Obfuscation of Big Subsets and Small Supersets as well as Conjunctions

Steven D. Galbraith and Trey Li

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

Abstract

We obfuscate the big subset and small superset functionalities in a very simple way. We prove both VBB and input-hiding in the standard model based on the subset product problems. Our security proofs are simple.

Let $n \in \mathbb{N}$ be the bit length, $t \in \mathbb{N}$ be the threshold indicating big/small, $x \in \{0,1\}^n$ be the characteristic vector of a set, with its hamming weight $|x|$ denoting the size of the set. Our obfuscation for x requires that $||x| - t| < n/2$. Note that a random x has hamming weight approximately $n/2$, hence this condition is for free most of the time.

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.

We also give a proof of input-hiding for the conjunction obfuscation by Bartusek et al. [5] (see Appendix A) and propose a new conjunction obfuscation based on the big subset and small superset obfuscation (see Appendix B). The security of our conjunction obfuscation is from our new assumption called the *twin subset product problem*.

1 Introduction

The goal of function obfuscation is to prevent a function from being recovered while preserving its functionality. 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 evasive functions. They are the kind of functions that are hard to find an accepting input. Examples of evasive functions include point functions, conjunctions, fuzzy distance matching, hyperplane membership functions, compute-and-compare functions, etc.

Two problems of interest are big subset functionality and small superset functionality [6, 4]. Let $t \leq n \in \mathbb{N}$ and $X \subseteq \{1, \dots, n\}$ be a set. A *big subset function* (BSF) $f_{n,t,X}(Y)$ takes as input a set Y and outputs 1 if Y is a subset of X and the size of Y is not smaller than t, or outputs 0 otherwise. Conversely, a *small superset function* (SSF) $f_{n,t,X}(Y)$ takes as input a set $Y \subseteq \{1, \dots, n\}$ and outputs 1 if Y is a superset of X and the size of Y is not larger than t, or outputs 0 otherwise.

In this paper we will use the following equivalent definitions. Let $x \in \{0,1\}^n$. 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.

Previous works for BSF or SSF obfuscation include [6] and [4]. 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 [7] 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 for the conjunction obfuscation by Bartusek et al. [5] and propose a new conjunction obfuscation based on the BSF and SSF obfuscation (see Appendix A and Appendix B, respectively). The security of our conjunction obfuscation is from our new assumption called the *twin subset product problem*.

1.1 Technical Overview

We explain our construction for BSFs as follows. The case of SSFs is similar.

Let $n, t \in \mathbb{N}$ with $t < n$. Let $x = (x_1, \dots, x_n) \in \{0,1\}^n$, and $r = |x| - t \in \mathbb{N}$. We require $r \leq n/2$. To obfuscate, the obfuscator samples n different small primes $p_1, \dots, p_n$ from $[2, B]$ for some $B \in \mathbb{N}$, and a 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 $(p_1, \dots, p_n, q, X)$ as the obfuscated function.

To evaluate with input $y = (y_1, \dots, y_n) \in \{0,1\}^n$, 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 $p_1, \dots, p_n$ one by one. If $x - y \in \{0,1\}^n$, which means y is a "subset" of x, then $|y| \geq t$ ensures that $|x - y| \leq r$, so E factors over $\{p_1, \dots, p_n\}$. 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.

2 Preliminaries

We call a binary string $x \in \{0,1\}^n$ the characteristic vector of a set. Its hamming weight $|x|$ represents the size of the set. Let C be a circuit, by $|C|$ we mean the size of C. Let a be a real number, 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 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

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

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

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 2.1 (Distributional Virtual Black-Box Obfuscator (VBB) [3, 12]). *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 μ such that for all $n \in \mathbb{N}$ and for all circuits $C \in \mathcal{C}$ with input size n we have

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

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., $|O(C)| \le p(|C|)$, 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 μ such that:

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

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 make sense to consider a definition that does not give the simulator black-box access to the circuit.

