Obfuscated Fuzzy Hamming Distance and Conjunctions from Subset Product Problems

Steven D. Galbraith(✉) and Lukas Zobernig

Department of Mathematics, The University of Auckland, Auckland, New Zealand
{s.galbraith,lukas.zobernig}@auckland.ac.nz

Abstract. We consider the problem of obfuscating programs for fuzzy matching (in other words, testing whether the Hamming distance between an n-bit input and a fixed n-bit target vector is smaller than some predetermined threshold). This problem arises in biometric matching and other contexts. We present a virtual-black-box (VBB) secure and input-hiding obfuscator for fuzzy matching for Hamming distance, based on certain natural number-theoretic computational assumptions. In contrast to schemes based on coding theory, our obfuscator is based on computational hardness rather than information-theoretic hardness, and can be implemented for a much wider range of parameters. The Hamming distance obfuscator can also be applied to obfuscation of matching under the ℓ_1 norm on $\mathbb{Z}^n$.

We also consider obfuscating conjunctions. Conjunctions are equivalent to pattern matching with wildcards, which can be reduced in some cases to fuzzy matching. Our approach does not cover as general a range of parameters as other solutions, but it is much more compact. We study the relation between our obfuscation schemes and other obfuscators and give some advantages of our solution.

1 Introduction

Program obfuscation is a major topic in cryptography. Since it was shown that *virtual black box (VBB) obfuscation* is impossible in general [3], a large amount of research has gone into constructing solutions in special cases. One special case that has attracted attention is evasive functions [2,37]. Evasive functions are programs for which it is hard to find an accepting input from black-box access to the program. There are some classes of evasive functions that are quite efficiently obfuscated, such as hyperplane membership [12], logical formulae defined by many conjunctions [9,10], pattern matching with wild cards [4,6], root of a polynomial system of low degree [2], compute-and-compare programs [26,44], and more [37].

More general obfuscation tools are less practical. For example, there are candidates for *indistinguishability obfuscation* [24], but the downside of all of these schemes is they are completely infeasible in terms of runtime and are only

D. Hofheinz and A. Rosen (Eds.): TCC 2019, LNCS 11891, pp. 81–110, 2019.
https://doi.org/10.1007/978-3-030-36030-6_4

able to deal with very simple programs. While general-purpose and efficient obfuscation is a "holy grail", the hope is that by restricting to a narrow class of programs we are able to construct customised and practical solutions.

Hamming Distance. One very natural class of evasive functions is fuzzy matching for the Hamming metric. Define the program $P_x(y)$, parametrised by $x \in \{0,1\}^n$ and a threshold $0 < r < n/2$, that determines whether or not the n-bit input y has Hamming distance at most r from x. Let D be a distribution on $\{0,1\}^n$. Then D determines a program collection $\mathcal{P} = \{P_x : x \leftarrow D\}$. For $\mathcal{P}$ to be an evasive program collection it is necessary that the distribution D has high Hamming ball min-entropy (see Definition 2.4; this notion is also known as fuzzy min-entropy in [23]). The uniform distribution on $\{0,1\}^n$ is an example of such a distribution.

We are interested in obfuscating the membership program $P_x(y)$, so that the description of P_x does not reveal the value x. Note that once an accepting input $y \in \{0,1\}^n$ is known then one can easily determine x by a sequence of chosen executions of P_x. Indeed, as with most other solutions to this problem, our scheme is essentially an error correcting code, and so the computation recovers the value x.

Fuzzy matching has already been treated by many authors and there is a large literature on it. For example, Dodis et al. [18–20] introduced the notion of *secure sketch* and a large number of works have built on their approach. They also show how to obfuscate proximity queries.

One drawback of the secure sketch approach is that the parameters are strongly constrained by the need for an efficient decoding algorithm. As discussed by Bringer et al. [11] this leads to "a trade-off between the correction capacity and the security properties of the scheme". In contrast, our scheme is based on computational hardness rather than information-theoretic hardness, and can be implemented for a much wider range of parameters.

A different solution to fuzzy matching was given by Karabina and Canpolat [33], based on computational assumptions related to the discrete logarithm problem. Note that they do not mention obfuscation or give a security proof. Wichs and Zirdelis [44] note that fuzzy matching can be obfuscated using an obfuscator for compute-and-compare programs.

Our Contribution. We give a full treatment of fuzzy matching based on computational assumptions. We present an extremely practical and efficient obfuscator, based on a natural number-theoretic computational assumption we call the *modular subset product problem*. In short, given $r < n/2$, distinct primes $(p_i)_{i=1,\ldots,n}$, a prime q such that $\prod_{i \in I} p_i < q$ for all subsets I of $\{1,\ldots,n\}$ of size r, and an integer $X = \prod_{i=1}^{n} p_i^{x_i} \mod q$ for some secret vector $x \in \{0,1\}^n$, the problem is to find x. We call the decisional version of the problem the *decisional modular subset product problem*: Distinguish between a modular subset product instance and uniformly random element of $(\mathbb{Z}/q\mathbb{Z})^*$. If $q \leq 2^n$, we conjecture that the statistical distance of the distribution $\prod_{i=1}^{n} p_i^{x_i} \mod q$ for uniform x

and the uniform distribution on $(\mathbb{Z}/q\mathbb{Z})^*$ is negligible. For $q > 2^n$ we conjecture that the distributions are computationally indistinguishable.

The modular subset product problem is similar to computational problems that have been used in previous works in cryptography [15].

In practice our scheme improves upon all previous solutions to this problem: It handles a wider range of parameters than secure sketches; it is 20 times faster than [33]; it is many orders of magnitude more compact than [13,44]; for full discussion see Sect. 7.5. Our solution is related to [33], but we think our approach is simpler and furthermore we give a complete security analysis.

We present two variants of our scheme. One is based only on the subset-product assumption but when r is very small, it admits the possibility of accepting an input y that is not within the correct Hamming ball. The second variant (and the one we present in the main body of this work) assumes the existence of a *dependent auxiliary input point function obfuscator* [5,7,8,37,43] and is perfectly correct. The key idea is to use the point function obfuscator to verify the Hamming ball center after error correction, see Sect. 7.3 for details. The auxiliary input in our case is the tuple $((p_i)_{i=1,\ldots,n}, q, X)$.

The following theorems are both special cases of Theorem 8.1, which we will prove later. The first one shows that, under the subset product assumption, our scheme gives a secure obfuscator for the uniform distribution over $\{0,1\}^n$ when n is sufficiently large.

Theorem 1.1. *Let $\lambda \in \mathbb{N}$ be a security parameter. Consider the family of parameters $(n, r) = (6\lambda, \lambda)$. Assuming the decisional modular subset product problem is hard and that there exists a dependent auxiliary input distributional VBB point function obfuscator, the Hamming distance obfuscator for a uniformly distributed $x \in \{0,1\}^n$ is distributional VBB secure.*

Taking $\lambda = 170$ gives parameters $(n, r) = (1020, 170)$, which are beyond the reach of practical secure sketches.

Under the further constraint $r < n/\log(n\log(n))$ we prove security under the discrete logarithm assumption. Taking $n = 1000$ this allows $r = 113$. This gives us the following

Theorem 1.2. *Let $\lambda > 20$ be a security parameter. Consider the family of parameters $(n, r) = (6\lambda, n/\log(n\log(n)))$. Under the discrete logarithm assumption and the assumptions of Theorem 5.1 and assuming that there exists a dependent auxiliary input distributional VBB point function obfuscator, the Hamming distance obfuscator for a uniformly distributed $x \in \{0,1\}^n$ is distributional VBB secure.*

Conjunctions. Another class of evasive functions are conjunctions. A conjunction on Boolean variables $b_1, \ldots, b_n$ is $\chi(b_1, \ldots, b_n) = \bigwedge_{i=1}^{k} c_i$ where each c_i is of the form b_j or $\neg b_j$ for some $1 \leq j \leq n$.

An alternative representation of a conjunction is called *pattern matching with wildcards*. Consider a vector $x \in \{0,1,\star\}^n$ of length $n \in \mathbb{N}$ where $\star$ is a special

wildcard symbol. Such an x then corresponds to a conjunction $\chi : \{0,1\}^n \to \{0,1\}$ which, using Boolean variables $b_1, \ldots, b_n$, can be written as $\chi(b) = \bigwedge_{i=1}^{n} c_i$ where $c_i = \neg b_i$ if $x_i = 0$, $c_i = b_i$ if $x_i = 1$, and $c_i = 1$ if $x_i = \star$.

Conjunction obfuscators have been considered before [4,6,9,10]. It is clear that, if the number r of wildcards is sufficiently smaller than $n/2$, one can reduce pattern matching with wildcards to fuzzy Hamming matching. Hence our solution also gives an alternative approach to obfuscating conjunctions that can be used for certain parameter ranges. We give a full security analysis and comparison to existing schemes.

Applications. Hamming distance obfuscators are interesting for a variety of applications. One major application is biometric matching, where a biometric reading is taken and matched with a stored template [32,36,41,42]. Since biometric readings (e.g., fingerprints, iris scans, or facial features) are a noisy process, the binary string representing the extracted features may not be an exact match for the template string.

Our Approach. We give a short summary of our Hamming distance obfuscator. Let $r, n \in \mathbb{N}$ with $r < n/2$. We wish to "encode" an element $x \in \{0,1\}^n$ so that it is hidden, and yet we will be able to recognise if an input $y \in \{0,1\}^n$ is within Hamming distance r of x. To do this we will sample a sequence of small distinct primes $(p_i)_{i=1,\ldots,n}$ (i.e., $p_i \neq p_j$ for $i \neq j$) and a small safe prime q such that $\prod_{i \in I} p_i < q/2$ for all $I \subset \{1, \ldots, n\}$ with $|I| \leq r$. The encoding of $x \in \{0,1\}^n$ is

$$X = \prod_{i=1}^{n} p_i^{x_i} \mod q$$

along with the primes $(p_i)_{i=1,\ldots,n}$ and q. Given an input $y \in \{0,1\}^n$ anyone can compute the encoding $Y = \prod_{i=1}^{n} p_i^{y_i} \mod q$ and then compute

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

for *errors* $\epsilon_i = x_i - y_i \in \{-1, 0, 1\}$. If y is close to x then almost all ϵ_i are zero, and so we are able to recover the errors ϵ_i using the continued fraction algorithm and factoring. We give the background theory and explanation in Sect. 6. In Remark 1 we explain that this technique also applies to matching vectors in $\mathbb{Z}^n$ under the ℓ_1-norm.

Note that these ideas have also been used in [39] where they are used to construct a *number theoretic error correcting code*. Some major differences to our scheme are: In [39] the parameters $(p_i)_{i=1,\ldots,n}$ and q are fixed for all messages; the encoding X is appended to each message. A similar application is given in [21] to construct a lattice with efficient bounded distance decoding.

Outline of This Work. Sections 2, 3, and 4 introduce basic notions in hamming distance/fuzzy matching and obfuscation. Section 5 introduces our new computational assumption. Section 6 gives background on continued fractions. Section 7 presents the obfuscator for fuzzy matching in the Hamming distance, including parameters and performance. Section 8 proves that the obfuscator is VBB secure and input-hiding. Section 9 briefly discusses our solution to obfuscating conjunctions.

