Big Subset and Small Superset Obfuscation

Steven D. Galbraith and Trey Li

Department of Mathematics, University of 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 big enough subset (resp. small enough superset) of X. Our purpose is to protect X from being known, 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 big subset functions (BSF) and small superset functions (SSF) respectively.
In this paper, we obfuscate BSF and SSF in a very simple and efficient way. We prove both virtual black-box (VBB) security and input-hiding security in the standard model 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 BSF and SSF 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 big subset function (BSF) (resp. small superset function (SSF)) is a function $f_X(Y)$ which takes as input a set $Y \subseteq S$ and accepts if Y is a big subset of X (resp. small superset of X), or rejects otherwise. For example, let $S = \{1, \ldots, 1000\}$, $X = \{1, \ldots, 800\}$, and let $t = 700$ (resp. $t = 900$) be a threshold value. Then $f_X(Y) = 1$ if $Y \subseteq X$ and the size of Y is at least 700 (resp. if $Y \supseteq X$ and the size of Y is at most 900).

Our goal is to protect X from being known, yet allow the users to be able to determine whether Y is a big subset (resp. small superset) 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 all $y \in \{0,1\}^n$. A simple obfuscator 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. 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 [5, 7], fuzzy Hamming distance matching [13], hyperplane membership functions [10], compute-and-compare functions [14, 20], etc. [17].

Two kinds of functions of interest are BSF and SSF, as we introduced at the beginning of the section. BSF was firstly introduced in [6] to better analyze and obfuscate conjunctions; while SSF was firstly introduced in [4] to construct public-key function-private encryption. Also, [6] and [4] are all the previous works for BSF or SSF obfuscation. Beullens and Wee [6] obfuscate BSF from a new knowledge assumption. Bartusek et al. [4] obfuscate SSF using similar techniques to [5]. The obfuscator in [5] is a dual scheme of Bishop et al.'s obfuscator [8] for conjunctions. The security proofs in both [6] and [4] are somewhat complicated.

Our contribution is to give new obfuscators for BSF and SSF for certain parameter ranges that are based on simpler and more standard computational assumptions, and that have simpler security proofs. Also we give a proof of input-hiding based on the discrete logarithm problem (DLP) for the conjunction obfuscation by Bartusek et al. [5] and propose a new conjunction obfuscation based on the BSF and SSF obfuscations (see Appendix A and Appendix B, respectively). The security of our conjunction obfuscation is from our new computational problem called the *twin subset product problem*, which is defined as given two subset product instances with respect to the same secret, to find the secret.

1.1 Technical Overview

In this paper we represent a set $X \subseteq \{1, \ldots, n\}$ by its characteristic vector $x \in \{0,1\}^n$. Hence a BSF is a function $f_{n,t,x}(y)$ which takes as input $y \in \{0,1\}^n$ and outputs 1 if $x - y \in \{0,1\}^n$ and $|y| \geq t$ (where $|y|$ denotes the Hamming weight of y), or outputs 0 otherwise. Similarly, an SSF is a function $f_{n,t,x}(y)$ which takes as input $y \in \{0,1\}^n$ and outputs 1 if $y - x \in \{0,1\}^n$ and $|y| \leq t$, or outputs 0 otherwise.

We explain our construction for BSF as follows. The case of SSF is similar.

The high level idea of our obfuscation 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 smooth primes $p_1, \ldots, p_n$ and a bigger prime modulus q so that if and only if an input $y \in \{0,1\}^n$ is a big subset of x (which implies that x and y have many bits in common) the product $\prod_{i=1}^n p_i^{x_i - y_i}$ has many primes p_i being canceled and is smaller than q and thus

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

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 $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, and $r = |x| - t \in \mathbb{N}$. We require $r \leq n/2$. To obfuscate, the obfuscator samples n different small

primes $p_1, \ldots, p_n$ from $[2, B]$ for some $B \in \mathbb{N}$, 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 $|y| \geq t$. If not, which means y is not "big", then it terminates and outputs 0. If $|y| \geq t$, then it further computes $Y = \prod_{i=1}^{n} p_i^{y_i} \pmod{q}$ and $E = XY^{-1} \pmod{q} = \prod_{i=1}^{n} p_i^{x_i - y_i} \pmod{q}$, and tries to factor E by dividing the primes $p_1, \ldots, p_n$ one by one. If $x - y \in \{0,1\}^n$, which means y is a "subset" of x, then $|y| \geq t$ ensures that $|x - y| \leq r$, then E factors over $\{p_1, \ldots, p_n\}$. If this is the case, then the function outputs 1. Otherwise, if $x - y \notin \{0,1\}^n$, which means y is not a "subset" of x, then with high probability E will not factor over $\{p_1, \ldots, p_n\}$, then the function outputs 0.

1.2 Organization

In Section 2 we introduce basic notions that will be used in this paper and define function obfuscation and evasive functions, as well as several entropies for the analysis of evasiveness in Section 4. In Section 3 we define BSF and SSF. In Section 4 we first define evasive BSF and SSF, which is the only type of BSF and SSF that can be obfuscated, then derive distributions of BSF and SSF that can be obfuscated by our method. In Section 5 we formally define the subset product problems and present the hardness assumptions that our obfuscation bases on. We also provide evidence to our hardness assumptions by reducing the discrete logarithm problem to both the high and low density case subset product problems. In Section 6 we present our obfuscation for BSF and SSF and prove distributional virtual black-box (VBB) security and input-hiding security based on the subset product problems. We discuss potential techniques for attacking our scheme in Section 6.4, and provide concrete parameters for our scheme in Section 6.5. Section 7 is a brief conclusion. Besides, Appendix A and Appendix B are of independent interests, where Appendix A is a proof of input-hiding for the conjunction obfuscation in [5], and Appendix B is a new obfuscation for conjunctions, which can be treated as another independent work.

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), $[a, b)$, $(a, b]$, and $[a, b]$, for $a, b \in \mathbb{R}$. We denote discrete intervals such as $\{a, \ldots, b\}$ in the same way as $[a, b]$, 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 $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

Let $\lambda \in \mathbb{N}$ be the security parameter. We say two distributions D_λ and E_λ are computational indistinguishable if for every probabilistic polynomial time (PPT) algorithm A, there exists a negligible function μ in λ such that

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

dented $D_\lambda \overset{c}{\approx} E_\lambda$. 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 the 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.

Definition 1 (Distributional Virtual Black-Box Obfuscator (VBB) [3, 20]). *Consider a family of circuits $\mathcal{C}$ and let O be a PPT algorithm, which takes as input a circuit $C \in \mathcal{C}$, a security parameter $\lambda \in \mathbb{N}$, and outputs a circuit $\tilde{C} \leftarrow O(1^\lambda, C)$. Let $\mathcal{D}$ be a class of distribution ensembles $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ that sample $(C, aux) \leftarrow D_\lambda$ with $C \in \mathcal{C}$ and aux some polynomial size auxiliary information. We say that O is an obfuscator for the distribution class $\mathcal{D}$ over the circuit family $\mathcal{C}$, if it satisfies the following properties:*

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

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

where the probability is over the coin tosses of O.

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

3. Distributional Virtual Black-Box: For every (non-uniform) polynomial size adversary A, there exists a (non-uniform) PPT simulator 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:

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

where the first probability is taken over the coin tosses of A and O, the second probability is taken over the coin tosses of S, $C.params$ 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 S^C has black-box access to the circuit C.

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

Definition 2 (Distributional-Indistinguishability [20]). *An obfuscator 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 S, such that for every distribution ensemble $D = \{D_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{D}$ that samples $(C, aux) \leftarrow D_\lambda$ with $C \in \mathcal{C}$, we have that*

$$(O(1^\lambda, C), aux) \stackrel{c}{\approx} (S(1^\lambda, C.params), aux),$$

where $(C, aux) \leftarrow D_\lambda$, and aux is some auxiliary information.

For convenience of use, we restate the definition in the following equivalent way.

Definition 3 (Distributional-Indistinguishability - Alternative Definition). *An obfuscator 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 S, such that for every PPT distinguisher B, for every distribution ensemble $D = \{D_\lambda\} \in \mathcal{D}$ that samples $(C, aux) \leftarrow D_\lambda$ with $C \in \mathcal{C}$, and every (non-uniform) polynomial size predicate $\varphi : C_\lambda \to \{0, 1\}$, there exists a negligible function $\mu(\lambda)$ such that:*

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

where the first probability is taken over the coin tosses of B and O, the second probability is taken over the coin tosses of B and S, and $aux' = (aux, \varphi(C))$.