Definition 2.2 (Distributional-Indistinguishability [12])**.** *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*

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

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 2.3 (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 μ such that:*

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

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 [12] 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 2.4 (Predicate Augmentation [3, 12]). *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 $D' = \{D'_\lambda\}$ where D'_λ samples $(C, aux) \leftarrow D_\lambda$, computes $aux' = (aux, \varphi(C))$ and outputs (C, aux').*

Theorem 2.5 (Distributional-Indistinguishability implies VBB [12]). *For any family of circuits $\mathcal{C}$ and a distribution class $\mathcal{D}$ over $\mathcal{C}$, if an obfuscator O satisfies distributional-indistinguishability (Definition 2.3) for the class of distributions $aug(\mathcal{D})$ then it also satisfies distributional-VBB security for the distribution class $\mathcal{D}$ (Definition 2.1).*

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

Definition 2.6 (Input-Hiding [1]). *An obfuscator O for a circuit collection $\mathcal{C} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$ is input-hiding if for every PPT adversary A there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and for every auxiliary input $aux \in \{0,1\}^{poly(\lambda)}$ to A:*

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

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

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

Definition 2.7 (Evasive Circuit Collection [1]). *A collection of circuits $\mathcal{C} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$ is evasive if there exists a negligible function μ 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 2.7 we made a distinction between the security parameter λ and the input length n, where [1] assumes that $\lambda = n$.

We also need the following four definitions to discuss evasiveness.

Definition 2.8 (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 2.9 (Hamming Ball Min-Entropy [10]). *(Also known as fuzzy min-entropy ([9], Definition 3).) Let $r < n \in \mathbb{N}$. The hamming ball min-entropy of a random variable X on $\{0,1\}^n$ is defined as*

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

where $\oplus$ denotes the XOR operation.

Definition 2.10 (Big Subset Min-Entropy). *Let $0 \leq t \leq n \in \mathbb{N}$. The big subset min-entropy of a random variable X on $\{0,1\}^n$ is defined as*

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

Definition 2.11 (Small Superset Min-Entropy). *Let $0 \leq t \leq n \in \mathbb{N}$. The small superset min-entropy of a random variable X on $\{0,1\}^n$ is defined as*

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

3 Big Subset and Small Superset Functionalities

We define big subset and small superset functionalities as follows.

Definition 3.1 (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 $f_{n,t,X} : P(\{1, \ldots, n\}) \rightarrow \{0,1\}$ 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 3.2 (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 $f_{n,t,X} : P(\{1, \ldots, n\}) \rightarrow \{0,1\}$ 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 3.3 (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 $f_{n,t,x} : \{0,1\}^n \rightarrow \{0,1\}$ 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 3.4 (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 BSFs and SSFs

In this section we discuss requirements for the evasiveness of BSFs and SSFs.

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 $n(\lambda), t(\lambda)$.

Definition 4.1 (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 BSFs (or SSFs). We say $\mathcal{C}$ is evasive if there exists a negligible function μ such that for every polynomial p, for every $\lambda \in \mathbb{N}$, and for every $y \in \{0,1\}^{n(\lambda)}$:*

$$\Pr_{x \leftarrow X_{n(\lambda)}} [f_{n(\lambda),t(\lambda),x}(y) = 1] \leq \mu(\lambda). \tag{3}$$

4.1 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 (3) 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 (3) 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 (3) 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 (3) holds if and only if $2^{t(\lambda)}/2^{\lambda} \leq 1/2^{\lambda}$, 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 are the most basic ones in the sense that they are obtained under the best possible distributions, namely the uniform distributions.

4.2 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 (3) 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 BSFs (or SSFs) 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 (3).

Definition 4.2 (Big Subset / Small Superset Evasive Distribution)**.** *Let $n(\lambda) \in \mathbb{N}$ and λ be the security parameter. 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 the big subset min-entropy (or small superset min-entropy, respectively) of X, as in Definition 2.10 (or in Definition 2.11, 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 BSFs (or SSFs), 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 none 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 BSFs as well as evasive SSFs than evasive fuzzy hamming distance matching functions.