2 Hamming Distance

We want to obfuscate the function that determines if an input binary vector is close to a fixed target. In our setting, one of the input vectors will be a secret and the other an arbitrary input. Let us first state some key definitions.

A natural property of a binary vector is its *Hamming weight* which is the number of non-zero elements of the vector. For $x \in \{0,1\}^n$, we will denote this by $w_H(x)$. The *Hamming distance* between two binary vectors $x, y \in \{0,1\}^n$ is then given by $d_H(x,y) = w_H(x-y)$. Finally, a *Hamming ball* $B_{H,r}(x) \subset \{0,1\}^n$ of radius r around a point $x \in \{0,1\}^n$ is the set of all points with Hamming distance at most r from x. We denote by $B_{H,r}$ the Hamming ball around an unspecified point.

2.1 Hamming Ball Membership over Uniformly Chosen Centers

We are interested in programs that determine if an input binary vector $y \in \{0,1\}^n$ is contained in a Hamming ball of radius r around some secret value x, i.e., if $y \in B_{H,r}(x)$. This problem is only interesting if it is hard for a user to determine such an input y, because if it is easy to determine values y such that $y \in B_{H,r}(x)$ and also easy to determine values y such that $y \notin B_{H,r}(x)$ then an attacker can easily learn x by binary search. So the first task is to find conditions that imply it is hard to find a y that is accepted by such a program. In other words, we need conditions that imply Hamming ball membership is an *evasive* problem. As we will see in Fig. 1, there are essentially three ways that this problem can become easy: if the Hamming balls are too big; if there are too few possible centers x; or if the centers x are clustered together.

Definition 2.1 (Evasive Program Collection). *Let $\mathcal{P} = \{\mathcal{P}_n\}_{n\in\mathbb{N}}$ be a collection of polynomial-size programs such that every $P \in \mathcal{P}_n$ is a program $P : \{0,1\}^n \to \{0,1\}$. The collection $\mathcal{P}$ is called* evasive *if there exists a negligible function ϵ such that for every $n \in \mathbb{N}$ and for every $y \in \{0,1\}^n$:*

$$\Pr_{P \leftarrow \mathcal{P}_n}[P(y) = 1] \leq \epsilon(n).$$

In short, Definition 2.1 means that a random program from an evasive collection $\mathcal{P}$ evaluates to 0 with overwhelming probability. Finally, we call a member $P \in \mathcal{P}_n$ for some $n \in \mathbb{N}$ an *evasive program* or an *evasive function.*

Hamming Ball Program Collection. Let $r, n \in \mathbb{N}$ with $0 < r < n/2$. For every binary vector $x \in \{0,1\}^n$ there exists a polynomial size program $P_x : \{0,1\}^n \to \{0,1\}$ that computes whether the input vector $y \in \{0,1\}^n$ is contained in a Hamming ball $B_{H,r}(x)$ and evaluates to 1 in this case, otherwise to 0. Any distribution on $\{0,1\}^n$ therefore gives rise to a distribution $\mathcal{P}_n$ of polynomial-size programs.

We first consider the uniform distribution on $\{0,1\}^n$, so that sampling $P \leftarrow \mathcal{P}_n$ means choosing x uniformly in $\{0,1\}^n$ and setting $P = P_x$. Since the condition $y \in B_{H,r}(x)$ is equivalent to $x \in B_{H,r}(y)$ we need to determine the probability that a random element lies in a Hamming ball. This is done in the next two lemmas. Note that if $r \geq n/2$ then a random element lies in the Hamming ball with probability $\geq 1/2$, which is why we are always taking $r < n/2$.

Lemma 2.1. *Let $n \in \mathbb{N}$, $x \in \{0,1\}^n$. The number of elements in a Hamming ball $B_{H,r}(x) \subseteq \{0,1\}^n$ of radius r is given by $h_r = |B_{H,r}| = \sum_{k=0}^{r} \binom{n}{k}$.*

Proof. This can be readily seen from the fact that for each $k \in [0, r]$ a vector has $\binom{n}{k}$ possible ways to be at Hamming distance of k from the origin point. □

Next we show that the probability for a randomly chosen element in $\{0,1\}^n$ to be contained in a Hamming ball $B_{H,r}$ is negligible if the parameters $r < n/2$ are chosen properly.

Lemma 2.2. *Let $\lambda \in \mathbb{N}$ be a security parameter and let $r, n \in \mathbb{N}$ such that $r \leq n/2 - \sqrt{n\lambda \log(2)}$. Fix a point $x \in \{0,1\}^n$. Then the probability that a randomly chosen vector $y \in \{0,1\}^n$ is contained in a Hamming ball of radius r around x satisfies $\Pr_{y \leftarrow \{0,1\}^n}[y \in B_{H,r}(x)] \leq 1/2^\lambda$.*

Proof. The total number of points in $\{0,1\}^n$ is given by 2^n. By Lemma 2.1 we thus have the probability of a randomly chosen vector $y \in \{0,1\}^n$ to be contained in a Hamming ball of radius r around a point x given by $h_r/2^n = 2^{-n} \sum_{k=0}^{r} \binom{n}{k}$. On the other hand, consider the cumulative binomial distribution for probability p which for $r \leq np$ is bounded[1] by

$$\Pr(X \leq r) = \sum_{k=0}^{r} \binom{n}{k} p^k (1-p)^{n-k} \leq \exp\left(-\frac{1}{2p} \frac{(np-r)^2}{n}\right).$$

Substitute $p = 1/2$ to find $\Pr(X \leq r) = h_r/2^n$. Hence, for $r < n/2 - \sqrt{n\lambda \log(2)}$ we have $h_r/2^n \leq \exp\left(-(n/2 - r)^2 / n\right) \leq 1/2^\lambda$ and the result follows. □

This result shows that Hamming ball membership over the uniform distribution is evasive when r is small enough.

Lemma 2.3. *Let $\lambda(n)$ be such that the function $1/2^{\lambda(n)}$ is negligible. Let $r(n)$ be a function such that $r(n) \leq n/2 - \sqrt{\log(2) n \lambda(n)}$. Let $\mathcal{P}_n$ be the set of programs that tests Hamming ball membership in $B_{H,r(n)}(x) \subseteq \{0,1\}^n$ over uniformly sampled $x \in \{0,1\}^n$. Then $\mathcal{P}_n$ is an evasive program collection.*

[1] Chernoff bound for binomial distribution tail [1,14].

Proof. We need to show that, for every $n \in \mathbb{N}$ and for every $y \in \{0,1\}^n$, $\Pr_{P \leftarrow \mathcal{P}_n}[P(y) = 1]$ is negligible. Note that

$$\Pr_{P \leftarrow \mathcal{P}_n}[P(y) = 1] = \Pr_{x \leftarrow \{0,1\}^n}[y \in B_{H,r(n)}(x)] = \Pr_{x \leftarrow \{0,1\}^n}[x \in B_{H,r(n)}(y)]$$

and this is negligible by Lemma 2.2. □

2.2 Hamming Ball Membership over General Distributions

Biometric templates may not be uniformly distributed in $\{0,1\}^n$, so it is important to have a workable theory for fuzzy matching without assuming that the input data is uniformly sampled binary strings. For example, in the worst case, there is only a small number of possible values $x \in \{0,1\}^n$ that arise, in which case taking y to be one of these x-values will show that Hamming ball membership is not evasive. More generally, as pictured in the right hand panel of Fig. 1, one could have many centers but if they are too close together then there might be values for y such that $\Pr_{P \leftarrow \mathcal{P}_n}[P(y) = 1]$ is not negligible.

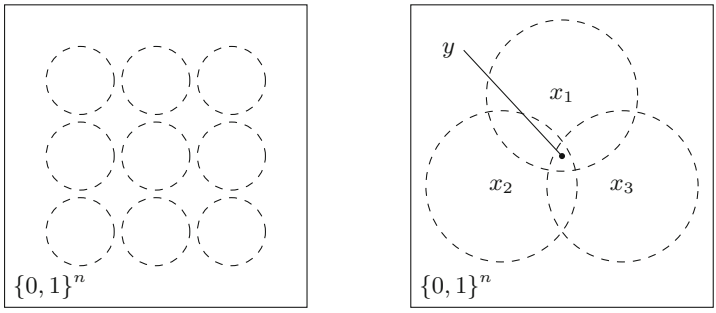


Fig. 1. Two example cases of Hamming ball distributions. The left side depicts the ideal distribution of Hamming ball centers. The right one shows what happens if the balls overlap.

Hence, for Hamming ball membership to be evasive, the centers x must be chosen from a "reasonably well spread" distribution. Before treating this in detail we give some definitions related to entropy of distributions in the computational sense.

Definition 2.2 (Min-Entropy). *The* min-entropy *of a random variable X is defined as* $\mathrm{H}_\infty(X) = -\log(\max_x \Pr[X = x])$. *The* (average) conditional min-entropy *of a random variable X conditioned on a correlated variable Y is defined as* $\mathrm{H}_\infty(X|Y) = -\log(\mathrm{E}_{y \leftarrow Y}[\max_x \Pr[X = x|Y = y]])$.

Definition 2.3 (Computational Indistinguishability). *We say that two ensembles of random variables $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are* computationally indistinguishable *and write $X \overset{c}{\approx} Y$ if for every (non-uniform) PPT*

distinguisher $\mathcal{A}$ *it holds that* $\Pr[\mathcal{A}(X_\lambda) = 1] - \Pr[\mathcal{A}(Y_\lambda) = 1] \leq \epsilon(\lambda)$ *where* $\epsilon(\lambda)$ *is some negligible function.*

Now suppose we have a distribution D_n on $\{0,1\}^n$, which defines a distribution P_n of Hamming ball membership programs. For the program collection to be evasive (i.e., to satisfy Definition 2.1), it is necessary that for any $y \in \{0,1\}^n$ we have $\Pr_{P \leftarrow \mathcal{P}_n}[P(y) = 1]$ being negligible. But note that

$$\Pr_{P \leftarrow \mathcal{P}_n}[P(y) = 1] = \Pr_{x \leftarrow D_n}[y \in B_{H,r}(x)] = \Pr_{x \leftarrow D_n}[x \in B_{H,r}(y)].$$

So the requirement for evasiveness is that this probability is negligible. In other words, we need that D_n has large min-entropy in the following sense.

Definition 2.4 (Hamming Ball Min-Entropy). *(Also known as* fuzzy min-entropy *[23, Definition 3].) The* Hamming ball min-entropy *of a random variable* X *on* $\{0,1\}^n$ *is defined to be*

$$\mathrm{H}_{H,\infty(X)} = -\log\left(\max_{y \in \{0,1\}^n} \Pr[X \in B_{H,r}(y)]\right).$$

For convenience, we give some necessary conditions to have Hamming ball min-entropy at least λ. Let $|D_n| = \{x \in \{0,1\}^n : \Pr(x \leftarrow D_n) > 0\}$ be the support of D_n. If for any $y \in \{0,1\}^n$

$$\frac{\left|\bigcup_{x \in |D_n|} B_{H,r}(x)\right|}{|B_{H,r}(y)|} < 2^\lambda$$

then we certainly do not have min-entropy at least λ. Hence at the very least it is required that points in D_n are well-spread-out, as pictured in the left-hand panel of Fig. 1.

Intuitively, we can say that if there are enough points in $|D_n|$ and if they are spread out such that the overlap between the Hamming balls is relatively small, then Hamming ball membership is an evasive problem.