It is shown in [20] that distributional-indistinguishability which works with $aux' = (aux, \varphi(C))$ implies distributional VBB which works with aux, 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 input.

Definition 4 (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 $(C, aux) \leftarrow D_\lambda$, computes $aux' = (aux, \varphi(C))$ and outputs (C, aux').*

Theorem 1 (Distributional-Indistinguishability implies VBB [20]). *For any family of circuits $\mathcal{C}$ and a distribution class $\mathcal{D}$ over $\mathcal{C}$, if an obfuscator O satisfies distributional-indistinguishability (Definition 3) for the class of distributions $\mathcal{D}' = aug(\mathcal{D})$, i.e., if there exists a (non-uniform) PPT simulator S, such that for every PPT distinguisher B, for every distribution ensemble $D' = \{D'_\lambda\}$ where D'_λ samples $(C, aux) \leftarrow D_\lambda$ with $C \in \mathcal{C}$, computes $aux' = (aux, \varphi(C))$ and outputs (C, aux'),*

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

then it also satisfies distributional-VBB security for the distribution class $\mathcal{D}$ (Definition 1).

Compared with VBB, input-hiding is a somehow more natural security notion for evasive function obfuscation.

Definition 5 (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 $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ that sample $C \leftarrow D_\lambda$ with $C \in \mathcal{C}$. An obfuscator O is input-hiding for the distribution class $\mathcal{D}$ over the circuit family $\mathcal{C}$ if for every PPT adversary A there exists a negligible function $\mu(\lambda)$ such that for every $\lambda \in \mathbb{N}$ and for every auxiliary input $aux \in \{0,1\}^{poly(\lambda)}$ to A:*

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

where the probability is taken over the random sampling of D_λ and the coin tosses of A and O.

2.2 Evasive Functions

Note that input-hiding is particularly defined for evasive functions, since there is no way one can hide the accepting inputs of a non-evasive function without changing its functionality.

Definition 6 (Evasive Circuit Collection [1]). *A collection of circuits $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ is evasive if there exists a negligible function $\mu(\lambda)$ such that for every polynomial p, for every $\lambda \in \mathbb{N}$, and for every $x \in \{0,1\}^n$ with $n = p(\lambda) \in \mathbb{N}$:*

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

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

Note that in Definition 6 we made a distinction between the security parameter λ and the input length n, where [1] assumes that $\lambda = n$.

2.3 Entropies

We define several entropies for the discussion of evasiveness in Section 4.

Definition 7 (Min-Entropy). *The min-entropy of a random variable X is defined as $H_\infty(X) = -\ln(max_x \Pr[X = x])$. The (average) conditional min-entropy of a random variable X conditioned on a correlated variable Y is defined as $H_\infty(X|Y) = -\ln(E_{y\leftarrow Y}[max_x \Pr[X = x|Y = y])$.*

Definition 8 (Conditional Hamming Ball Min-Entropy [13]). *(Also known as conditional fuzzy min-entropy ([12], Definition 3).) Let $r < n \in \mathbb{N}$. The Hamming ball min-entropy of random variables X and Y on $\{0,1\}^n$ is defined as*

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

where $\oplus$ denotes the XOR operation.

Definition 9 (Conditional Big Subset Min-Entropy). *Let $0 \leq t \leq n \in \mathbb{N}$. The conditional big subset min-entropy of a random variable X and Y on $\{0,1\}^n$ is defined as*

$$H_{Sub,\infty}(X|Y) = -\ln\left(max_{y\in\{0,1\}^n} \Pr[X - y \in \{0,1\}^n, |y| \geq t|Y]\right).$$

Definition 10 (Conditional Small Superset Min-Entropy). *Let $0 \leq t \leq n \in \mathbb{N}$. The small superset min-entropy of a random variable X and Y on $\{0,1\}^n$ is defined as*

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

3 Big Subset and Small Superset Functionalities

We define big subset and small superset functions as follows.

Definition 11 (Big Subset Function (BSF) [6]). *For each $n \in \mathbb{N}$, we define the class of big subset functions C_n to be the class of functions parametrized by (n, t, X), where $X \subseteq \{1, \ldots, n\}$, and $t \in \mathbb{N}$ is a threshold with $0 \leq t \leq n$. A big subset function is a function $f_{n,t,X} : P(\{1, \ldots, n\}) \to \{0, 1\}$ that on input a set $Y \subseteq \{1, \ldots, n\}$ outputs 1 if $Y \subseteq X$ and $|Y| \geq t$, or outputs 0 otherwise, where P denotes the power set.*

Definition 12 (Small Superset Function (SSF) [4]). *For each $n \in \mathbb{N}$, we define the class of small superset functions to be the class of functions C_n parametrized by (n, t, X), where $X \subseteq \{1, \ldots, n\}$, and $t \in \mathbb{N}$ is a threshold with $0 \leq t \leq n$. A small superset function is a function $f_{n,t,X} : P(\{1, \ldots, n\}) \to \{0, 1\}$ that on input a set $Y \subseteq \{1, \ldots, n\}$ outputs 1 if $X \subseteq Y$ and $|Y| \leq t$, or outputs 0 otherwise, where P denotes the power set.*

Following are equivalent definitions that will be used in the rest of the paper.

Definition 13 (Big Subset Function (BSF) - Alternative Definition). *For each $n \in \mathbb{N}$, we define the class of big subset functions to be the class of functions C_n parametrized by (n, t, x), where $x \in \{0,1\}^n$, and $t \in \mathbb{N}$ is a threshold with $0 \leq t \leq n$. A big subset function is a function $f_{n,t,x} : \{0,1\}^n \to \{0,1\}$ that on input $y \in \{0,1\}^n$ outputs 1 if $x - y \in \{0,1\}^n$ and $|y| \geq t$, or outputs 0 otherwise.*

Definition 14 (Small Superset Function (SSF) - Alternative Definition). *For each $n \in \mathbb{N}$, we define the class of small superset functions C_n to be the class of functions parametrized by (n, t, x), where $x \in \{0,1\}^n$, and $t \in \mathbb{N}$ is a threshold with $0 \leq t \leq n$. A small superset function is a function $f_{n,t,x} : \{0,1\}^n \to \{0,1\}$ that on input $y \in \{0,1\}^n$ outputs 1 if $y - x \in \{0,1\}^n$ and $|y| \leq t$, or outputs 0 otherwise.*

4 Evasiveness of BSF and SSF

Now we discuss what kinds of BSF and SSF can be obfuscated. We conclude that only "evasive" BSF and SSF are possible to obfuscate. To see this, just to notice that the secret x is immediately leaked once a big subset or a small superset of x is found.

The attack for BSF is as follows (the case of SSF is similar): Let y be a big subset of x found by the attacker. The attacker then flips the 0's of y one by one and queries the oracle of $f_{n,t,x}$ (i.e., the obfuscated function of $f_{n,t,x}$). If the y with a 0-position flipped is still a big subset of x, then the corresponding position in x is a 1; otherwise the corresponding bit of x is 0. Running through all 0's in y the attacker learns x. In particular, if all the flipped y's are rejected, then the attacker learns that $x = y$.

In the following we discuss the requirements for evasive BSF and SSF.

4.1 Evasive BSF and SSF

We first define evasive BSF and evasive SSF. Let $n \in \mathbb{N}$ be the bit length of a set $x \in \{0,1\}^n$, and $t \in \{0, \ldots, n\}$ be the threshold indicating big/small. Let $\lambda \in \mathbb{N}$ be the security parameter such that $n = p(\lambda)$ for some polynomial p. Note that n, t are functions in λ, we therefore sometimes denote them as $n(\lambda), t(\lambda)$ for clarity.

Definition 15 (Evasive BSF/SSF). *Let $\{X_{n(\lambda)}\}_{n(\lambda)\in\mathbb{N}}$ be an ensemble of distributions over $\{0,1\}^{n(\lambda)}$. Let $\mathcal{C} = \{C_{n(\lambda),t(\lambda)}\}_{n(\lambda)\in\mathbb{N}}$ with $C_{n(\lambda),t(\lambda)} = \{f_{n(\lambda),t(\lambda),x}\}_{x\leftarrow X_{n(\lambda)}}$ be the corresponding collection of BSF (or SSF). We say $\mathcal{C}$ is evasive if there exists a negligible function $\mu(\lambda)$ such that for every polynomial p, for every $\lambda \in \mathbb{N}$, and for every $y \in \{0,1\}^{n(\lambda)}$:*

$$\Pr_{x\leftarrow X_{n(\lambda)}}[f_{n(\lambda),t(\lambda),x}(y) = 1] \leq \mu(\lambda). \tag{4}$$

4.2 Uniform Distributions

Now we consider the requirements for evasiveness. Let us start with the case where $\{X_{n(\lambda)}\}_{n(\lambda)\in\mathbb{N}}$ are uniform distributions.