Nonetheless, our obfuscation for BSFs and SSFs 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 BSFs and SSFs in the rest of the paper.

Definition 4.3 (Hamming Distance Evasive Distribution [10])**.** *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*

$\{0,1\}^{n(\lambda)}$. *We say that X is hamming distance evasive if the hamming ball min-entropy of X (as in Definition 2.9 with $r(\lambda) := ||X| - t(\lambda)| < n(\lambda)$) 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 is at least $1/2^{\lambda}$, i.e.,

$$\Pr_{y \leftarrow \{0,1\}^{n(\lambda)}}[|x \oplus y| \leq r(\lambda)] \leq \frac{1}{2^{\lambda}},$$

we require that

$$r(\lambda) \leq \frac{n(\lambda)}{2} - \sqrt{\lambda n(\lambda) \ln 2} \tag{4}$$

(for a proof of this for the uniform distribution $X_{n(\lambda)}$, see Lemma 2 in [10]). The hamming distance evasive distribution gives the gap $||x| - t(\lambda)|$ the same domain as $r(\lambda)$.

5 Computational Assumptions

In the following we state the subset product problems (SP) defined in [10]. Then we reduce SP from DLP based on a seemingly looser conjecture than [10]. We show that even in a very low density case (e.g., $q = n(\lambda)2^{n(\lambda)}$), the reduction still works. In the following definitions, n, r, t are always functions in λ, denoted $n(\lambda)$, $r(\lambda)$, $t(\lambda)$, respectively; and we assume $n(\lambda) \geq \lambda$.

Definition 5.1 (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 $h = g^a$.*

Definition 5.2 (Subset Product Problem (SP) [10]). *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 5.3 (Decisional-Subset Product Problem (d-SP) [10]). *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 exist a binary string $(x_1, \ldots, x_{n(\lambda)}) \in \{0,1\}^{n(\lambda)}$ such that $X = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$.*

Assumption 5.4 (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 $\mu(\lambda)$.*

Definition 5.5 ($(n(\lambda), r(\lambda), B(\lambda)$-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 (4), 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 5.6 (Hard 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 $\mu(\lambda)$.*

Note that requiring Inequality (4) in Definition 5.5 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 (4). In particular, Assumption 5.6 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 [10]. Also, we refer to the discussion regarding Equation (9.3) in [10] to justify why such a safe prime q in Definition 5.5 exists.

Assumption 5.7 (Hard 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.,*

$$\left| \Pr[A^{D_0} = 1] - \Pr[A^{D_1} = 1] \right| \leq \mu(\lambda). \tag{5}$$

It is shown in [10] that the hardness of SP (as in Assumption 5.6) reduces from the hardness of DLP (as in Assumption 5.4), assuming the following conjecture.

Conjecture 5.8 ([10]). *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 $q \le 2^{n(\lambda)}$ and $X_{n(\lambda)}$ being uniform are not necessary. The reduction works as long as the number n_{SP} of subset products $\prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ satisfies $n_{\mathrm{SP}}/q \ge p(\lambda)$ for some polynomial p. For this, looser conditions like $q = 2^{n(\lambda)}p(\lambda)$ together with high min-entropy $X_{n(\lambda)}$ over $\{0,1\}^{n(\lambda)}$ should be sufficient.

In the following we give a new conjecture based on the looser conditions, and state the theorem in terms of the new conjecture.

Conjecture 5.9. *Let $\lambda, n(\lambda), r(\lambda), B(\lambda) \in \mathbb{N}$. Let $(p_1, \dots, p_{n(\lambda)}, q, \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q)$ be the $(n(\lambda), r(\lambda), B(\lambda))$-SP distribution with the extra condition that $q \le 2^{n(\lambda)}p(n(\lambda))$ 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 $(x_1, \dots, x_{n(\lambda)}) \in \{0,1\}^{n(\lambda)}$ is $\ge q/p(n(\lambda))$.*

