbb{Z}_{q-1}$, obtaining $\log_g(p_i) \pmod{q-1}$ for all $i \in \{1, \ldots, n\}$.

Lastly we sample $b \leftarrow \{1, \ldots, q-1\}$, compute $hg^b \pmod q$, and call $\mathcal{A}$ to solve it. With at most $p(n)$ extra samples of b, we expect one more relation $\log_g(h) + b \equiv \sum_{i=1}^n x_i \log_g(p_i) \pmod{q-1}$ with $x \in \{0,1\}^n$. Then $\log_g(h) = \sum_{i=1}^n x_i \log_g(p_i) - b \pmod{q-1}$. □

5 Obfuscation

The obfuscator is the following.

Algorithm 1 SSF Obfuscator

Input: $n, t, r, w \in \mathbb{N}$, $x \in \{0,1\}^n$ with $r := t - w \leq n/2 - \sqrt{\lambda n \ln 2}$
Output: $((p_1, \ldots, p_n) \in \mathbb{N}^n, q \in \mathbb{N}, X \in \mathbb{Z}_q^*)$

1: sample distinct primes $p_1, \ldots, p_n$ from $\{2, \ldots, B\}$ where $B = 3n \ln n$
2: sample safe prime q from $\{B^r, \ldots, 3B^r\}$
3: compute $X = \prod_{i=1}^{n} p_i^{x_i} \mod q$
4: **return** $((p_1, \ldots, p_n), q, X)$

Note that in Algorithm 1 we require $r \leq \frac{n}{2} - \sqrt{\lambda n \ln 2}$ due to Inequality (6).

The following factoring algorithm (Algorithm 2) is a sub-procedure of the evaluation algorithm (Algorithm 3).

Algorithm 2 Factor

Input: $n \in \mathbb{N}$, $(p_1, \ldots, p_n) \in \mathbb{N}^n$, $a \in \mathbb{N}$
Output: 0 or 1

1: **for** $i = 1, \ldots, n$ **do**
2: **if** $p_i \mid a$ **then** $a \leftarrow a/p_i$
3: **end for**
4: **return** 1 **if** $a = 1$ **else** 0

The evaluation algorithm is the following.

Algorithm 3 SSF Evaluation (with embedded data $(p_1, \ldots, p_n) \in \mathbb{N}^n$, $q \in \mathbb{N}, X \in \mathbb{Z}_q^*$)

Input: $y \in \{0,1\}^n$
Output: 0 or 1

1: $F \leftarrow 0$
2: **if** $w \leq |y| \leq t$ **then**
3: compute $Y = \prod_{i=1}^{n} p_i^{y_i} \pmod q$
4: compute $E = YX^{-1} \pmod q$
5: compute $F \leftarrow \mathsf{Factor}(n, (p_1, \ldots, p_n), E)$
6: **end if**
7: **return** 1 **if** $F = 1$ **else** 0

5.1 Correctness

Note that the inputs y with $|y| < w$ or $|y| > t$ will always be correctly rejected. We therefore only discuss the case where $w \leq |y| \leq t$.

Let $E = YX^{-1} \pmod q = \prod_{i=1}^{n} p_i^{e_i} \pmod q$ with $e = (e_1, \ldots, e_n) = y - x \in \{-1, 0, 1\}^n$. If y is a small superset of x, then $e \in \{0,1\}^n$ and $|e| \leq r$, hence

$\prod_{i=1}^{n} p_i^{e_i} < B^r < q$. This means E is a product of primes in $\{p_1, \ldots, p_n\}$ hence will be reduced to 1 in Factor and y will be correctly accepted by Algorithm 3.

If y is not a small superset of x, then it will either (1) result in some E which contains a prime factor not in $\{p_1, \ldots, p_n\}$ or $e \notin \{0,1\}^n$; or (2) result in some E such that E is still a product of primes in $\{p_1, \ldots, p_n\}$. The former case will be correctly rejected by Algorithm 3. The latter case will be falsely accepted. We therefore call a $y \in \{0,1\}^n$ a *false positive* if it is not a small superset of x but is accepted by Algorithm 3.

Avoiding False Positives Using Lattice Arguments Now we discuss how to avoid false positives.

Let y be a false positive. We have that $E = \prod_{i=1}^{n} p_i^{y_i - x_i} \pmod q = \prod_{i=1}^{n} p_i^{e_i} \pmod q$ with $\prod_{i=1}^{n} p_i^{e_i} < q$ and $e = (e_1, \ldots, e_n) \in \{0,1\}^n$. I.e., $\prod_{i=1}^{n} p_i^{y_i - x_i - e_i} = 1 \pmod q$ with $y - x - e \neq 0$. This implies a nonzero short vector $z \in \{-2, -1, 0, 1\}^n$ of length $\leq 2\sqrt{n}$ in the lattice

$$L = \left\{ z \in \mathbb{Z}^n \;\middle|\; \prod_{i=1}^{n} p_i^{z_i} = 1 \pmod q \right\}.$$