For BSF, if $|y|< t(\lambda)$, then Inequality (4) always holds since y will never be a "big" subset of any x. If $|y|\geq t(\lambda)$, then there are at most $2^{n(\lambda)-t(\lambda)}$ many x such that y is a subset of x, Inequality (4) holds if and only if $2^{\lambda-t(\lambda)}/2^{n(\lambda)} \leq 1/2^{\lambda}$, i.e. $t(\lambda) \geq \lambda$. Therefore in the case where $\{X_n\}_{n\in\mathbb{N}}$ are uniform distributions, the BSF family $\mathcal{C}$ is evasive if and only if

$$t(\lambda) \geq \lambda.$$

Similarly, for SSF, if $|y|> t(\lambda)$, then y will never be a "small" superset of any x hence Inequality (4) always holds. If $|y|\leq t(\lambda)$, then there are at most $2^{t(\lambda)}$ many x such that y is a superset of x, Inequality (4) holds if and only if $2^{t(\lambda)}/2^{n(\lambda)} \leq 1/2^{\lambda}$, i.e., $t(\lambda) \leq n(\lambda) - \lambda$. Hence in the case where $\{X_n\}_{n\in\mathbb{N}}$ are uniform distributions, the SSF family $\mathcal{C}$ is evasive if and only if

$$t(\lambda) \leq n(\lambda) - \lambda.$$

Note that the above requirements for $t(\lambda)$ are the most basic ones in the sense that they are obtained under the best possible distributions, namely the uniform distributions.

4.3 General Distributions

We now consider the case where $\{X_{n(\lambda)}\}_{n(\lambda)\in\mathbb{N}}$ are general distributions.

Let us first explain what exactly Inequality (4) means. In words, it means that for every $y \in \{0,1\}^{n(\lambda)}$, an x sampled from the distribution $X_{n(\lambda)}$ has negligible probability that y is a big subset (or small superset in the case of SSF) of x. Intuitively, this requires that in the space $\{0,1\}^{n(\lambda)}$, the number of points x representing BSF (or SSF) is large enough and at the same time they are "well spread out" in the sense that big subset relations (or small superset relations, respectively) between points occur sparsely and evenly in the space. Rigorously, the following requirement implies Inequality (4).

Definition 16 (Big Subset / Small Superset Evasive Distribution). *Let $n(\lambda) \in \mathbb{N}$ and λ be the security parameter. Let $0 \leq t(\lambda) \leq n(\lambda) \in \mathbb{N}$. Let $X = \{X_{n(\lambda)}\}_{n(\lambda)\in\mathbb{N}}$ be an ensemble of distributions over $\{0,1\}^{n(\lambda)}$. We say that X is big subset evasive (or small superset evasive, respectively) if for all auxiliary inputs aux, the conditional big subset min-entropy (or conditional small superset min-entropy, respectively) of X conditioned on aux, as in Definition 9 (or in Definition 10, respectively), is at least λ.*

Note that asking for a big subset or 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 BSF (or 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 $c \in \{0,1\}^{n(\lambda)-2}$. We can see that x and y have very small Hamming distance $|x \oplus y| = 2$, but neither of them is a subset or a superset of the other.

This means that in the same space $\{0,1\}^{n(\lambda)}$, there are more evasive BSF as well as evasive SSF than evasive fuzzy Hamming distance matching functions.

Nonetheless, our obfuscation for BSF and SSF has to work under the stronger requirement, namely the requirement for evasive Hamming distance matching. This is because an attacker can always recover the secret x from its encoding by merely finding a "close" set and not necessary to find a big subset or a small superset.

We therefore use the following definition for evasiveness of BSF and SSF in the rest of the paper.

Definition 17 (Hamming Distance Evasive Distribution [13]). *Let $\lambda \in \mathbb{N}$ be the security parameter and $n(\lambda), t(\lambda) \in \mathbb{N}$. Let $X = \{X_{n(\lambda)}\}_{n(\lambda)\in\mathbb{N}}$ be an ensemble of distributions over $(\{0,1\}^{n(\lambda)}, \{0,1\}^{poly(\lambda)})$, where $X_{n(\lambda)}$ outputs $x \in \{0,1\}^{n(\lambda)}$, and some auxiliary information $aux \in \{0,1\}^{poly(\lambda)}$ about x. We say that X is Hamming distance evasive if for all $n(\lambda) \in \mathbb{N}$, the conditional Hamming ball min-entropy of $X_{n(\lambda)}$ conditioned on aux (as in Definition 8 with $r(\lambda) := ||X_{n(\lambda)}| - t(\lambda)| < n(\lambda)$) is at least λ.*

Note that this requirement of evasiveness already implies a wide range of parameters in the sense of the largest obfuscatable gap between $|x|$ and $t(\lambda)$. Specifically, for the negligible probability that a uniform $y \leftarrow \{0,1\}^{n(\lambda)}$ falls into the radius-r Hamming ball of x, i.e., $\Pr_{y\leftarrow\{0,1\}^{n(\lambda)}}[|x \oplus y| \leq r(\lambda)] \leq 1/2^{\lambda}$, we require that

$$r(\lambda) \leq \frac{n(\lambda)}{2} - \sqrt{\lambda n(\lambda) \ln 2} \tag{5}$$

(for a proof of this for the uniform distribution $X_{n(\lambda)}$, see Lemma 2 in [13]). Notice that this $r(\lambda)$ has a rather large domain, and now the Hamming distance evasive distribution (as in Definition 17) gives the gap $||x| - t(\lambda)|$ the same domain as this $r(\lambda)$.

5 Computational Assumptions

In this section we define the subset product problems and give evidence to their hardness.

In the following definitions, n, r, t are always functions in λ, denoted $n(\lambda)$, $r(\lambda)$, $t(\lambda)$, respectively; and we assume that $n(\lambda) \geq \lambda$.

5.1 Discrete Logarithm Problem

Following are the definition of DLP and its hardness assumption, one can find them in any standard textbook.

Definition 18 (Discrete Logarithm Problem (DLP)). *Let $N \in \mathbb{N}$. Let G of order N be a finite group written in multiplicative notation. The discrete logarithm problem is the following. Given $g, h \in G$ to find a (if it exists) such that $h = g^a$.*

Assumption 2 (Hard DLP) *Let $\lambda \in \mathbb{N}$. Let $\mathbb{Z}_q^*$ be the multiplicative group of integers modulo q with $q = 2p + 1 \geq 2^\lambda$ 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 all $\lambda \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that for all PPT algorithms A, the probability that A solves the DLP (g, g^a) is not greater than $\mu(\lambda)$.*

5.2 Subset Product Problems and Hardness Assumptions

The subset product problems are also classic problems. We cite [13] for its first application to obfuscation.

Definition 19 (Subset Product Problem (SP) [13]). *The subset product problem is the following. Given $n(\lambda)+1$ primes $(p_1, \ldots, p_{n(\lambda)}, q)$ and an integer $X \in \mathbb{Z}_q^*$, find a subset of the p_i's (if one exists) that multiply to X modulo q, or equivalently, find a binary string $(x_1, \ldots, x_{n(\lambda)}) \in \{0,1\}^{n(\lambda)}$ such that $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$.*

Definition 20 (Decisional-Subset Product Problem (d-SP) [13]). *The decisional-subset product problem is the following. Given $n(\lambda)+1$ primes $(p_1, \ldots, p_{n(\lambda)}, q)$ and an integer $X \in \mathbb{Z}_q^*$, decide if there exists a binary string $(x_1, \ldots, x_{n(\lambda)}) \in \{0,1\}^{n(\lambda)}$ such that $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$.*

In order to state the hardness of SP, we first define the $(n(\lambda), t(\lambda), B(\lambda))$-SP distribution.