Definition 2.5 (Hamming Distance Evasive Distribution). *Consider an ensemble of distributions* D_λ *over* $\{0,1\}^{n(\lambda)}$*, call it* $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$*. Let* $r(\lambda) < n(\lambda)/2$ *be some function. We say that* D *is* Hamming distance evasive *if the Hamming ball min-entropy of* D_λ *for Hamming balls in* $\{0,1\}^{n(\lambda)}$ *of radius* $r(\lambda)$ *(as in Definition 2.4) is at least* λ*.*

3 Conjunctions

Similar to Sect. 2, we will first give basic definitions regarding conjunctions and then determine necessary conditions for a given conjunction to be evasive.

Definition 3.1 (Conjunction/Pattern Matching With Wildcards). *Let $n \in \mathbb{N}$ and let $x \in \{0,1,\star\}^n$ where $\star$ is a special* wildcard *symbol. Such an x then corresponds to a* conjunction $\chi : \{0,1\}^n \to \{0,1\}$ *which, using a vector of Boolean variables $b = (b_1, \ldots, b_n)$, can be written as $\chi(b) = \bigwedge_{i=1}^{n} c_i$ where $c_i = \neg b_i$ if $x_i = 0$, $c_i = b_i$ if $x_i = 1$, and $c_i = 1$ if $x_i = \star$. Denote by $W_x = \{i | x_i = \star\}$ the set of all wildcard positions and let $r = |W| \in \mathbb{N}$ be the number of wildcards.*

Note that a priori the input is considered a plaintext and directly visible to the evaluating party. The wildcard positions of an obfuscated conjunction are only secret as long as no matching input is known. Once such an input is presented to the evaluator, it is straightforward to work out all wildcard positions in time linear in the input length: Simply flip each input bit and check whether this changed input still matches, in which case the flipped position must be a wildcard.

Lemma 3.1. *Let $\lambda \in \mathbb{N}$ be a security parameter and let $r < n/2 \in \mathbb{N}$ such that $r \leq n - \lambda$. Fix a conjunction χ corresponding to a vector $x \in \{0,1,\star\}^n$ such that $r = |\{i | x_i = \star\}|$. Then the probability that χ evaluates to true on a randomly chosen vector $y \in \{0,1\}^n$ satisfies $\Pr_{y \leftarrow \{0,1\}^n}[\chi(y) = 1] \leq 1/2^\lambda$.*

Proof. The total number of points in $\{0,1\}^n$ is given by 2^n. We thus have the probability of a randomly chosen vector $y \in \{0,1\}^n$ to be matched by χ to be $\Pr_{y \leftarrow \{0,1\}^n}[\chi(y) = 1] = 2^r/2^n$. This probability is upper-bounded by $1/2^\lambda$ if $r \leq n - \lambda$.

Lemma 3.1 shows that all conjunctions which have their non-wildcard values uniformly distributed over $\{0,1\}^{n-r}$ are evasive. For general distributions we need to consider the following

Definition 3.2 (Conjunction Evasive Distribution). *Consider an ensemble $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ of distributions D_λ over $\{0,1,\star\}^{n(\lambda)}$ with $r(\lambda)$-many wildcards for functions $r(\lambda) < n(\lambda)$. We say that D is* conjunction evasive *if the min-entropy of D_λ is at least λ.*

4 Obfuscation Definitions

Our ultimate goal is to prove that our obfuscators are distributional *virtual black box* (VBB) secure. For this we first state the definition of such a distributional VBB obfuscator.

Definition 4.1 (Distributional Virtual Black-Box Obfuscator [2,3]). *Let $\mathcal{P} = \{\mathcal{P}_n\}_{n \in \mathbb{N}}$ be a family of polynomial-size programs with input size n and let $\mathcal{O}$ be a PPT algorithm which takes as input a program $P \in \mathcal{P}$, a security parameter $\lambda \in \mathbb{N}$ and outputs a program $\mathcal{O}(P)$ (which itself is not necessarily in $\mathcal{P}$). Let $\mathcal{D}$ be a class of distribution ensembles $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ that sample $P \leftarrow D_\lambda$ with $P \in \mathcal{P}$. The algorithm $\mathcal{O}$ is a* VBB obfuscator for the distribution class $\mathcal{D}$ *over the program family $\mathcal{P}$ if it satisfies the following properties:*

- *Functionality preserving: There exists a negligible function* $\epsilon(\lambda)$ *such that for all* $P \in \mathcal{P}$
$$1 - \Pr\left[\forall x \in \{0,1\}^n : P(x) = \mathcal{O}(P)(x)\right] \leq \epsilon(\lambda)$$
where the probability is over the coin tosses of $\mathcal{O}$.
- *Polynomial slowdown: For every* $\lambda \in \mathbb{N}$ *and* $P \in \mathcal{P}$, *we have* $|\mathcal{O}(P)| \leq \text{poly}(|P|, \lambda)$.
- *Virtual black-box: For every (non-uniform) polynomial size adversary* $\mathcal{A}$, *there exists a (non-uniform) polynomial size simulator* $\mathcal{S}$ *with oracle access to* P, *such that for every* $D = \{D_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{D}$, *and every (non-uniform) polynomial size predicate* $\varphi : \mathcal{P} \to \{0,1\}$:
$$\left| \Pr_{P \leftarrow D_\lambda, \mathcal{O}, \mathcal{A}} \left[\mathcal{A}(\mathcal{O}(P)) = \varphi(P)\right] - \Pr_{P \leftarrow D_\lambda, \mathcal{S}} \left[\mathcal{S}^P(|P|) = \varphi(P)\right] \right| \leq \epsilon(\lambda)$$
where $\epsilon(\lambda)$ *is a negligible function.*

In simple terms, Definition 4.1 states that a VBB obfuscated program $\mathcal{O}(P)$ does not reveal anything more than would be revealed from having black box access to the program P itself.

A definition that is more convenient to work with is *distributional indistinguishability.*

Definition 4.2 (Distributional Indistinguishability [44]). *An obfuscator* $\mathcal{O}$ *for the distribution class* $\mathcal{D}$ *over a family of programs* $\mathcal{P}$ *satisfies* distributional indistinguishability *if there exists a (non-uniform) PPT simulator* $\mathcal{S}$ *such that for every distribution ensemble* $D = \{D_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{D}$ *the following distributions are computationally indistinguishable*
$$(\mathcal{O}(P), \alpha) \overset{c}{\approx} (\mathcal{S}(|P|), \alpha) \tag{4.1}$$
where $(P, \alpha) \leftarrow D_\lambda$. *Here* α *denotes some auxiliary information.*

Note that the sampling procedure for the left and right side of Eq. (4.1) in Definition 4.2 is slightly different. For both we sample $(P, \alpha) \leftarrow D_\lambda$ and for the left side we simply output $(\mathcal{O}(P), \alpha)$ immediately. On the other hand, for the right side we record $|P|$, discard P and finally output $(\mathcal{S}(|P|), \alpha)$ instead.

It can be shown that distributional indistinguishability implies VBB security under certain conditions. To see this, we first require the following.

Definition 4.3 (Predicate Augmentation [44]). *For a distribution class* $\mathcal{D}$, *its* augmentation under predicates $\text{aug}(\mathcal{D})$ *is defined as follows: For any (non-uniform) polynomial-time predicate* $\varphi : \{0,1\}^* \to \{0,1\}$ *and any* $D = \{D_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{D}$, *the class* $\text{aug}(\mathcal{D})$ *indicates the distribution* $D' = \{D'_\lambda\}_{\lambda \in \mathbb{N}}$ *where* D'_λ *samples* $(P, \alpha) \leftarrow D_\lambda$, *computes* $\alpha' = (\alpha, \varphi(P))$ *and outputs* (P, α'). *Here* α *denotes some auxiliary information.*

Theorem 4.1 (Distributional Indistinguishability Implies VBB [10]). *For any family of programs $\mathcal{P}$ and a distribution class $\mathcal{D}$ over $\mathcal{P}$, if an obfuscator satisfies distributional indistinguishability (Definition 4.2) for the class of distributions* $\mathrm{aug}(\mathcal{D})$ *then it also satisfies distributional VBB security for the distribution class $\mathcal{D}$ (Definition 4.1).*

Lastly, we also want to prove that our obfuscator is *input hiding*. For this we state the definition of an input hiding obfuscator.

Definition 4.4 (Input Hiding Obfuscator [2]). *An obfuscator $\mathcal{O}$ for a collection of evasive programs $\mathcal{P}$ is* input hiding, *if for every PPT adversary $\mathcal{A}$ there exists a negligible function ϵ such that for every $n \in \mathbb{N}$ and for every auxiliary input $\alpha \in \{0,1\}^{\mathrm{poly}(n)}$ to $\mathcal{A}$:*

$$\Pr_{P \leftarrow \mathcal{P}_n}[P(\mathcal{A}(\alpha, \mathcal{O}(P))) = 1] \leq \epsilon(n),$$

where the probability is also over the randomness of $\mathcal{O}$.

To summarise, Definition 4.4 states that given the obfuscated program $\mathcal{O}(P)$ it is hard to find an input that evaluates to 1.

5 Computational Assumptions

In this section we introduce our new computational assumptions. We start with the definition of a safe prime.

Definition 5.1 (Safe Prime, Sophie Germain Prime). *A prime q is called a* safe prime *if q is of the form $q = 2p + 1$ for a prime p. The prime p is then called a* Sophie Germain prime.

Problem 5.1 (Distributional Modular Subset Product Problem). *Let $r < n \in \mathbb{N}$, D be a distribution over $\{0,1\}^n$. Given a sequence of distinct primes $(p_i)_{i=1,\ldots,n}$, a safe prime q such that*

$$\prod_{i \in I} p_i < \frac{q}{2} < (1 + o(1)) \max\{p_i\}^r \text{ for all } I \subset \{1, \ldots, n\} \text{ with } |I| \leq r \tag{5.1}$$

and an integer

$$X = \prod_{i=1}^{n} p_i^{x_i} \mod q \tag{5.2}$$

*for some vector $x \leftarrow D$, the (r, n, D)-*distributional modular subset product problem *($MSP_{r,n,D}$) is to find x.*

We also state a decisional version of the problem.

Problem 5.2 (Decisional Distrib. Modular Subset Product Problem). *Let $r < n \in \mathbb{N}$, D be a distribution over $\{0,1\}^n$. Define the distribution*

$$\mathcal{D}_0 = ((p_i)_{i=1,\ldots,n}, q, X)$$

where $(p_i)_{i=1,\ldots,n}$ are distinct primes, q satisfies Eq. (5.1), and X satisfies Eq. (5.2) for some vector $x \leftarrow D$. Define the distribution

$$\mathcal{D}_1 = ((p_i)_{i=1,\ldots,n}, q, X')$$

where $(p_i)_{i=1,\ldots,n}$ and q are as in $\mathcal{D}_0$, but

$$X' \leftarrow (\mathbb{Z}/q\mathbb{Z})^*.$$

Then the (r,n,D)-decisional distributional modular subset product problem *(D-MSP$_{r,n,D}$) is to distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$. In other words, given a sample from $\mathcal{D}_b$ for uniform $b \in \{0,1\}$, the problem is to determine b.*

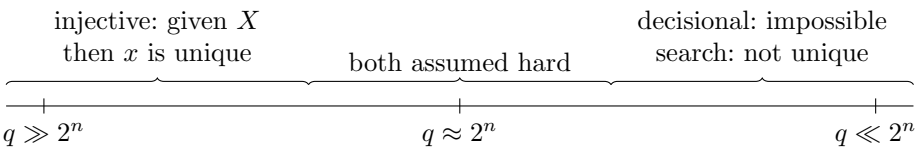
We believe these computational problems are hard whenever the fuzzy matching problem itself is evasive. Precisely we make the following conjecture that covers all possible distributions D.