To avoid false positives, it is sufficient that the shortest vector in the above lattice is longer than $2\sqrt{n}$. If the primes $p_1, \ldots, p_n$ are sufficiently random, which means that the lattice is sufficiently random, then we can employ the Gaussian heuristic to estimate the length of the shortest vector as

$$\lambda_1 \sim \sqrt{\frac{n}{2\pi e}} \operatorname{vol}(L)^{\frac{1}{n}}.$$

Also, by the first isomorphism theorem, the volume of the lattice $\operatorname{vol}(L)$ is given by the size of the image $|\operatorname{im} \phi|$ of the group morphism

$$\phi : \mathbb{Z}^n \to \mathbb{Z}_q^*,$$

$$(x_1, \ldots, x_n) \mapsto \prod_{i=1}^{n} p_i^{x_i} \pmod q$$

whose kernel defines L. Hence

$$\operatorname{vol}(L) \leq \varphi(q) = q - 1,$$

where φ is the Euler totient function. The equality holds if and only if $\{p_1, \ldots, p_n\}$ generates $\mathbb{Z}_q^*$. So

$$\lambda_1 \sim \sqrt{\frac{n}{2\pi e}} \operatorname{vol}(L)^{\frac{1}{n}} \leq \sqrt{\frac{n}{2\pi e}} (q-1)^{\frac{1}{n}} < \sqrt{\frac{n}{2\pi e}} q^{\frac{1}{n}}.$$

If we take $\lambda_1 = \sqrt{\frac{n}{2\pi e}} q^{\frac{1}{n}}$ and $q \sim (n \ln n)^r$, for $\lambda_1 > 2\sqrt{n}$ we require that

$$r > \frac{n \ln(2\sqrt{2\pi e})}{\ln(n \ln n)}. \tag{8}$$

If we satisfy this condition on r then heuristically there are no false positives.

Evidence for the Gaussian Heuristic in These Lattices To provide evidence for the Gaussian heuristic on the relation lattice L, we give some experimental results. Due to the limitation of computational resources, we only work with small parameters such as $n = 20$ or 30 or 40, $r = \lfloor \frac{n}{\ln n} \rfloor$ (which is an appropriate choice as we will be discussing in the later section about parameters), and $B = 3n \ln n$.

Let λ_1 denote the length of the shortest vector in a lattice and let γ denote the Gaussian heuristic. For each $n = 20$ or 30 or 40, we create 1000 lattices L from random subset products, calculate the proportion of lattices that λ_1/γ falls into the 20 intervals $[0.0, 0.1), [0.1, 0.2), \ldots, [1.9, 2.0]$, respectively. The results are as follows.

When $n = 20$, $r = \lfloor \frac{n}{\ln n} \rfloor$, $B = 3n \ln n$, the sequence of proportions is:

$$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \frac{9}{20}, \frac{11}{20}, 0, 0, 0, 0, 0, 0, 0, 0).$$

When $n = 30$, $r = \lfloor \frac{n}{\ln n} \rfloor$, $B = 3n \ln n$, the sequence of proportions is:

$$(0, 0, 0, 0, 0, 0, 0, 0, \frac{2}{1000}, \frac{26}{1000}, \frac{399}{1000}, \frac{557}{1000}, \frac{16}{1000}, 0, 0, 0, 0, 0, 0, 0).$$

When $n = 40$, $r = \lfloor \frac{n}{\ln n} \rfloor$, $B = 3n \ln n$, the sequence of proportions is:

$$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \frac{29}{1000}, \frac{702}{1000}, \frac{269}{1000}, 0, 0, 0, 0, 0, 0, 0).$$

We can see that for most cases $\lambda_1/\gamma \in [1.0, 1.2]$, which means that the Gaussian heuristic is quite close to the true length of the shortest vectors most of the time. Also λ_1 tends to be larger than γ, which gives more confidence in Inequality (8) to avoid false positives.

Dealing with False Positives by Hashing Another way to deal with false positives is to use a hash function or a point function obfuscator. Let us take hash as an example. To avoid false positives, all we need to do is to compute and output an extra value $h = H(x)$ in Algorithm 1, where H is a collision resistant hash function modeled as a random oracle; and in Factor, store the factors of E in a list F and replace "return 1" with "return F"; also in Algorithm 3, add process to recover x from F and compare its hash value against $H(x)$. If y is a small superset of x, then the factors of E will tell the positions of the distinct bits between x and y, then one can recover x by flipping y at those positions. Otherwise if y is a false positive, then doing so will give a wrong $x' \neq x$ which can be detected by comparing the hash values.

5.2 Parameters For Secure Obfuscation

Restrictions for the parameters λ, n, t, r, and q are as follows.

(1) For evasiveness, the basic requirement is Inequality (5).

(2) For the hardness of finding a y close to x such that it decodes (which will recover x), we require r to be small enough, i.e., the Hamming ball of any x should be small enough. This requires $r(n) \leq n/2 - \sqrt{\lambda n \ln 2}$. (Inequality (6)).

(3) To avoid false positives without using a hash function or a point function obfuscator, we require $r > \frac{n \ln(2\sqrt{2\pi e})}{\ln(n \ln n)}$ (Inequality (8)).

From (2) and (3) we have that

$$\frac{n \ln(2\sqrt{2\pi e})}{\ln(n \ln n)} < r(n) \leq \frac{n}{2} - \sqrt{n\lambda \ln 2}.$$

Notice that

$$\frac{n \ln(2\sqrt{2\pi e})}{\ln(n \ln n)} \prec \frac{n}{\ln n} \prec \frac{n}{\ln \ln n} \prec \frac{n}{2} - \sqrt{n\lambda \ln 2},$$

both $r(n) \sim \frac{n}{\ln n}$ and $r(n) \sim \frac{n}{\ln \ln n}$ are possible functions for r, where by $f \sim g$ we mean $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$ and by $f \prec g$ we mean $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

We take $r(n) = \lfloor \frac{n}{\ln n} \rfloor$. Then the condition $r \leq \frac{n}{2} - \sqrt{n\lambda \ln 2}$ gives

$$\begin{aligned} & \sqrt{n\lambda \ln 2} \leq n \left(\frac{1}{2} - \frac{1}{\ln n} \right) \\ \iff & \lambda \leq \frac{n}{\ln 2} \left(\frac{1}{2} - \frac{1}{\ln n} \right)^2 \\ \impliedby & \lambda \leq \frac{n}{6}, \end{aligned}$$

where for the last line we assume $n \geq 1024$.