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

Conjecture 5.10. *Let $\lambda, n(\lambda), p_1, \dots, p_{n(\lambda)}, q$ be as defined in Definition 5.5. Let $\mathbb{Z}_q^* = \langle g \rangle$ be a DLP group generated by g as defined in Assumption 5.4. Let p be a polynomial. Let $a_1, \dots, a_{p(\lambda)} \leftarrow \mathbb{Z}_q^*$ such that $g^{a_k} = \prod_{i=1}^{n(\lambda)} p_i^{x_{k,i}} \pmod q$ for some $x_k = (x_{k,i}, \dots, x_{k,n(\lambda)}) \in \mathbb{Z}_2^{n(\lambda)}$, for all $k \in \{1, \dots, p(\lambda)\}$. Then for every p, for every $n(\lambda)$, there exists a polynomial function p' such that the probability that $\{x_1, \dots, x_{p(\lambda)}\}$ are linearly independent over $\mathbb{Z}_{q-1}$ is $\ge 1/p'(\lambda)$.*

Theorem 5.11. *Assuming Conjecture 5.9, Conjecture 5.10, Assumption 5.7 , and suppose there exists an SP instance $(n(\lambda), r(\lambda), p_1, \dots, p_n, q, X)$ defined in Assumption 5.6 with $q \le 2^{n(\lambda)}p(n(\lambda))$ 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 5.4) 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 $(p_1, \dots, p_{n(\lambda)}, q, X)$ defined in Assumption 5.6 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 $\{1, \dots, q-1\}$, then call A to solve $(p_1, \dots, p_{n(\lambda)}, q, g^a)$. Note that g^a is uniform over $\mathbb{Z}_q^*$. By Assumption 5.7, the distribution of $(p_1, \dots, p_{n(\lambda)}, q, g^a)$ is close to the distribution of SP instances. Therefore A can solve $(p_1, \dots, p_{n(\lambda)}, q, g^a)$ for an x (if exists) such that $g^a = \prod_{i=1}^{n(\lambda)} p_i^{x_i} \pmod q$ with overwhelming probability. Since $q \le 2^{n(\lambda)}p(n(\lambda))$, by Conjecture 5.9 we have that $n_{\mathrm{SP}}/q \ge p(n(\lambda))$. 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 5.10, 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 $\log_g(h) = \sum_{i=1}^{n(\lambda)} x_i \log_g(p_i) - b \pmod{q-1}$. $\square$

5.1 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 (6)). Since a random x has hamming weight approximately $n(\lambda)/2$, due to Assumption 5.6 which requires Inequality (4), 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 (4), 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.

So we have two choices: (1) typically sample x from the whole space $\{0,1\}^{n(\lambda)}$ and ignore the extreme cases of x, which occur with negligible probability, or (2) explicitly remove the extreme cases and sample x from the subset of $\{0,1\}^{n(\lambda)}$ with $|x|$ lying in some sub-interval of $[0, n(\lambda)]$, or with $|x| = m$ for some $m \in [0, n(\lambda)]$. In this paper we take the first choice.

6 Construction

We now present our construction.

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: $((p_1, \dots, p_n) \in \mathbb{N}^n, q \in \mathbb{N}, X \in \mathbb{Z}_q^*)$
1: sample distinct primes $p_1, \dots, p_n$ 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 $((p_1, \dots, p_n), q, X)$
```

Note that we require $||x| - t| \leq \frac{n}{2} - \sqrt{\lambda n \ln 2}$ by Inequality (4). 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 $\geq \frac{1}{n+1}$.

Algorithm 2 Factor

```
Input: $n \in \mathbb{N}$, $(p_1, \dots, p_n) \in \mathbb{N}^n$, $a \in \mathbb{N}$
Output: 0 or 1
1: for $i = 1, \dots, n$ do
2:     if $p_i | a$ then $a \leftarrow a / p_i$
3: end for
4: return 1 if $a = 1$ else 0
```

Algorithm 3 Eval (with embedded data $(p_1, \dots, p_n) \in \mathbb{N}^n$, $q \in \mathbb{N}, X \in \mathbb{Z}_q^*$; 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 $F \leftarrow$ Factor$(n, (p_1, \dots, p_n), E)$
6: end if
7: return 1 if $F = 1$ else 0
```

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 $e = (e_1, \dots, e_n) = x - y \in \{-1, 0, 1\}^n$. If y is a big subset of x, then $e \in \{0,1\}^n$ with $|e| \le |x| - t \le r$, since $\prod_{i=1}^{n} p_i^{e_i} < B^r < q$. This means E is a product of primes in $\{p_1, \dots, p_n\}$ 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 $\{p_1, \dots, p_n\}$ 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 $\{p_1, \dots, p_n\}$. 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*.