Conjecture 1. Fix $r < n/2 \in \mathbb{N}$. If D is a distribution on $\{0,1\}^n$ with Hamming ball min-entropy at least λ (i.e. D is a Hamming distance evasive distribution in the sense of Definition 2.5) then solving D-MSP$_{r,n,D}$ (Problem 5.2) requires $\Omega(\min\{2^\lambda, 2^{n/2}\})$ operations.

Note that Problem 5.2 only makes sense if the two distributions are different. In the case $2^n \gg q$ we conjecture that the values $X = \prod_{i=1}^n p_i^{x_i} \mod q$ are distributed close to uniformly if x is sampled uniformly, and so it makes no sense to ask for a distinguisher between this distribution and the uniform distribution. For the proof of Theorem 5.1 we need a more precise version of this statement, so we make the following conjecture that we believe is very reasonable.

Conjecture 2. Let $r, n, (p_i)_{i=1,\ldots,n}, q$ be as in Problem 5.1, with the extra condition that $q \leq 2^n$. Let D be the uniform distribution on $\{0,1\}^n$. Then the statistical distance of the distribution $\prod_{i=1}^n p_i^{x_i} \mod q$ over $x \leftarrow D$ and the uniform distribution on $(\mathbb{Z}/q\mathbb{Z})^*$ is negligible.

The situation is summarised in the following diagram. The left hand side is the *low-density* case. The right hand side is the *high-density* case where for every value X there are likely (multiple) solutions. As can be seen in Fig. 2 our interest reaches over all density cases.



A "search-to-decision" reduction in the low-density or density one case (i.e., $2^n < q$) is possible by borrowing techniques from [31,38]. One can obtain the result that a decision oracle for Problem 5.2 in this case can be used to solve the search problem Problem 5.1 in polynomially many queries to the decision oracle. This gives further evidence that Problem 5.2 should be hard (recall that the assumption makes no sense in the high-density case).

5.1 Algorithms

We now consider algorithms for Problem 5.1. If the Hamming weight of x is too small, or if q is too large, then it might happen that $\prod_{i=1}^{n} p_i^{x_i} < q$ and hence Problem 5.1 can be solved by factoring X over the integers. This is the low-density case. More generally, an approach to Problem 5.1 is to guess some x_i for $i \in I$ and try to factor $X \prod_{i \in I} p_i^{-x_i} \mod q$. It can be shown that if $q = O((n \log(n))^r)$ and if x is sampled from a distribution with large Hamming ball min-entropy then this approach does not lead to an efficient attack. In short, the requirement that the Hamming ball membership program is evasive already implies that such an attack requires exponential time.

We now consider algorithms that are appropriate in general. There is an obvious meet-in-the-middle algorithm: Let $m = \lfloor n/2 \rfloor$. Given X we compute a list L of pairs (z, Z) where $Z = \prod_{i=1}^{m} p_i^{z_i} \mod q$ for all $z \in \{0,1\}^m$. Then for all $z' \in \{0,1\}^{n-m}$ compute $Z' = X \prod_{i=1}^{n-m} p_{m+i}^{-z'_i} \mod q$ and check if Z' is in L. If there is a match then we have found $x = z \| z'$. This attack requires $O(2^{n/2})$ operations. It follows that n must be sufficiently large for the problem to be hard.

5.2 Hardness

We now give evidence that Problems 5.1 and 5.2 are hard in the high-density case. Our argument is based on ideas from index calculus algorithms in finite fields. We prove that if one can solve Problem 5.1 (in the medium-/high-density case) in time T then can solve the discrete logarithm problem (DLP) in $(\mathbb{Z}/q\mathbb{Z})^*$ in time $poly(T)$. Note that this result gives at best a subexponential hardness guarantee, and does not say anything about post-quantum security. A similar computational assumption (called the *very smooth number discrete log*) was considered in [15], where a similar reduction to the discrete logarithm problem is also given.

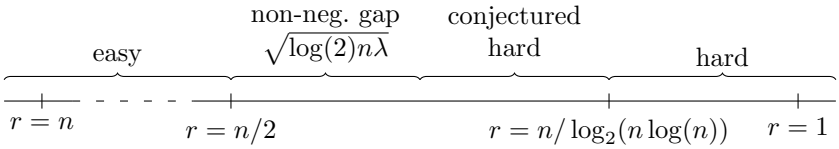
Theorem 5.1. *Fix $r, n \in \mathbb{N}$ such that $r < n/2$. Let q be prime such that $q \leq 2^n$ and $(p_i)_{i=1,\ldots,n}$ be a sequence of distinct primes such that $p_i \in [2, O(n \log(n))]$. Assume Conjecture 2 holds and suppose $MSP_{r,n,D}$ (Problem 5.1) can be solved with probability 1 in time T. Then there is an algorithm to solve the DLP in $(\mathbb{Z}/q\mathbb{Z})^*$ with expected time $\tilde{O}(nT)$.*

Proof. Let $g, h \in \mathbb{Z}_q^*$ be a DLP instance and let $\mathcal{A}$ be an oracle for Problem 5.1 that runs in time T and succeeds with probability 1. Let g be a generator of $(\mathbb{Z}/q\mathbb{Z})^*$ so that its order is $M = q - 1$. Choose random $1 < a < q$ and compute

$C = g^a \mod q$. Call $\mathcal{A}$ on C. Due to the assumptions in the theorem, with probability bounded below by a constant, $\mathcal{A}$ succeeds and outputs a solution x. Store (a, x). Note that each relation implies a linear relation $a \equiv \sum_{i=1}^{n} x_i \log_g(p_i) \mod M$. Repeat until we have n linearly independent relation vectors x, and hence use linear algebra to solve for $\log_g(p_i)$. Finally, choose a random b and set $C = hg^b \mod q$. Call $\mathcal{A}$ on C to get, with high probability, one more relation (b, y). Knowing $\log_g(p_i)$ we now compute $\log_g(h) = -b + \sum_i y_i \log_g(p_i)$. □

The above proof can be generalised to any group whose order is known. When $q = (n \log(n))^r$ then the condition $2^n \geq q$ boils down to $r < n/\log_2(n \log(n))$. Hence, when $r < n/\log_2(n \log(n))$ the hardness of Problem 5.1 follows from the discrete log assumption.

It follows that Problem 5.1 has a spectrum of difficulty, ranging from easy in the extreme low-density case to hard in the medium-/high-density case. We visualise the situation below.



All index calculus algorithms for factoring and discrete logarithms are based on smoothness. A typical situation is to generate certain random elements x modulo N (or p), and check if they are *equal* to $\prod_i p_i^{e_i}$ for primes p_i less than some bound. If one could efficiently compute a smooth product $\prod_i p_i^{e_i}$ that is *congruent* to x modulo N (or p) then factorization and discrete logarithm algorithms would be revolutionised (and classical public key crypto broken).

Our subset product problem is slightly different, since we impose the restriction $e_i \in \{0, 1\}$. But we still believe any fundamentally new algorithmic approach to the problem would likely lead to major advances. The only algorithms we know for this problem are "combinatorial" (in other words, requiring some kind of brute-force search), apart from when the density is extremely low and we can just factor. Note that our parameter choices (e.g. in Fig. 2) are very far from such low density (as we require $r < n/2$ by Lemma 2.3).

We briefly discuss the relation with lattice problems in the next section and in Sect. 7.3. Our feeling is that the subset product problem is not really a lattice problem but a number-theoretical problem. As evidence, references [21,39] use similar number theory ideas to solve coding/lattice type problems. Nevertheless, any new algorithms to solve Problem 5.1 would have implications in lattices, such as giving an improvement on the work of [21].

Ultimately, we are making a new assumption based on our experience and knowledge. A similar assumption was made in [15]. We hope this work will inspire further study of these problems.

5.3 Post-Quantum Security

To the best of our knowledge there exists no classical nor quantum algorithm that efficiently solves either of Problem 5.1 or Problem 5.2 in general.

Consider an adversary that has access to a quantum computer for computing discrete logarithms. Given an encoding $((p_i)_{i=1,\ldots,n}, q, X)$ of a secret $x \in \{0,1\}^n$, the adversary may then turn it into a modular subset sum instance $\log_g(X) = \sum_i x_i \log_g(p_i) \mod q-1$. Such a modular subset sum problem may be classified by its density d, see [16,34]. In our case, when x is chosen from the uniform distribution, the density is $d = n/\log_2(q)$.

There is a polynomial time algorithm for low-density subset sum instances where $d < 0.645$ [34] which was later improved to $d < 0.941$ [16]. This algorithm requires access to a perfect lattice oracle (just using LLL [35] is not enough). It is furthermore assumed that the lattice oracle is *perfect*.

In our case, we can give an estimate for when we expect post-quantum security. By the prime number theorem, we have $q \sim (n \log n)^r$. Thus we can estimate the density by $d \sim n/(r \log_2(n \log n))$. To ensure density of $d > 1$ we can require

$$r < \frac{n}{\log_2(n \log n)} = r_{\mathrm{PQ}}(n). \tag{5.3}$$

Hence we conjecture post-quantum hardness of the modular subset product problem when $r < r_{\mathrm{PQ}}(n)$, and potentially even for slightly larger values for r.

6 Continued Fractions

The background can be found in any number theory textbook, such as [27]. Consider a rational number $x \in \mathbb{Q}$. It has a finite *continued fraction* representation of the form

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots + \cfrac{1}{a_N}}}$$

for $a_i \in \mathbb{N}$. We define the notation $x = [a_0, a_1, a_2, \ldots, a_N]$ for such a representation. In the more general case of $x \in \mathbb{R}$ such a representation also exists, though it is not necessarily finite.

We call the fractions h_i/k_i for an index $i \in \mathbb{N}$ defined by the recursion

$$\begin{aligned} h_i &= a_i h_{i-1} + h_{i-2}, \quad h_{-1} = 1, \quad h_{-2} = 0, \\ k_i &= a_i k_{i-1} + k_{i-2}, \quad k_{-1} = 0, \quad k_{-2} = 1 \end{aligned} \tag{6.1}$$

the *convergents* of x.

Theorem 6.1 (Diophantine Approximation [30]). *Let $\alpha \in \mathbb{R}$ then there exist fractions $p/q \in \mathbb{Q}$ such that $\left|\alpha - \frac{p}{q}\right| < \frac{1}{\sqrt{5}q^2}$. If, on the other hand, there exist $p/q \in \mathbb{Q}$ such that $\left|\alpha - \frac{p}{q}\right| < \frac{1}{2q^2}$, then p/q is a convergent of α.*

To find the continued fraction representation it is useful to review the extended Euclidean algorithm first.