Hence a possible function family for the uniform distribution $B_{n,w}$ is $(n = 6\lambda, r = \lfloor n/\ln(n) \rfloor)$. In terms of t, it is $(n = 6\lambda, t = w + \lfloor n/\ln(n) \rfloor)$. A concrete setting is: $\lambda = 128$; $n = 1024$; $t = 659$; $w = 512$; $B = 8161$ (the 1024-th prime, 13 bits); $q \approx 2B^r$ (about 1912 bits); X_n has conditional Hamming ball min-entropy λ. Note that this requirement on X_n is easy to achieve with the settings of n, w and t, because n is much larger than λ and there is a big gap between a λ min-entropy distribution and the uniform distribution. Even when we consider auxiliary information which reduces the entropy a little bit, it is still easy to have a λ min-entropy distribution conditioned on the auxiliary information.

Note that an elementary requirement is that $w > r$ since otherwise the encoding of x, namely $\prod_{i=1}^{n} p_i^{x_i} \pmod q$ will always be factorable and x will be exposed immediately. Also notice that $r(n) \sim n/\ln(n) \sim \pi(n)$, namely the function for r is the prime counting function.

6 Security Proofs

The security is based on hardness assumptions that are slightly different from Assumption 2 and 3. We consider SP and d-SP over points $x \in \{0,1\}^n$ with fixed Hamming weight $w \approx n/2$ and with auxiliary information given.

The following assumption serves the proof of input-hiding, which involves some global auxiliary information $\alpha \in \{0,1\}^{poly(\lambda)}$ about the whole function family.

Assumption 8 (Hard SP with Global Auxiliary Information) *Let X_n be a distribution on $B_{n,w}$ where $n/2 - n/8 \leq w \leq n/2 + n/8$. Let $\alpha \in \{0,1\}^{poly(\lambda)}$ be auxiliary information. For every PPT algorithm A, for every $\lambda \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that the probability that A, provided with α, solves an SP sampled from the (n, r, B, X_n)-SP distribution is not greater than $\mu(\lambda)$.*

We then state the hard d-SP assumption which serves the proof of DVBB. Different from Assumption 8 where there is only one α for the entire function family, the following Assumption 9 assumes that d-SP is hard even given auxiliary information α about the specific x sampled from X_n. Furthermore, for convenience in proving distributional-indistinguishabiliy, we define the d-SP problem in the "predicate-augmentation" style (as in Definition 5), namely to define it over a distribution D'_b which outputs $\alpha' = (\alpha, \varphi(x))$ instead of just α, for any (non-uniform) polynomial size predicate $\varphi : X_n \to \{0,1\}$, where $b \in \{0,1\}$.

Assumption 9 (Hard d-SP with Local Auxiliary Information) *Fix a (non-uniform) polynomial time predicate $\varphi : \{0,1\}^n \to \{0,1\}$. Let X_n be a distribution on $B_{n,w} \times \{0,1\}^{poly(\lambda)}$ which samples (x, α) with α some auxiliary information about x that satisfies Definition 12 (i.e., the conditional Hamming ball min-entropy of the distribution X_n conditioned on α is still at least λ). Let $X'_n = (x, \alpha')$ be a distribution on $B_{n,w} \times \{0,1\}^{poly(\lambda)} \times \{0,1\}$, where $\alpha' = (\alpha, \varphi(x))$. Let $D'_0 = (p_1, \ldots, p_n, q, X, \alpha')$ be the (n, r, B, X_n)-SP distribution corresponding to X'_n. Let D'_1 be D'_0 with $X = \prod_{i=1}^n p_i^{x_i} \pmod q$ replaced by uniformly sampled $X' \leftarrow \mathbb{Z}_q^*$, but all other terms the same. Then for every PPT algorithm A, for every $\lambda \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that*

$$\left| \Pr_{d_0 \leftarrow D'_0}[A(d_0) = 1] - \Pr_{d_1 \leftarrow D'_1}[A(d_1) = 1] \right| \leq \mu(\lambda). \tag{9}$$

6.1 Input-Hiding

Now we show input-hiding from the hardness of SP.

Theorem 10. *Let n, t, r, B satisfy Definition 15, the Gaussian Heuristic and Inequality (8). Then assuming the hardness of SP (Assumption 8), the SSF obfuscator given by Algorithm 1 is input-hiding.*

Proof. Let $(p_1, \ldots, p_n, q, X)$ with $X = \prod_{i=1}^n p_i^{x_i} \pmod q$ for some unknown $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ be an SP instance defined in Assumption 8, and $aux \in \{0,1\}^{poly(\lambda)}$ be some auxiliary information. Let A be a PPT algorithm that breaks input-hiding of the obfuscation given by Algorithm 1-3. Then we solve the SP as follows. We directly call A on input $((p_1, \ldots, p_n, q, X), aux)$. Since

r satisfies Inequality (6), i.e., there are no false positives, A will return a small superset y of x such that $E = YX^{-1} \pmod q = \prod_{i=1}^{n} p_i^{y_i - x_i} \pmod q = \prod_{i=1}^{n} p_i^{e_i}$ with $e = (e_1, \ldots, e_n) \in \{0,1\}^n$. Then we can factor E to get e and recover x by flipping y at the positions i such that $e_i = 1$. □

6.2 DVBB

We show DVBB from the hardness of d-SP.