Definition 21 ($(n(\lambda), t(\lambda), B(\lambda))$-SP Distribution). *Let $\lambda, n(\lambda), t(\lambda), B(\lambda) \in \mathbb{N}$ with $n(\lambda) \geq \lambda$ polynomial in λ, $t(\lambda) < n$, and $B(\lambda)$ larger than the $n(\lambda)$-th prime. Let $X_{n(\lambda)}$ be a distribution over $\{0,1\}^{n(\lambda)}$ with Hamming ball min-entropy λ. Let $(x_1 \ldots, x_{n(\lambda)}) \leftarrow X_{n(\lambda)}$ such that $r(\lambda) := ||x| - t(\lambda)|$ satisfies Inequality (5). Let $p_1, \ldots, p_{n(\lambda)}$ be distinct primes sampled uniformly from the primes in $[2, B(\lambda)]$ and q be a random safe prime in $[B(\lambda)^{r(\lambda)}, (1 + o(1))B(\lambda)^{r(\lambda)}]$. Then we call the distribution $(p_1, \ldots, p_{n(\lambda)}, q, X)$ with $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ the $(n(\lambda), t(\lambda), B(\lambda))$-SP distribution.*

Note that requiring Inequality (5) in Definition 21 is to ensure that for a center point $x \in \{0,1\}^{n(\lambda)}$, the probability that a uniform $y \leftarrow \{0,1\}^{n(\lambda)}$ falls into the radius-r Hamming ball of x is at least $1/2^{\lambda}$, as we discussed with regard to Inequality (5). In particular, Assumption 3 is false if $r(\lambda) > n(\lambda)/2$ since one can easily guess a string within Hamming distance $n(\lambda)/2$ of x, and then solve SP using the decoding method in [13]. Also, we refer to the discussion regarding Equation (9.3) in [13] to justify why such a safe prime q in Definition 21 exists.

In the following we state our hardness assumption for the proof of input-hiding in Section 6.3. Note that the definition of input-hiding (Definition 5) involves some auxiliary information $aux \in \{0,1\}^{poly(\lambda)}$. We therefore put the auxiliary information in the assumption. Since this kind of auxiliary information is some "fixed" information for all $x \leftarrow X_n$ with the same n, giving it out should not change the hardness of the problem too much.

Assumption 3 (Hard SP) *Let $\lambda, n(\lambda), t(\lambda), B(\lambda) \in \mathbb{N}$. Let D be an $(n(\lambda), t(\lambda), B(\lambda))$-SP distribution. Then for every PPT algorithm A, there exists a negligible function $\mu(\lambda)$ such that for every $n(\lambda) \in \mathbb{N}$, for every auxiliary input $aux \in \{0,1\}^{poly(\lambda)}$, the probability that A solves the SP $(p_1, \ldots, p_{n(\lambda)}, q, \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q)$ over the distribution D with aux given is not greater than $\mu(\lambda)$.*

We then state the hard d-SP assumption which serves the proof of distributional VBB of our obfuscation. Different from Assumption 3 where there is only one aux for each $n(\lambda)$, the following Assumption 4 assumes that d-SP is hard even given auxiliary information aux about the specific x output by $X_{n(\lambda)}$. Furthermore, for convenience in proving distributional-indistinguishabiliy, we define the d-SP problem in the "predicate-augmentation" style (as in Definition 4), namely to define it over a distribution D'_b which outputs $aux' = (aux, \varphi(x))$ instead of just aux, for any (non-uniform) polynomial size predicate $\varphi : X_{n(\lambda)} \to \{0,1\}$, where $b \in \{0,1\}$. The assumption is as follows.

Assumption 4 (Hard d-SP) *Let $\lambda, n(\lambda), t(\lambda), B(\lambda) \in \mathbb{N}$. Let $D_0 = (p_1, \ldots, p_{n(\lambda)}, q, X, aux)$ be the $(n(\lambda), t(\lambda), B(\lambda))$-SP distribution which outputs some extra auxiliary information aux about x that satisfies Definition 17 (i.e., the conditional Hamming ball min-entropy of the distribution $X_{n(\lambda)}$ conditioned on aux is still at least λ; also note that aux is essentially output by $X_{n(\lambda)}$). Let φ be any (non-uniform) polynomial size predicate. Let $D'_0 = (p_1, \ldots, p_{n(\lambda)}, q, X, aux')$ be a distribution which samples $(p_1, \ldots, p_{n(\lambda)}, q, X, aux) \leftarrow D_0$, computes $aux' = (aux, \varphi(x))$ and outputs $(p_1, \ldots, p_{n(\lambda)}, q, X, aux')$. Similarly, let $D_1 = (p_1, \ldots, p_{n(\lambda)}, q, X', aux)$ be the same distribution as D_0 with $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ replaced by $X' \leftarrow \mathbb{Z}_q^*$ and D'_1 be the same distribution as D'_1 with $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ replaced by $X' \leftarrow \mathbb{Z}_q^*$. Then for every PPT algorithm A, there exists a negligible function $\mu(\lambda)$ such that for every $n(\lambda) \in \mathbb{N}$, given polynomially many instances from D'_b with $b \in \{0,1\}$, the difference of the probabilities that A out-*

puts $b' = b$ and A outputs $b' \neq b$ is not greater than $\mu(\lambda)$. I.e.,

$$\left| \Pr_{(p_1,\ldots,p_n,q,X,aux')\leftarrow D_0'}[A(p_1,\ldots,p_n,q,X,aux')=1] - \Pr_{(p_1',\ldots,p_n',q',X',aux')\leftarrow D_1'}[A(p_1',\ldots,p_n',q',X',aux')=1] \right| \leq \mu(\lambda). \quad (6)$$

5.3 Evidence to the Hardness of SP and d-SP

Now we turn to give some evidence to the hardness of SP and d-SP. Specifically, we reduce DLP to SP, and SP to d-SP.

We consider SP and d-SP without auxiliary information and over a slightly different distribution which is with respect to a fixed r instead of an $r := ||x|-t|$ dependent on $|x|$. Intuitively, these minor differences should not change our confidence in the hardness of the problems too much. Take SP as an example, the auxiliary information only decreases the entropy of x by a tiny amount. Leaking one or two bits of x does not make the task of finding the whole x any easier. Also, if SP is hard with respect to r, then it should still be hard when r is defined to be $||x|-t|$, as long as $||x|-t|$ satisfies the same requirements that r satisfies.

We state the definition and the assumptions in the following.

Definition 22 ($(n(\lambda), r(\lambda), B(\lambda))$-SP Distribution). *Let $\lambda, n(\lambda), r(\lambda), B(\lambda) \in \mathbb{N}$ with $n(\lambda) \geq \lambda$ polynomial in λ, $r(\lambda)$ satisfying Inequality (5), and $B(\lambda)$ larger than the $n(\lambda)$-th prime. Let $X_{n(\lambda)}$ be a distribution over $\{0,1\}^{n(\lambda)}$ with Hamming ball min-entropy λ. Let $(x_1 \ldots, x_{n(\lambda)}) \leftarrow X_{n(\lambda)}$ and let $p_1, \ldots, p_{n(\lambda)}$ be distinct primes sampled uniformly from the primes in $[2, B(\lambda)]$. Let q be any safe prime in $[B(\lambda)^{r(\lambda)}, (1+o(1))B(\lambda)^{r(\lambda)}]$. Then we call the distribution $(p_1, \ldots, p_{n(\lambda)}, q, X)$ with $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ the $(n(\lambda), r(\lambda), B(\lambda))$-SP distribution.*

Assumption 5 (Hard SP without *aux* (SP′)) *Let $\lambda, n(\lambda), r(\lambda), B(\lambda) \in \mathbb{N}$. Then for every $n(\lambda) \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that for every PPT algorithm A, the probability that A solves the SP $(p_1, \ldots, p_{n(\lambda)}, q, \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q)$ over the $(n(\lambda), r(\lambda), B(\lambda))$-SP distribution is not greater than $\mu(\lambda)$.*

Assumption 6 (Hard d-SP without *aux* (d-SP′)) *Let $\lambda, n(\lambda), r(\lambda), B(\lambda) \in \mathbb{N}$. Let $D_0 = (p_1, \ldots, p_{n(\lambda)}, q, X)$ be the $(n(\lambda), r(\lambda), B(\lambda))$-SP distribution and $D_1 = (p_1, \ldots, p_{n(\lambda)}, q, X')$ be the same distribution with $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ replaced by $X' \leftarrow \mathbb{Z}_q^*$. Then for every $n(\lambda) \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that given polynomially many instances from D_b with $b \in \{0,1\}$, for every PPT algorithm A, the difference of the probabilities that A outputs $b' = b$*

and A outputs $b' \neq b$ is not greater than $\mu(\lambda)$. I.e.,

$$\left| \Pr_{(p_1,\ldots,p_n,q,X)\leftarrow D_0}[A(p_1,\ldots,p_n,q,X)=1] - \Pr_{(p'_1,\ldots,p'_n,q',X')\leftarrow D_1}[A(p'_1,\ldots,p'_n,q',X')=1] \right| \leq \mu(\lambda). \quad (7)$$

For the hardness of d-SP$'$, we can refer to [16] and [18] for a search-to-decision reduction for the knapsack problems, which can be applied to reduce SP$'$ to d-SP$'$. In the following we reduce DLP to SP$'$.