Extended Euclidean Algorithm. For a pair of integers a, b, the extended euclidean algorithm finds integers x, y such that $ax + by = \gcd(a, b)$. The algorithm proceeds as follows: First, it initialises variables r_i, s_i, t_i for $i = 0, 1$ as

$$r_0 = a, \quad r_1 = b, \quad s_0 = 1, \quad s_1 = 0, \quad t_0 = 0, \quad t_1 = 1.$$

Then it iteratively produces the sequence

$$r_{i+1} = r_{i-1} - q_i r_i, \quad s_{i+1} = s_{i-1} - q_i s_i, \quad t_{i+1} = t_{i-1} - q_i t_i. \tag{6.2}$$

Here r_{i+1} and q_i are found using Euclidean division ($r_{i-1} = q_i r_i + r_{i+1}$) such that $0 \leq r_{i+1} < |r_i|$. Finally, the algorithms stops when $r_{i+1} = 0$.

It can be shown that the worst-case runtime of the extended Euclidean algorithm is of the order $O(\log(b))$ assuming that $b < a$; the average runtime is of a similar order [17,28].

Finding Convergents. Comparison of Equation (6.1) with Equation (6.2) shows that the convergents of a fraction $p/q \in \mathbb{Q}$ are exactly produced by the integers s_i, t_i (up to signs) in the steps of the extended Euclidean algorithm applied to p and q. Thus the runtime for computing the continued fraction representation is essentially the same as that of the extended Euclidean algorithm. Furthermore, we see that the number of convergents is linear in the input size.

7 Obfuscating Hamming Distance

In this section we will present our Hamming distance obfuscator in detail and then give some examples for parameter choices. The obfuscator is given in Sect. 7.1.

The Hamming Distance Obfuscator. Let $r, n \in \mathbb{N}$ with $r < n/2$. Choose a random sequence of small distinct primes $(p_i)_{i=1,\ldots,n}$ (i.e., $p_i \neq p_j$ for $i \neq j$). By the prime number theorem it suffices to randomly sample each p_i from the interval $[2, O(n \log(n))]$. Choose then a safe prime q such that $\prod_{i \in I} p_i < q/2$ for all $I \subset \{1, \ldots, n\}$ with $|I| \leq r$. The prime q should be sampled to satisfy the bound $q/2 < (1 + o(1)) \max\{p_i\}^r$ as in Eq. (5.1). We refer to the discussion regarding Eq. (9.3) to justify why we may assume that such a suitable safe prime exists.

To encode an element $x \in \{0, 1\}^n$, publish

$$X = \prod_{i=1}^{n} p_i^{x_i} \mod q \tag{7.1}$$

along with the list of primes $(p_i)_{i=1,\ldots,n}$ and q. Note that, for this encoding to hide x, we require that $w_H(x) > r$ and $\prod_{i=1}^{n} p_i^{x_i} > q$.

Given another element $y \in \{0,1\}^n$ we can now check if $y \in B_{H,r}(x)$ using the encoding X. First we compute $Y = \prod_{i=1}^{n} p_i^{y_i} \mod q$ from which we can find

$$E = XY^{-1} \mod q = \prod_{i=1}^{n} p_i^{x_i - y_i} \mod q = \prod_{i=1}^{n} p_i^{\epsilon_i} \mod q \tag{7.2}$$

where $\epsilon_i \in \{-1, 0, 1\}$. We show in Lemma 7.1 that if $y \in B_{H,r}(x)$ then we are able to recover the errors ϵ_i using continued fraction decomposition and factoring.

Legendre Symbol. Recall that the *Legendre symbol* $(\frac{a}{q})$ is $+1$ if a is a non-zero square modulo q and -1 if a is a non-zero non-square. It is multiplicative in the sense that $(\frac{ab}{p}) = (\frac{a}{p})(\frac{b}{p})$. Thus for X as in Eq. (7.1) the Legendre symbol $(\frac{X}{q})$ is equal to the product $\prod_i (\frac{p_i}{q})^{x_i}$, which reveals a linear equation in the secret $(x_i)_{i=1,\ldots,n}$. In other words, the encoding X leaks one bit of information about x. Note that this does not violate the definition of VBB security: since the primes p_i are chosen randomly by the obfuscator, we cannot fix in advance a predicate and compute it using the Legendre symbol.

Why a Safe Prime. As mentioned, the Legendre symbol leaks a linear equation in (x_i). If there were other small prime divisors of $q-1$ then one could extend this idea to get further linear equations. Hence we choose q to be a safe prime to ensure that only the single bit of leakage arises. An alternative solution would be to square X, but this would mean we need to use larger parameters to do fuzzy matching with Hamming weight r, so we prefer to use the minimal parameters.

7.1 Obfuscator and Obfuscated Program

To be precise, for every pair of integers $r < n/2 \in \mathbb{N}$ and every binary vector $x \in \{0,1\}^n$ there exists a polynomial size program $P_x : \{0,1\}^n \to \{0,1\}$ that computes whether the input vector $y \in \{0,1\}^n$ is contained in a Hamming ball $B_{H,r}(x)$ and evaluates to 1 in this case, otherwise to 0. Denote the family of all such programs with $\mathcal{P}$.

The Hamming distance obfuscator $\mathcal{O}_H : \mathcal{P} \to \mathcal{P}'$ takes one such program $P_x \in \mathcal{P}$ and uses Algorithm 7.2 to output another polynomial size program in a different family denoted by $\mathcal{P}'$. In our case this is the decoding algorithm along with the polynomial size elements $(p_i)_{i=1,\ldots,n}$, q and $X \in \mathbb{Z}/q\mathbb{Z}$.

We furthermore require a dependent auxiliary input point function obfuscator [7,8] that we call $\mathcal{O}_{PT}$. Let $R_z : \{0,1\}^n \to \{0,1\}$ be a program that takes an input $y \in \{0,1\}^n$ and outputs 1 if and only if $y = z$. The point function obfuscator outputs an obfuscated version $\mathcal{O}_{PT}(R_z)$ of R_z. In addition to the output of Algorithm 7.2, our obfuscator $\mathcal{O}_H$ also outputs $Q = \mathcal{O}_{PT}(R_x)$.

As the decoding algorithm is a universal algorithm, we will simply denote the *obfuscated program* $\mathcal{O}_H(P)$ with the tuple $((p_i)_{i=1,\ldots,n}, q, X, Q)$. During the execution of the obfuscated program, Algorithm 7.4 is run on $(n, (p_i)_{i=1,\ldots,n}, q, X, y)$

and returns either $\perp$ (in which case the program returns 0) or a candidate value x'. The obfuscated program then outputs $Q(x')$, which is 1 if and only if $x' = x$. Formally, the obfuscated program is given in Algorithm 7.1.

Algorithm 7.1. Obfuscated Program (with embedded data $(p_i)_{i=1,\ldots,n}, q, X, Q$)

procedure EXECUTE($y \in \{0,1\}^n$)
  $x' =$ DECODE$(n, (p_i)_{i=1,\ldots,n}, q, X, y)$
  if $x' = \perp$ **then return** 0
  return $Q(x')$
end procedure

The encoder (Algorithm 7.2) receives as an input the distance threshold r, the vector size n and the target vector x. It then outputs the encoding represented by a triple $((p_i)_{i=1,\ldots,n}, q, X)$.

Algorithm 7.2. Encoding (Obfuscation)

procedure ENCODE($r < n/2 \in \mathbb{N}$; $x \in \{0,1\}^n$)
  sample a random sequence of distinct primes $(p_i)_{i=1,\ldots,n}$ from $[2, O(n \log(n))]$
  sample small safe prime q such that $\forall I \subset \{1,\ldots,n\}$ with $|I| \leq r$, $\prod_{i \in I} p_i < q/2$
  compute $X = \prod_{i=1}^{n} p_i^{x_i} \mod q$
  return $((p_i)_{i=1,\ldots,n}, q, X)$
end procedure

The constrained factoring algorithm (Algorithm 7.3) factors an input number using a fixed list of primes and outputs the factors respectively fails if the input is composite with factors that are not in the list of primes.

Algorithm 7.3. Constrained Factoring

procedure CFACTOR$(n, (p_i)_{i=1,\ldots,n}, x \in \mathbb{N})$
  set $F = \{\}$
  for $i = 1, \ldots, n$ **do**
    if $p_i \mid x$ **then** append p_i to F and reduce $x \leftarrow x/p_i$
  end for
  return F **if** $x = 1$ **else** $\perp$
end procedure

The decoder (Algorithm 7.4) receives as an input an encoding in the form of a triple $((p_i)_{i=1,\ldots,n}, q, X)$ and a test vector. It then attempts to decode the triple and outputs the original target vector or fails if the test vector was not within the required distance threshold.

Algorithm 7.4. Decoding (Executing the obfuscated program)

procedure DECODE$(n, (p_i)_{i=1,\dots,n}, q \in \mathbb{N};\ X \in \mathbb{Z}/q\mathbb{Z};\ y \in \{0,1\}^n)$
  compute $Y^{-1} = \prod_{i=1}^{n} p_i^{-y_i} \mod q$
  compute $E = XY^{-1} \mod q$
  compute the continued fraction representation of E/q, with convergents C
  for all $h/k \in C$ **do**
    $F \leftarrow \text{CFACTOR}(n, (p_i)_{i=1,\dots,n}, k)$, $F' \leftarrow \text{CFACTOR}(n, (p_i)_{i=1,\dots,n}, kE \mod q)$
    if $F \neq \perp$ and $F' \neq \perp$ **then**
      let $m = (0, \dots, 0) \in \{0,1\}^n$ be the zero vector
      for $i = 1, \dots, n$ **do**
        if $p_i \in F \cup F'$ **then** set $m_i = 1$
      end for
      return $y \oplus m$
    end if
  end for
  return $\perp$
end procedure

7.2 Decoding

In this section we will analyse decoding complexity and efficiency. For decoding we have to factor the product Eq. (7.2). First, we note that it can be written as ND^{-1} modulo q, or in other words $ED = N + sq$, $N = \prod_{i=1}^{n} p_i^{\mu_i}$, and $D = \prod_{i=1}^{n} p_i^{\nu_i}$ for some $s \in \mathbb{Z}$ and where now $\mu_i, \nu_i \in \{0,1\}$, $\mu_i \nu_i = 0$ for all i. By expanding E/q into a continued fraction we are then able to recover s/D from one of the convergents h_i/k_i for some $i \in \mathbb{N}$ under the condition that $ND < q/2$. Hence decoding always succeeds since we have chosen the primes $(p_i)_{i=1,\dots,n}$ and q such that $ND = \prod_{i \in I} p_i < q/2$ for some $I \subset \{1, \dots, n\}$ with $|I| \leq r$.

Lemma 7.1 (Correctness). *Consider the algorithms* ENCODE *(Algorithm 7.2) and* DECODE *(Algorithm 7.4). For every $r < n/2 \in \mathbb{N}, x \in \{0,1\}^n$, for every $((p_i)_{i=1,\dots,n}, q, X) \leftarrow \text{ENCODE}(r, n, x)$ and for every $y \in \{0,1\}^n$ such that $d_H(x, y) < r$ it holds that $\text{DECODE}(n, (p_i)_{i=1,\dots,n}, q, X, y) = x$.*

Proof. To see why we require $ND < q/2$, note that there exists an $s \in \mathbb{Z}$ such that $ED - sq = N$. Therefore $\left|\frac{E}{q} - \frac{s}{D}\right| = \frac{N}{qD}$. Now Theorem 6.1 asserts us that s/D is a convergent of E/q if $\left|\frac{E}{q} - \frac{s}{D}\right| < \frac{1}{2D^2}$ and so we find the requirement $ND < q/2$.