Theorem 11. *Let X_n be a distribution over $B_{n,w} \times \{0,1\}^{poly(\lambda)}$ with conditional (on α) Hamming ball min-entropy λ. Then assuming Assumption 9, the obfuscation given by Algorithm 1 - 3 is DVBB (with heuristic correctness if we use the lattice technique to avoid false positives).*

Proof. Functionality preservation and polynomial slowdown are shown in the proof of Theorem 10. Now we show distribuional VBB. We show distributional-indistinguishability, which implies DVBB by Theorem 1. Fix a predicate φ. For every circuit $C \leftarrow C_\lambda$ (which contains the secret $x \leftarrow X_n$), let $O(1^\lambda, C) = (p_1, \ldots, p_n, q, X)$ be the obfuscated function of C. We define a simulator S which works as follows: S takes $\pi = (n, t, B)$ samples n primes $p'_1, \ldots, p'_n$ and a modulus q' in the same way as O, and samples $X' \leftarrow \mathbb{Z}_q$. Denote $S(1^\lambda, \pi) = (p'_1, \ldots, p'_n, q', X')$. We will show that the two probabilities in Inequality (3) equal to the two probabilities in Inequality (9) respectively.

For the first equality, we have that for every PPT distinguisher $\mathcal{A}$, for every $\lambda \in \mathbb{N}$,

$$\Pr_{(x,\alpha') \leftarrow X'_n}[\mathcal{A}(p_1, \ldots, p_n, q, X, \alpha') = 1] = \Pr_{d_0 \leftarrow D'_0}[\mathcal{A}(d_0) = 1],$$

where $d_0 = (p_1, \ldots, p_n, q, X, \alpha')$ and both probabilities are over the randomness of $x, p_1, \ldots, p_n$, q and α'. This holds simply from the definition of D'_0 (as in Assumption 9).

Replace x with C, X'_n with D'_λ, and $p_1, \ldots, p_n, q, X$ with $O(1^\lambda, C)$ we have that

$$\Pr_{(C,\alpha') \leftarrow D'_\lambda}[\mathcal{A}(O(1^\lambda, C), \alpha') = 1] = \Pr_{d_0 \leftarrow D'_0}[\mathcal{A}(d_0) = 1], \tag{10}$$

where the first and the second probabilities are the first probabilities of Inequality (3) and Inequality (9) respectively.

For the second equality, we have that for every PPT distinguisher $\mathcal{A}$, for every $\lambda \in \mathbb{N}$,

$$\Pr_{(x,\alpha') \leftarrow X'_n}[\mathcal{A}(p'_1, \ldots, p'_n, q', X', \alpha') = 1] = \Pr_{d_1 \leftarrow D'_1}[\mathcal{A}(d_1) = 1],$$

where $d_1 = (p'_1, \ldots, p'_n, q', X', \alpha')$ and the probability is over the randomness of $x, p'_1, \ldots, p'_n, q'$, X' and α'. This holds from the definition of D'_1 (as in Assumption 9). Note that the α' in both probabilities are the same α' as in Equation

(10), which is the auxiliary information about the unique real x sampled at the beginning of the game. In particular the α' in d_1 is not generated by the simulator but copied from the left hand side.

Replace x with C, X'_n with D'_λ, and $p'_1, \ldots, p'_n, q', X'$ with $S(1^\lambda, \pi)$ we have that

$$\Pr_{(C,\alpha')\leftarrow D'_\lambda}[\mathcal{A}(S(1^\lambda,\pi),\alpha')=1] = \Pr_{d_1\leftarrow D'_1}[\mathcal{A}(d_1)=1], \tag{11}$$

where the first and the second probabilities are the second probabilities of Inequality (3) and Inequality (9) respectively.

By Assumption 9, there exists a negligible function $\mu(\lambda)$ such that the difference between the right hand sides of Equation (10) and Equation (11) is not greater that $\mu(\lambda)$. Therefore the difference between the left hand sides of Equation (10) and Equation (11) is not greater that $\mu(\lambda)$. I.e., Inequality (3) holds. This completes the proof. □

7 Attacks

As we mentioned earlier, having an accepting y that is not a false positive one can recover x by flipping the corresponding bits of y according to the factors of E. And to recover x, it is not necessary to find a small superset of x, but a "close" set. Hence we discuss an attack based on the following theorem.

Theorem 12 (Diophantine Approximation [15]). *Let $\alpha \in \mathbb{R}$ then there exist fractions $a/b \in \mathbb{Q}$ such that $\left|\alpha - \frac{a}{b}\right| < \frac{1}{\sqrt{5}b^2}$. If, on the other hand, there exists $a/b \in \mathbb{Q}$ such that $\left|\alpha - \frac{a}{b}\right| < \frac{1}{2b^2}$, then a/b is a convergent of α.*

The attack based on Theorem 12 is as follows. Having an input y such that the Hamming distance between x and y is bounded by r, we compute $E = XY^{-1} \pmod q = \prod_i p_i^{x_i - y_i} \pmod q = UV^{-1} \pmod q$, where UV^{-1} is the lowest terms of XY^{-1} modulo q with $U = \prod_{i=1}^n p_i^{u_i}$ and $V = \prod_{i=1}^n p_i^{v_i}$, for $u_i, v_i \in \{0,1\}$. We have that $EV - kq = U$ hence $\left|\frac{E}{q} - \frac{k}{V}\right| = \frac{U}{qV}$. By Theorem 12, if $UV < \frac{q}{2}$, then $\frac{k}{V}$ is a convergent of $\frac{E}{q}$. Finding this convergent from the continued fraction of $\frac{E}{q}$ is efficient. So we have V and k, and thus $U = EV - kq$. We then factor U and V to find all different bits between x and y, and recover x by flipping y accordingly.