6.1.1 Dealing with False Acceptances

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 $e = (e_1, \dots, e_n) \in \{0,1\}^n$. 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 $\le 2\sqrt{n}$ 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 $p_1, \dots, p_n$ 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 $|\mathrm{im}\ \phi|$ of the group morphism

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

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

whose kernel defines L. Hence

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

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

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

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

$$r > \frac{n \ln(2\sqrt{2\pi e})}{\ln(n \ln n)}. \tag{6}$$

To summarize, if we require that r satisfy Inequality (6) then heuristically there are no false acceptances. Note that Inequality (6) 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:

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

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 on Inequality (6) to avoid false acceptances.

6.1.2 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 6.1. *Let X_n be a distribution over $\{0,1\}^n$ with hamming ball min-entropy λ. Then assuming Assumption 5.7, the obfuscation given by Algorithm 1 - 3 is VBB-secure.*

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 VBB due to Theorem 2.5. For every circuit $C \in \mathcal{C}$, let $O(1^\lambda, C) = (p_1, \ldots, p_n, q, X)$ be the obfuscated function of C. We create a simulator S which works as follows: S takes $C.params = (n, t, B)$ and samples n primes $p'_1, \ldots, p'_n$ and a modulus q' in the same way as O. Denote $S(1^\lambda, C.params) = (p', q', X')$. Note that $O(1^\lambda, C)$ belongs to the distribution D_0 in Assumption 5.7 and $S(1^\lambda, C.params)$ belongs to the distribution D_1. Hence by Assumption

5.7, we have that, for every $n \in \mathbb{N}$, there exists a negligible function μ such that for every PPT distinguisher A,

$$\left| \Pr_{O(1^\lambda,C)\leftarrow D_0}[A(O(1^\lambda,C),aux)=1] - \Pr_{S(1^\lambda,C.params)\leftarrow D_1}[A(S(1^\lambda,C.params),aux)=1] \right| \leq \mu(\lambda), \quad (7)$$

where aux is some auxiliary information.

Also, note that

$$\Pr_{(C,aux)\leftarrow D_\lambda}[A(O(1^\lambda,C),aux')=1] = \Pr_{O(1^\lambda,C)\leftarrow D_0}[A(O(1^\lambda,C),aux)=1],$$

and

$$\Pr_{(C,aux)\leftarrow D_\lambda}[A(S(1^\lambda,C.params),aux')=1] = \Pr_{S(1^\lambda,C.params)\leftarrow D_1}[A(S(1^\lambda,C.params),aux)=1], \quad (8)$$

where $aux' = (aux, \varphi(C))$, and $\varphi : C_\lambda \to \{0,1\}$ is any (non-uniform) polynomial time predicate. I.e., the advantage in the distributional-indistinguishability game (Definition 2.3) equals the advantage in Assumption 5.7. We therefore have

$$\left| \Pr_{(C,aux)\leftarrow D_\lambda}[A(O(1^\lambda,C),aux')=1] - \Pr_{(C,aux)\leftarrow D_\lambda}[A(S(1^\lambda,C.params),aux')=1] \right| \leq \mu(\lambda). \quad (9)$$

This completes the proof. □

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

Theorem 6.2. *Let $n,t,r,B \in \mathbb{N}$ satisfy Definition 5.5 and Inequality (4). Then assuming Conjecture 5.9, Conjecture 5.10, Assumption 5.7, and the hardness of SP (Assumption 5.6), 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 5.6. 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 $(p_1,\ldots,p_n,q,X)$. Since r satisfies Inequality (4), i.e., there is 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 theorems for SSF.