A reduction from DLP to the high density case SP$'$ was given in [13]. However, what we truly care about is the low density case SP$'$, which is the case that our obfuscation really relies on. In the following we give a reduction from DLP to both high and low density case SP$'$.

The reduction in [13] is based on the following conjecture, which restricts the SP$'$ instances to be in the high density case.

Conjecture 1 ([13]). Let $\lambda, n(\lambda), r(\lambda), B(\lambda) \in \mathbb{N}$. Let $(p_1,\ldots,p_{n(\lambda)}, q, \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q)$ be the $(n(\lambda), r(\lambda), B(\lambda))$-SP distribution with the extra condition that $q \leq 2^{n(\lambda)}$. Let $X_{n(\lambda)}$ be the uniform distribution on $\{0,1\}^{n(\lambda)}$. Then the statistical distance of the distribution of $\prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ and the uniform distribution on $\mathbb{Z}_q^*$ is negligible.

In fact the conditions that $q \leq 2^{n(\lambda)}$ and that $X_{n(\lambda)}$ being uniform are not necessary. The reduction works as long as the number $n_{\mathrm{SP}'}$ of subset products $\prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ satisfies $n_{\mathrm{SP}'}/q \geq p(\lambda)$ for some polynomial p. This is because the purpose of Conjecture 1 is to make sure that we can sample a subset product X from $\mathbb{Z}_q^*$ in polynomial time. For this, much looser conditions like $q = 2^{n(\lambda)}p(\lambda)$ together with merely "high" min-entropy $X_{n(\lambda)}$ over $\{0,1\}^{n(\lambda)}$ are sufficient.

In the following we give a new conjecture based on the looser conditions, and reduce DLP to SP$'$ based on the new conjecture.

Conjecture 2. Let $\lambda, n(\lambda), r(\lambda), B(\lambda) \in \mathbb{N}$. Let $(p_1,\ldots,p_{n(\lambda)}, q, \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q)$ be the $(n(\lambda), r(\lambda), B(\lambda))$-SP distribution with the extra condition that $q \leq 2^{n(\lambda)}$ $p(n(\lambda))$ for some polynomial p. Then the number of elements in $\mathbb{Z}_q^*$ being a subset product $\prod_{i=1}^{n} p_i^{x_i} \pmod q$ with $(x_1,\ldots,x_{n(\lambda)}) \in \{0,1\}^{n(\lambda)}$ is $\geq q/p(n(\lambda))$.

Note that the looser requirement $q \leq 2^{n(\lambda)}p(n(\lambda))$ in Conjecture 2 permits SP$'$ instances in a very low density case.

For the proof of Theorem 7 we also need the following conjecture, which states that when writing different DLP group elements g^a in terms of subset products $\prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ with respect to some random primes $p_1,\ldots,p_{n(\lambda)}$, the exponent strings $x = (x_1,\ldots,x_{n(\lambda)}) \in \mathbb{Z}_2^{n(\lambda)}$ with respect to different g^a have high probability to be linearly independent. This makes sense if we think about the "kind of" uniform distribution of the subset products $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$

over $\mathbb{Z}_q^*$, and the randomness of the primes $p_1, \ldots, p_{n(\lambda)}$. Here we state the conjecture.

Conjecture 3. Let $\lambda, n(\lambda), p_1, \ldots, p_{n(\lambda)}, q$ be as defined in Definition 21. Let $\mathbb{Z}_q^* = \langle g \rangle$ be a DLP group generated by g as defined in Assumption 2. Let p be a polynomial. Let $a_1, \ldots, a_{p(\lambda)} \leftarrow \mathbb{Z}_q^*$ such that $g^{a_k} = \prod_{i=1}^{n(\lambda)} p_i^{x_{k,i}} \pmod q$ for some $x_k = (x_{k,i}, \ldots, x_{k,n(\lambda)}) \in \mathbb{Z}_2^{n(\lambda)}$, for all $k \in \{1, \ldots, p(\lambda)\}$. Then for every p, for every $n(\lambda)$, there exists a polynomial function p' such that the probability that $\{x_1, \ldots, x_{p(\lambda)}\}$ are linearly independent over $\mathbb{Z}_{q-1}$ is $\geq 1/p'(\lambda)$.

Theorem 7. *Assuming Conjecture 2, Conjecture 3, Assumption 6 , and suppose there exists an SP′ instance $(n(\lambda), r(\lambda), p_1, \ldots, p_n, q, X)$ defined in Assumption 5 with $q \leq 2^{n(\lambda)} p(n(\lambda))$ can be solved with overwhelming probability in time T. Then there is an algorithm to solve the DLP in $\mathbb{Z}_q^*$ (as defined in Assumption 2) with overwhelming probability in expected time $O(p(\lambda)T)$, for some polynomial p.*

Proof. Let A be a PPT algorithm that solves the SP′ $(p_1, \ldots, p_{n(\lambda)}, q, X)$ defined in Assumption 5 and let (g, h) with $g, h \in \mathbb{Z}_q^*$ be a DLP instance defined in $\mathbb{Z}_q^*$. Then we solve the DLP as follows. Sample a uniform a from $\{1, \ldots, q-1\}$, then call A to solve $(p_1, \ldots, p_{n(\lambda)}, q, g^a)$. Note that g^a is uniform over $\mathbb{Z}_q^*$. By Assumption 6, the distribution of $(p_1, \ldots, p_{n(\lambda)}, q, g^a)$ is close to the distribution of SP′ instances. Therefore A can solve $(p_1, \ldots, p_{n(\lambda)}, q, g^a)$ for an x (if one exists) such that $g^a = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ with overwhelming probability. Since $q \leq 2^{n(\lambda)} p(n(\lambda))$, by Conjecture 2 we have that $n_{\mathrm{SP}'}/q \geq p(n(\lambda))$. Therefore we expect that after $n(\lambda)p(n(\lambda))$ samples, we have $n(\lambda)$ such a's such that g^a are subset products with $x \in \{0,1\}^{n(\lambda)}$. Also by Conjecture 3, with at most $n(\lambda)p(\lambda)p'(\lambda)$ samples of a, we expect $n(\lambda)$ such $x \in \{0,1\}^n$ which are linearly independent over $\mathbb{Z}_{q-1}$, where $p'(\lambda)$ is some polynomial. We therefore have $n(\lambda)$ linearly independent relations $a \equiv \sum_{i=1}^{n(\lambda)} x_i \log_g(p_i) \pmod{q-1}$ over $\mathbb{Z}_{q-1}$. By solving the equations we have $\log_g(p_i) \pmod{q-1}$ for all $i \in \{1, \ldots, n(\lambda)\}$. Lastly we sample $b \leftarrow \{1, \ldots, q-1\}$, compute $hg^b \pmod q$, and call A to solve it. With at most $p(\lambda)$ extra samples of b, we expect one more relation $\log_g(h) + b \equiv \sum_{i=1}^{n(\lambda)} x_i \log_g(p_i) \pmod{q-1}$ with $x \in \{0,1\}^{n(\lambda)}$. Then $\log_g(h) = \sum_{i=1}^{n(\lambda)} x_i \log_g(p_i) - b \pmod{q-1}$.

Unfortunately, due to parameter restriction, we cannot apply Theorem 7 to reduce the security of our obfuscator to the hardness of DLP. Regardless of the auxiliary information and the difference between the $(n(\lambda), t(\lambda), B(\lambda))$-SP Distribution and the $(n(\lambda), r(\lambda), B(\lambda))$-SP Distribution, the main barrier comes from the perfect correctness of our scheme. If we want to use Theorem 7, then we cannot avoid false acceptances (as defined in Section 6.1), which requires the parameter $r := ||x| - t|$ to lie in a range that is disjoint with the range that Theorem 7 requires. This is why our construction is based on SP and d-SP, and not DLP. Theorem 7 only serves as evidence to why SP and d-SP are "generally" hard.

5.4 Parameters for Obfuscation

We discuss how the parameters in SP and d-SP affect the parameters in our obfuscation.