Moreover, the following theorem shows a way to push the continued fraction algorithm beyond the naive limits given by Theorem 12.

Theorem 13 (Extended Legendre Theorem [11]). *Let α be an irrational number, let the fractions $\frac{p_i}{q_i} \in \mathbb{Q}$ be its continued fraction, and let a, b be coprime nonzero integers satisfying the inequality $\left|\alpha - \frac{a}{b}\right| < \frac{c}{b^2}$, where c is a positive real number. Then $(a,b) = (rp_{m+1} \pm sp_m, rq_{m+1} \pm sq_m)$, for some nonnegative integers m, r and s such that $rs < 2c$.*

By Theorem 13 one can always find a and b by tuning c, which gets rid of the limitation of $\left|\alpha - \frac{a}{b}\right| < \frac{1}{2b^2}$. But this adds exponential overhead so does not greatly improve the attack.

8 Conclusion

We obfuscate small superset and big subset functions using the subset product problem, which is a more trustworthy assumption than the previous works. Our construction is very simple and highly efficient. The correctness is simply based on the uniqueness of integer factoring. We give security proofs for both input-hiding and DVBB.

Acknowledgement

We thank the Marsden Fund of the Royal Society of New Zealand for funding this research, and the reviewers for suggestions.

References

1. Barak, B., Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O., Sahai, A.: Obfuscation for evasive functions. In: Lindell, Y. (ed.) Theory of Cryptography. pp. 26–51. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Annual International Cryptology Conference (CRYPTO). pp. 1–18. Springer (2001)
3. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
4. Bartusek, J., Carmer, B., Jain, A., Jin, Z., Lepoint, T., Ma, F., Malkin, T., Malozemoff, A.J., Raykova, M.: Public-key function-private hidden vector encryption (and more). In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 489–519. Springer International Publishing, Cham (2019)
5. Bartusek, J., Lepoint, T., Ma, F., Zhandry, M.: New techniques for obfuscating conjunctions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. pp. 636–666. Springer International Publishing, Cham (2019)
6. Beullens, W., Wee, H.: Obfuscating simple functionalities from knowledge assumptions. In: Lin, D., Sako, K. (eds.) Public-Key Cryptography – PKC 2019. pp. 254–283. Springer International Publishing, Cham (2019)
7. Bishop, A., Kowalczyk, L., Malkin, T., Pastro, V., Raykova, M., Shi, K.: A simple obfuscation scheme for pattern-matching with wildcards. In: Annual International Cryptology Conference (CRYPTO). pp. 731–752. Springer (2018)
8. Bishop, A., Kowalczyk, L., Malkin, T., Pastro, V., Raykova, M., Shi, K.: A simple obfuscation scheme forpattern-matching with wildcards. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 731–752. Springer International Publishing, Cham (2018)
9. Canetti, R.: Towards realizing random oracles: Hash functions that hide all partial information. In: Kaliski, B.S. (ed.) Advances in Cryptology — CRYPTO '97. pp. 455–469. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
10. Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of hyperplane membership. In: Theory of Cryptography Conference (TCC). pp. 72–89. Springer (2010)

11. Dujella, A.: A variant of Wieners attack on RSA. Computing **85**(1-2), 77–83 (2009)
12. Fuller, B., Reyzin, L., Smith, A.: When are fuzzy extractors possible? In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016. pp. 277–306. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
13. Galbraith, S.D., Zobernig, L.: Obfuscated fuzzy Hamming distance and conjunctions from subset product problems. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. pp. 81–110. Springer International Publishing, Cham (2019)
14. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS). pp. 612–621 (2017)
15. Hurwitz, A.: Über die angenäherte darstellung der irrationalzahlen durch rationale brüche. Mathematische Annalen **39**(2), 279–284 (1891)
16. Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. Journal of cryptology **9**(4), 199–216 (1996)
17. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004. pp. 20–39. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
18. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Annual Cryptology Conference. pp. 465–484. Springer (2011)
19. Wee, H.: On obfuscating point functions. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing. pp. 523–532. ACM (2005)
20. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS). pp. 600–611. IEEE (2017)

A Proof of Input-hiding of [5] from DLP

Here we prove input-hiding of the conjunction obfuscation in [5] from the hardness of DLP. This security property is not studied in [5].