Theorem 6.3. *Let $n, t, r, B \in \mathbb{N}$ satisfy Definition 5.5 and Inequality (4). Then assuming Conjecture 5.9, Conjecture 5.10, Assumption 5.7, and the hardness of SP (Assumption 5.6), 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 6.4 (Diophantine Approximation [11]). *Let $\alpha \in \mathbb{R}$ then there exist fractions $p/q \in \mathbb{Q}$ such that $\left|\alpha - \frac{a}{b}\right| < \frac{1}{\sqrt{5}b^2}$. If, on the other hand, there exist $a/b \in \mathbb{Q}$ such that $\left|\alpha - \frac{a}{b}\right| < \frac{1}{2b^2}$, then a/b is a convergent of α.*

An attack based on Theorem 6.4 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 6.4, 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 6.4.

Theorem 6.5 (Extended Legendre Theorem [8]). *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 6.5 one can always find a and b by tuning c, which gets rid of the limitation from $\left|\alpha - \frac{a}{b}\right| < \frac{1}{2b^2}$.

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 4.1, 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 (4)).

(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 (6)).

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

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

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.

References

[1] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. "Obfuscation for Evasive Functions". In: *Theory of Cryptography*. Ed. by Yehuda Lindell. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 26–51. ISBN: 978-3-642-54242-8.

[2] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. "On the (im) possibility of obfuscating programs". In: *Annual International Cryptology Conference (CRYPTO)*. Springer. 2001, pp. 1–18.

[3] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs". In: *Advances in Cryptology — CRYPTO 2001*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 1–18. ISBN: 978-3-540-44647-7.

[4] James Bartusek, Brent Carmer, Abhishek Jain, Zhengzhong Jin, Tancrède Lepoint, Fermi Ma, Tal Malkin, Alex J. Malozemoff, and Mariana Raykova. "Public-Key Function-Private Hidden Vector Encryption (and More)". In: *Advances in Cryptology – ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Cham: Springer International Publishing, 2019, pp. 489–519. ISBN: 978-3-030-34618-8.

[5] James Bartusek, Tancrède Lepoint, Fermi Ma, and Mark Zhandry. "New Techniques for Obfuscating Conjunctions". In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Cham: Springer International Publishing, 2019, pp. 636–666. ISBN: 978-3-030-17659-4.

[6] Ward Beullens and Hoeteck Wee. "Obfuscating Simple Functionalities from Knowledge Assumptions". In: *Public-Key Cryptography – PKC 2019*. Ed. by Dongdai Lin and Kazue Sako. Cham: Springer International Publishing, 2019, pp. 254–283. ISBN: 978-3-030-17259-6.

[7] Allison Bishop, Lucas Kowalczyk, Tal Malkin, Valerio Pastro, Mariana Raykova, and Kevin Shi. "A Simple Obfuscation Scheme forPattern-Matching with Wildcards". In: *Advances in Cryptology – CRYPTO 2018*. Ed. by Hovav Shacham and Alexandra Boldyreva. Cham: Springer International Publishing, 2018, pp. 731–752. ISBN: 978-3-319-96878-0.

[8] Andrej Dujella. “A variant of Wieners attack on RSA”. In: *Computing* 85.1-2 (2009), pp. 77–83.

[9] Benjamin Fuller, Leonid Reyzin, and Adam Smith. “When Are Fuzzy Extractors Possible?” In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 277–306. ISBN: 978-3-662-53887-6.

[10] Steven D. Galbraith and Lukas Zobernig. “Obfuscated Fuzzy Hamming Distance and Conjunctions from Subset Product Problems”. In: *Theory of Cryptography*. Ed. by Dennis Hofheinz and Alon Rosen. Cham: Springer International Publishing, 2019, pp. 81–110. ISBN: 978-3-030-36030-6.