In the obfuscation, for BSF we have $r(\lambda) = |x| - t(\lambda)$ (or $r(\lambda) = t(\lambda) - |x|$ for SSF) if $||x| - t(\lambda)| \geq \frac{n \ln(2\sqrt{2\pi e})}{\ln(n \ln n)}$ (see Algorithm 1 and Inequality (8)). Since a random x has Hamming weight approximately $n(\lambda)/2$, due to Assumption 3 which requires Inequality (5), we have $r(\lambda) \approx n(\lambda)/2 - t(\lambda) \leq n(\lambda)/2 - \sqrt{\lambda n(\lambda) \ln 2}$ (or $r \approx t(\lambda) - n(\lambda)/2 \leq n(\lambda)/2 - \sqrt{\lambda n(\lambda) \ln 2}$ for SSF). Hence we require

$$t(\lambda) \geq \sqrt{\lambda n(\lambda) \ln 2} \geq \lambda,$$

(or $t(\lambda) \leq n(\lambda) - \sqrt{\lambda n(\lambda) \ln 2} \leq n(\lambda) - \lambda$ for SSF).

On the other hand, under the requirement of Inequality (5), if we choose $t(\lambda)$ based on the average value of $|x|$, i.e., $n(\lambda)/2$, then for some extreme cases of x (like $|x| \approx n(\lambda)$ for BSF, or $|x| \approx 0$ for SSF), the obfuscated function might not have perfect correctness. For example, when $t(\lambda) = n(\lambda)/2 - \sqrt{\lambda n(\lambda) \ln 2}$, for a large set x such that $|x| = n(\lambda) - \sqrt{\lambda n(\lambda) \ln 2}$, a big subset y of x with $|x| - |y| > n(\lambda)/2 - \sqrt{\lambda n(\lambda) \ln 2}$ might be falsely rejected. The example for SSF is similar: when $t(\lambda) = n(\lambda)/2 + \sqrt{\lambda n(\lambda) \ln 2}$, for a small set x such that $|x| = \sqrt{\lambda n(\lambda) \ln 2}$, then a small superset y of x with $|y| - |x| > n(\lambda)/2 - \sqrt{\lambda n(\lambda) \ln 2}$ might not decode correctly. However, this should not be a problem since x is sampled from a high entropy distribution $X_{n(\lambda)}$, its Hamming weight $|x|$ is approximately $n/2$ and the extreme x's occur with negligible probability.

6 Construction

We now present our obfuscator. The construction is very simple so we do not give too much explanation here. The readers can refer to Section 1.1 for the intuition.

Algorithm 1 Obf (for both BSF and SSF)

Input: $n \in \mathbb{N}$, $t \in \mathbb{N}$, $x \in \{0,1\}^n$ with $||x| - t| \leq n/2 - \sqrt{\lambda n \ln 2}$
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 $[2, B]$ where $B \in O(n \ln n)$
2: sample safe prime q from $[B^r, (1 + o(1))B^r]$ where $r := max\{||x| - t|, \frac{n \ln(2\sqrt{2\pi e})}{\ln(n \ln n)}\}$
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 $||x| - t| \leq \frac{n}{2} - \sqrt{\lambda n \ln 2}$ by Inequality (5). Also note that from the size of q, one can guess r hence $|x|$. But this is expected, since $|x| \in [0, n]$, one can always guess $|x|$ with probability $\geq \frac{1}{n+1}$.

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** $p_i | a$ **then** $a \leftarrow a/p_i$
3: **end for**
4: **return** 1 **if** $a = 1$ **else** 0

Following is the evaluation algorithm.

Algorithm 3 Eval (with embedded data $(p_1, \ldots, p_n) \in \mathbb{N}^n$, $q \in \mathbb{N}, X \in \mathbb{Z}_q^*$; for both BSF and SSF)

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

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

Note that compared with the other use of the subset product problem, i.e., [13], our evaluation algorithm (i.e., Algorithm 3) only bases on the uniqueness of integer factoring and saves the process of computing continued fractions. This makes our decoding process extremely simple.

6.1 Correctness

Let us take BSF as an example to analyze the correctness. The analysis for SSF is similar. Note that the inputs y with $|y| < t$ will always be correctly rejected. We therefore only discuss the case where $|y| \geq t$.

Let $E = XY^{-1} \pmod q = \prod_{i=1}^{n} p_i^{e_i} \pmod q$ with $e = (e_1, \ldots, e_n) = x - y \in \{-1, 0, 1\}^n$. If y is a big subset of x, then $e \in \{0,1\}^n$ with $|e| \leq |x| - t \leq r$, since $\prod_{i=1}^{n} p_i^{e_i} < B^r < q$. This means E is a product of primes in $\{p_1, \ldots, p_n\}$ hence will be reduced to 1 in Factor and therefore y will be correctly accepted by Eval.

If y is not a big subset of x, then it will either (1) result in some E such that E contains a prime factor not in $\{p_1, \ldots, p_n\}$ or $e \notin \{0,1\}^n$ (i.e., E is not square-free); 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 Eval. The latter case will be falsely accepted. We therefore call a $y \in \{0,1\}^n$ such that it is not a big subset (or not a small superset in the case of SSF) of x but accepted by Eval a *false acceptance*.

Dealing with False Acceptances by Lattices We now discuss how to deal with false acceptances to achieve perfect correctness.

Let y be a false acceptance. We have that $E = \prod_{i=1}^n p_i^{x_i - y_i} \pmod q = \prod_{i=1}^n p_i^{e_i} \pmod q$ with $\prod_{i=1}^n p_i^{e_i} < q$ and $e = (e_1, \ldots, e_n) \in \{0,1\}^n$. I.e., $\prod_{i=1}^n p_i^{x_i - y_i - e_i} = 1 \pmod q$ with $x - y - e \neq 0$. 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 acceptances, we can require the shortest vector in the above lattice to be larger 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

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

To summarize, if we require that r satisfy Inequality (8) then heuristically there are no false acceptances. Note that Inequality (8) is not a serous restriction, as the problem are most interesting when $r = ||x|-t|$ is large.

Besides, to provide evidence for the precision of the Gaussian heuristic when applied to the relation lattice L, we did some experiments. 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 Section 6.5), 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 acceptances.

Dealing with False Acceptances by Hashing Another way to deal with false acceptances is to use a hash function or a point function obfuscation. Let us take hash as an example. To avoid false acceptances, all we need to do is to compute and output an extra value $h = H(x)$ in Obf, where H is some hash function; and in Factor, store the factors of E in a list F and replace "return 1" with "return F"; also in Eval, add process to recover x from F and compare its hash value against $H(x)$. If y is a big subset (or small superset in the case of SSF) of x, then the factors of E will tell the positions of distinct bits between x and y, then one can recover x by flipping y at those positions. Otherwise if y is a false acceptance, then doing so will give a wrong $x' \neq x$ which can be detected by checking its hash value.

6.2 Efficiency

In the obfuscating algorithm Obf (Algorithm 1), we need to sample $n+1$ primes, perform $n-1$ modular multiplications of integers of size $< q$. Therefore the time

complexity of the obfuscation is linear in the number of modular multiplications of integers of size $< q$.

Again, in the evaluation algorithm Eval (Algorithm 3), we need $n-1$ modular multiplications of integers of size $< q$ to compute Y, and 1 inversion, 1 modular multiplication of integers of size $< q$ to compute E, also n inversions and n modular multiplications of integers of size $< q$ to run Factor (Algorithm 2). Therefore the time complexity of the evaluation is also linear in the number of modular multiplications of integers of size $< q$.

6.3 Security

Theorem 8. *Let X_n be a distribution over $\{0,1\}^n$ with Hamming ball min-entropy λ. Then assuming Assumption 4, the obfuscation given by Algorithm 1 - 3 is distributional VBB.*

Proof. (1) In Section 6.1 we have shown correctness.

(2) In Section 6.2 we have shown polynomial slowdown compared to the original function.

(3) We now show distributional-indistinguishability, which implies distributional VBB due to Theorem 1. 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 with as follows: S takes $C.params = (n, t, B)$ and samples $x' \leftarrow X_n$ such that $r' := ||x'|-t|$ satisfies Inequality (5). S then samples n primes $p'_1, \ldots, p'_n$ and a modulus q' in the same way as O except that it samples q' from $[B^{r'}, (1+o(\lambda))B^{r'}]$ based on its own r'. Denote $S(1^\lambda, C.params) = (p'_1, \ldots, p'_n, q', X')$. We will show that the two probabilities in Inequality (3) equal to the two probabilities in Inequality (7) respectively.