Definition 17. *(Bartusek et al.'s Scheme [5]). The conjunction obfuscation in [5] is as follows:*

- $\mathsf{Setup}(n)$. *Let G be a group of prime order $q > 2^n$ with generator g. Let $B := B_{n+1,2n,q}$, where*

$$B_{n+1,2n,q} = \begin{bmatrix} 1 & 2 & \dots & 2n \\ 1 & 2^2 & \dots & (2n)^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2^{n+1} & \dots & (2n)^{n+1} \end{bmatrix} \tag{12}$$

- $\mathsf{Obf}(\mathsf{pat} \in \{0,1,*\}^n)$. *Set $e \in \mathbb{Z}_q^{2n\times 1}$ such that $e_{2i-1} = e_{2i} = 0$ if $\mathsf{pat}_i = *$, or $e_{2i-b} \leftarrow \mathbb{Z}_q$ and $e_{2i-(1-b)} = 0$ if $\mathsf{pat}_i = b$, for $b \in \{0,1\}$. Output*

$$v = g^{B\cdot e} \in G^{n+1}.$$

– $\mathsf{Eval}(B \in \mathbb{Z}^{(n+1)\times(2n)}, v \in G^{n+1}, x \in \{0,1\}^n)$. *Define* B_x *according to* x *to be the* $(n+1) \times n$ *matrix with column* j *set as* $(B_x)_j := (B)_{2j-x_j}$. *Solve* $tB_x = 0$ *for a non-zero vector* $t \in \mathbb{Z}_q^{1\times(n+1)}$. *Compute*

$$w = \prod_{i=1}^{n+1} v_i^{t_i}$$

and accept if and only if $w = 1$.

Correctness For an input $(B, \mathsf{Obf}(\mathsf{pat}), x)$ of Eval, Eval outputs $w = \prod_{i=1}^{n+1}(\mathsf{Obf}(\mathsf{pat}))^{t_i}$, which equals 1 if x satisfies pat.

Theorem 14. *Suppose the Discrete Logarithm Problem (DLP) is hard (Assumption 4). Then the conjunction obfuscation in Definition 17 is input-hiding.*

Proof. Let $(B, v = g^{B\cdot e})$ be defined as in Definition 17. We show that if there exists an algorithm A that given $(B, v = g^{B\cdot e})$ returns an accepting input $x \in \{0,1\}^n$ with probability $\mu(n)$, then there exists an algorithm A' that solves DLP.

For a DLP instance $(g, h = g^a)$ with g a generator of a group G and $a \in \{0, \ldots, |G|-1\}$, A' first generates matrix B as Equation (12) shows, samples $s_1, \ldots, s_{n+1} \leftarrow \{0, \ldots, |G|-1\}$ and computes

$$u = (g^{s_1}, \ldots, g^{s_{n+1}}).$$

A' then samples $r_i \leftarrow \{0, \ldots, |G|-1\}$ for all $i \in [n+1]$ and computes

$$u' = (h^{r_1} g^{s_1}, \ldots, h^{r_{n+1}} g^{s_{n+1}}),$$

and invokes A with (B, u') to get x' such that

$$\Pr\Big[\prod_{i=1}^{n+1}(h^{r_i} g^{s_i})^{t'_i} = 1\Big] = \mu(n)$$

for some function $\mu(n)$ and for any $t' = (t'_1, \ldots, t'_{n+1}) \neq 0$ computed as follows: A' first creates $B_{x'}$ according to x', then solves $t' B_{x'} = 0$ for a non-zero t'.

Note that

$$\prod_{i=1}^{n+1}(h^{r_i} g^{s_i})^{t'_i} = 1$$
$$\iff g^{a\cdot\sum_{i=1}^{n+1} r_i t'_i} \cdot g^{\sum_{i=1}^{n+1} s_i t'_i} = 1.$$

A' then solves the DLP $(g, h = g^a)$ for a as

$$a' = \frac{-\sum_{i=1}^{n+1} s_i t'_i}{\sum_{i=1}^{n+1} r_i t'_i}.$$

Note that r_i are independent of all other parameters during the process. In particular, r_i are independent of t'_i. Hence $\sum_{i=1}^{n+1} r_i t'_i \neq 0$ with probability $(|G|-1)/|G|$ for some polynomial p. Therefore the probability of solving the DLP is $\mu'(n) = \mu(n) \cdot (|G|-1)/|G|$. By the hardness of DLP, $\mu'(n)$ is negligible. Hence $\mu(n)$ is negligible.

B New Obfuscation for Conjunctions

We now show how to use the techniques for BSF and SSF obfuscation to obfuscate conjunctions in a highly efficient way. Compared with [13], the advantage of the obfuscation in this paper is that it does not rely on continued fractions hence is much simpler and more efficient.

B.1 New Definition of Conjunctions

Conjunctions are also called *pattern matching with wildcards*. Typically they are defined as functions $f_{n,r,x}(y)$ that, holding a secret string $x \in \{0,1,*\}^n$ with $r < n \in \mathbb{N}$ wildcards, take as input binary strings $y \in \{0,1\}^n$ and output 1 if y matches all non-wildcard positions of x, or output 0 otherwise. We define $|x|$ to be the number of 1's in the string.

We give an equivalent definition based on the big subset and small superset functionalities. The intuition is to do both big subset and small superset tests so that any unmatched bit at the non-wildcard positions will be exposed.

Our new definition is as follows.

Definition 18. *A conjunction (or pattern matching with wildcards function) is a function $f_{n,r,x}(y)$ which holds a secret string $x \in \{0,1,*\}^n$ with $r < n \in \mathbb{N}$ wildcards, takes as input a binary string $y \in \{0,1\}^n$ and outputs 1 if y is a subset of x with all wildcards being replaced by 1 and at the same time y is a superset of x with all wildcards being replaced by 0, or outputs 0 otherwise.*

By this new definition, we generically reduce the conjunction obfuscation problem to the BSF and SSF obfuscation problems. However, the catch is that the two instances are "correlated" and the previous security analysis is not sufficient.