For each convergent h_i/k_i of E/q, k_i respectively $(k_i E \mod q)$ can be factored separately using the p_i to recover the ν_i and μ_i from which $x \in \{0,1\}^n$ can then finally be recovered using $y \in \{0,1\}^n$. This all works assuming $y \in B_{H,r}(x)$ since then the factors of N and D will be unique (of multiplicity 1) and contained in the sequence $(p_i)_{i=1,\dots,n}$. If now $y \notin B_{H,r}(x)$ then with high probability (dependent on r, n) the factors of N and D will not be unique and/or not contained in $(p_i)_{i=1,\dots,n}$ in which case the decoding fails. □

Remark 1. Note that our decoding algorithm can also be used to solve the problem of matching distance in $\mathbb{Z}^n$ under the ℓ_1 norm. If $X = \prod_i p_i^{x_i} \mod q$ is an encoding of $\mathbf{x} \in \mathbb{Z}^n$ and if $\mathbf{y} \in \mathbb{Z}^n$ is such that $\|\mathbf{x} - \mathbf{y}\|_1 \leq r$ then by taking continued fractions and factoring still reveals the error vector $\mathbf{e} = \mathbf{x} - \mathbf{y} \in \mathbb{Z}^n$.

Decoding Efficiency. We will now argue that decoding E is efficient. Assuming that $E = XY^{-1}$ for some $x, y \in \{0,1\}^n$ such that $d_H(x,y) < r$, one of the convergents h_i/k_i will yield s/D. From Sect. 6 we know that in our case the number of convergents we have to factor is of the order $O(\log(q))$ in the worst case. Because we fixed a list of small primes p_i beforehand, we can test for a proper convergent h_i/k_i and simultaneously factor N and D efficiently. Thus decoding is of the order $O(n \log(q))$ in the number of (modular) multiplications/divisions. By the prime number theorem we may take $q \sim (n \log n)^r$ and thus decoding is also of the order $O(nr \log(n \log n))$.

7.3 Avoiding False Accepts

We define a *false accept* to be an input y that is far from x but such that E has a smooth product representation. Recall that the obfuscator $\mathcal{O}_H$ defined in Sect. 7.1 additionally outputs $Q = \mathcal{O}_{PT}(R_x)$ which is used to prevent such false accepts. We will explain in this section that the additional step can be omitted if r is chosen such that

$$r > \log(2\sqrt{2\pi e}) \frac{n}{\log(n \log(n))} = r_f(n). \tag{7.3}$$

Let y be a false accept, which means $XY^{-1} \equiv \prod_i p_i^{\epsilon_i} \mod q$ as in Eq. (7.2), where $\epsilon_i \in \{-1, 0, 1\}$. Then $\prod_{i=1}^n p_i^{x_i - y_i - \epsilon_i} = 1 \mod q$ where $-2 \leq x_i - y_i - \epsilon_i \leq 2$. It follows that there is a non-zero vector in the lattice $\Lambda = \{x \in \mathbb{Z}^n \mid \prod_{i=1}^n p_i^{x_i} = 1 \mod q\}$ with norm bounded by $2\sqrt{n}$.

Treating the lattice Λ as a random lattice, the Gaussian heuristic (see [29, Section 7.5.3]) estimates the size of the shortest non-zero vector in Λ as $\lambda_1 \sim \sqrt{\frac{n}{2\pi e}} \mathrm{vol}(\Lambda)^{\frac{1}{n}} \leq \sqrt{\frac{n}{2\pi e}}(q-1)^{\frac{1}{n}}$, where equality holds if the p_i generate $(\mathbb{Z}/q\mathbb{Z})^*$. We use this bound to argue that there is no vector of length bounded by $2\sqrt{n}$, and hence no false accept, Indeed, to have $\lambda_1 > 2\sqrt{n}$ we need $\sqrt{\frac{n}{2\pi e}}(n \log(n))^{\frac{r}{n}} > 2\sqrt{n}$ and so

$$(n \log(n))^{\frac{r}{n}} > 2\sqrt{2\pi e}. \tag{7.4}$$

Equation (7.4) assumes that $q \sim (n \log(n))^r$, i.e. the size of the primes p_i is as small as possible. If we want to be able to use a smaller r we may also choose the primes $p_i > n \log(n)$.

7.4 Example Parameters

The parameters of the Hamming distance obfuscator can be chosen fairly flexibly. We want to emphasize that a priori any vector size $n \in \mathbb{N}$ is possible. The

actual security level of the obfuscator depends on the error parameter $r < n/2$ which we expect to be fixed by the demands of the application. Note that it is naturally bounded by Lemma 2.2. Assuming a uniform distribution of possible target vectors x, the bit-security (meaning logarithm to base 2 of the expected number of operations to find an accepting input) of a parameter set (r, n) can be calculated using $\lambda_{r,n} = -\log_2(h_r/2^n)$ where h_r is defined in Lemma 2.1. We give some example parameter sets along with their bit-security in Fig. 2.

r	$\lfloor \lambda_{r,n} \rfloor$	$\lfloor \log_2(q) \rfloor$
155	64	1804
128	102	1490
32	346	372

(a) $n = 512$ ($r_f(n) = 134$, $r_{\mathrm{PQ}}(n) = 44$)

r	$\lfloor \lambda_{r,n} \rfloor$	$\lfloor \log_2(q) \rfloor$
306	128	3915
256	199	2546
64	686	818

(b) $n = 1024$ ($r_f(n) = 244$, $r_{\mathrm{PQ}}(n) = 80$)

Fig. 2. Example parameter sets for obfuscated Hamming distance with $r < n/2$ and bit-security parameter $\lambda_{r,n}$. We estimate the size of q by $q \sim (n \log n)^r$. When $r > r_f(n)$ (see Eq. (7.3)) we do not expect false accepts, and so do not need to use the point obfuscator. When $r < r_{\mathrm{PQ}}(n)$ (see Eq. (5.3)) then the scheme is conjectured to be post-quantum secure (as long as the point obfuscator is post-quantum secure).

7.5 Performance

For completeness, we have implemented (an unoptimised version of) the Hamming distance obfuscator using the C programming language and conducted experiments on a desktop computer (Intel(R) Core(TM) i7-4770 CPU 3.40 GHz). We take $n = 511$ and $r = 85$ (i.e. $\lfloor \lambda_{85,511} \rfloor = 185$ and $\lfloor \log_2(q) \rfloor = 989$) to allow comparison with [33,42]. We measured the time to produce and decode 1000 obfuscation instances. We found an average encoding time of 52 ms and an average decoding time of 14 ms. In comparison Karabina and Canpolat [33] found 100 ms for encoding and 350 ms for decoding respectively on a similar computer (Intel(R) Xeon(R) CPU E31240 3.30 GHz). It is possible to further speed up encoding by choosing a good safe prime generating algorithm and decoding can be parallelised in the factoring steps instead of attempting to factor after computing each new convergent. Note that the data $((p_i)_{i=1,\ldots,n}, q, X)$ can be stored in less than one kilobyte.

An interactive model of program obfuscation called *token based obfuscation* was first considered by Goldwasser et al. [25] and an LWE based implementation was presented by Chen et al. [13]. They found experimentally that "For the case of the Hamming distance threshold of 3 and 24-bit strings, the TBO construction requires 213 GB to store the obfuscated program." Obfuscation took 72.6 min. Of course, the parameters $(n, r) = (24, 3)$ are much too small for the function to be evasive. Further, this problem is easily solved using a secure sketch. In comparison our scheme can be implemented with realistic parameters like $(n, r) = (511, 85)$ and requires less than a kilobyte of storage and less than a second to run. Clearly the ring-LWE approach in [13] is orders of magnitude worse than in our scheme. Also for comparison, the scheme of Bishop et al. [6]

(using the optimised variant in [4]) requires $n+1$ elements of a group with a hard discrete logarithm problem. With $n = 511$ and a group of size 2^{256} this would require at least 16 kilobytes to store the program.

7.6 Polynomial Ring Variant

There is a variant of our Hamming distance obfuscator that uses a polynomial ring over a finite field to encode binary vectors. Note that this variant works analogously for the conjunction obfuscator.

Let k be a field and let $R = k[z]$. The idea is to replace $\mathbb{Z}/q\mathbb{Z}$ by R/Q where the ideal $Q = (q(z))$ is generated by some suitable irreducible polynomial $q \in R$ of sufficiently large degree. For the ground field we may take a finite field of suitable order.

Given $r < n/2 \in \mathbb{N}$, encoding a target vector $x \in \{0,1\}^n$ follows the same process as before. We choose a random sequence of small distinct irreducible polynomials $(p_i)_{i=1,\ldots,n}$ in R and an irreducible polynomial q such that

$$\sum_{i \in I} \deg(p_i) < \deg(q)$$

for all $I \subset \{1,\ldots,n\}$ with $|I| \leq r$. To encode $x \in \{0,1\}^n$, publish $X = \prod_{i=1}^{n} p_i^{x_i} \bmod q$ along with the polynomials $(p_i)_{i=1,\ldots,n}$ and q. Given another element $y \in \{0,1\}^n$ we can check if $y \in B_{H,r}(x)$ using the encoding X. Again, to recover the errors ϵ_i we use continued fraction decomposition and factoring, though now in R. We refer to the full version of this work for details.

Comparison to $\mathbb{Z}/q\mathbb{Z}$ Case. Using polynomials has several advantages: The ground field k can be of small order since the order of R/Q is given by $|R/Q| = |k|^{\deg(q)}$ and thus controllable by the size of $q \in R$. We may furthermore choose a compact representation of the irreducibles $(p_i)_{i=1,\ldots,n}$ and q to shrink encoding size and speed up computation. Working out an exact comparisons of parameters, encoding sizes, and computational aspects we leave as a future open question.

8 Security

Here we analyse the security of our Hamming distance obfuscator. We will show distributional VBB security and that the obfuscator is input-hiding. Our results will depend on the hardness of the *distributional modular subset product problem* that was introduced in Sect. 5.

8.1 Security of the Obfuscator

To show that the Hamming distance obfuscator is a distributional VBB obfuscator, we need to show that it satisfies all the properties of Definition 4.1. Note that Definition 4.1 for VBB obfuscation is given in asymptotic terms with respect to a security parameter λ. On the other hand, Problems 5.1 and 5.2 are given in terms

of explicit parameters $r, n \in \mathbb{N}$. Thus in the following, let the parameters $r, n \in \mathbb{N}$ be implicitly dependent on the security parameter λ, i.e. $r = r(\lambda), n = n(\lambda)$. Taking $(n(\lambda), r(\lambda)) = (16\lambda, \lambda)$ gives Theorem 1.1 of the introduction.