For the first equivalence relation, we have that for every $n \in \mathbb{N}$, for every PPT distinguisher A and for every predicate φ,

$$\Pr_{(x,aux')\leftarrow X'_{n(\lambda)}}[A(p_1, \ldots, p_n, q, X, aux') = 1] = \Pr_{(p_1,\ldots,p_n,q,X,aux')\leftarrow D'_0}[A(p_1, \ldots, p_n, q, X, aux') = 1],$$

where both probabilities are over the randomness of $x, p_1, \ldots, p_n$ and q. This holds simply from the definition of D'_0 (as in Assumption 4).

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

$$\Pr_{(C,aux')\leftarrow D'_\lambda}[A(O(1^\lambda, C), aux') = 1] = \Pr_{(p_1,\ldots,p_n,q,X,aux')\leftarrow D'_0}[A(p_1, \ldots, p_n, q, X, aux') = 1], \quad (9)$$

where the first and the second probabilities are the first probabilities of Inequality (3) and Inequality (7) respectively.

For the second equivalence relation, we have that for every $n \in \mathbb{N}$, for every PPT distinguisher A and for every predicate φ,

$$\Pr_{(x,aux')\leftarrow X'_{n(\lambda)}}[A(p'_1,\ldots,p'_n,q',X',aux')=1] = \Pr_{(p'_1,\ldots,p'_n,q',X',aux')\leftarrow D'_1}[A(p'_1,\ldots,p'_n,q',X',aux')=1],$$

where the probability is over the randomness of $x, p'_1, \ldots, p'_n, q'$ and X'. This holds from the definition of D'_1 (as in Assumption 4).

Replace x with C, $X'_{n(\lambda)}$ with D'_λ, and $p'_1, \ldots, p'_n, q', X'$ with $S(1^\lambda, C.params)$ we have that

$$\Pr_{(C,aux')\leftarrow D'_\lambda}[A(S(1^\lambda,C.params),aux')=1] = \Pr_{(p'_1,\ldots,p'_n,q',X',aux')\leftarrow D'_0}[A(p'_1,\ldots,p'_n,q',X',aux')=1], \quad (10)$$

where the first and the second probabilities are the second probabilities of Inequality (3) and Inequality (7) respectively.

By Assumption 4, there exists a negligible function $\mu(\lambda)$ such that the difference between the right hand sides of Equation (9) and Equation (10) is not greater that $\mu(\lambda)$. Therefore the difference between the left hand sides of Equation (9) and Equation (10) is not greater that $\mu(\lambda)$. I.e., Inequality (3) holds. This completes the proof.

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

Theorem 9. *Let $n, t, r, B \in \mathbb{N}$ satisfy Definition 21 and Inequality (5). Then assuming Conjecture 2, Conjecture 3, Assumption 4, and the hardness of SP (Assumption 3), the BSF obfuscation given by Algorithm 1-3 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 3, 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 to break $((p_1, \ldots, p_n, q, X), aux)$. Since r satisfies Inequality (5), i.e., there are no false acceptances, A will return a big subset y of x such that $E = XY^{-1} \pmod q = \prod_{i=1}^n p_i^{x_i - y_i} \pmod q = \prod_{i=1}^n p_i^{e_i} \pmod q$ 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 $e_i = 1$.

Similarly, we have the following dual theorem for SSF.

Theorem 10. *Let $n, t, r, B \in \mathbb{N}$ satisfy Definition 21 and Inequality (5). Then assuming Conjecture 2, Conjecture 3, Assumption 4, and the hardness of SP (Assumption 3), the SSF obfuscation given by Algorithm 1-3 is input-hiding.*

6.4 Attacks

As we mentioned earlier, having an accepting y 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 big subset or a smaller superset of x, but a "close" set.

Theorem 11 (Diophantine Approximation [15]). *Let $\alpha \in \mathbb{R}$ then there exist fractions $p/q \in \mathbb{Q}$ such that $|\alpha - \frac{a}{b}| < \frac{1}{\sqrt{5}b^2}$. If, on the other hand, there exists $a/b \in \mathbb{Q}$ such that $|\alpha - \frac{a}{b}| < \frac{1}{2b^2}$, then a/b is a convergent of α.*

An attack based on Theorem 11 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 $|\frac{E}{q} - \frac{k}{V}| = \frac{U}{qV}$. By Theorem 11, if $UV < \frac{q}{2}$, then $\frac{k}{V}$ is a convergent of $\frac{E}{q}$. Finding this convergent from the continued fraction of 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 correspondingly.

Moreover, the following theorem shows a way to push the continued fraction algorithm beyond the naive limits given by Theorem 11.

Theorem 12 (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 $|\alpha - \frac{a}{b}| < \frac{c}{b^2}$, 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 12 one can always find a and b by tuning c, which gets rid of the limitation from $|\alpha - \frac{a}{b}| < \frac{1}{2b^2}$.

6.5 Parameters

In this section we discuss function families that can be obfuscated using our method. Restrictions for the parameters λ, n, t, r, and q are as follows.

(1) By Section 4.2, for uniform x's, the basic requirements for evasiveness are $t \geq \lambda$ for BSF, and $t \leq n - \lambda$ for SSF.
(2) For the hardness of finding a y close to x such that it decodes (which will recover x), we require $r := ||x|-t|$ to be small enough, i.e., the Hamming ball of any x to be small enough. This gives $r(n) \leq n/2 - \sqrt{\lambda n \ln 2}$. (Inequality (5)).
(3) To avoid false acceptances (as we discussed this can also be handled using a hash), we need $r > \frac{n \ln(2\sqrt{2\pi e})}{\ln(n \ln n)}$ (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

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

both $r(n) \sim \frac{n}{\ln n}$ and $r(n) \sim \frac{n}{\ln\ln n}$ are possible functions for r. 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 is $(n = 6\lambda, r = \lfloor \frac{n}{\ln(n)} \rfloor)$. In terms of t (take the average weight $\frac{n}{2}$ of a random x), it is $(n = 6\lambda, t = \frac{n}{2} - \lfloor \frac{n}{\ln(n)} \rfloor)$ for BSF and $(n = 6\lambda, t = \frac{n}{2} + \lfloor \frac{n}{\ln(n)} \rfloor)$ for SSF. Note that an elementary requirement is that $|x| > r$ 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 note that $r(n) \sim \frac{n}{\ln n} \sim \pi(n)$, namely our function for r is the prime counting function.

Based on the above analysis, a concrete setting for the BSF/SSF obfuscation is:

BSF: ($\lambda = 128$; $n = 1024$; $t = 365$; $B = p_{1024}$ (the 1024-th prime); X_n has Hamming ball min-entropy λ);

SSF: ($\lambda = 128$; $n = 1024$; $t = 659$; $B = p_{1024}$ (the 1024-th prime); X_n has Hamming ball min-entropy λ).

7 Conclusion

We obfuscate big subset and small superset functionalities using the subset product problems. 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 VBB and input-hiding in the standard model.

We conclude the limitations of our solution as follows.

Let $n \in \mathbb{N}$ be the bit length, $t \in \mathbb{N}$ be the threshold indicating big/small, and $x \in \{0,1\}^n$ be the characteristic vector of a set $X \subseteq \{1,\ldots,n\}$, with its Hamming weight $|x|$ indicating the size of the set X. Our obfuscation for x requires that $||x|-t| < n/2$. However, this is hardly a restriction since a random x has Hamming weight $|x|$ approximately $n/2$ (i.e., with very low probability $|x|$

is far away from $n/2$). Hence the condition $||x|-t| < n/2$ is for free most of the time.

Also, our obfuscation requires Hamming distance evasiveness, which is stronger than big subset and small superset evasiveness. Though, this requirement already implies a fairly large family of functions to obfuscate, as we discussed at the end of Section 4.3.

Bibliography

[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 23. *(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*

$$B_{n+1,2n,q} = \begin{bmatrix} 1 & 2 & \dots & 2n \\ 1 & 2^2 & \dots & (2n)^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2^{n+1} & \dots & (2n)^{n+1} \end{bmatrix} \tag{11}$$

- $\mathsf{Obf}(\mathsf{pat} \in \{0,1,*\}^n)$. *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* $(B_x)_j := (B)_{2j-x_i}$. *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.

The idea of the scheme is that the obfuscated function accepts only when B_x is correctly created, which cannot be completed without knowing the pattern pat.