B.2 Evasiveness

Let $w \in \{0,1\}^n$ be x with the wildcard positions set to 0, and $z \in \{0,1\}^n$ be x with the wildcard positions set to 1. We use two functions to define the pattern matching with wildcards function $f_{n,r,x}(y)$, which are the SSF $f_{n,t,w}(y)$ with $|w| = |x|$, $t = |x| + r$, and the BSF $f_{n,t',z}(y)$ with $|z| = |x| + r$, $t' = |x|$. If we further convert the BSF into an SSF, it is the SSF $f_{n,\bar{t'},\bar{z}}(\bar{y})$ with $|\bar{z}| = n - |z| = n - |x| - r$ and $\bar{t'} := n - t'$, where $\bar{z}$ is the complement of z. For convenience, in the following we denote the two SSF as f_1 and f_2 respectively, and denote t and $\bar{t'}$ as t_1 and t_2 respectively.

Suppose w and z are uniform. By the approximation form of Inequality (5), for evasivenesses of both f_1 and f_2, we require both $t_1^{|w|}/n^{|w|} \leq 1/2^\lambda$ and $t_2^{|\bar{z}|}/n^{|\bar{z}|} \leq 1/2^\lambda$. Plug in $|w| = |x|$, $|\bar{z}| = n - |x| - r$, $t_1 = |x| + r$ and $t_2 = n - |x|$ we have $[(|x|+r)/n]^{|x|} \leq 1/2^\lambda$ and $[(n-|x|)/n]^{n-|x|-r} \leq 1/2^\lambda$. Suppose $\lambda \leq n/2$ and $|x| = n/2$, the second inequality gives $r \leq n/2 - \lambda$. This also satisfies the first inequality if we plug in $\lambda \leq n/2$ and $|x| = n/2$. Hence for an intuitive impression of the evasive parameters, we can think of $\lambda \leq n/2$, $|x| \approx n/2$ and $r \leq n/2 - \lambda$.

B.3 Construction

We first explain our obfuscation at a high level.

Let $n, r, B \in \mathbb{N}$ with $r < n/2$ and $B \in O(n \ln n)$. Let $f_{n,r,x}(y)$ be a pattern matching with wildcards function with $x \in \{0, 1, *\}^n$ the pattern and r the number of wildcards.

To obfuscate, we derive two binary strings $w, z \in \{0, 1\}^n$ from x, where w is x with the wildcard positions set to 0, and z is x with the wildcard positions set to 1. We then choose two sequences of small primes, $(p_1, \ldots, p_n)$ and $(l_1, \ldots, l_n)$, from $\{2, \ldots, B\}$, and two primes q and s from $\{B^r, \ldots, (1 + o(1))B^r\}$. Then we encode w and z into two different subset products as:

$$X_1 = \prod_{i=1}^{n} p_i^{w_i} \pmod{q},$$

$$X_2 = \prod_{i=1}^{n} l_i^{z_i} \pmod{s}.$$

Then we output $(p_1, l_1, \ldots, p_n, l_n, q, s, X_1, X_2)$ as the obfuscated function. The obfuscating algorithm is as follows.

Algorithm 4 Conjunction Obfuscator

Input: $n \in \mathbb{N}$, $r \in \mathbb{N}$, $x \in \{0, 1, *\}^n$
Output: $((p_1, l_1, \ldots, p_n, l_n) \in \mathbb{N}^{2n}, q, s \in \mathbb{N}, X_1, X_2 \in \mathbb{Z}_q^*)$

1: set w as x with $* \leftarrow 0$, set z as x with $* \leftarrow 1$, and set $\bar{z}$ as the complement of z
2: sample distinct primes $p_1, l_1, \ldots, p_n, l_n$ from $\{2, \ldots, B\}$ where $B \in O(n \ln n)$
3: sample safe primes q, s from $\{B^r, \ldots, (1 + o(1))B^r\}$
4: compute $X_1 = \prod_{i=1}^{n} p_i^{w_i} \mod q$ and $X_2 = \prod_{i=1}^{n} l_i^{\bar{z}_i} \mod s$
5: **return** $(p_1, l_1, \ldots, p_n, l_n, q, s, X_1, X_2)$

To evaluate with an input $y \in \{0, 1\}^n$, we compute

$$Y_1 = \prod_{i=1}^{n} p_i^{y_i} \pmod{q},$$

$$Y_2 = \prod_{i=1}^{n} l_i^{\bar{y}_i} \pmod{s},$$

and $E_1 = Y_1 X_1^{-1} \pmod{q}$, $E_2 = Y_2 X_2^{-1} \pmod{s}$. We then use Algorithm 2 to factor E_1 using the primes $p_1, \ldots, p_n$, and factor E_2 using $l_1, \ldots, l_n$. If both E_1 and E_2 factor successfully, then output 1, otherwise output 0. The evaluation algorithm is as follows.