Theorem 8.1. *Let $(n(\lambda), r(\lambda))$ be a sequence of parameters for $\lambda \in \mathbb{N}$. Let $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of Hamming distance evasive distributions (as in Definition 2.5). Suppose that D-MSP$_{r,n,D}$ (Problem 5.2) is hard and that $\mathcal{O}_{PT}$ is a dependent auxiliary input distributional VBB point function obfuscator. Then the Hamming distance obfuscator $\mathcal{O}_H$ is a distributional VBB obfuscator.*

Proof. The obfuscator is functionality preserving by Lemma 7.1. It is also clear that the obfuscator causes only a polynomial slowdown when compared to an unobfuscated Hamming distance calculation since the evaluation algorithm runs in time polynomial in all the involved parameters.

By Theorem 4.1 it is sufficient to show that there exists a (non-uniform) PPT simulator $\mathcal{S}$ such that, for every distribution ensemble $D = \{D_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{D}$, it holds that (where α denotes any auxiliary information, if required)

$$(\mathcal{O}_H(P), \alpha) \overset{c}{\approx} (\mathcal{S}(|P|), \alpha).$$

Let $\mathcal{D}$ now be a class of distribution ensembles such that each $D \in \mathcal{D}$ is a Hamming distance evasive distribution as in Definition 2.5. We will construct the simulator $\mathcal{S}$. First the simulator $\mathcal{S}$ takes as input $|P|$ and determines the parameters $r, n \in \mathbb{N}$. Then it runs Algorithm 8.1 which will generate the first half of the eventual output.

Algorithm 8.1. Encoding Simulator

procedure SIMULATEENCODE($r < n/2 \in \mathbb{N}$)
  sample random sequence of distinct primes $(p_i)_{i=1,\ldots,n}$ from $[2, O(n \log(n))]$
  sample small safe prime q such that $\forall I \subset \{1, \ldots, n\}$ with $|I| \leq r$: $\prod_{i \in I} p_i < q/2$
  sample $X' \leftarrow \mathbb{Z}/q\mathbb{Z}$ uniformly
  return $((p_i)_{i=1,\ldots,n}, q, X')$
end procedure

Lastly, the simulator $\mathcal{S}$ samples a uniformly random Q' from the codomain of $\mathcal{O}_{PT}$ (using the simulator $\mathcal{S}_{\text{PT}}$ that exists due to the assumption of $\mathcal{O}_{PT}$ being a distributional VBB obfuscator). The simulator $\mathcal{S}_{\text{PT}}$ receives the auxiliary information $((p_i)_{i=1,\ldots,n}, q, X')$ as additional input, provided by the top-level simulator $\mathcal{S}$.

Denote the simulator output by the tuple $((p_i)_{i=1,\ldots,n}, q, X', Q')$. It is clear that $\mathcal{S}$ is polynomial-time since Algorithm 8.1 is too. Finally, assuming that Problem 5.2 is hard, a real obfuscation $((p_i)_{i=1,\ldots,n}, q, X, Q)$ obtained from the Hamming distance obfuscator $\mathcal{O}_H$ described in Sect. 7.1 and the simulator output are computationally indistinguishable:

$$((p_i)_{i=1,\ldots,n}, q, X, Q) \overset{c}{\approx} ((p_i)_{i=1,\ldots,n}, q, X', Q'). \tag{8.1}$$

This completes the proof. □

Remark 2. As noted in Sect. 7.3, the obfuscator $\mathcal{O}_H$ can be modified to omit the point obfuscation step. Hence Theorem 8.1 can be restated without requiring a distributional VBB obfuscator $\mathcal{O}_{PT}$ by assuming an ensemble of Hamming distance evasive distributions $\{D_\lambda\}_{\lambda\in\mathbb{N}}$ that satisfy Eq. (7.3).

Next, we will show that the Hamming distance obfuscator is input hiding according to Definition 4.4.

Theorem 8.2. *Let $(n(\lambda), r(\lambda))$ be parameters satisfying $r > r_f(n)$ (recall Eq. (7.3)). Let $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble of Hamming distance evasive distributions (as in Definition 2.5). Suppose that $MSP_{r,n,D}$ (Problem 5.1) is hard. Then the Hamming distance obfuscator $\mathcal{O}_H$ is input hiding.*

Proof. The ensemble $\{D_\lambda\}_{\lambda\in\mathbb{N}}$ of Hamming distance evasive distributions induces an ensemble of programs $\{\mathcal{P}_n\}_{n\in\mathbb{N}}$ (see Sect. 7.1). Suppose there exists a PPT adversary $\mathcal{A}$ such that the success probability is bounded by

$$\Pr_{P\leftarrow\mathcal{P}_n}[P(\mathcal{A}(\alpha, \mathcal{O}_H(P))) = 1] \leq g(n)$$

for some function $g(n)$ (recall Definition 4.4 of an input hiding obfuscator).

We will now construct an algorithm $\mathcal{A}'$ that solves Problem 5.1 given $\mathcal{A}$ with success probability bounded by $g(n)$. Let $((p_i)_{i=1,\ldots,n}, q, X)$ be an instance of Problem 5.1. Since $r > r_f(n)$, this instance uniquely defines $x \in \{0,1\}^n$ such that $X = \prod_{i=1}^n p_i^{x_i} \mod q$, and hence defines a program P. Then $((p_i)_{i=1,\ldots,n}, q, X)$ is a correct obfuscation of P. The algorithm $\mathcal{A}'$ runs the adversary on $\mathcal{A}$ on $((p_i)_{i=1,\ldots,n}, q, X)$ and $\mathcal{A}$ outputs a vector $y \in \{0,1\}^n$ that is accepted by P with probability $g(n)$. Note that in Definition 4.4 the adversary outputs a valid input for P, not $\mathcal{O}_H(P)$. Hence, y is close to x as $r > r_f(n)$. Finally, $\mathcal{A}'$ decodes X given y using Algorithm 7.4 and thus outputs x with probability $g(n)$.

But we assumed that Problem 5.1 is hard and hence $g(n)$ is negligible. □

9 Obfuscating Conjunctions

In this section we describe a new obfuscator for conjunctions, based on the Hamming distance obfuscator of Sect. 7. Recall the notation $\chi(x)$ from Definition 3.1.

We first give a generic reduction of pattern matching with wildcards to hamming distance. Let $x \in \{0,1,\star\}^n$ be a pattern and let r be the number of wildcards. Let $x' \in \{0,1\}^n$ be any string such that $x'_i = x_i$ for all non-wildcard positions $1 \leq i \leq n$. Then it is clear that any $y \in \{0,1\}^n$ that satisfies the pattern has Hamming distance at most r from x'. The problem is that there are many other vectors y that have Hamming distance at most r from x' but which do not satisfy the pattern. Further, pattern matching with wildcards can be evasive with r as large as $n - \lambda$ where λ is a security parameter (e.g., $n = 1000$ and $r = 900$), while Hamming distance is not evasive if $r > n/2$). So it is clear that this is not a general reduction of obfuscating conjunctions to fuzzy matching.

However, in certain parameter ranges (where $r < n/2$) one can consider using fuzzy matching to give an approach to obfuscating conjunctions. As we will explain in this section, our scheme has some advantages over the generic reduction because inputs y that match the pattern are more easily identified than vectors y that are close to x' in the Hamming metric but do not match the pattern. Indeed, we will explain that, for certain parameter ranges, our approach is much more compact than other solutions to the conjunction problem.

The Conjunction Obfuscator. Let $n \in \mathbb{N}$ and $x \in \{0,1,\star\}^n$. Choose a random sequence of small distinct primes $(p_i)_{i=1,\ldots,n}$ (i.e. $p_i \neq p_j$ for $i \neq j$). It suffices to randomly sample each p_i from the interval $[(n\log(n))^2, ((n+1)\log(n+1))^2]$. Denote by $W_x = \{i \,|\, x_i = \star\}$ the set of indices such that x_i is a wildcard. Assume we can choose a safe prime q such that

$$\prod_{i \in W_x} p_i < \frac{q}{2} < \prod_{i \in W_x \cup \{j\}} p_i \tag{9.1}$$

for all $j \in \{1,\ldots,n\} \setminus W_x$. Set $r = |W_x|$; we furthermore require that $r < n/2$ as we will see shortly.

To encode x, consider the map $\sigma : \{0,1,\star\} \to \{-1,0,1\}$ that acts in the following fashion

$$0 \mapsto -1, \quad 1 \mapsto 1, \quad \star \mapsto 0.$$

Publish then

$$X = \prod_{i=1}^{n} p_i^{\sigma(x_i)} \mod q$$

along with the list of primes $(p_i)_{i=1,\ldots,n}$ and the modulus q. Note that, for this encoding to hide x, we require $\prod_{i=1}^{n} p_i^{\sigma(x_i)} > q$.

Given a vector $y \in \{0,1\}^n$ such that $\chi(y) = 1$, we compute $Y = \prod_{i=1}^{n} p_i^{\sigma(y_i)} \mod q$ from which we can immediately find

$$E = XY^{-1} \mod q = \prod_{i=1}^{n} p_i^{\sigma(x_i)-\sigma(y_i)} \mod q = \prod_{i=1}^{n} p_i^{\epsilon_i} \mod q \tag{9.2}$$

where $\epsilon_i \in \{-1,0,1\}$. We then recover the errors ϵ_i using continued fraction decomposition and factoring. The errors ϵ_i directly correspond to the wildcard positions W_x.

If $\chi(y) \neq 1$ then $y_i \neq x_i$ in some non-wildcard positions, i.e. Eq. (9.2) includes values $\epsilon_i \in \{-2,2\}$ and so decoding fails with high probability. The fact that incorrect inputs give factors $p_i^{\pm 2}$ in the product (while wildcard positions introduce simply p_i) is a nice feature that makes our scheme more secure than the generic transformation of conjunctions to Hamming matching. It means we are not reducing conjunctions to Hamming distance, but to a weighted ℓ_1-distance on $\mathbb{Z}$, where the non-wildcard positions are weighted double. Hence, even if an

attacker guesses some wildcard positions (and so does not include the corresponding p_i in their product Y), the value $XY^{-1} \mod q$ has $p_i^{\pm 2}$ terms for each incorrect non-wildcard position and so the attacker still needs to correctly guess the correct bits in most non-wildcard positions.

Obfuscator and Obfuscated Program. The conjunction obfuscator works as follows: For every conjunction $x \in \{0,1,\star\}^n$ with $|W_x| < n/2$ there exists a polynomial size program $P : \{0,1\}^n \to \{0,1\}$ that computes whether the input vector $y \in \{0,1\}^n$ matches x and evaluates to 1 in this case, otherwise to 0. Denote the family of all such programs with $\mathcal{P}$.

The conjunction obfuscator $\mathcal{O}_C : \mathcal{P} \to \mathcal{P}'$ takes one such program $P \in \mathcal{P}$ and outputs another polynomial size program in a different family $\mathcal{P}'$. In our case this is the decoding algorithm along with the polynomial size elements $(p_i)_{i=1,\ldots,n}$, q and $X \in \mathbb{Z}/q\mathbb{Z}$. The obfuscator also outputs $Q = \mathcal{O}_{PT}(R_{x'})$ where x' denotes the vector x with the wildcards replaced with 0.

We again identify the obfuscated program with the tuple $((p_i)_{i=1,\ldots,n}, q, X, Q)$. The obfuscated program outputs 1 if evaluation succeeds for an input $y \in \{0,1\}^n$ and if the program Q, when executed on the decoded conjunction with its wildcards replaced with 0, outputs 1, else the output is 0. See the full version of this work for details.