Theorem 13. *Suppose the Discrete Logarithm Problem (DLP) is hard (Assumption 2). Then the conjunction obfuscation in Definition 23 is input-hiding.*

Proof. Let $(B, v = g^{B\cdot e})$ be defined as in Definition 23. We show that if there exists an algorithm A that solves $(B, v = g^{B\cdot e})$ for an accepting input $x \in \{0,1\}^n$, 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_n and $a \in \{0, \ldots, |G_n|-1\}$, A' first generates matrix B as Equation (11) shows, samples $s_1, \ldots, s_{n+1} \leftarrow \mathbb{Z}_q$ and computes

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

A' then samples $r_i \leftarrow \{0, \ldots, |H|-1\}$ for all $i \in [n+1]$, inserts h^{r_i} into u to get

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

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 $1/p(n)$ for some polynomial p. Therefore the probability of solving the DLP is

$$\Pr[a' = a] = \mu(n)(1 - p(n)) = \mu'(n).$$

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 24. *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.*

Note that in Definition 24 we only require subset and superset, not "big" subset or "small" superset. In fact, requiring both subset and superset implicitly requires big subset and small superset. This is because if y is a superset of x with $*$ being replaced by 0, then it must have Hamming weight $|y| \geq |x|$. Also, if y is a subset of x with $*$ being replaced by 1, then it must have Hamming weight $|y| \leq |x| + r$.

So by the 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

Note that we used two functions to define the pattern matching with wildcards function $f_{n,r,x}(y)$, which were the BSF $f_{n,t,w}(y)$ with $|w|= |x|+r$, $t = |x|$, and the SSF $f_{n,t',z}(y)$ with $|z|= |x|$, $t' = |x|+r$.

Suppose w and z are uniform. (In fact this is not true but it is convenient for us to derive some basic results.) By Section 4.2, for the evasiveness of both $f_{n,t,w}$ and $f_{n,t,z}$, we need both $t \geq \lambda$ and $t' \leq n - \lambda$. Remind that $t = |x|$ and $t' = |x|+r$, we have that $|x|\geq \lambda$ and $|x|+r \leq n - \lambda$. Take the average case of $|x|$, i.e., $|x|\approx \frac{n-r}{2}$, we have that $\frac{n-r}{2} \geq \lambda$ and $\frac{n-r}{2} + r \leq n - \lambda$. Solve for r to get

$$r \leq n - 2\lambda.$$

This is automatically satisfied since we originally require $r \leq \frac{n}{2}$ to avoid an easy guess of y close to w or z by Hamming distance $\frac{n}{2}$. Also in our typical settings for BSF and SSF obfuscation, r is approximately $\frac{n}{\ln n}$ (as discussed in Section 6.5).

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 $f_{n,r,x}(y)$ with $x \in \{0, 1, *\}^n$ be a pattern matching with wildcards function with r 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 1, and z is x with the wildcard positions set to 0. We then choose two sequences of small primes, $(p_1, \ldots, p_n)$ and $(l_1, \ldots, l_n)$, from $[2, B]$, and two primes q and s from $[B^r, (1 + o(1)B^r)]$. 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 Obf

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

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

$$Y_2 = \prod_{i=1}^{n} l_i^{y_i} \pmod{s},$$

and $E_1 = X_1 Y_1^{-1} \pmod{q}$, $E_2 = Y_2 X_2^{-1} \pmod{s}$. 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 Eval (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 $Y_2 = \prod_{i=1}^{n} l_i^{y_i} \pmod{s}$
2: compute $E_1 = X_1 Y_1^{-1} \pmod{q}$ 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 $U_1 = \prod_{i=1}^{n} p_i^{w_i}$, $U_2 = \prod_{i=1}^{n} l_i^{z_i}$, $V_1 = \prod_{i=1}^{n} p_i^{y_i}$, and $V_2 = \prod_{i=1}^{n} l_i^{y_i}$ as the pure products without modular reduction. Then $E_1 = U_1/V_1 \pmod{q}$, $E_2 = V_2/U_2 \pmod{s}$. The correctness of our obfuscation is as follows.

If x and y match at all non-wildcard positions, then w is a superset of y and z is a subset of y, so $E_1 = U_1/V_1$ 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 U_1/V_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. Note that false acceptances are possible. In the following we discuss them.

Dealing with False Acceptances by Lattices We define a *false acceptance* to be a $y \in \{0,1\}^n$ such that y is not a matching pattern to x yet both $U_1/V_1 < q$ and $V_2/U_2 < s$ are integers.

In other words, a false acceptance means the same y gives short vectors in two lattices simultaneously. We show in the following that if we choose $p_i = l_i$ for all $i \in \{1, \ldots, n\}$ and $q = s$, then false acceptances can be easily avoided. The only drawback is that these settings will leak the wildcard positions since

the attacker can factor $X_1X_2^{-1}$ to get the primes corresponding to the wildcard positions. We give our lattice analysis as follows.

Note that some false acceptances can be rejected by adding rules in Algorithm 5 to check if the resulting error vector e and y both have 1's at the same positions. If so, then y is not a good input and should be rejected. Then we only need to deal with the other cases of false acceptances. In the following we call the other cases the second case of false acceptances.

Let $p_i = l_i$ for all $i \in \{1, \ldots, n\}$ and $q = s$. Let y be a second case false acceptance. Then $E_1 = \prod_{i=1}^{n} p_i^{w_i - y_i} = \prod_{i=1}^{n} p_i^{e_{1,i}} \pmod q$ implies a nonzero short vector $u = w - y - e_1 \in \{-2, -1, 0, 1\}^n$ with $e_1 \in \{0, 1\}^n$, $|e_1| \leq r$ in the lattice

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

Similarly, E_2 implies a nonzero short vector $v = y - z - e_2 \in \{-2, -1, 0, 1\}^n$ with $e_2 \in \{0, 1\}^n$, $|e_2| \leq r$ in the same lattice. We therefore have

$$\prod_{i=1}^{n} p_i^{u_i + v_i} = 1 \pmod q.$$

Note that $w = z + e$ for some $e \in \{0, 1\}^n$ with $|e| = r$. So $u + v = (w - y - e_1) + (y - z - e_2) = (z + e - y - e_1) + (y - z - e_2) = e - e_1 - e_2 \in \{-2, -1, 0, 1\}$. We have that

$$|u + v| \leq 2\sqrt{3r}.$$

To avoid false acceptances, it is sufficient to let the shortest vector λ_1 in L to be

$$\lambda_1 \sim \sqrt{\frac{n}{2\pi e}}(q-1)^{\frac{1}{n}} \approx \sqrt{\frac{n}{2\pi e}} q^{\frac{1}{n}} > 2\sqrt{3r},$$

which gives

$$r > \frac{\frac{1}{2} n \ln \frac{8r\pi e}{n}}{\ln(n \ln n)} \tag{12}$$

if we take $q = (n \ln n)^r$. If we further take $r = \frac{n}{\ln n}$, we have

$$\frac{1}{\ln n} > \frac{\frac{1}{2} n \ln \frac{8r\pi e}{\ln n}}{\ln(n \ln n)}.$$

Solve for n we have

$$n > 123.025.$$

I.e., when $n > 123$, false acceptances are naturally avoided, and r can be chosen in the general way as $r = \frac{n}{\ln n}$ as we derived in Section 6.5. Hence we have the following family of parameters:

$$(n > 123, r = \frac{n}{\ln n}).$$

But as we mentioned, this is only for the scheme that will leak the wildcard positions. For our true scheme where no p_i is a l_i and $q \neq s$, we can still hope that in the low density case, each subset product $X \in \mathbb{Z}_q^*$ uniquely defines an $x \in \{0,1\}^n$.

Efficiency It is not hard to verify that the obfuscation and the evaluation have time complexity linear in the number of modular multiplications of primes of size $< q$ and $< s$, respectively. In particular, our obfuscation is even more efficient than the scheme in [13]. One of the drawbacks of [13] is that it uses continued fractions, which makes the evaluation process complicated. The scheme in this paper successfully cut 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 25 (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 19) 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 14 (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 3 with $|x \oplus x'| < r$. Then for every $n \in \mathbb{N}$, there exists a negligible function $\mu(\lambda)$ such that for every PPT algorithm A, the probability that A solves any one of the SPs is not greater than $\mu(\lambda)$.*

Now we prove input-hiding security.

Theorem 15 (Input-hiding of Conjunction Obfuscation). *Assuming the hardness of TSP (Assumption 14), 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 $e_i = 1$.