Algorithm 5 Conjunction Evaluation (with embedded data $(p_1, l_1, \ldots, p_n, l_n) \in \mathbb{N}^{2n}$, $q, s \in \mathbb{N}, X_1, X_2 \in \mathbb{Z}_q^*$

Input: $y \in \{0,1\}^n$
Output: 0 or 1

1: compute $Y_1 = \prod_{i=1}^n p_i^{y_i} \pmod q$ and $Y_2 = \prod_{i=1}^n l_i^{\bar{y}_i} \pmod s$
2: compute $E_1 = Y_1 X_1^{-1} \pmod q$ and $E_2 = Y_2 X_2^{-1} \pmod s$
3: compute $F_1 \leftarrow \mathsf{Factor}(n, (p_1, \ldots, p_n), E_1)$ and $F_2 \leftarrow \mathsf{Factor}(n, (l_1, \ldots, l_n), E_2)$
4: **return** 1 **if** $F_1 = F_2 = 1$ **else** 0

Correctness For convenience, let us denote $U_1 = \prod_{i=1}^n p_i^{w_i}$, $U_2 = \prod_{i=1}^n l_i^{\bar{z}_i}$, $V_1 = \prod_{i=1}^n p_i^{y_i}$, and $V_2 = \prod_{i=1}^n l_i^{\bar{y}_i}$ as the pure products without modular reduction. Then $E_1 = V_1/U_1 \pmod q$, $E_2 = V_2/U_2 \pmod s$. The correctness of our obfuscation is as follows.

If x and y match at all non-wildcard positions, then y is a small superset of w and a big subset of z (i.e., y is a small superset of w and $\bar{y}$ is a small superset of $\bar{z}$), so $E_1 = V_1/U_1$ is an integer $< q$, and $E_2 = V_2/U_2$ is an integer $< s$, then both the factorings of E_1 and E_2 will succeed and the obfuscation will output 1 correctly. Otherwise if there is any non-wildcard position at which x and y do not match, then at least one of V_1/U_1 and V_2/U_2 is a proper rational. With high probability the corresponding E will not factor over the original list of primes. Then the factoring will fail and the obfuscation will output 0 correctly. Notice that false positives are possible. In the following we discuss them.

Avoiding False Positives Using Lattice Arguments We define a *false positive* to be a $y \in \{0,1\}^n$ such that y is not a matching pattern to x yet both $V_1/U_1 < q$ and $V_2/U_2 < s$ factor successfully.

Note that a false positive $y \in \{0,1\}^n$ gives a short vector $u = y - w - e_1 \in \{-2, -1, 0, 1\}^n$ in the lattice

$$L_1 = \left\{ u \in \mathbb{Z}^n \;\middle|\; \prod_{i=1}^n p_i^{u_i} = 1 \pmod q \right\},$$

and a short vector $v = \bar{y} - \bar{z} - e_2 \in \{-2, -1, 0, 1\}^n$ in the lattice

$$L_2 = \left\{ v \in \mathbb{Z}^n \;\middle|\; \prod_{i=1}^n \ell_i^{v_i} = 1 \pmod s \right\},$$

where $e_1, e_2 \in \{0,1\}^n$ are "good" error vectors meaning that the corresponding products E_1, E_2 factor successfully.

In other words, a false positive means that the same y gives short vectors in the two lattices L_1, L_2 simultaneously. To avoid false positives, it is sufficient (more than enough) to avoid short vectors in both lattices. Notice that this is implied by the lattice analysis for SSF in Section 5. Hence the restriction to r is the same as Inequality 8.

Efficiency It is not hard to see that the obfuscation and the evaluation are just about twice those of the SSF obfuscation (i.e., Algorithm 1 and Algorithm 3, respectively). In particular, our obfuscation is more efficient than the scheme in [13]. A drawback of [13] is that it uses continued fractions, which makes the evaluation process complicated. Our scheme cuts off the procedures that deal with continued fractions, resulting in a very simple algorithm.

Security We prove input-hiding security of our scheme. The security is based on the following new computational problem.

Definition 19 (Twin Subset Product Problem (TSP)). *The twin subset product problem is defined as the following. Given two instances* $(p_1, \ldots, p_n, q, X)$ *and* $(l_1, \ldots, l_n, s, X')$ *of SP (as in Definition 13) from two Hamming close points* $x \in \{0,1\}^n$ *and* $x' \in \{0,1\}^n$*, respectively, to find any one of* x *and* x'*.*

The new hardness assumption is as follows.

Assumption 15 (Hard TSP) *Let* $r < n/2 \in \mathbb{N}$ *and* $((p_1, \ldots, p_n, q, X), (l_1, \ldots, l_n, s, X'))$ *be a TSP with two SPs as in Assumption 8 with* $|x \oplus x'| < r$*. Then for every PPT algorithm* A*, for every* $\lambda \in \mathbb{N}$*, there exists a negligible function* $\mu(\lambda)$ *such that the probability that* A *solves any one of the SP is not greater than* $\mu(\lambda)$*.*

Now we prove input-hiding security.

Theorem 16 (Input-hiding of Conjunction Obfuscation). *Assuming the hardness of TSP (Assumption 15), the conjunction obfuscation given by Algorithm 4 - 5 is input-hiding.*

Proof. Let $((p_1, \ldots, p_n, q, X), (l_1, \ldots, l_n, s, X'))$ be a TSP with respect to the secret sets x and x', respectively. Without loss of generality, let x be a superset of x'. Let A be a PPT algorithm that solves the obfuscation given Algorithm 4 - 5. We solve the TSP as follows. We query A on the TSP above. Since the TSP is exactly an obfuscation instance, A can solve for a $y \in \{0,1\}^n$ which is a subset of x and a superset of x'. We then compute and factor $E = XY^{-1} \pmod q = \prod_{i=1}^{n} p_i^{x_i - y_i} \pmod q$ to obtain the error vector $e = x - y \in \{0,1\}^n$. Then we can recover x by flipping y at the positions i such that $e_i = 1$. □