Parameters. The same considerations regarding the use of a safe prime and decoding efficiency as in Sect. 7 apply here. Let us now argue that a safe prime q which is bounded as in Eq. (9.1) exists. We use the following heuristic: The density of Sophie Germain primes is given by $\pi_{\mathrm{SG}}(n) \sim 2Cn/\log^2(n)$ for a constant $2C \approx 1.32032$ [40]. An asymptotic inverse is given by $n\log^2(n)$ and so we can expect the m-th Sophie Germain prime to be of size approximately $m\log^2(m)$. Hence, assuming that the p_i are sampled from $[(n\log(n))^2, ((n+1)\log(n+1))^2]$, we require that there exists an index $m \in \mathbb{N}$ such that

$$((n+1)\log(n+1))^{2r} < m\log^2(m) < (n\log(n))^{2(r+1)} \tag{9.3}$$

which, heuristically, we may convince ourselves to hold by considering the exponential nature of the bounding expressions in r. We refer to the full version of this work for details.

9.1 Relation to Hamming Distance

Our conjunction obfuscator construction is related to our Hamming distance obfuscator (cf. Sect. 7) and thus exhibits several limitations.

Firstly, the construction limits the number of wildcards $|W_x| < n/2$.

Secondly, due to the construction, the problem of finding a match to $x \in \{0,1,\star\}^n$ reduces to the problem of finding a vector $y \in \{-1,0,1\}^n \subset \mathbb{Z}^n$ such that $\|\sigma(x) - y\|_1 < |W_x|$. Note that we took the representatives of $\mathbb{Z}/3\mathbb{Z}$ to be $\{-1,0,1\}$ such that the wildcard primes never appear as factors of X. We may

compute the number of possible vectors in an ℓ_1-ball of radius r ($0 \leq r \leq n(q-1)$ for $\mathbb{Z}/q\mathbb{Z}$) using $|B_{1,q,r}| = \sum_{k=0}^{r} \left\langle {n \atop k} \right\rangle_q$ where $\left\langle {n \atop k} \right\rangle_q = \sum_{i=0}^{n} (-1)^i \binom{n}{i} \binom{k+n-1-iq}{n-1}$ is the *q-nomial triangle*. The upper limit of the sum may actually be taken as $\lfloor (k+n-1)/q \rfloor$ instead of n. The symbol $\left\langle {n \atop k} \right\rangle_q$ counts the number of compositions of k into n parts p_i such that $0 \leq p_i \leq q-1$ for each p_i [22].

Finally, an input conjunction $x \in \{0, 1, \star\}^n$ needs to be evasive. Assuming a uniform conjunction, this will be the case if $|B_{1,3,r}|/3^n < 1/2^\lambda$ is negligible.

9.2 Parameter Choices

In Sect. 9.1 we have learned that the possible parameter choices of the conjunction obfuscator are more limited. Assuming a uniform and evasive conjunction distribution, we find from Lemma 3.1 and Sect. 9.1 that the bit-security is given by $\lambda_{r,n} = \min\{n - r, -\log_2(|B_{1,3,r}|/3^n)\}$. On the other hand, the conjunction obfuscators given in [4,6] allow for a wider range of $r < n - O(\log(n))$ at the cost of assuming a generic group model.

Brakerski et al. [10] give no estimate of encoding size or security parameters for their graded coding scheme based obfuscator. Chen et al. [13] found experimentally that "The TBO of 32-bit conjunctions is close to being practical, with a total evaluation runtime of 11.6 ms, obfuscation runtime of 5.1 min, and program size of 11.6 GB for a setting with more than 80 bits of security." This program size and obfuscation time is orders of magnitude worse than the encoding size of our scheme or the schemes of [4,6].

9.3 Security

Showing security of the conjunction obfuscator works in essentially the same way as for the Hamming distance obfuscator in Sect. 8.1. Note that Problem 5.1 respectively Problem 5.2 also makes sense when the distribution D is considered to be over $\{-1, 0, 1\}^n$ instead of $\{0, 1\}^n$. Note that Remark 2 applies to Theorem 9.1 as well.

Theorem 9.1. *Let $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of conjunction evasive distributions (as in Definition 3.2). Suppose that D-MSP$_{r,n,D}$ (Problem 5.2) with the distribution D over $\{-1, 0, 1\}^n$ is hard and that $\mathcal{O}_{PT}$ is a dependent auxiliary input distributional VBB point function obfuscator. Then the Conjunction obfuscator $\mathcal{O}_C$ is a distributional VBB obfuscator.*

Theorem 9.2. *Let $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of conjunction evasive distributions (as in Definition 3.2). Suppose that MSP$_{r,n,D}$ (Problem 5.1) with the distribution D over $\{-1, 0, 1\}^n$ is hard for $r > r_f(n)$ (recall Eq. (7.3)). Then the Conjunction obfuscator $\mathcal{O}_C$ is input hiding.*

10 Conclusion

We have introduced a new special purpose obfuscator for fuzzy matching under Hamming distance as well as a new special purpose obfuscator for conjunctions. We have shown that our obfuscators are virtual-black-box secure and input hiding, based on the search and decision versions of the distributional modular subset product problem. We believe our obfuscators are post-quantum secure. The Hamming distance obfuscator can cover a wider range of parameters than previous solutions based on secure sketches.

Open problems include finding optimal parameters. More speculative open problems include obfuscating fuzzy matching with respect to edit distance or other metrics.

Acknowledgements. We thank Trey Li for several corrections and comments. We thank the Marsden Fund of the Royal Society of New Zealand for funding this research, and the reviewers for suggestions.

References

1. Alon, N., Spencer, J.H.: The Probabilistic Method. Wiley, New York (1992)
2. Barak, B., Bitansky, N., Canetti, R., Kalai, Y.T., Paneth, O., Sahai, A.: Obfuscation for evasive functions. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 26–51. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_2
3. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
4. Bartusek, J., Lepoint, T., Ma, F., Zhandry, M.: New techniques for obfuscating conjunctions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11478, pp. 636–666. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_22
5. Bellare, M., Stepanovs, I.: Point-function obfuscation: a framework and generic constructions. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 565–594. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_21
6. Bishop, A., Kowalczyk, L., Malkin, T., Pastro, V., Raykova, M., Shi, K.: A simple obfuscation scheme for pattern-matching with wildcards. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 731–752. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_25
7. Bitansky, N., et al.: The impossibility of obfuscation with auxiliary input or a universal simulator. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 71–89. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_5
8. Bitansky, N., Paneth, O.: Point obfuscation and 3-round zero-knowledge. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 190–208. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_11
9. Brakerski, Z., Rothblum, G.N.: Obfuscating conjunctions. J. Cryptol. **30**(1), 289–320 (2017)

10. Brakerski, Z., Vaikuntanathan, V., Wee, H., Wichs, D.: Obfuscating conjunctions under entropic ring LWE. In: 2016 ACM Conference on Innovations in Theoretical Computer Science, pp. 147–156. ACM (2016)
11. Bringer, J., Chabanne, H., Cohen, G., Kindarji, B., Zemor, G.: Theoretical and practical boundaries of binary secure sketches. IEEE Trans. Inf. Forensics Secur. **3**(4), 673–683 (2008)
12. Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of hyperplane membership. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 72–89. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_5
13. Chen, C., Genise, N., Micciancio, D., Polyakov, Y., Rohloff, K.: Implementing token-based obfuscation under (ring) LWE. Cryptology ePrint Archive, Report 2018/1222 (2018). https://eprint.iacr.org/2018/1222
14. Chernoff, H., et al.: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Ann. Math. Stat. **23**(4), 493–507 (1952)
15. Contini, S., Lenstra, A.K., Steinfeld, R.: VSH, an efficient and provable collision-resistant hash function. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_11
16. Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: Improved low-density subset sum algorithms. Comput. Complex. **2**(2), 111–128 (1992)
17. Dixon, J.D.: The number of steps in the Euclidean algorithm. J. Number Theory **2**(4), 414–422 (1970)
18. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. SIAM J. Comput. **38**(1), 97–139 (2008)
19. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_31
20. Dodis, Y., Smith, A.: Correcting errors without leaking partial information. In: STOC 2005, pp. 654–663. ACM (2005)
21. Ducas, L., Pierrot, C.: Polynomial time bounded distance decoding near minkowski's bound in discrete logarithm lattices. Des. Codes Crypt. **87**(8), 1737–1748 (2019)
22. Fielder, D.C., Alford, C.O.: Pascal's triangle: top gun or just one of the gang? In: Bergum, G.E., Philippou, A.N., Horadam, A.F. (eds.) Applications of Fibonacci Numbers, pp. 77–90. Springer, Dordrecht (1991). https://doi.org/10.1007/978-94-011-3586-3_10
23. Fuller, B., Reyzin, L., Smith, A.: When are fuzzy extractors possible? In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 277–306. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_10
24. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM J. Comput. **45**(3), 882–929 (2016)
25. Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: STOC 2013, pp. 555–564. ACM (2013)
26. Goyal, R., Koppula, V., Waters, B.: Lockable obfuscation. In: FOCS 2017, pp. 612–621. IEEE (2017)

27. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 4th edn. Oxford University Press, Oxford (1975)
28. Hensley, D.: The number of steps in the Euclidean algorithm. J. Number Theory **49**(2), 142–182 (1994)
29. Hoffstein, J., Pipher, J., Silverman, J.H.: An Introduction to Mathematical Cryptography. UTM. Springer, New York (2014). https://doi.org/10.1007/978-1-4939-1711-2
30. Hurwitz, A.: Über die angenäherte darstellung der irrationalzahlen durch rationale brüche. Math. Ann. **39**(2), 279–284 (1891)
31. Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. J. Cryptol. **9**(4), 199–216 (1996)
32. Jain, A.K., Nandakumar, K., Nagar, A.: Biometric template security. EURASIP J. Adv. Signal Process. **2008**, 113 (2008)
33. Karabina, K., Canpolat, O.: A new cryptographic primitive for noise tolerant template security. Pattern Recogn. Lett. **80**, 70–75 (2016)
34. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. J. ACM **32**(1), 229–246 (1985)
35. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982)
36. Li, Q., Sutcu, Y., Memon, N.: Secure sketch for biometric templates. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 99–113. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_7
37. Lynn, B., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_2
38. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_26
39. Brier, E., Coron, J.-S., Géraud, R., Maimuţ, D., Naccache, D.: A number-theoretic error-correcting code. In: Bica, I., Naccache, D., Simion, E. (eds.) SECITC 2015. LNCS, vol. 9522, pp. 25–35. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27179-8_2
40. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press, New York (2009)
41. Sutcu, Y., Li, Q., Memon, N.: Protecting biometric templates with sketch: theory and practice. IEEE Trans. Inf. Forensics Secur. **2**(3), 503–512 (2007)
42. Tuyls, P., Akkermans, A.H.M., Kevenaar, T.A.M., Schrijen, G.-J., Bazen, A.M., Veldhuis, R.N.J.: Practical biometric authentication with template protection. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 436–446. Springer, Heidelberg (2005). https://doi.org/10.1007/11527923_45
43. Wee, H.: On obfuscating point functions. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 523–532. ACM, New York (2005)
44. Wichs, D., Zirdelis, G.: Obfuscating compute-and-compare programs under LWE. In: FOCS 2017, pp. 600–611. IEEE (2017)