[11] Adolf Hurwitz. “Über die angenäherte Darstellung der Irrationalzahlen durch rationale Brüche”. In: *Mathematische Annalen* 39.2 (1891), pp. 279–284.

[12] Daniel Wichs and Giorgos Zirdelis. “Obfuscating compute-and-compare programs under LWE”. In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 600–611.

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 A.1. *(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*

$$B_{n+1,2n,q} = \begin{bmatrix} 1 & 2 & \dots & 2n \\ 1 & 2^2 & \dots & (2n)^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2^{n+1} & \dots & (2n)^{n+1} \end{bmatrix} \tag{10}$$

- $\mathsf{Obf}(\mathsf{pat} \in \{0,1,*\}^n)$. *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 A.2. *Suppose the Discrete Logarithm Problem (DLP) is hard (Assumption 5.4). Then the conjunction obfuscation in Definition A.1 is input-hiding.*

Proof. Let $(B, v = g^{B \cdot e})$ be defined as in Definition A.1. 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 (10) 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

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

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 that how to use the techniques in this paper to obfuscate conjunctions in a highly efficient way. In particular, the obfuscation here is even more efficient than the scheme in [10], which is already very efficient. In a high level, our scheme only bases on uniqueness of integer factoring without using continued fraction. We therefore can cut off from the decoding algorithm in [10] the procedures that deal with continued fractions, resulting in a very simple algorithm.

Conjunctions are also called pattern matching with wildcards. Typically it is defined as a function $f_{n,r,x}(y)$ that holding 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 matches all non-wildcard positions of x, or outputs 0 otherwise. We define $|x|$ to be the number of 1's in the string.

We give an equivalent definition based on 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. A 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 the above definition we only require subset and superset, not big subset or small superset. In fact, requiring both 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 big subset and small superset obfuscation problems. However, the catch is that the two instances are "correlated" and the previous security analysis is not sufficient.

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. We explain our obfuscation as follows.

To obfuscate, we derive two binary strings $w, z \in \{0,1\}^n$ from x, where w is x with the wildcard positions set 1, and z is x with the wildcard positions set 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 output $(p_1, l_1, \ldots, p_n, l_n, q, s, X_1, X_2)$ as the obfuscated function.

To evaluate with 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.

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 the 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.

B.1 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 4.1, 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 $\frac{n}{\ln n}$.

B.2 Construction

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 $(p_1, l_1, \ldots, p_n, l_n, q, s, X_1, X_2)$
```

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. The correctness is similar to the BSF and SSF obfuscation. In the following we mainly talk about false acceptances.

As before, let us denote $U_1 = \prod_{i=1}^n p_i^{w_i}$, $U_2 = \prod_{i=1}^n l_i^{z_i}$, $V_1 = \prod_{i=2}^n p_i^{y_i}$, and $V_2 = \prod_{i=1}^n l_i^{y_i}$. Also $E_1 = U_1/V_1 \pmod q$ and $E_2 = V_2/U_2 \pmod q$.

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. But the drawback is that this will leak the wildcard positions since the attacker can factor $X_1 X_2^{-1}$ to get the primes corresponding to the wildcard positions.

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 $i \in \{1, \dots, n\}$ 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 $e_1 \in \{0,1\}^n$, $|e_1| \le r$ 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 $e_2 \in \{0,1\}^n$, $|e_2| \le r$ 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

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

To avoid false acceptances, it is sufficient to let the shortest vectorλ_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

$$r > \frac{\frac{1}{2} n \ln \frac{8r\pi e}{n}}{\ln(n \ln n)} \tag{11}$$

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.

Security. We prove input-hiding of our scheme.

Definition B.1 (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 5.2) 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'.*

Assumption B.2 (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 5.6 with $|x \oplus x'| < r$. Then for every $n \in \mathbb{N}$, there exists a negligible function μ such that for every PPT algorithm A, the probability that A solves any one of the SPs is not greater than $\mu(\lambda)$.*

Theorem B.3 (Input-hiding of Conjunction Obfuscation). *Assuming the hardness of TSP (Assumption B.2), 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 secrete 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 $e_i = 1$. □